ROBUST SCHEDULING METHODOLOGY TO REDUCE RISK IN AEROSPACE PRODUCTION SYSTEMS

A Dissertation
Presented to
The Academic Faculty

by

Dennis J.L. Siedlak

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Aerospace Engineering

Georgia Institute of Technology
December 2016

ROBUST SCHEDULING METHODOLOGY TO REDUCE RISK IN AEROSPACE PRODUCTION SYSTEMS

Approved by:

Professor Dimitri Mavris,
Committee Chair
School of Aerospace Engineering
Georgia Institute of Technology

Professor Daniel Schrage
School of Aerospace Engineering
Georgia Institute of Technology

Professor Graeme Kennedy
School of Aerospace Engineering
Georgia Institute of Technology

Doctor Olivia Pinon-Fischer
School of Aerospace Engineering
Georgia Institute of Technology

Don Farr
Boeing Research & Technology
The Boeing Company

Date Approved: 7 November 2016

ACKNOWLEDGEMENTS

I would like to thank the many people who have helped me throughout this dissertation process. First, Prof. Dimitri Mavris, my advisor and thesis committee chairman. Thank you for bringing me to ASDL and giving me the freedom and opportunity to explore this dissertation topic. I have truly learned a great deal throughout this experience. Further, I would like to thank the rest of my thesis committee, Prof. Daniel Schrage, Prof. Graeme Kennedy, Dr. Olivia Pinon-Fischer, and Don Farr, for their input and support during the process.

To Olivia especially, thank you for the support and encouragement throughout. You helped me so much to formulate my ideas and, most importantly, graduate in a reasonable amount of time. Thank you for reading everything when it was not very intelligible and helping to make this document somewhat readable. Also, to Don, thank you for supporting the project that led to this thesis and providing guidance throughout. I would also like to thank everyone else within Boeing who supported the project, especially Phil and Scott.

I would also be remiss without thanking my family for the support and encouragement throughout these last 4 years. Particularly, thanks to my parents for enabling me to pursue my doctorate while also spurring me to finish. Also, to my sister, for graduating from law school in May 2017 and pushing me to get a job before her. It was nice to know that I had support but also a clear goal.

Thanks to my friends at Georgia Tech. I enjoyed meeting everyone and having people to bounce ideas off of. I'm also glad we were able to talk about things other than Aerospace (I was a bit worried after quals). Specifically, thanks to Seth for being my roommate (and fellow Pittsburgh fan) the past 3 years. I truly value everyone's

friendship and hope to continue being friends throughout our lives.

Ultimately, I have enjoyed *many* parts of this PhD process, and I am excited to move forward with the experiences and friendships that I have gained. Thank you again to everyone who has supported me in one way or another, and I look forward to seeing and taking part in what everyone accomplishes!

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

NOMENCLATURE

Δf	Change in objective function value between the current and candidate point
G	Average normalized distance to the true Pareto frontier
S	Multi-objective optimization spacing
T_k	Current temperature for simulated annealing algorithm
$x_{criticality}$	Mission criticality limit of a sensor for it to be re-installed after a failure
x_{labor}	Number of technicians available to install sensors
x_i	The primary production process during which sensor i is installed
AMS/ABM	Agent-based Simulation/Modeling
API	Application Program Interface
CAD	Computer-Aided Design
CPM	Critical Path Method
DES	Discrete-event Simulation
DoD	Department of Defense
ERP	Enterprise Resource Planning
FIFO	First In, First Out
GA	Genetic Algorithm
IE	Industrial Engineer
IPPD	Integrated Product and Process Development

LAI	Lean Aircraft Initiative
LP	Linear Programming
MC	Monte Carlo
MCDM	Mulit-Criteria Decision Making
ME	Manufacturing Engineer
MWR	Most Work Remaining
NSGA-II	Non-dominated Sorting Genetic Algorithm II
OEM	Original Equipment Manufacturer
OOP	Object-Oriented Programming
PERT	Program Evaluation and Review Technique
PLM	Product Lifecycle Management
PORRTSS	Production Optimization to Reduce Risk Through Simulation-based Scheduling
QA	Quality Assurance
RP	Pareto frontier relative improvement
SA	Simulated Annealing
SD	System Dynamics
SPTF	Shortest Processing Time First
TOPSIS	Technique for Order Preference by Similarity to Ideal Solution
TS	Tabu Search
WIP	Work-in-Process

SUMMARY

The knowledge and value gained from collecting data and being able to monitor vehicles' performance, safety, reliability, etc. have resulted in a sharp increase in the number of sensors being installed on modern aerospace vehicles. Sensor installations, which are commonly performed manually, lead to increased risk for installation errors and quality issues. These disruptions, in turn, contribute to the program cost overruns, increased schedule risk, and production delays seen throughout the industry. As such, reducing the risk and impact of manual installation tasks on aerospace production flows is becoming increasingly important for such highly schedule- and cost-constrained vehicles.

Robust scheduling methodologies, which aim to build schedules with reduced risk of cost or time overruns by minimizing the impact of disruptions, have the potential to meet the requirements of these scheduling problems. Despite the benefit to be gained by implementing robust, detailed project scheduling methodologies, traditional, deterministic strategies still tend to dominate the industry. Two research challenges must be overcome to support the implementation of such robust scheduling techniques in an industrial setting: First, the project scheduling methodologies in use today struggle to model and optimize real-world systems. The increasing complexity of modern aerospace vehicles is only going to exacerbate these difficulties. Second, the transition of new planning and scheduling practices from academia to an industrial setting is commonly challenging. Moreover, this transition is not generally discussed alongside the development of new methods.

To address these challenges, this dissertation focuses on the development, implementation, and evaluation of a new planning methodology, named PORRTSS: Production Optimization to Reduce Risk Through Simulation-based Scheduling. A representative case study is used to test the methodology's capability to model a real production environment and search for improved scheduling options. The case study involves planning sensor installation processes within a provided production schedule to reduce the risk of production delays. Traditional scheduling techniques provide a strong framework to plan and optimize, at a medium level of detail, the completion of primary production processes (e.g. structural assembly, system integration, etc.). However, fully defining the interactions and logic required to evaluate the impact resulting from the sensor installations in this scheduling framework is challenging. The discrete-event simulation paradigm simplifies the definition of these production rules and constraints; however, DES models commonly require too much detail, modeling effort, and optimization time/resources to be useful during pre-production planning. The developed methodology addresses this gap by integrating the process optimization strengths of scheduling with the modeling flexibility of simulation. This enables the fast generation of a limited fidelity simulation that can evaluate the impact of sensor installations to support simulation-based schedule optimization.

Even with an optimization framework in place, the deployment of scheduling methodologies developed in academia to an industrial setting remains challenging. A primary barrier that limits the implementation of developed scheduling practices is poor interactions between the system and the human planners. The developed methodology works to overcome these challenges by: 1) increasing the transparency of the planning process, 2) improving collaboration among the stakeholders, and 3) enabling the stakeholders to directly modify the sensor installation plan. Increased transparency and improved collaboration is achieved by developing a decision-support

tool that provides both system- and detailed-level views of the planning results. Finally, this research does not claim to provide *the* answer, but instead, recognizes that there may be additional "soft" constraints. As such, it also provides planners with the capability to make manual modifications to the optimized production plans. This ultimately leads to a more implementable and beneficial planning methodology when compared to the many rigid methods developed in academia.

The PORRTSS methodology begins by identifying process constraints contained within a provided medium-level production plan. This schedule is accepted as truth, and the identified process constraints are utilized to automatically generate a baseline discrete-event simulation. The DES model contains process logic to control sensor installation processes, and using this logic, the simulation can estimate the impact of a parametrically defined sensor installation plan.

With a parametric model in place, a multi-objective, meta-heuristic optimization algorithm (Non-dominated Sorting Genetic Algorithm-II) is linked to the simulation. The optimizer sets the locations within the primary production plan during which each sensor is installed. The flexible nature of the optimization routine enables the inclusion of a variety of objective functions, including process time and heuristic risk metrics. Once convergence is achieved, the resulting non-dominated points are fed into a data analysis and decision support environment.

The decision making system is included to support the implementation of the methodology. An initial system-level view and ranking algorithm enable SMEs to quickly identify points of interest. These can then be compared in more detail to identify similarities and differences between the selected plans. A detailed Gantt chart is also utilized to improve transparency and help understand the reasons for potential problems. Finally, the user is able to make manual modifications to a selected plan to include any additional knowledge or understanding.

With the PORRTSS methodology in place, the following experiments are performed to test its ability to overcome the aforementioned research challenges. Focusing on the first research challenge, the difficulty in modeling and optimizing the systems of interest, the appropriateness of the simulation logic and model generation strategy is tested. This is accomplished by generating a model from a schedule for a major subassembly of a "real-world" aerospace vehicle. The simulation is shown to appropriately match the baseline schedule and estimate the impact of parametrically defined sensor installations. The appropriateness of the optimization integration is then tested by linking the NSGA-II to the simulation model. An initial experiment conducted with deterministic simulation evaluations is shown to improve the objectives of interest in a short time, which demonstrates that the optimization strategy is effective for the problem of interest. This experiment is then expanded to investigate the impact of directly evaluating the risk in a schedule. This is accomplished by allowing for uncertainty in the simulation and running multiple replication per case to estimate the output distribution of the process time. The optimization is then seeded with generation 500 from the deterministic run and evaluated with this additional information. When comparing the optimization runs with and without the robustness information, the optimization considering robustness is shown to make immediate improvements to the population, especially in the process time risk. While this indicates that the additional information is effectively utilized by the optimization routine, a significant number of further runs are needed to make a generalized conclusion. Despite this, these experiments demonstrate that the developed methodology is able to effectively model and efficiently optimize the sensor installation plan.

To address the second research challenge, strategies to improve the "implementability" of the developed methodology are investigated. To test the scalability of the

methodology, the modeling and optimization strategy are applied to a range of problem sizes and complexities. The results demonstrate that the methodology is capable of handling models representative of the largest size expected to be seen in an industrial setting. Alternative, point-solution optimization algorithms are also investigated to attempt to improve the optimization's speed. These are shown to perform adequately when optimizing the deterministic model; however, when considering the stochastic model, their performance does not appear to leverage the schedule risk evaluations provided. Therefore, it cannot be shown that the point-solution algorithms expand the applicability of the methodology.

A final experiment is conducted to investigate whether the decision-making environment increases the acceptance and deployability of the methodology. Results generated by the NSGA-II for a real-world planning problem are propagated to the decision-making tool. This tool and results are then provided to the industrial engineers, manufacturing engineers, and avionics experts to down-select to a final plan for execution. Following this exercise, the SMEs confirmed the feasibility of the provided plans and leveraged the decision-making system to down-select and compare scenarios of interest. Overall, the real-world implementation demonstrates that the inclusion of the decision-support environment increased transparency and acceptance of the methodology.

From these results, the PORRTSS methodology is shown to overcome the two identified research challenges. The modeling and optimization strategy enables the automatic generation and evaluation of feasible alternative installation plans. The inclusion of heuristic robustness metrics and process time risk enables the identification of more robust installation plans. Then, the transparency and freedom provided by the decision-support system is shown to increase the approachability and deployability of the methodology. Ultimately, this methodology enables the replacement of

a manual planning process with one that can better estimate and reduce the system-level impact of small installation steps, which can be large contributors to the time to complete a schedule.

CHAPTER I

INTRODUCTION

Aerospace systems of the late 1990s and 21st century have seen a sharp increase in complexity (as seen in Figure 1) compared to the programs of the Cold War era [204]. As an example, the F-35 has 3 to 8 times the operational capability of an F-16. However, to achieve such increased capabilities requires 130 subsystems with 90% of its functions managed by software. This is in contrast to the 15 subsystems with 40% software managed functions on the F-16 [21]. As seen in Figure 2, the increase in complexity has been accompanied by considerable cost overruns [11, 21, 204] and schedule delays for many major programs executed during this century, including the F-35 Joint Strike Fighter [143], Boeing 787, and Airbus 380 [133].

Despite the implementation of methodologies such as Product Lifecycle Management (PLM) [168], Integrated Product and Process Development (IPPD), and the

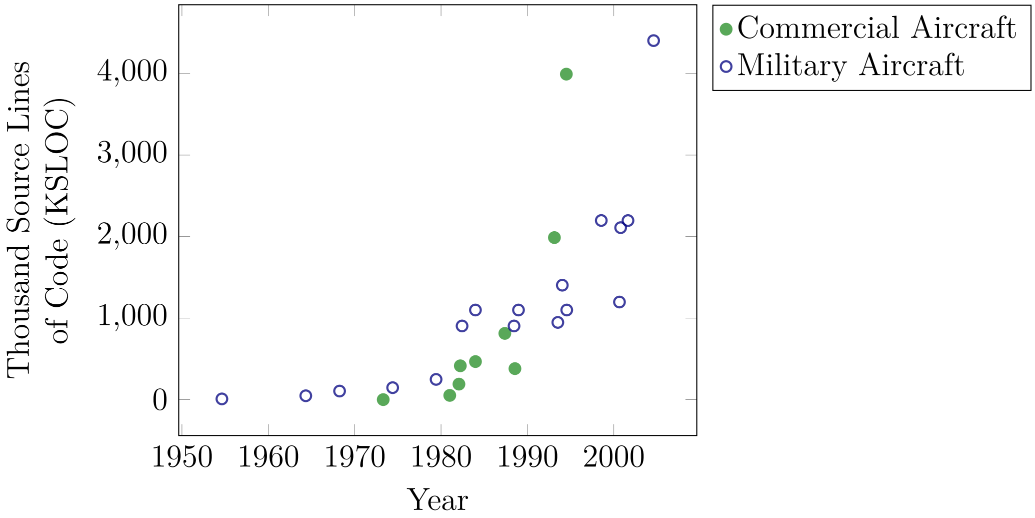


Figure 1: Complexity of Aerospace Systems (as Measured by Source Lines of Code) [5]

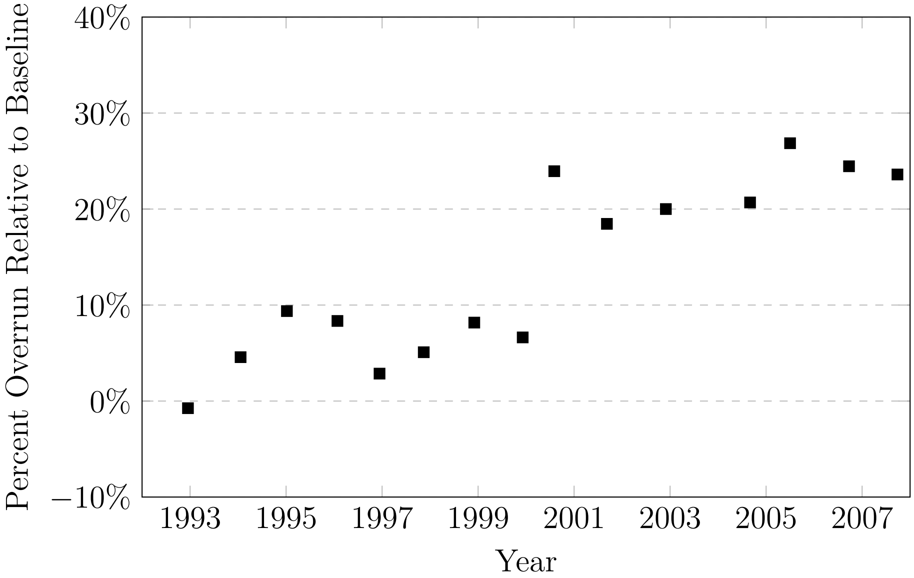


Figure 2: Total Cost Overruns for Major US Department of Defense Aerospace Programs Since 1993. Reproduced from [35].

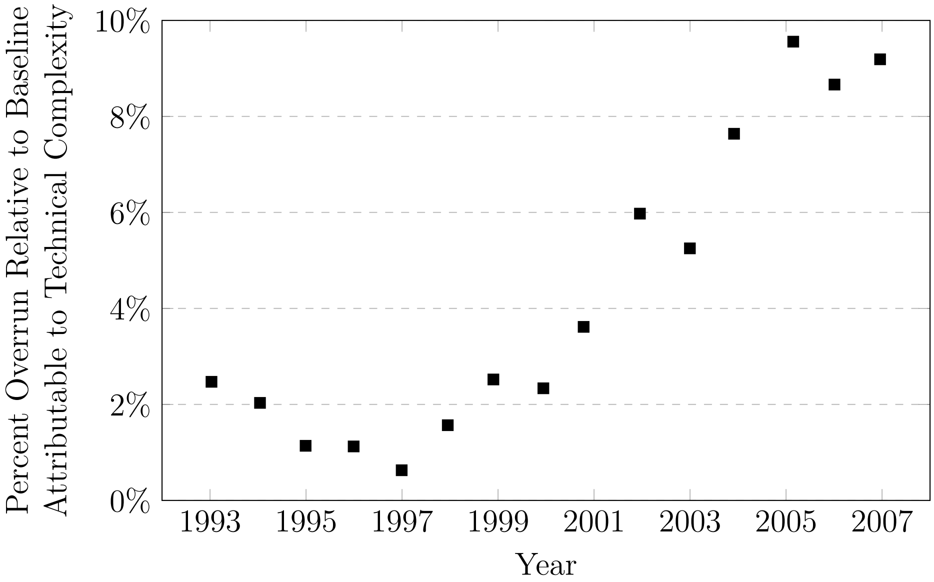


Figure 3: Total Cost Overruns for Major US Department of Defense Aerospace Programs Attributable to Technical Complexity. Reproduced from [35].

Lean Aircraft Initiative (LAI) [184], which were developed to control costs of increasingly complex systems [2], the average percent overrun attributable to complexity for the U.S. Department of Defense (DoD) portfolio has grown steadily in the last 2 decades. As seen in Figure 3, complexity has accounted for an increasing percentage of cost overruns since 1997. By 2007, complexity had accounted for approximately 1/3 of the observed cost overruns in the portfolio [35]. In addition, the average delay to delivery of initial operating capabilities past the baseline for DoD programs was 28.9 months in 2014 [173], which is up from the 21 months average in 2007 [35]. This leads to the following observation:

Observation 1

Increasing system complexity is contributing to rising program delays and cost overruns.

This increase in complexity leads to significant challenges during manufacturing and production [58, 60, 137]. Problems occurring during the completion of tasks and processes involved in the delivery of goods and services contribute to the observed cost overruns and schedule delays. Increasingly complex systems require tighter manufacturing tolerances to achieve the desired performance [35]. Tight tolerances, in turn, lead to increased quality assurance (QA) inspections, more frequent tool replacement and calibration, and higher rates of scrap and rework [26].

Furthermore, the knowledge and value gained from collecting data and being able to monitor vehicles' performance, safety, reliability, etc. have resulted in a sharp increase in the number of sensors being installed on modern aerospace vehicles (Table 1) [85, 131, 141, 158, 169]. Integrating those sensors requires that additional production steps be dedicated to their installation [13]. For complex aerospace vehicles, such as space launch vehicles or commercial aircraft, sensor installations are usually performed manually and present many challenges in terms of accessibility and precedence constraints [116]. With manual installations also come even further risk for installation errors and quality issues [43, 171], all of which contribute to production disruptions. Hence, while integrating sensors onto a vehicle provides valuable data, it also contributes to the increasing complexity of the newer generations of aerospace vehicles, and the resulting program cost overruns [11, 21, 204], increased risk, and production delays [13, 58, 171]. As such, reducing the risk and impact of manual installation tasks on low volume aerospace production flows is becoming increasingly important for such highly schedule- and cost-constrained vehicles [41, 176].

Table 1: Examples of Increased Sensors and Electronics on Current Generation Aerospace Vehicles

System	**Sensors & Electronics Employed**	**Purpose of Including Electronics**
Airbus 380	25,000 Sensors [162] requiring 100,000 wires [43]	• Better anticipated maintenance • Faster delivery of replacement parts • Improved on-time performance • Reduced flight and maintenance costs [3]
Boeing 787	Generates about 500 GB of data throughout a flight [162]	• Improved Maintenance and operational decisions • Streamline data management [74]
Space Launch System (SLS)	Over 700 development instrumentation channels collecting data [82]	• Gather information for: – Testing systems – Ground operations – Navigation and guidance – Validating environmental and vehicle models (during the first flight) [82]
Orion Multi-Purpose Crew Vehicle	Approximately 1200 sensors on-board during Exploration Flight Test 1 [65]	• Provide information about conditions aboard the vehicle throughout the flight [65] • Validate models • Test systems [1]

The production problems experienced by the Airbus 380 exemplify some of these challenges. The A380, which has 100,000 wires (40,000 more than Airbus's next largest A340-600), experienced delivery delays of about 2 years due to wiring issues. Fuselage sections were supposed to be delivered to the final integration site with all of the electrical cabling installed; however, when they arrived, generally only half of the work had been completed. In addition, many wires were too short because the

German and French design teams used different versions of the same CAD software. The planning of the cable installation work orders also was often not done correctly such that cables that were installed one day needed to be removed on the next day [43].

The aerospace industry has recognized the problems emerging from increased complexity and has responded by devoting significant effort to generating optimized production plans that deterministically sequence production tasks [34, 199]. Techniques such as linear programming and heuristic based optimization have proven effective and, when used alongside techniques such as PLM and IPPD, helped to bring about the successful programs of the 1990s [61, 79, 148, 168]. However, the successful design and scheduling strategies from the 1990s have not brought about similar successes in the first decade of the 21st century. For OEMs, it has been shown that as systems' complexity continues to increase, especially related to the number and complexity of electronic systems and sensors, the traditional schedule optimization techniques are losing effectiveness [13]. This leads to the following observation:

Observation 2

Increasingly complex systems are challenging traditional schedule optimization approaches.

The following section further discusses the reasons for the lack of effectiveness of traditional scheduling approaches and how this partially contributes to cost and schedule overruns.

1.1 Lack of Schedule Robustness

Scheduling is defined as "an optimization approach by which limited resources are allocated over time among parallel and sequential activities [15]." System complexity can be measured by the quantity of components within the system [83, 204]. Hence, as system complexity increases, OEMs are faced with an increasing number of sensors or other subcomponents to install during the production process. As seen in Table

1, every Boeing 787 flight generates approximately 500 GB of data [55], and the Orion Multi-Purpose Crew Vehicle gathered data from approximately 1200 sensors during its Exploration Flight Test 1 [65]. In the past, the installations of sensors and electronic systems were accomplished during a planned integration phase, which was acceptable because the number of these components was relatively small. However, it is shown that the increasing complexity of modern systems leads to an increasing number of installation tasks and processes required to produce the system [21, 43]. Hence, as the installation of sensors and electronic components begins to represent a more significant block of time, this traditional approach is becoming less viable in the context of highly cost- and schedule constrained-programs.

Schedule disruptions, such as machine failure, job priority or due date change, quality issue, variation in processing time, or operator absenteeism [19, 81, 113], increase as process steps are added to the manufacturing system. Traditional approaches to mitigate these problems, such as planning slack time in the process [79] or adding additional emergency capacity [76], are becoming infeasible as the schedule and cost constraints tighten. Therefore, identifying opportunities to reduce the risk (e.g. the probability of a cost or time overrun) within the schedule while limiting the additional time and cost required to complete the process is becoming vital.

Robust design methodology has the potential to meet the requirements of these scheduling problems. Building from the work of Taguchi, the goal of robust design is to design a system that is insensitive to noise factors that are difficult or impossible to affordably control [24, 102, 177]. Robust scheduling aims to build schedules that minimize the impact of disruptions and reduce differences between the plan and execution [14, 113, 115]. For the purposes of this work, *risk* is defined as the probability that a schedule requires more time, money, or resources to complete than planned or available. The *robust scheduling methodology* is developed to apply the ideas from robust design and robust scheduling to reduce this risk. In the language of robust design, the control variables define the schedule and resource allocation. The noise variables include individual process time variation, random quality issues arising during the schedule's completion, supplier delays, etc. Real processes never go completely according to plan, and by not accounting for situations when processes take longer than planned, resources go off-line, or quality problems arise, the identified detailed, optimized schedules can quickly become impractical or infeasible to implement [19,79,108,149]. In the case of the A380, for example, implementing a schedule planned to meet the delivery deadlines did not prove effective when the wiring issues arose and required additional effort. While most problems cannot be solely attributed to a lack of robustness, improving the process planning to better account for uncertainty has the potential to reduce cost, schedule overruns, and risk [41, 176]. This leads to the following assertion:

Assertion 1

Robust scheduling techniques can help alleviate some of the increasing costs and delays experienced by the aerospace industry.

In other words, to help reduce system costs and delivery delays, robust design principles should be integrated with traditional scheduling practices. In order to get a better understanding of the issues and challenges of doing so, it is important to first discuss current scheduling strategies and their limitations.

1.1.1 Current Scheduling Strategy Overview

The scheduling strategy for a static (the number of jobs or tasks is fixed), deterministic (all processing times and resource availability is known and fixed) [61] problem commonly used throughout industry today is summarized in Figure 4. The process starts by defining a set of tasks, required resources, and precedence relations. This information is then fed into a schedule model that is able to take a sequence of tasks and estimate the time to complete the schedule along with its associated costs [149].

This model is linked to a deterministic optimization routine that is able to generate a well-performing schedule that respects any manpower or resource constraints. This optimizer commonly leverages linear programming and/or heuristic rules to optimize the sequence and does so without accounting for any variation in task completion time or resource capacity [8, 79, 214].

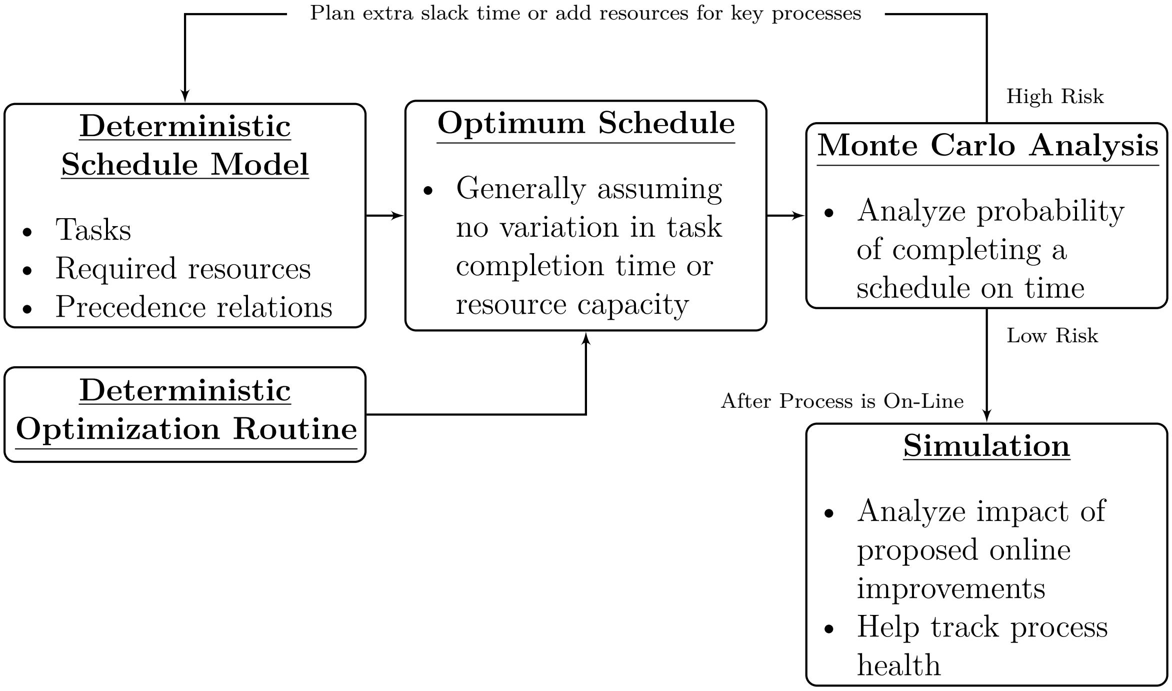

Plan extra slack time or add resources for key processes
High Risk
Deterministic Schedule Model
Tasks
Required resources
Precedence relations
Optimum Schedule
Generally assuming no variation in task completion time or resource capacity
Monte Carlo Analysis
Analyze probability of completing a schedule on time
Low Risk
Deterministic Optimization Routine
After Process is On-Line
Simulation
Analyze impact of proposed online improvements
Help track process health

Figure 4: Current Schedule Modeling Strategy

After a trial optimum schedule is generated, it may be run though a Monte Carlo analysis to estimate the probability of completing the schedule on time [138]. If the analysis concludes that the schedule has a large amount of risk (typically defined as a probability of tardiness [81, 200]), the planner will return to the initial schedule model to include additional slack time or resources around key processes [79]. This slack time, which is sometimes used to indicate the robustness of the schedule [8], gives additional time for critical tasks to be completed while providing down time for key resources to accept any unexpected jobs that may arise [79, 101]. With the additional time planned, the optimization is rerun and another round of Monte Carlo analysis is performed. If the schedule is then deemed to be low risk, the schedule is implemented. By planning this extra time or resources, the risk of completing the schedule within the projected time may decrease, however it does not necessarily decrease the total time to completion (the decision maker may have added to the nominal schedule completion time to increase robustness).

If the process being scheduled is repeated often (such as in a production environment), then a simulation of the process may be built after it comes online. The simulation may be used to analyze proposed improvements to the production flow in an effort to convince management to incorporate necessary changes [6, 79]. Hence, the simulation, by better accounting for the uncertainty present in the process and providing a clearer picture of the interactions within the process flow, allows the planner to assess and articulate the impact the online modifications may have on cost and risk [18, 57, 58, 91, 99, 100, 110, 119]. However, because the process is already online and large resources are fixed, significant changes and reconfigurations are usually either not considered or would be extremely expensive/time-consuming to implement [106, 107].

The previous sections presented the limitations of current scheduling strategies and further discussed the need to better account for uncertainty and robustness to decrease cost and risk. Despite the acknowledged shortcomings of deterministic scheduling, such practices still dominate the industry [128]. Consequently, barriers need to be identified and alleviated to support the implementation of robust scheduling.

1.1.2 Challenges to Integrating Robust Design Principles with Scheduling Strategies

Limitations to better deploying robust scheduling within an industrial setting are:

- Scheduling typically relies on analytical models. However, many “real-world” problems or systems cannot be described using a closed-form, mathematical formulation [17,18,23,57,79,100,110]. Hence, even modeling complicated problems using traditional scheduling techniques can be difficult [79].

- For problems or systems that can be modeled, the vast majority of software tools and developed methodologies used for scheduling purposes today focus on the deterministic problem [128]. As such, they fail to perform the uncertainty analysis required to quantify a system's robustness [115]. This is a significant limitation because schedules that are optimized "with respect to deterministic assumptions deteriorate quickly with the introduction of uncertainty [108]."

- The learning curve for any new tools is typically very high [96]. Traditionally, scheduling and simulation have been viewed as separate disciplines [79], so integrating any type of simulation tool useful for robust scheduling methodologies often requires significant training.

- Integrating simulation/optimization results into existing scheduling, accounting, and other enterprise programs is also difficult. Significant effort has been devoted to integrating current scheduling tools with companies' enterprise resource planning (ERP) software so that the impact of any changes to the schedule can be quickly understood and propagated. Because the current systems are so intertwined, completely replacing the currently implemented scheduling software could lead to significant integration challenges [134, 142].

- Many schedule optimization routines currently in use are geared towards deterministic optimization. These routines have limited applicability to problems containing uncertainty, so they must be modified or replaced to accommodate robustness analysis [19].

- While a wide variety of scheduling strategies have been developed within academia, only a small amount of advances in scheduling have been transitioned to practice in the "real world" [60, 128, 185]. One reason for this disconnect is that a significant amount of scheduling research has been focused on mathematically rigorous solutions to very specific problems as opposed to the development of more deployable, approachable scheduling systems [60,128,214].

These challenges are discussed and substantiated in the following chapter. However, one can already note that the challenges to integrating new tools to enable robust design within scheduling are primarily organizational and training/experience related as opposed to technological. While methods leveraging simulation to account for uncertainty may have been limited by computing power in the past, simulation cases no longer prohibitively strain computing budgets [62]. Additionally, discrete-event simulation has continued to mature, and automating model generation, linking to external optimization routines, and quickly generating any required statistics is now immediately possible [99,100]. This leads to the final assertion upon which this thesis is based:

Assertion 2

The lack of widely accepted robust scheduling practices is primarily due to implementation challenges. There is no inherent technological barrier to their application.

1.2 Summary

This chapter discussed the need to develop scheduling and planning methods that help reduce the costs of producing and operating increasingly complex aerospace systems. In particular, one effect contributing to the schedule and cost overruns is the significant increase in processes that must be completed when producing such aerospace systems. These additional tasks challenge traditional scheduling practices for OEMs because completing them during a single integration production step is no longer a viable option.

Furthermore, a lack of schedule robustness is identified as a contributor to the observed delays and cost overruns. Schedule disruptions are shown to cause significant problems for schedules optimized using traditional, deterministic methods. Therefore, a method that can better account and plan for uncertainty to decrease process time, cost, and risk is needed.

Finally, six challenges to integrating tools to enable robust scheduling practices with existing scheduling practices have been identified. These challenges are primarily organizational and training related; however, there are no inherent technological barriers to implementing simulation during the pre-production planning phase.

Throughout this chapter, two assertions were made that form the backbone of this thesis. Assertion 1 contends that integrating robust design techniques with scheduling can help to alleviate the cost overruns and schedule delays observed in industry. Assertion 2 claims that the barriers to implementing robust design principles within scheduling are primarily organizational. The following chapter will build on these assertions to describe the problem to be addressed and further define the scope and objective of this thesis.

CHAPTER II

PROBLEM DEFINITION

This chapter discusses further the challenges associated with the assertions made in Chapter 1. As such, it follows the structure of the first chapter and begins by addressing the first assertion:

Assertion 1

Robust scheduling techniques can help alleviate some of the increasing costs and delays experienced by the aerospace industry.

2.1 Introduction

As discussed in Chapter 1, the increase in system complexity is shown to be a contributor to the schedule and cost overruns see throughout the aerospace industry [35,136]. The schedule problem's complexity, which increases with "multiple part types made in the same facility/line, numerous manufacturing steps (300-500 steps is not uncommon), batch processing, very complex equipment that leads to high levels of preventive maintenance and downtime, and multiple levels of sub-assemblies [58]", increases alongside the complexity of the system to produce [58]. Limitations in current scheduling practices, especially related to modeling capabilities and system robustness, also become exacerbated by increased system complexity [61]. The following review of selected simulation and scheduling methods helps to highlight 1) the failures of common scheduling techniques to efficiently manage systems of increased complexity, and 2) the potential when integrating improved robust scheduling and simulation practices to address the observed cost and schedule overruns.

This review is divided into first a review of common scheduling practices and then a review of typical uses for simulation models. This delineation is used because scheduling and simulation are typically considered to be associated with different domains with their own problems and challenges [79]. Scheduling relies "on mathematical techniques and heuristic methods to allocate limited resources to activities that have to be done [148]," whereas simulation is used to describe the time-dependent flow of entities or information [79] within systems too complex for analytical models [109]. This difference has contributed to the disconnect between the two fields even though both are essentially working towards the same goals: reducing cost, improving efficiency, etc. As such, by examining the strengths and weaknesses of both scheduling and simulation practices, it is expected that an improved, hybrid scheduling strategy can be developed.

2.2 Review of Scheduling Strategies

Scheduling strategies can be categorized based on a few distinguishing features. Figure 5 illustrates the hierarchy of these features. The first distinction is whether the model is stochastic or deterministic. Deterministic scheduling deals with the case when all data in the scheduling model (e.g., process times, capacity, etc.) is known in advance with no uncertainty [53]. A stochastic model relaxes this constraint by allowing for uncertainty in processing times, machine breakdowns, workforce availability, etc. [77]. Another distinction can be made between static and dynamic models. Static problems assume that, while there may be uncertainty in processing time or resource availability, the jobs to be completed will not change because of additional orders or cancellations and the job ready times are known and fixed [61, 79]. The dynamic problem allows for jobs to arrive at random during the schedule execution period [61] and seeks to develop a scheduling system that is able to react to disruptions [145]. In this way, the stochastic, dynamic scheduling problem is the closest to the real world [41].

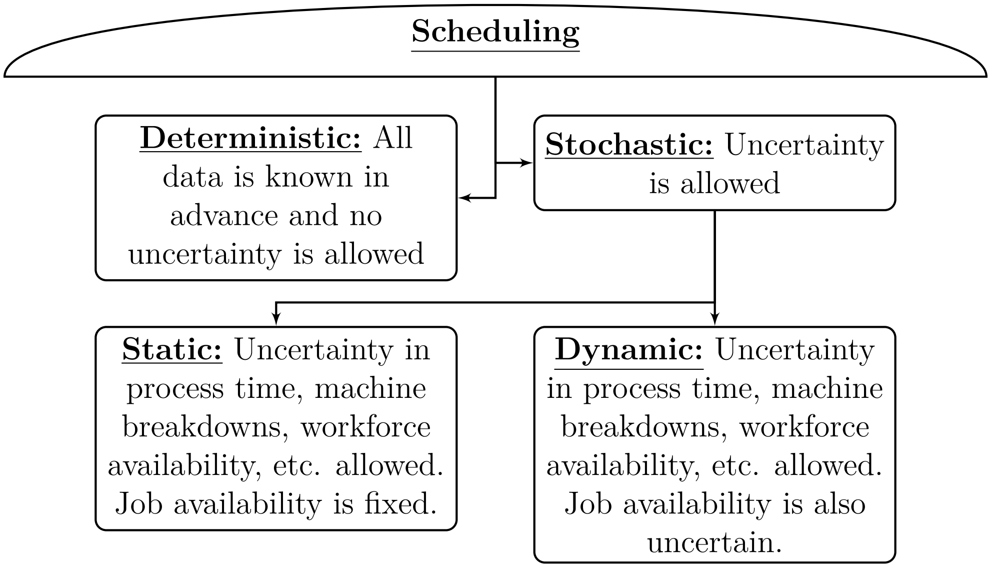

Scheduling
Deterministic: All data is known in advance and no uncertainty is allowed
Stochastic: Uncertainty is allowed
Static: Uncertainty in process time, machine breakdowns, workforce availability, etc. allowed. Job availability is fixed.
Dynamic: Uncertainty in process time, machine breakdowns, workforce availability, etc. allowed. Job availability is also uncertain.

Figure 5: Scheduling Classification Hierarchy

These distinctions are important because they have a significant impact on the type of schedule optimization algorithms that are capable of solving the problem [61]. Static, deterministic problems fall under the umbrella of deterministic combinatorial optimization [61], while problems that are dynamic and/or stochastic typically require a heuristic or metaheuristic optimization approach for all but the simplest of problems [27]. Examples of algorithms typically used to solve deterministic scheduling problems are linear programs [148], branch and bound methods, or heuristics (such as the shifting bottleneck) [123]. Stochastic models, on the other hand, typically leverage some type of Monte-Carlo or discrete-event simulation combined with a metaheuristic search algorithm (genetic algorithm, simulated annealing, tabu search, etc.) [15]. With this structure in mind, the review will begin with static, deterministic scheduling strategies.

2.2.1 Static, Deterministic Scheduling Strategies

Even though the problems to be addressed in this thesis are clearly stochastic, investigating deterministic schedule modeling and optimization approaches helps to inform solution strategies for stochastic problems [61]. There are two main areas within deterministic scheduling that are important to investigate for this thesis. The first area is the schedule representation and optimization methods applied with specific focus on any heuristics used to guide the optimization. Since deterministic scheduling has been studied the most of any branch of scheduling, reviewing heuristics identified in the literature can help to guide the development of a robust scheduling system. Secondly, a study of the objective functions used throughout the literature illustrate potential key performance indicators for use in experiments evaluating the proposed techniques.

Deterministic schedule models are typically represented as a sequence of jobs that require resources for completion and are subject to precedence constraints [148]. These models form the basis for many familiar project scheduling techniques such as the Critical Path Method (CPM) or the Program Evaluation and Review Technique (PERT). The CPM seeks to optimize the schedule by minimizing the critical path, which is the sequence of jobs that have no slack time [140]. Slack time being the time that a specific task can be delayed without affecting the project's completion time, a delay along the critical path directly impacts the project's completion time. However, if a process off the critical path is delayed, the schedule may still be completed on time by absorbing the delay in the scheduled slack time. PERT is very similar to CPM except that it begins to address stochasticity by computing the critical path for optimistic, expected, and pessimistic process completion times [44]. These techniques have been used for project schedule evaluation since their first implementation by the U.S. Navy during the late 1950s to help control the development of the Polaris missile [10].

The deterministic schedules used for the CPM and/or PERT are typically represented as disjunctive graphs (as seen in Figure 6) [60]. Disjunctive graphs contain a set of jobs to be performed displayed on nodes. Conjunctive arcs (solid arrows) connect operations in sequence, while disjunctive arcs (dashed lines) connect jobs that may not be performed concurrently (e.g., if two jobs require the same equipment to be performed) [42]. This type of representation is chosen because it lends itself to combinatorial optimization techniques [16], which will be discussed in the next section.

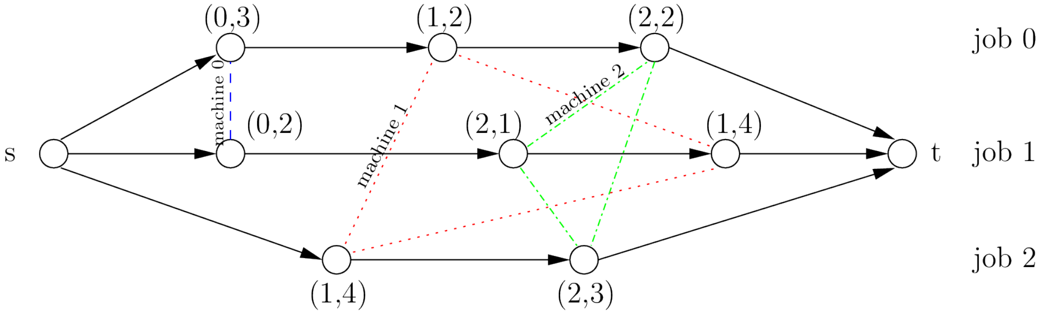


Figure 6: Example Disjunctive Graph for a Job Shop with 3 Machines and 3 Jobs [4]

2.2.1.1 Deterministic Combinatorial Schedule Optimization

One of the first combinatorial optimization strategies utilized to solve static, deterministic scheduling problems involves linear programming [79]. Deterministic scheduling is characterized by a large number of potential solutions that satisfy the problem's constraints and must be intelligently evaluated to determine the "best" feasible solution [136]. In the case of manufacturing, the best solution is typically that which either minimizes cost or maximizes profit [64]. As the name suggests, the objective function and all constraints must be formulated using linear functions, which may then be solved via algorithms like the simplex or interior-point methods [117, 148]. The standard form for a linear program is:

$$
\begin{aligned}
&\text{Minimize:} \quad F(\mathbf{X}) = \sum_{i=1}^{n} c_i X_i \\
&\text{Subject to:} \\
&\qquad \sum_{i=1}^{n} a_{ij} X_i = b_j \qquad j = 1 \dots m \\
&\qquad X_i \geq 0 \qquad\qquad\quad i = 1 \dots n
\end{aligned}
\tag{1}
$$

where F is the objective function, $\mathbf{X}$ is a vector of design variables (e.g. the sequence of jobs to complete), c_i are the linear cost coefficients, and a_{ij} and b_j define the set of constraints [194].

Due to the increasing complexity of planning tasks (more jobs to plan, resources to seize, and/or constraints to fulfill), many of the exact combinatorial optimization techniques used to solve scheduling problems become computationally infeasible [29, 136]. This is because many scheduling problems belong to the NP-Hard class of problems, i.e. problems that are characterized by sharp increases in possible solutions with small increases in problem size [28, 33, 136]. As a consequence, many heuristic methods have been developed to find "good" solutions to scheduling problems in a relatively short amount of time [136]. Reference [136] provides a vast overview of scheduling heuristics for many problems. The following paragraphs review some of the most popular examples.

The first set of heuristics to investigate are constructive algorithms, which build up a solution to the scheduling problem sequencing jobs via simple rules [29, 61]. For schedules with n jobs to be sequentially processed on two machines, Johnson's Algorithm seeks to plan the jobs with the shortest process time on the first machine first. By doing so, the algorithm ensures that the second machine begins processing orders as soon as possible to minimize make-span (maximum process time) [61]. This algorithm does produce the optimal schedule (in terms of process time) for limited cases; however, heuristics using similar rules are applied to more complex problems as well [61, 136].

Many other heuristic rules that are shown to produce good schedules have been discussed in the literature. These rules, typically called selection or dispatching rules, select the next job to process from a list of available jobs. Similar to Johnson's Algorithm, the Shortest Processing Time First (SPTF) rule selects the job with the shortest process time to process first. This frees the resource as quickly as possible to limit its constraint on the system. The Most Work Remaining (MWR) rule processes the job with the most processing time remaining to avoid getting too far behind on any one job. However, it is sometimes better to process the job with the least total work remaining to complete jobs and free up the system. Finally, if the system is complex and the computation resources are available, the planners may choose to implement a Monte Carlo approach where the jobs are planned randomly and the best of many possible schedules is chosen.

These heuristic methods attempt to use empirical results to generate good schedules [61, 136]. These rules represent a very common way to generate production schedules, however they do have significant limitations [214]. While producing high performing schedules in some instances, there are other cases where the same dispatching/heuristic rules perform very poorly. Worse, it is not currently possible to predict the performance of a specific set of rules before applying them to a system [84, 214].

Another set of heuristic methods seek to incrementally improve an initial sequence through neighborhood searches. One such strategy exchanges pairs of jobs until no exchange improves the solution. In this situation, the proposed solution is a local optimum, and if the problem is convex, the solution is the global optimum [136]. Generally, neighborhood search techniques, such as the pairwise exchange method, investigate solutions "in the neighborhood" of a seed solution for improvements. Bottleneck methods, which are some of the most popular approaches in present use, identify the bottleneck resource (i.e. the most constraining) and then seek to free that machine or resource as much as possible. If the bottleneck shifts to another resource, the algorithm will then move its focus to reduce the load on the new bottleneck [136]. The algorithms presented here, while not covering all proposed schedule optimization strategies, serve to present a range of different overarching methods that have been investigated throughout the literature. While these processes are popular, the direct optimization methods struggle with even moderately sized problems while heuristic approaches have unpredictable performance when applied to new problems. The following section further discusses the various objective functions typically employed in deterministic scheduling problems. The objectives discussed are extended to further include robust scheduling considerations in Section 2.2.2.3.

2.2.1.2 Overview of Objective Functions for Deterministic Scheduling Problems

Planners ideally seek to maximize profit, however quantifying total profit is typically difficult. As such, the objective functions utilized for deterministic schedules typically focus on minimizing some form of process time. This is based on the assumption that cost scales linearly with time; so minimizing time minimizes the process cost [61] and indirectly maximizes profit. The two most direct forms of time to investigate are the maximum process time and average process time. These measure the amount of time that a component is being processed or a job is being performed to complete the schedule. The maximum process time is also called the process make-span [61, 148]. Other possible objective functions relate to the products' due dates. If there is a bonus for completing a job early, the company may choose to minimize the total or maximum lateness (difference between the due date and completion date). If there is no bonus for early completion but a penalty for lateness, the company may minimize the total or maximum tardiness, which is 0 if the job is early or on time but becomes positive if the job is late [61, 148]. A final set of performance measures typically seen in scheduling relates to factory efficiency. For example, minimizing setup times or costs usually helps to reduce make-span or improve throughput [148]. Minimizing the work-in-process (WIP) inventory is also important to free up equipment and reduce system congestion [61, 148]. Finally, some planners attempt to improve the system's robustness by maximizing the schedule's slack time; this allows for extra time or capacity in case some processes do not occur as planned [8, 79]. Maximizing the Critical Ratio, which is the ratio of the time until the due date to the process time remaining, is also suggested as a helpful measure to reduce risk [25, 136].

The previous paragraph has not sought to provide a complete list of schedule planning objective functions; instead, the goal was to provide an overview of the various areas of interest along with the implications and reasoning for the selection of objective functions. A summary of the objectives discussed can be found in Table 2.

As discussed, static, deterministic scheduling is suitable for many problems and certainly represents an improvement above the experience-based rules-of-thumb used in the past. Despite this, literature sources asserting that even with its strong theoretical basis, deterministic scheduling generally fails to address the practical concerns of the industry [128, 145]. This is because while increased process automation, better data management, and improved supply chain coordination have worked to make the deterministic assumption more valid [61], the increased uncertainty due to the complexity of many modern products has surpassed the gains from automation [60, 136] and increased the need to consider stochasticity [128]. Additionally, the manual installation tasks typical of more complex systems do not lend themselves to automation [116], which limits the potential for improvement brought by investing in advanced and costly robotic capabilities. Hence, there is a clear need to consider risk and uncertainty when generating schedules for many modern systems.

The deterministic scheduling and optimization strategies presented here have shortcomings when dealing with significant stochasticity. While these shortcomings

Table 2: Examples of Common Objective Functions for Deterministic Scheduling Problems

Objective Function	Description	Benefits
Process time	The amount of time that a component was processed to complete the schedule	Cost typically scales with process time [61].
Minimize lateness	Due date − completion date	Seeks to complete products before their due date to receive a benefit for early delivery [61, 148].
Minimize tardiness	$\text{Cost} = \begin{cases} 0, & \text{delivery date} \leq \text{due date} \\ \text{penalty}, & \text{delivery date} > \text{due date} \end{cases}$	If there is only a penalty for late delivery, simply seek to meet all delivery dates [61, 148].
Minimize setup time	Depending on the flow of products through the system, some machines may require setup or changeover time to process a new job.	Improves utilization rates and reduces non-value added time. Typically leads to a reduction in process time [148].
Minimize work-in-process (WIP)	Amount of jobs or components currently within the system.	Helps to free resources and reduce system congestion [61, 148].
Maximize schedule slack time	Maximize the time between processes while maintaining due dates and process constraints	Improves the schedule's robustness by adding additional time to absorb schedule disruptions [8, 79].
Maximize the total critical ratio	$\text{CR} = \frac{\text{Time until delivery date}}{\text{Process time remaining}}$	Attempts to reduce risk by completing jobs with longer processing times further from their due dates [25, 136].

have been somewhat addressed by adding an objective function to increase the slack time between processes, this heuristic objective is fairly simplistic and typically ignores many other sources of available information [79]. The following section explores scheduling strategies that allow for various forms of non-deterministic behavior.

2.2.2 Non-Deterministic Scheduling Strategies

Non-deterministic/stochastic scheduling allows for uncertainty in the processing time of jobs, machine availability, etc. [61]. Non-deterministic schedules can be classified as either *static* or *dynamic*. Static, non-deterministic schedules consider uncertainty in processing time, technician availability, etc. However, the jobs to be completed are assumed constant [61]. Dynamic scheduling allows for the arrival and availability of jobs to be uncertain [61,79]. By allowing for these forms of uncertainty, the developed models can more closely approximate the real world [148]. However, by including uncertainty, many of the optimization strategies developed for deterministic types of problems are no longer valid [61]. This section reviews some of the most prominent stochastic scheduling techniques to help identify strategies that could help to alleviate the delays discussed in Assertion 1.

Before proceeding through this section, a distinction between predictive and reactive schedules must be made. A *predictive* schedule is the initial plan developed by a scheduling algorithm that accounts for resource constraints [59, 89]. A *reactive* scheduling system then monitors the production environment and decides how to modify the predictive schedule to deal with unexpected events such as machine breakdowns or process time variation [156]. This section concentrates on stochastic predictive model generation, which is the focus of the proposed robust scheduling methodology. Reactive scheduling strategies are addressed briefly as a means to support the development of simulation model logic.

Even though predictive schedules are rarely if ever executed perfectly, defining a good baseline schedule is an important step towards completing a project or process. Indeed, the baseline schedule is important for determining resource allocation, coordinating with suppliers, and approximating future work for employees [187]. The focus of stochastic, predictive scheduling has primarily been on developing a more robust initial schedule (sometimes called a *proactive* schedule [93]), which is a schedule that is insensitive to noise and requires fewer adjustments [187]. Many methods, along with many different measures of robustness, have been developed to help determine a robust schedule. These are discussed and evaluated in the following sections.

In particular, Section 2.2.2.1 first investigates the models typically used for stochastic scheduling. Section 2.2.2.2 then provides an overview of the algorithms typically employed to optimize such schedules. Section 2.2.2.3 then provides an overview of the objectives and strategies used for robust schedule optimization. Finally, Section 2.2.2.4 presents a review of common reactive strategies used to repair schedules when disruptions occur.

2.2.2.1 Stochastic Schedule Model Architecture

The nature of the models used for schedule experimentation is important to discuss to facilitate an understanding of their capabilities and limitations. Generally, a model is a "stripped down, simplified and abstract representation of an otherwise complex, detailed and broad reality [60]." In manufacturing systems, real objects, such as machines, technicians, material, and tooling, are typically the focus of the modeling effort [60]. The purpose of a model is to test ideas [67], so models must be built with that purpose in mind and contain appropriate details to achieve this goal [60, 68]. Depending on the goal of the scheduling process and the types and significance of uncertainty, various methods to include stochasticity have been employed.

Mathematical models "aim to describe the different aspects of the real world, their interactions, and their dynamics through mathematics [153]." This is the type of formulation primarily used in deterministic scheduling where the problem can be formulated as a constrained optimization problem [61]. While most applicable to deterministic scheduling, if noise variables or uncertainty distributions can be defined for the problem, Monte Carlo analysis can add an element of stochasticity and provide a means to evaluate the probability of cost overrun for a project [182]. Traditionally, Monte Carlo simulations are used to evaluate the risk within a determined schedule or evaluate the accuracy of the initial forecast [195]. However, the results from the Monte Carlo simulation generally are not used to drive the optimization, and are typically relegated to a post-optimization risk analysis. In cases where Monte Carlo analysis is incorporated within the optimization, the results are generally used to estimate an average value for a deterministic algorithm [163].

Disjunctive graph models, while commonly reserved for deterministic analysis [61], can be utilized for stochastic analysis by adding uncertainty to process times. These models can then be optimized via heuristic methods [73, 183]. While this approach does include stochasticity, the uncertainty is required to be independent, which means that the methods cannot directly account for disruptions. By assigning independent uncertainties to each process time, this approach is unable to account for relationships between processes. For example, a quality problem encountered during one step may impact the process time of another. As such, graphical or network schedule models may not be sufficient for robustness investigations that are concerned with mitigating the impact of specific disruptions.

Most scheduling systems utilize these types of models for their investigation. These models are commonly classified by the number of jobs, number of machines, and other assumptions used to define the system. However, this research focuses on investigating a general methodology linking scheduling and simulation, so these classifications are superfluous. The interested reader may find further information about the classification schemes in [28, 61, 148, 149]. With the types of stochastic schedule models in mind, the following section describes a selection of the optimization algorithms typically employed to solve stochastic scheduling problems.

2.2.2.2 Optimization Algorithms for Stochastic Scheduling

As previously mentioned, one of the first methods to account for uncertainty was PERT. The technique assigns time distributions to each process task and then calculates the project's completion time based on optimistic, expected, and pessimistic process times [44]. In general, PERT simply replaces a deterministic value with one determined from the distribution and calculates the critical paths using that value [112]. There are a few disadvantages that make PERT unsuitable for the complex systems of today. Generally, PERT relies on task relationships that are very inflexible. Therefore, in instances when some disruption occurs, there is no way for the model to adapt as the real world system would. Additionally, the disruptions are not directly captured and are instead rolled up into subjective time estimates, which are typically over optimistic [112]. Finally, PERT identifies the critical path, but task sequences that are close to critical could easily shift onto the critical path based on the process time uncertainty. These tasks are not initially identified as critical, and may, therefore, not receive the same focus as the ones identified as critical [120].

Moving on from the basic PERT method, there has been much work in utilizing metaheuristic optimization algorithms to produce predictive schedules under uncertainty [27]. A metaheuristic search is a process that uses an overall heuristic to guide "the operation of one or more subordinate heuristics (which may be from a local search process, to a constructive process of random solutions) to efficiently produce quality solutions for a problem [70]." As such, metaheuristics typically take a solution or solutions that are generated via a problem specific heuristic and guide its incremental improvement [155]. Summaries of the various heuristic schedule generation methods can be found in references [60,61,136] and are discussed in Section 2.2.1.1. A benefit of metaheuristic methods is that they are problem agnostic, and can, therefore, be quickly and easily applied to a wide variety of problems [146].

Metaheuristics, and their underlying heuristic methods, are sometimes called approximate approaches because they cannot guarantee that a global optimum has been found. However, they generally find "good" solutions relatively quickly [28,79]. This is in contrast to the mathematical programming methods described in the previous section that can typically prove that the optimum was found [61]. Despite not being able to prove an optimum, metaheuristic optimization methods are popular for scheduling problems with uncertainty. This is due to the fact that the mathematical programming methods (linear programming, branch & bound, etc.) designed for the deterministic problems are feasible only for small stochastic problems and require significant computational effort [27,79]. Essentially, schedulers operating in a "real-world" setting prefer finding a "good" schedule in a reasonable amount of time to finding the optimal schedule in an exorbitant amount of time [15]. The following paragraphs discuss three metaheuristic algorithms typically used in scheduling, Tabu Search (TS), Simulated Annealing (SA), and Genetic Algorithms (GA), along with specific scheduling applications relevant to this research.

Tabu Search (TS) algorithms search for improvements within a neighborhood around an initial solution [136]. The algorithm will always move to the best candidate solution. To avoid becoming trapped within a local optimum, a "tabu" list of disallowed moves is maintained that is designed to encourage exploration and prevent the algorithm from cycling back to a local optimum. The best solution is always stored in case no better solution is found, and the algorithm is typically stopped after a certain amount of time (or moves) after an improved solution has been found [69,136,146]. The TS algorithm may also be parallelized by distributing evaluations of points within the neighborhood or simply running the base algorithm with different starting points on multiple processors to speed convergence [178]. One drawback of TS is that it must move to the best candidate point (not just an improved point), which makes evaluating the neighborhood costly [136]. However, this drawback can be alleviated by increasing parallelization.

The tabu search metaheuristic has been applied to scheduling problems by modifying schedules constructed by some heuristic method [203]. For manufacturing scheduling problems, the search typically involves adding, removing, or swapping jobs planned within the schedule according to the rules of the TS [155, 183, 203]. Because the effectiveness of the search is based on the initial schedule, the strategies generally apply multiple heuristics or random permutations to a starting schedule generated by a heuristic method [183]. The TS method has been shown to be much faster than traditional deterministic algorithms as problem size increases [27].

Tabu search can also be modified for multi-objective optimization purposes, which is important when considering the competing objectives typically found in scheduling problems [180]. One method is to combine the objectives into a single weighted objective function and use the single objective algorithm to solve the problem with varying weights to explore the Pareto frontier. While this is a simple method, it has the drawback that weighted-sum methods cannot find concave portions of the Pareto frontier [90]. Another approach utilizes the tabu concept to disallow recently visited points while saving optimal or near-optimal points in a separate list to intensify the search. In this way, points can be compared based on Pareto dominance, and in instances where multiple candidates are Pareto-equivalent, the algorithm will randomly choose the next point from the equivalent candidates. All of the equivalent points are included in the list of points for intensification. A more detailed description of this strategy may be found in reference [90].

Simulated Annealing (SA) is an optimization technique inspired by the annealing process of metals [136]. SA is another neighborhood search method that aims to avoid becoming trapped in a local optimum [12]. The algorithm evaluates points within the neighborhood of the current design point. If a candidate point is better than the current point, the algorithm moves to the new point. If the candidate is worse than the current point, the algorithm moves to the new point if a random number sample is less than the value of an equation of the same form as Equation 2. In this equation, Δf describes the difference between the current point and the candidate point's objective function evaluation. Therefore, as Δf increases, the probability to accept the worse point decreases. T_k is the current "temperature" of the algorithm. In the early stages of the algorithm, the temperature is high to improve the probability of moving to a worse candidate point to encourage exploration of the space. As the optimization progresses, the temperature is decreased according to an annealing schedule to gradually hone in on the (hopefully) global optimum [12].

$$p\left(\Delta f\right) = \exp\left(\frac{-\Delta f}{T_k}\right) \tag{2}$$

The simulated annealing algorithm can be modified to take advantage of parallel computing. An immediate option is to simply run the algorithm with multiple starting points in parallel to converge to different points along the Pareto frontier. Alternatively, instead of evaluating and comparing single solutions, one may evaluate many solutions within the neighborhood and then apply Equation 2 to each candidate solution. The next point then may be chosen randomly from the candidates weighted by Equation 2 [136].

Simulated annealing has been applied to scheduling problems successfully in much the same way as the tabu search. Generally, the search begins by generating an initial feasible schedule via a heuristic method [180]. The algorithm then proceeds to investigate the neighborhood around the initial schedule by swapping, adding, or removing jobs. SA has been applied to the job shop scheduling problem and shown to outperform heuristic methods designed specifically for the problem [191].

References [27, 174] describe multiple applications of simulated annealing to various job shop scheduling, facility layout, logistics, and other project scheduling problems.

Simulated annealing can also be applied to multi-objective problems. A typical strategy is to combine the objective functions into a single, weighted criterion. The SA is then used to solve the problem with variations on the weightings to find different regions of the Pareto frontier [180]. Another approach relies on the fact that SAs find multiple global optima along the Pareto Frontier with equal probability. Therefore, if the objective function is the Pareto domination rank of a point, the algorithm should have an equal probability of finding any point along the Pareto frontier. Hence, this strategy starts the algorithm at multiple points and collects each optimal point found to construct the Pareto frontier [139].

A more direct exploration of the Pareto frontier is possible with simulated annealing. Instead of using a single composite objective, the SA can select new points based on the points' Pareto rank [139]. To encourage exploration along the frontier, the algorithm moves to a trial point if it is a better or equal rank. Movement to a dominated point is controlled by Equation 2 where Δf is found based on a composite objective function. Various weighting types are discussed in reference [139]. This strategy could prove useful if no information about the relative importance of the objectives is available.

The final metaheuristic schedule optimization algorithm to be discussed is the *Genetic Algorithm* (GA). By simulating the Darwinian natural selection process, the GA guides a population of points towards an optimum [12]. The general strategy begins by generating an initial population of random design points. The initial population may also include population members created via other heuristic methods to seed the population with good designs. After evaluating the "fitness" (objective function value) of each member of the population, some members of the population are chosen for reproduction based on their relative fitness. Upon selection of two parent points, child points are created by genetic crossover. Similar to biological reproduction, crossover entails swapping portions of each parent's design bit string (called a chromosome) to form the children. After the crossover, the new designs may undergo a mutation phase where random binary bits in the design bit string are flipped. This ensures that genetic diversity is maintained while also searching in the neighborhood of current solutions to move towards the global optimum. A more detailed description of typical genetic algorithms can be found in references [12, 194].

Genetic algorithms are extremely amenable to parallel computation. Upon determining a new population to evaluate, each member may be calculated independently and distributed to any number of computing cores. GAs typically require many function evaluations to arrive at the optimum point, so parallelization is extremely important, especially when the objective function evaluation is relatively slow. For this reason, combining GAs with surrogate modeling techniques is a popular strategy [194].

The overall genetic algorithm strategy can be extended to multi-objective optimization as well. A popular multi-objective genetic algorithm is the Non-dominated Sorting Genetic Algorithm II (NSGA-II). This algorithm uses non-domination to sort the potential parents. Each group of non-dominated points are then further ranked based on their crowding distance, which is essentially a solution's "closeness" to another population member in the same non-dominated rank. By including the crowding distance, the algorithm attempts to maintain diversity and evenly sample the Pareto frontier [50]. A review of multi-objective optimization techniques applied from 1991 to 1999 found that 70% of articles utilize GAs as the primary optimization routine, which speaks to their extreme flexibility and wide applicability [92]. Hence, because of its natural multi-objective formulation and evidenced applicability to a wide range of problems, the NSGA-II is a strong candidate for the problem at hand.

With an understanding of the schedule modeling architectures capable of accounting for stochasticity and some of the optimization methods available, the following section describes how the strategies are commonly implemented to identify robust, predictive schedules.

2.2.2.3 Review of Robust Schedule Generation Strategies

As mentioned previously, a primary purpose for including uncertainty within scheduling is to generate schedules that are robust [180]. Schedules generated with robustness considerations are sometimes classified as *proactive* schedules [93,180]. Proactive/ robust schedules are created in order to protect the schedule from disruptions that may occur during execution and minimize the need to repair the schedule during operation [93]. Within the literature, many scheduling strategies have been proposed that claim to produce robust schedules by optimizing different objective functions. A review of the most popular strategies is provided.

The most common approaches to improve schedule robustness are to include idle time (also called buffer or slack time) in the schedule and make additional resources available when disruptions occur [8, 79, 81, 93]. The idle time provides a buffer if there are delays in the schedule. Idle time has the additional benefit of providing open capacity to process an unexpected, urgent job [79]. Extra resources in the form of machines or personnel can be brought online to help recover from disruptions as well [79,81]. Different strategies for choosing the amount of buffer time to include have been proposed. Critical chain scheduling focuses on ensuring project completion by the due date by allocating buffer to jobs on the critical chain [80, 93]. Other systems include buffer in front of processes that would incur a large cost if delayed [93] or processes closer to the end of the schedule that have a larger possibility of having a disruption [46]. Furthermore, maximizing the minimum of slack times or the ratio of slack time to the corresponding activity's duration has been proposed to help reduce the variation in make-span and schedule instability [105]. Reference [79] also lists some other qualitative considerations for robust scheduling including identifying potential problems, scheduling difficult jobs first to improve learning, and keeping the most flexible resources free.

Another proposed strategy to improve schedule robustness is to generate multiple, well-performing schedules during the generation phase. Then, as the process is being completed and disruptions occur, the company may switch to a schedule (or partial schedule) that is unaffected by the disruption. While helping to mitigate the problems caused by schedule disruptions, this strategy, called contingent scheduling, is focused more on flexibility than robustness. In other words, the algorithm does not search for a specific schedule that is robust, but instead provides many alternatives that can potentially handle various disruptions [93].

With any of these robust schedule generation strategies, the quality of the proposed solution may be measured in multiple ways. Reference [8] suggests simply using the total amount of slack time in the schedule as an indicator of robustness. Two additional measures, typically called *schedule stability* and *quality robustness*, are commonly found in the literature. Schedule stability (sometimes called solution robustness) is related to the difference between the predictive schedule and the realized schedule [80, 93, 105, 126, 213]. This may be quantified by the difference between planned and actual start times, the number of processes that are disturbed, or the number of times a process must be re-planned [80, 105, 126, 213]. The schedule can be simulated multiple times to estimate the proposed schedule's stability [48, 93].

As opposed to schedule stability, which assumes that a variation from the plan will degrade the schedule's performance, quality robustness deals directly with the uncertainty's impact on the objective functions (e.g. cost, make-span, etc.) [93]. Because the end goal of creating a robust schedule is to minimize the impact of disruptions, many sources in the literature suggest optimizing the expected value of a chosen objective function as is common in stochastic resource-constrained scheduling [93, 170]. Another related measure is the customer service level, which seeks to minimize the probability that the completion date exceeds the due date [48,93]. Risk-averse entities completing high-cost projects commonly require an 80% probability to remain within schedule and/or budget to pursue a project [87]. As such, the quality robustness of a project schedule can be quantified by the 80th quantile of an objective function of interest.

It is desirable to improve both the schedule stability (to minimize re-tasking the workforce, change paperwork, etc.) and quality robustness (to optimize the expected value of the objective function(s)) [48, 126, 180]. Reference [186] reports on a study that optimizes a composite objective for schedule stability and quality robustness. Others have introduced a non-dominated, multi-objective approach to design robust schedules. Pareto optimal solutions that simultaneously minimize a schedule performance function, like make-span, along with the function's standard deviation have been developed [218]. This particular study reports results for various flow shop scheduling problems that compare the robust scheduling scheme with a deterministic optimization strategy for various levels of time uncertainty. While the deterministic methodology is able to produce slightly better best-case scenario schedules when compared to the robust scheme, the robust scheme's average performance is better for most experiments and its standard deviation is always better. Finally, as expected, the results become even more pronounced as the degree of uncertainty increases, which indicates that the more uncertainty contained within a process, the more benefits to be expected from robust scheduling [218].

As presented throughout this section, developing and implementing robust scheduling strategies help to reduce the impact of process time uncertainty on an overall schedule. Generally, the strategies seek to place buffer time within a schedule using various heuristics or to leverage optimization approaches to minimize schedule disruptions that are mainly due to process time uncertainty. A generated schedule's performance is typically evaluated through Monte-Carlo simulation by drawing from distributions set on process times. Finally, results have demonstrated that by including robustness considerations, the generated schedules are able to better cope with disruptions, hence leading to schedules with better average performance with reduced risk. The following section briefly discusses reactive scheduling strategies that are used to repair predictive schedules when disruptions occur during the schedule's execution.

2.2.2.4 Review of Reactive Scheduling Strategies

Reactive scheduling strategies modify the predictive/proactive schedule as disruptions occur during the schedule's execution [93]. Generally, if the initial schedule was generated with a proactive approach, there will be enough buffer in the system such that small, expected variation in process time can simply be absorbed by the original schedule. However, more significant disruptions, such as equipment breakdowns or technician absenteeism, could lead to disruptions that cannot be absorbed. These are the cases where reactive scheduling strategies must be employed to ensure the schedule remains feasible and within the time and budget allowable [93].

A few general rescheduling strategies exist to ensure the schedule remains feasible. One strategy involves right-shifting each operation following a disturbance [79]. This means that the processes affected by the disruption are delayed for as long as necessary for the schedule to become feasible again. Because no jobs are re-sequenced or moved to alternate resources, this strategy does not take advantage of any additional capacity that may be available [60, 79]. The next strategy, called partial rescheduling, resequences jobs at stations that are impacted by the disruption [60, 79]. This allows the scheduler to take advantage of the fact that while a delay may occur in one section of the schedule, other jobs and processes can still be completed. Therefore, once the resource comes back online, it may be beneficial to change the job processing order [79]. Finally, at the extreme, the entire schedule can be regenerated when problems occur. This is rarely done in practice because the rescheduling optimization may take too much time to complete. Additionally, there are costs from "schedule nervousness," which results from constant re-tasking on the production floor [60, 79, 136, 152]. In general, a partial rescheduling strategy that reduces the impact on the schedule's performance while also limiting the changes to the original schedule is preferred. This is because it takes advantage of new information about the current state of the production line while limiting the severity of changes to the schedule [93, 136].

To avoid further delays on the production floor, any rescheduling strategy must be implemented quickly. As previously mentioned, this typically will eliminate complete re-optimization procedures that could take hours to complete. Similarly, rescheduling only some portions of the flow, using metaheuristics or other optimization procedures, could still be too time consuming. In such instances, dispatching rules or other heuristic methods described previously can be used to correct the schedule. The rules utilize real-time information to reschedule jobs around the disruption (machine breakdown) [136]. In the extreme case where no schedule optimization is used and the work is planned completely by dispatching rules, disruptions are inherently considered by the system by the rules in place [79]. However, while this flexibility is desirable, dispatching rules are typically outperformed by other strategies [214].

2.2.3 Discussion and Summary

This section has reviewed aspects of scheduling strategies that are relevant to this dissertation. While in no way a complete review of all modeling, optimization, and evaluation possibilities, the section intends to provide a foundation to understand the myriad of approaches developed throughout the last 60-70 years. In deterministic scheduling, all data is know and assumed to be fixed. With stochastic scheduling, uncertainty in various parts of the problem is allowed. Stochastic scheduling problems can be either static or dynamic. Static problems allow for uncertainty stemming from process times variation, breakdowns, etc. but assumes that the jobs are all known and fixed. Dynamic scheduling removes this final constraint to allow for jobs to arrive at varying points during the schedule's execution.

Stemming from the various scheduling assumptions, models, and constraints, a number of solution strategies have been developed within the literature. Table 3 provides an overview of the 4 main categories of solution strategies presented throughout this chapter. Exact strategies, such as linear programming and branch & bound methods, are among the first methods developed and have a large amount of research behind their development. However, the types of problems that are solvable using exact methods are limited to simple, deterministic problems [136, 214]. As problems began to grow in complexity, heuristic methods, which cannot guarantee an optimum but have proven effective in many situations, began to be developed [136]. These methods typically use general rules-of-thumb to develop a schedule. The initial schedule may be further refined through various local search methods like pairwise swaps or the shifting bottleneck strategy. While being an important improvement, the heuristic methods tend to have unpredictable performance and can also become trapped in local optima fairly easily. To alleviate this, metaheuristic methods that guide lower level heuristic methods to help better explore the space and avoid becoming trapped in local optima have been developed. These methods have proven effective at providing "good" solutions to very complex problems at the expense of increased runtime over heuristic methods. Finally, for extremely dynamic cases when predictive scheduling is not possible, dispatching or priority ranking procedures can be applied to select the next job or operation in real-time based on proven rules.

These strategies essentially apply some of the constructive heuristic methods in real time. Schedule repair techniques fall within this category.

While a wide variety of scheduling strategies have been developed within academia, only a small amount of advances in scheduling have been transitioned to practice in the real world [45, 60, 126, 128, 185, 201]. One reason for this disconnect is that a significant amount of scheduling research has been focused on mathematically rigorous solutions to very specific problems as opposed to developing more general, approachable scheduling systems [60, 214]. Additionally, implementing an optimization routine that fulfills the practical needs of an organization without a planner's intervention is extremely difficult. Optimizers must be fed a rigid objective function that may be difficult to quantify exactly. This in turn could lead to solutions that have good objective function values but are not feasible or cannot capture all possible options. For instance, defining an explicit cost for missing a due date may be difficult to quantify without detailed knowledge of the situation that only a human planner can have [84]. Typically, the scheduling systems are unable to account for this type of information and integrate a planner's knowledge [45, 79, 126, 127, 207], which causes planners to sometimes abandon the use of advanced systems [60].

Another difficulty preventing the widespread adoption of planning systems, especially those able to account for uncertainty and provide robust solutions, is that many "real-world" problems cannot be effectively described using traditional methods [79]. This is because scheduling typically relies on analytical models; however, in many cases, it is simply not possible to describe the system in a closed form, mathematical statement [17, 18, 57, 79, 100, 110]. Because of this limitation, the simplifications and assumptions required to model many real world systems with analytical models strip away much of the model's effectiveness [18, 57, 167]. This, combined with the rigid, black-box optimization methods described previously, tend to dissuade many planners from fully and effectively implementing scheduling strategies in an

Table 3: Summary of Schedule Solution Strategies

	Exact	**Heuristic**	**Metaheuristic**	**Dispatching Rules**
Applicable to:	• Deterministic	• Deterministic • Static stochastic	• Deterministic • Static stochastic	• Deterministic • Static stochastic • Dynamic stochastic
Example algorithms and rules:	• Linear programming (LP) • Branch & Bound • Mixed integer programming	• Constructive algorithms • Monte Carlo (MC) • Pairwise swap • Shifting bottleneck	• Tabu search (TS) • Simulation annealing (SA) • Genetic algorithm (GA)	• Shortest processing time first (SPTF) • Most work remaining (MWR) • First in, first out (FIFO)
General benefits:	• Can guarantee optimum for some problems • Significant body of research and methods to draw from	• Typically the fastest solution strategies (especially constructive algorithms) • Can be combined together (create an initial solution with a constructive algorithm and then refine it with pairwise swaps or shifting bottleneck) • Well performing in certain cases	• Effectively explores large search spaces while utilizing information derived from lower level heuristics • Capable of escaping local optima • Generally provides better solutions than heuristic methods without the extraordinary time required for exact methods • Strategies are typically problem agnostic • Strategies are generally conducive to multi-objective optimization and parallelization	• Simplest methods to implement • Best approach for highly dynamic situation where predictive schedules are not useful or quickly become obsolete
General disadvantages:	• Quickly become computationally infeasible for even relatively small problems • Can only support deterministic optimization • Cannot directly optimize for robustness	• Unable to guarantee exact optimum is found • No such heuristic can be applied successfully to all problems and there is no way to know how it will perform a priori • Typically become trapped in local optimum easily	• Typically require *many* function calls to reach a “good” solution • Unable to guarantee exact optimum is found • More difficult to implement and tune the search parameters than lower level heuristics	• Does not necessarily produce a formal schedule • Difficult to predict performance of rules before implementation

increasingly complex environment. In other words, insufficient interfaces between planners and scheduling systems has contributed to the lack of widespread implementation [45, 60, 198, 205].

In summary, traditional scheduling procedures struggle to model modern systems that require effective scheduling to remain within schedule and under budget. This leads to the first research question to be addressed in this thesis:

Research Question 1

How can the challenges of implementing scheduling techniques be overcome to provide a system capable of producing robust schedules to reduce cost and delays?

This research question is the backbone of this thesis. There are a few key points within the question that will be addressed in the following paragraphs to formulate specific sub-research questions.

The first point of discussion deals with the specific challenges to implementation that need to be overcome. Most of the identified challenges stem from a disconnect between formal scheduling and "real-world" applications. Specifically, the difficulty that many schedule modeling methodologies have in effectively representing complex systems needs to be addressed. Effectiveness, in this case, entails quickly modeling the system with the correct amount of detail such that it can be used in optimization. This leads to the first sub-research question:

Research Question 1.1: Which modeling techniques can be applied to effectively model increasingly complex production systems for use in the proposed schedule optimization methodology?

While effectively modeling a system is a step forward, a model is not useful unless it can be used to make decisions [67]. Furthermore, a scheduling system must not only provide the ability to make decisions but also should facilitate the automation of the decisions via an optimization routine [60]. Hence, any modeling techniques must have the capability to interface with an optimization routine to efficiently find a solution. This leads to the next sub-research question:

Research Question 1.2: Which optimization technique(s) should be implemented to adjust the developed model effectively to search for optimal schedules?

In addition to having the ability to work with the developed scheduling model, the optimization routine must also have the capability to search for schedules with improved robustness. As such, any optimization routine selected must be able to accept objective functions beyond mathematical relationships that are able to quantify the schedule's robustness as discussed in Section 2.2.2.3. This leads to the following research question:

Research Question 1.3: How can the selected optimization technique(s) be utilized to improve the schedule's robustness?

The following sections discuss the formulated research questions and develop hypotheses to guide the experimentation and development of a methodology that supports the implementation of robust scheduling techniques. In particular, Section 2.3 examines potential simulation techniques that can potentially be leveraged to model complex systems. Upon selecting a candidate simulation framework, Section 2.4 will further explore the chosen framework's typical applications within scheduling.

2.3 Techniques for Modeling Complex Systems

Traditional schedule modeling techniques struggle to appropriately represent the increasing complexity encountered in modern systems. Hence, over the past couple decades, various forms of simulation have become increasingly popular as a means to represent complex systems to enable intelligent decision making and address some of the shortcomings of traditional schedule modeling strategies [18,57,58,91,99,100,110, 119]. There are 3 primary forms of simulation that are common in operations research (OR): (1) Discrete-event Simulation (DES), (2) Agent-based Simulation/Modeling (ABS/ABM), and (3) System Dynamics (SD) [31,119]. The following sections briefly investigate each to determine which is best suited to model the complex manufacturing systems of interest. Proper selection of a modeling strategy is important to ensure time and resources are devoted to the technology with the best potential to solve the problem at hand [190].

The potential simulation frameworks' appropriateness is judged on the following criteria:

- The modeling paradigm shall be capable of capturing the required levels of complexity to be relevant to modern manufacturing systems.
- The simulation framework should be as simple to implement as possible. This will enable easier transitions by planners who may not be completely familiar with the modeling structure. The framework should also facilitate automated model generation strategies. In doing so, the developed scheduling system has a better chance of being adopted by "real-world" practitioners.
- The simulation structure shall interface with optimization algorithms to enable the search for robust schedules.
- The modeling standard shall limit runtime to facilitate the number of cases required by the optimization.
- The simulation shall be capable of evaluating the robustness of a system (schedule stability and quality robustness).

2.3.1 Discrete-Event Simulation

Discrete-event simulation is a candidate modeling methodology to effectively represent complex systems of interest in this thesis. Discrete-event simulation is "the modeling of systems in which the state variable changes only at a discrete set of points in time" and "are analyzed by numerical methods [18]." Numerical methods entail running the model to approximate the solution; in contrast, analytic schedule models contain objective functions that can be mathematically evaluated. This mathematic evaluation may involve simple linear relationships or more complicated probability theory and calculus, however in all cases, the objective function has a closed-form solution [18]. Because DES models must be "run" to obtain an estimation of the objective [18], many optimization routines available for deterministic schedule optimization are not applicable to simulation models.

Even though discrete-event simulation removes the ability to use many efficient optimization algorithms, it is still an extremely popular modeling paradigm. This is because DES is able to better model "real-world" systems that are too complex for analytical models [100, 110]. DES is also designed for modeling manufacturing and queuing systems. As such, many of the modeling constructs are useful for a study in scheduling. Consequently, discrete-event simulation is a strong candidate to model the complex systems described in Research Question 1.

2.3.2 Agent-based Modeling

Agent-based Modeling (also referred to as Agent-based Simulation or Agent-base Modeling and Simulation) is a modeling framework particularly suited to representing systems "comprised of autonomous, interacting agents [118]." In the generally agreed terminology, agents are independent components within the system that can make decisions based on interactions with other agents [118]. Agents can represent any component in the system, from physical, manufactured components to technicians and machines. The decisions made by the agents can be based on preset rules (such as the dispatching rules discussed previously) or may be adaptable based on observed behavior [118]. Finally, an agent has goals that it is trying to attain (such as maximizing its efficiency or working to minimize process time) and works toward that goal [118].

Agent-based modeling relies on a monotonically advancing simulation clock to track progress [151]. At each clock tick, each agent is able to move, complete tasks, and make decisions based on the available information. Selecting the appropriate time step is one of the most important aspects of ABM: if the time steps are too small, the model runtime is too large without significant gains in accuracy; however steps that are too large can lead to a loss of accuracy and ability to represent the real system [151]. Run-time is highly dependent on this selection and can quickly increase as the required granularity of the simulation increases [151].

ABM has been be applied to production modeling and queuing systems [31,36,56, 111,119,135,151]. In general, ABM is a good way to model highly dynamic systems where decisions must be made at many different levels of the organization [135]. Supply chain dynamics or the selection of workers or machines to use to process a job are some typical uses of ABM in the manufacturing sector.

However, for queuing and network systems, many constructs that are implicit in a DES must be created for ABM. For instance, there must be physical room in the environment for agents to stack up in front of a server or machine to form a queue, so accurate physical spacing is a must for accurate ABM [151]. Reasons such as this means that ABMs are typically more difficult to construct than DESs for queuing systems [119, 151]. However, its flexibility means that ABM is also a candidate to model and optimize the schedule for complex systems.

2.3.3 System Dynamics

The final commonly employed modeling construct is system dynamics. In a SD model, “the real-world processes are represented in terms of stocks, [. . .] flows between these stocks, and information that determines the values of the flows [31].” The stocks can represent anything from raw material supplies to information, money, and workforce effort allocation [31, 150]. System dynamics models are developed as a system of differential equations that operate on continuous, aggregate stocks. As such, there are no distinguishable individuals in the stocks or queues [31].

System dynamics models are best used to model high-level properties of a system [31, 91, 100, 119]. The review paper by Jahangirian et al. indicates that SD is primarily applied to high level domains such as strategy development, project management, and supply chain management [91]. In more detailed domains, such as production planning, resource allocation, and scheduling, there were either no papers on SD applications to these fields or the number of SD papers were significantly less than those leveraging ABM or DES [91]. Therefore, because SD is too abstract to effectively model the details of a discrete manufacturing environment, it is not a suitable technique for further investigation. As such, the following section focuses on a comparison of DES and ABM to decide which framework to carry forward.

2.3.4 Comparison of Discrete-event Simulation and Agent-based Modeling

The two candidate modeling techniques that possess the required characteristics for application to this thesis are discrete-event simulation and agent-based modeling. Table 4 provides a summary of the frameworks’ capabilities compared to the metrics of interest. The table includes System Dynamics for completeness, but because SD is not applicable for this study, it is not further considered. DES’s strength lies in the fact that it inherently contains many modeling constructs and classes to model manufacturing and queuing networks, which typically makes modeling these environments easier in DES than ABM [119,151]. While not having these built in constructs, AMBs are much more flexible than traditional DES and can better model systems containing many dynamic decisions [119].

Table 4: Summary of Modeling Frameworks

	Discrete-Event Simulation	**Agent-Based Modeling**	**System Dynamics**
Capture high levels of complexity	●	●	◔
Ease of implementation	●	◓	◕
Automated model generation	◕	◓	Not-applicable: SD is not able to capture the granularity required.
Interface with optimization	●	●	●
Reduced runtime	◓	◔	●
Evaluate robustness	●	●	◓

The proposed application requires an approach that can be automated to some extent (to reduce the modeling burden) and can be easily linked to an optimization routine. Because DESs are more inherently amenable to production modeling, this should lead to a more easily defined process to automate the model building. Additionally, the model must effectively interact with an optimization routine. The optimizer defines a plan to be followed, and then the model should be able to take this plan and approximate how the system performs. This capability can be better incorporated into a DES than an ABM because ABM typically approaches scheduling by allowing the individual agents to make process decisions (somewhat like dispatching rules). Requiring an ABM to follow a defined schedule removes much of the learning and adaptation that motivates the use of ABM. Finally, ABM requires a sacrifice of accuracy to reduce runtime because of its continuous clock formulation. DES, on the other hand, is able to maintain accuracy and granularity because it only calculates properties and functions at required times [151], so DES should be able to limit runtime better than ABM.

One aspect of ABM that would be useful is to dynamically make decisions when disruptions occur on the factory floor. Because the optimizer has to develop a plan assuming that there are no disruptions, allowing agents to decide when to rework a component would be helpful to ensure that the schedule is followed as it would be in the "real world." Modern simulation software, however, typically contains aspects from all three paradigms [100]. Therefore, while discrete-event simulation is the primary paradigm used throughout this thesis, ideas and constructs from ABM can also be leveraged where appropriate.

With discrete-event simulation chosen as the primary modeling framework, the following section provides an overview of relevant discrete-event simulation modeling concepts. Unless otherwise specified, "simulation" henceforth refers specifically to discrete-event simulation. Section 2.4.1.1 then discusses common uses for simulation and addresses optimization approaches regularly used in conjunction with simulation. Next, it presents the advantages and disadvantages of simulation. Section 2.4.2 concludes with the formulation of hypotheses aimed at answering Research Question 1.

2.4 Overview of Discrete-Event Simulation

Discrete event simulation models do not have a closed form, mathematical solution. Therefore, to understand how the models are used to estimate the response of a system to varying input parameters, a description of the fundamental modeling parameters is presented. One initial note is that most modern simulation languages embrace object-oriented programming to increase code re-usability and ease modeling [23, 57, 100]. As such, many of the modeling concepts describe different types of objects, classes, and interfaces typically found in simulations. The interested reader may consult any number of discrete-event simulation references [18, 57, 99, 100, 110] to gain further knowledge about the typical formulations and detailed overviews of simulation concepts. The following overview highlights some of the concepts relevant to this thesis.

The first modeling concepts of interest are called *entities*. Entities typically represent components, people, or information that flow through the modeled system [99, 100]. Throughout the process, the entities may change state (e.g. raw material is machined into a sub-component). The states are tracked via attributes that are attached to each entity. The entities also track statistics about their movement through the system (such as flow time or waiting time). These statistics can be collected and summarized at the end of the simulation run.

Throughout the simulation run, the entities request the use of *resources* to complete various processes [110]. In the manufacturing sense, these resources can be machines, technicians, transporters, etc. Like entities, resources also have various states (such as Busy, Idle, or Off-shift). These states are also tracked through the simulation run.

The final primary components of any simulation are the *queues* present in the model. Discrete-event simulation works by processing lists (queues) of entities using resources according to specified rules and logic. Any entity proceeding through the process moves by joining queues of other entities waiting for resources. The primary modeling challenge is then to correctly link every queue and resource to provide a path for entities to follow that mimics the real world system.

2.4.1 Common Discrete-event Simulation Implementation Process

The primary use for discrete-event simulation is to evaluate the performance of proposed changes to systems that are too complex to model analytically [110]. Because simulation is typically stochastic, each run only produces an estimate of the objective function. As such, using single runs for optimization is commonly viewed as bad practice when considering uncertainty, and running multiple replications to produce statistics for optimization is typically seen as too time consuming, even for modern computers [110].

The generally applied simulation development and implementation process is outlined in Figure 7. This process involves first collecting data and documenting assumptions. The team then decides if a simulation using the data and assumptions described within an assumptions document is valid and detailed enough to accomplish the study's objective. Next, the actual simulation is coded and verified against logic or programming errors. Finally, the simulation is validated against the real-world system [110].

Once the model has been validated, the simulation can be exercised to determine the effectiveness of proposed improvements to the system. The first step is to design experiments that test the various proposed alternatives [18, 110]. This step also involves determining the appropriate number of replications to run, warm-up period length, and the length of time to simulate [18]. Once this is accomplished, the experiments are run and the data generated is collected and analyzed. The final recommendations are then documented and discussed before the final solution is implemented [18, 110].

As discussed, simulation studies typically follow a rigorous development, verification, validation, and experimentation process to determine results. Discrete-event simulation studies, in particular, are typically formulated to assess the effectiveness of a relatively small number of considered alternatives [18, 99, 100, 110]. These alternatives often represent large, strategic decisions (like whether or not to buy an expensive piece of equipment), and therefore, companies are willing to invest significant resources into investigating the decision [37, 58]. The following section will discuss how simulation is utilized to schedule production processes. Note that the common

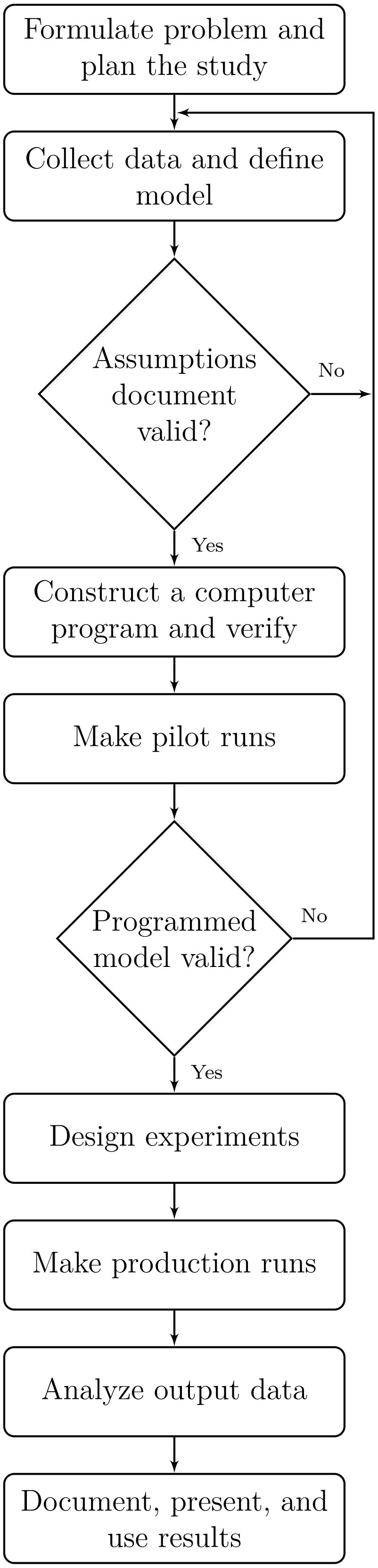

Formulate problem and plan the study
Collect data and define model
Assumptions document valid?
No
Yes
Construct a computer program and verify
Make pilot runs
Programmed model valid?
No
Yes
Design experiments
Make production runs
Analyze output data
Document, present, and use results

Figure 7: Steps in a simulation study. Reproduced from reference [110].

implementation of simulation for scheduling does not consider uncertainty when planning the schedule. It also commonly does not optimize the predictive schedule and instead focuses on evaluating various dispatching rules, which may not provide an optimal solution.

2.4.1.1 Scheduling via Simulation

In addition to being used to evaluate alternative system designs, discrete-event simulation is utilized to generate production schedules [17, 37, 79, 84, 100, 167]. This is accomplished by first instantiating the simulation to the current state of the production environment. This current state includes any work in progress, current machine states (e.g. online, in maintenance, etc.), and orders to be fulfilled. With the current state running, dispatching rules are applied to determine the order in which to complete jobs through the factory [17, 37, 79, 84, 100, 167]. To create a schedule, all uncertainty within the model is turned off; this allows the model to represent the expected system performance [100]. The model is then run from the current state and executes jobs based on the defined dispatching/selection rules. Throughout execution, the model tracks the start and completion of jobs along with the resources being utilized. With this information, the model then exports the schedule that can be distributed to the production floor [17, 100, 167].

While removing uncertainty may be acceptable when the scheduling time horizon is small [79], doing so often leads to overly optimistic schedules as the horizon expands because disruptions are more likely to occur [100]. However, because the simulation likely includes more details than an algorithmic schedule model, the schedule produced from simulation is likely to be more accurate [100]. Similar to some robust scheduling methods, the proposed schedule's robustness could be estimated based on the buffer time within the schedule. This approach, however, lacks the ability to effectively measure the schedule's quality robustness. Therefore, by replicating the schedule with the simulation's stochasticity turned on, the riskiness of meeting deadlines can be evaluated [100]. In this way, a job with low slack time proceeding through a well controlled process has lower risk than one with a high amount of slack time but a poorly controlled process. Assessing this risk and the impact of many disruptions that may occur during schedule execution is a strength of this "risk-based planning and scheduling" approach [100].

Scheduling jobs utilizing dispatching rules within simulations does have some benefits over the algorithmic dispatching rules methods discussed in Section 2.2.1.1. This method allows the planner to model more complex systems and apply more detailed dispatching rules than the more traditional methods [100]. Additionally, simulation allows for the rules and priorities to be easily modified to generate many potential schedules [84]. Then, based on the planner's requirements, the "best" schedule can be chosen among the candidates [84], and by replicating each schedule multiple times, the schedule risk can be analyzed and traded [100].

There are, however, still many disadvantages to this approach. The first being that there is still no way to know if a dispatching policy produces a "good" schedule, so there may be significant trial and error involved to determine a solution [84]. Additionally, this analysis is typically driven by the materials resource planning (MRP) system that determines material and capacity levels. In some cases, this release is infeasible, and no set of priority rules can make it feasible [84]. Finally, this approach develops a schedule by applying rules to a deterministic model. If there is significant variability in the system, the generated schedule may not be very applicable to the realized system [84]. In such cases, replicating the schedule can show the planner that the developed schedule may perform poorly, but in this scheduling paradigm, this information is generally not used to guide the search for a more robust schedule.

While the previously mentioned strategy is the most common way for simulation to be used to schedule processes within industry, simulation has increasingly been paired with metaheuristic optimization routines to improve a process while also considering variability [62, 79]. In this instance, the simulation is used to estimate the performance of a design by running multiple replications through the system. Then, the metaheuristic optimization routine provides candidate solutions to test through the simulation [62, 63, 79]. The typical simulation-optimization procedure is outlined in Figure 8. In this paradigm, the optimization routine selects candidate solutions and feeds them to the discrete-event simulation. These candidate solutions commonly consist of tuning heuristic selection rules within the simulation [79, 104, 161]. The simulation is then run for a specified number of replications to enable the statistical estimation of the system's performance. The performance metrics are then fed back to the optimization routine where the process is repeated until some convergence criteria or maximum computational effort is reached [62, 63].

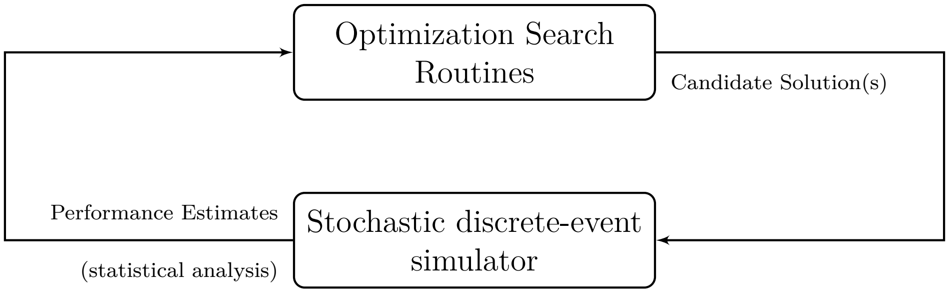


Figure 8: Optimization for simulation framework. Reproduced from reference [62].

The simulation-based optimization framework described above has the potential to generate robust, proactive schedules for problems too complex for traditional, algorithmic scheduling strategies [62]. Even with this potential, there are challenges that prevent fuller implementation. First, even with the continuing increase in computational power, computational time is still seen as a limiting factor for the implementation of simulation-based optimization [38, 39, 62, 110]. Scheduling problems typically exacerbate this problem because of their high dimensionality and combinatorial nature [62]. Additionally, similar to the problems with applying many scheduling advancements in industry, there is typically no general approach to successfully implement simulation-based optimization to "real-world" problems. Therefore, implementing simulation fully within industry, especially for simulation-based schedule optimization, has significant room for advancement [79].

2.4.2 Discussion

Section 2.4 has described the general formulation of discrete-event simulation methodologies and discussed their common implementations. Discrete-event simulation is primarily seen as a tool to be used to evaluate large-scale decisions (such as whether or not to buy a new machine) or potential changes to production strategies (such as a modification to workforce levels). The simulation paradigm is also used to generate manufacturing schedules. Primarily, this is accomplished by initiating the simulation to the current factory state and then running the simulation with different dispatching rules to generate a schedule. In this instance, the uncertainty in the simulation is turned off initially to create the expected schedule, and then the resulting trial schedules can be replicated with the uncertainty to predict risk. While this strategy enables scheduling for complex systems that may not have been possible with other scheduling systems, it still relies on dispatching rules that have uncertain performance. Even with the post-schedule generation risk analysis, the method may not be able to find a robust solution if the proposed dispatching rules are not appropriate for the situation.

The implementation of simulation-based optimization begins to merge many of the benefits of simulation with the usefulness of metaheuristic optimization routines. Simulation is able to estimate the performance of systems too complex for algorithmic approaches, and by running replications of a schedule, can estimate the robustness of a solution (by investigating the solution's variance or upper quantile). Coupled with a metaheuristic optimization routine, the simulation can be guided to a more robust solution by using knowledge gained from replicating a trial point. However, metaheuristic optimizers typically require many function calls to find a good solution, so even with ever increasing computational speeds, the typical implementation using a detailed simulation may not be fully viable.

Through this discussion, hypotheses can be formulated in response to the research questions posed in the previous section. The first sub-hypothesis (in response to Research Question 1.1) details how to effectively model modern, complex production systems for scheduling purposes:

Research Question 1.1: Which modeling techniques can be applied to effectively model increasingly complex production systems for use in the proposed schedule optimization methodology?

Hypothesis 1.1: If discrete-event simulation is leveraged, then increasingly complex scheduling environments can be modeled effectively such that the information required for use in a selected optimization routine can be captured.

This hypothesis focuses on the fact that discrete-event simulation is a proven paradigm for modeling highly complex systems. By leveraging simulation, simplifying assumptions that could cause schedule models to be incapable of effectively modeling complex systems do not need to be made. Additionally, modern simulation packages have embraced object-oriented programming, which can reduce modeling effort and improve reusability. Better user interfaces help to reduce the learning curve for planners new to simulation. Taken together, the discrete-event simulation paradigm should prove to be an efficient modeling framework to model complex systems at the level of detail required for this study.

With a modeling framework capable of effectively representing the systems of interest, the research will turn to improving the system through optimization. To address this, the following hypothesis is proposed:

Research Question 1.2: Which optimization technique(s) should be implemented to adjust the developed model effectively to search for optimal schedules?

Hypothesis 1.2: If a metaheuristic optimization routine is linked to the developed discrete-event simulation schedule model, then installation plans with improved performance can be efficiently identified.

This hypothesis focuses on the ability of the simulation to provide estimates of objective functions for use in optimization. A metaheuristic optimization routine is chosen as the most likely to succeed because it only relies on knowledge of the objective function, which can be easily extracted from simulation models. Metaheuristics, as discussed previously, are also commonly amenable to parallelization and can be multi-objective. With the class of optimization algorithm selected, the following sub-hypothesis can be developed:

Research Question 1.3: How can the selected optimization technique(s) be utilized to improve the schedule's robustness?

Hypothesis 1.3: If the optimization routine and model can estimate robustness related responses (quality robustness) and support multi-objective optimization, then the methodology will be capable of finding robust schedules.

There are two main parts of this hypothesis that must be true to enable the methodology to find robust schedules. First, the model to be optimized must have the ability to evaluate the robustness of a proposed solution. This means that deterministic schedule models, in addition to being too simple for many complex systems of interest today, cannot provide the required information for use in robust optimization. Fortunately, simulation techniques are inherently designed to include uncertainty and are, therefore, appropriate for this application.

The chosen optimization routine also must support multi-objective optimization. This is because many scheduling decisions involve competing objectives which must be balanced to provide a robust solution. With a multi-objective formulation, one could directly include some of the various forms of robustness measures as discussed in section 2.2.2.3. Furthermore, heuristic metrics can be included alongside the direct estimation of quality robustness. Also, by including schedule stability or flow time variation as objective functions along with traditional objectives such as average cost or make-span, the optimizer could directly search for robust solutions. This is in contrast with many strategies that apply heuristic objectives, such as increasing buffer time, as a surrogate for true robust objective functions.

These three developed hypotheses then lead to the overall hypothesis for this portion of the research:

Research Question 1: How can the challenges of implementing scheduling techniques be overcome to provide a system capable of producing robust schedules to reduce cost and delays?

Hypothesis 1: If a schedule is modeled at the appropriate level of detail via discrete-event simulation and optimized with a multi-objective, metaheuristic algorithm, then the methodology is capable of improving the robustness of complex systems' schedules.

This hypothesis stresses the need to develop both models and appropriate optimization frameworks in order to improve the robustness of complex systems' schedules. The modeling must be performed at the correct level of detail. Too much detail increases the time and resources required for model creation and inflates run-time. However, too little detail may not represent the system well enough and can fail to provide the necessary information to make decisions. By combining the DES model with the optimization routines described previously, the hypothesis predicts that the methodology can generate robust schedules.

2.5 Implementation Challenges

In the previous sections, various scheduling, simulation, and optimization techniques were reviewed. Individually, most if not all of the described techniques are relatively mature and have been implemented in some fashion. One also notices that the increase in computational speed has facilitated the deployment of more advanced and computationally intensive scheduling methods. Starting from the efficient but scope-limited exact methods to the broader but complexity-limited heuristic and metaheuristic schedule optimization models to the simulation-based optimization methods that can handle highly complex systems, the methods advance alongside the gain in computational power [62]. Hence, as stated in Assertion 2, the technological challenges preventing the effective modeling and simulation of complex production and manufacturing systems can be overcome.

Assertion 2

The lack of widely accepted robust scheduling practices is primarily due to implementation challenges. There is no inherent technological barrier to their application.

The following section, therefore, focuses on the implementation challenges identified in Section 2.2.3, specifically the need to reduce implementation time, reduce the time to identify quality solutions, improve the link between scheduling and simulation practitioners, and better incorporate the human decision maker's knowledge.

The challenges specified in Assertion 2 encompass multiple barriers that have prevented recent advancements from being implemented in industry. The first challenge, as discussed in Section 2.2.3, is that many academic advances have either been too mathematical or too specific for widespread implementation. Furthermore, these methods are commonly very rigid, which limits the ability to take advantage of the human planner's knowledge. Also, while studies and recommendations detailing the implementation of scheduling systems have been performed, the focus was primarily placed on database design and other activities related to materials resource planning [148]. Research into the design of systems and methodologies that implement new advances in scheduling is generally lacking [60].

Developed optimization strategies are also commonly designed for too specific of an application or are not transparent enough to allow users to modify an algorithm for his or her own application [62]. Improving these shortcomings by developing a more general modeling and optimization framework is a desirable feature to help improve industry adoption [62]. It is documented that the implementation of problem-specific solution strategies can be very time consuming [146], so a more generalized framework that is able to reduce the problem solving time, in both implementation and solution identification, can help to convince management to invest in the simulation study [58].

The final large implementation challenge stems from the fact that many manufacturing planners, who are typically well versed in algorithmic modeling and optimization methods, generally view simulation as a tool that is separate from their domain [79]. As such, it may be difficult for the planners to appreciate the benefit that simulation-based strategies can provide. This is related to the "Big Challenge" identified by Fowler for better integrating modeling and simulation with manufacturing: greater acceptance of modeling and simulation in industry [58]. Hence, providing an interface between the traditional scheduling paradigm and simulation-based methods may help to overcome this challenge.

These identified implementation challenges reinforce the proposition that a developed scheduling system must not only provide high quality schedules, but must also be deployable to the "real world." This leads to the following research question:

> **Research Question 2**: Does a methodology that improves the interface between scheduling, simulation, and the human planners better address the needs of the planners?

This overall research question motivates the development of a "deployable" scheduling system. The implementation challenges described above must be overcome to encourage acceptance and deployment of the methodology. Two quantifiable challenges (the need to reduce the problem setup time and the time to identify quality solutions) lead to the development of the following research questions. Following the investigation of these sub-research questions, further requirements to better link scheduling and simulation practices are investigated in response to Research Question 2. The first sub-research question investigates the possible means to reduce the methodology's implementation time:

> **Research Question 2.1**: How can the methodology's setup time and effort be reduced to encourage further adoption within industry?

The first sub-research question is important to encourage further adoption within the industry because constructing the DES model is a very time consuming process [58]. A reduction in problem solving time helps to increase the time available for problem solving iterations and reduce the financial burden on the company [58]. In doing so, reducing the problem definition and setup time could help to increase simulation's acceptance within the industry by making its implementation less costly.

While reducing the problem definition and simulation setup time is important to improve the strategy's usefulness and reduce its initial barrier to implementation, the methodology must ultimately provide meaningful results within a reasonable time horizon. As such, the following research question is developed:

Research Question 2.2: How can the effectiveness of the methodology in terms of solution quality and computation time be improved to make implementation of a simulation-based scheduling methodology economically viable and operationally feasible?

The following section addresses Sub-research Questions 2.1 and 2.2 by investigating potential ways to reduce the methodology's implementation time. Hypotheses are then formulated to address the research questions.

2.5.1 Methodology Implementation Time Reduction

The implementation time for the proposed modeling and optimization framework involves first modeling the complex system of interest and then incorporating the model into an optimization framework. The modeling process follows a process similar to that shown in Figure 7 but with optimization integration inserted. To facilitate a discussion of potential efficiency and effort-reduction opportunities, the steps are grouped into the following phases [58]:

1. Model Design
2. Model Development
3. Model Deployment

The following subsections discuss potential ways to improve efficiency during each phase of the simulation study. For the purposes of this work, the integration with the optimization routine is included within the Model Deployment phase of the study.

2.5.1.1 Potential Improvements During Model Design

Model design identifies issues to investigate throughout the study, plans the execution of the project, and develops a conceptual model [58]. One of the primary goals of this phase is to identify the level of complexity to include in the model [37, 58]. The driving force behind the selection of the level of detail to model is the identification of the questions to be answered through the study [18, 37]. The inclusion of more detail than required to support the project's goals should be limited because it adds to the time required to gather preliminary observations and data about the system, increases the amount of programming and debugging required, and increases the runtime, and cost, required to obtain a solution [57]. Therefore, ensuring that the details and assumptions developed during model design are appropriate is critical to ensure that the model is efficiently developed during the subsequent phase [18, 37, 57].

2.5.1.2 Potential Improvements During Model Development

The model development phase involves building the model, verifying the model's logic, and then validating its outputs against actual available data [18, 58]. Fowler et al. identified the model building phase as an area that requires a significant amount of effort and has room for efficiency improvements [58]. One more recent improvement to simulation languages that has improved model building efficiency is object-oriented programming (OOP) [58, 100]. Simulation languages that fully leverage the object-oriented paradigm help to improve "models' reliability, robustness, reusability, extensibility, and maintainability [100]." By using objects to create high level constructs, such as servers, the expertise required to create the simulation models is lowered [58, 100]. Finally, because objects enable complex problems to be reduced to smaller blocks, OOP helps to facilitate the verification of the final model [58, 100].

The next step for improvement is to leverage the object-oriented paradigm to automate the creation of the simulation model. There is a plethora of information available in modern manufacturing environments, from process routing to the time to complete tasks throughout the factory [58]. Leveraging this information to automatically build the simulation model inherently speed up the model construction time. Furthermore, if previous work dedicated to identifying process constraints and preferred work flows completed during a traditional scheduling exercise can be leveraged, much of the effort spent gathering data and modeling complex resource constraints may be reduced. Automated model generation should also help in the verification process. By eliminating the potential for human error during model creation, a model automatically generated from individually verified components should have fewer to no issues stemming from integrating the components [58]. As such, an automated model generation scheme can, after an initial modeling investment, greatly speed the model development phase and help alleviate the financial hurdle to implementing a simulation study.

2.5.1.3 Potential Improvements During Model Deployment

The main focus for efficiency gains during the model deployment phase is the execution time of the model [58]. For this thesis, the time required to optimize the system is also included. Therefore, focus is placed on reducing the time required for a single simulation replication as well as the computing time and the number of replications required to optimize the system. The primary means to reduce single run execution time, as discussed previously, is to limit the amount of detail captured by the model [58]. Hence, selecting the correct level of fidelity and focusing on efficiency throughout the modeling effort is essential to this goal.

The first technique to reduce the time to optimize the system is parallelization. Modern simulation software includes the ability to distribute individual replications across multiple computing cores or even multiple computers over a network. With the abundance of computing power commonly available today, this relatively simple addition can greatly speed any optimization run. As such, maximizing the problem's parallelization is an important first step to reducing the time for optimization.

Another approach to improving optimization speed is to intelligently decide on the number of replications required for each trial solution [62]. Traditionally, the number of replications is decided *a priori* in an effort to populate the response distributions with enough information to estimate the system's performance to a specified confidence level [160]. During optimization, however, this approach is typically inefficient as replications are wasted evaluating solutions that are clearly sub-optimal [62]. Indeed, the goal of optimization is to find the "best" design, not to provide an exact estimate of the performance of the optimum. As such, much computational effort is wasted evaluating the performance of designs that are clearly inferior to others [39,62]. Therefore, developing a procedure to intelligently run replications so that the algorithm spends most of its time evaluating "good" solutions can help to improve the optimization's efficiency [39, 62].

While the previous set of techniques focuses on reducing the time required for the optimization routine by reducing the time to evaluate solutions, the following techniques attempt to reduce the number of solutions that must be evaluated. If the optimization technique employed is an evolutionary algorithm, seeding the initial population with "good" points may help reduce the number of generations required [66, 147]. For scheduling problems, the initial population can be seeded with points generated through various heuristic or dispatching procedures [66, 147]. Combining these solutions with randomly generated solutions can help to ensure that the algorithm still explores the design space while having well-performing initial points to exploit. This procedure also ensures that an evolutionary algorithm with elitism (the best designs are always kept) never perform worse than a heuristic method used to create some of the initial population members [66].

Beyond improving the initial population, some potential modifications to the search strategy can also be made to improve the convergence of scheduling problems. One such strategy is to hybridize a genetic algorithm with an intermediate local search [66]. This is a promising strategy to hone in on a solution because while genetic algorithms are good at exploring the design space, they typically have difficulty converging [66]. As such, a local search can be used to fine tune individual population members after the genetic operations have been completed. In this way, the genetic algorithm can handle the global search, while the local search is used to speed the convergence of individual population members [66].

Similar to the previous strategy, the mutation strategy of a genetic algorithm may be modified utilizing information beyond the fitness function [147]. In a scheduling problem, delays and disruptions typically occur on the most highly utilized (or bottleneck) machine. Any process information that shows where delays or disruptions are occurring could be used to guide the potential jobs or sequences for mutation. For example, *intelligent mutation* strategies are developed such that if a child is selected for mutation, a job being processed on a machine with the maximum workload is moved to one with the minimum workload [147].

Finally, in the case that a solution is required very quickly or the problem is too large that population-based optimization is not feasible even with parallelization, point-based optimization algorithms may be used. While these strategies do not intrinsically explore the space as well as population based methods do [29], they can potentially arrive at solutions more quickly. The effectiveness of the point-based optimization strategies compared to the NSGA-II are contingent on the problem's runtime (both in terms of single replication runtime and the number of replications required), complexity, and *a priori* knowledge of relative objective functions' importance.

With these potential improvements to optimization strategies applicable to scheduling delineated, the following section discusses requirements to overcome the "soft" implementation challenges, namely, the need to improve the link between scheduling and simulation while also better incorporating the human decision maker into the methodology.

2.5.1.4 Improved Link between Scheduling and Simulation

A potential means to improve the link between scheduling and simulation is to incorporate strategies developed for visual, collaborative decision making. In recent years, collaborative decision making and data visualization have been recommended as important methods to help identify solutions to complex, multi-faceted problems [78, 97, 98, 179, 181]. Furthermore, many problems faced in todays world require a multi-disciplinary team to arrive at a solution that satisfies all sides of the issue [78,168,184,212]. In situations where the users may have different backgrounds and preferences, each user may prefer or be more comfortable with one view over another [78]. When multiple views or representations are required, ensuring that each are interconnected is necessary to support collaboration across disciplines [78]. Therefore, identifying views and decision making strategies familiar to each user can help to facilitate collaboration amongst the stakeholders.

It is observed that "it seems necessary for viable scheduling systems to combine the best of historical human expertise, theoretical or mathematical knowledge, and the common sense of the current user [136]." Human schedulers are seen to have superior capabilities in the following areas [79,84,127,207]:

Objective Flexibility: As discussed previously, fully describing and capturing the objectives of a scheduling problem mathematically is extremely difficult. Humans, however, "can cope with many stated, non-stated, incomplete, erroneous, and outdated goals and constraints [79]." Furthermore, even without the use of stochastic scheduling techniques, the human can potentially identify schedules that are robust to disruptions [207].

Communication: Humans can communicate and negotiate with operators and customers. This can provide alternative solutions that a rigid optimization cannot identify.

Intution: Humans can better integrate past experiences and knowledge into their scheduling decisions. From knowing which operators to assign to a job to planning job A on machine 2 instead of machine 1, the human can implement this knowledge without directly quantifying the impact to the optimization routine.

Hence, an effective data visualization and user interface is likely necessary to support the human decision maker [45,60,79,125,198,201,205].

The purpose of providing this decision support tool is two-fold. First, providing a means to down-select among promising scenarios and potentially modify selected plans enables the human planners to utilize their experience to supplement the models, objectives, and constraints used to generate the solutions [60]. Secondly, the decision support tool can greatly improve the transparency of the solution process. Increased trust through increased transparency in the outputs of the tool are important to encouraging its usage [60,206]. Especially for critical decisions with high uncertainty and tight constraints, it is critical that the planner feels that he or she is in control and not using a "black-box" to obtain a solution [206]. Hence, providing an appropriate decision support tool to incorporate human experience and increase the transparency of the solution process can potentially enhance the understanding of results from a simulation exercise and improve the link between simulation and scheduling practitioners.

The following section formulates hypotheses for Research Question 2 and its sub-research questions. These hypotheses guide the design of experiments formed to test whether the features discussed increase the "deployability" of the proposed methodology.

2.6 Discussion

The previous section has discussed some strategies found within the literature that could help to address implementation issues preventing simulation from being more utilized by manufacturing planners to optimize schedules with uncertainty. A primary issue is that simulation studies are commonly seen as too time consuming to be worth the investment. Additionally, there is a more “social” challenge curtailing the use of simulation in the industry: a general lack of acceptance of simulation as an appropriate problem solving strategy [58]. The improvements discussed throughout this section to improve the model design, development, deployment, and analysis phases have the potential to help overcome the identified challenges. Therefore, the following hypothesis in response to Research Question 2.1 is formulated:

> **Research Question 2.1:** How can the methodology’s setup time and effort be reduced to encourage further adoption within industry?
>
> **Hypothesis 2.1:** If the advanced object-oriented nature of modern discrete-event simulation packages is leveraged to help automate model generation and if metaheuristic algorithms are appropriately implemented to increase the optimization’s flexibility, then the methodology’s implementation time and effort will be reduced.

Hypothesis 2.1 stresses reducing modeling time to improve the time to implement simulation for manufacturing scheduling. The hypothesis states that the object-oriented nature of modern simulation languages can allow for automated model generation. It also states that the problem-agnostic trait of metaheuristic algorithms can support the development of general optimization frameworks, thereby enabling them to be applied fairly quickly. This hypothesis posits that by leveraging these features, the time required to implement the proposed simulation-based optimization study can be reduced.

While reducing the implementation time is valuable to help reduce the cost of initiating the methodology, the computational time required to identify a feasible and "good" solution must also be reasonable. In other words, the methodology must be capable of producing a solution in time to be usable within the planning horizon. Therefore, based on the discussion about strategies that have the potential to improve the model deployment time (Section 2.5.1.3), the following hypothesis is formulated:

Research Question 2.2: How can the effectiveness of the methodology in terms of solution quality and computation time be improved to make implementation of a simulation-based scheduling methodology economically viable and operationally feasible?

Hypothesis 2.2: If alternative optimization strategies are implemented, then the methodology can be used to explore and exploit the solution space quickly enough to make implementation feasible and viable for a wider range of time and resource constraints and solution quality requirements.

Hypothesis 2.2 recognizes that, in order to improve computation time enough to make the methodology feasible for use in a time restricted environment, the execution efficiency must be addressed at many levels. The first portion of the hypothesis recognizes that an important part of reducing the computation time is to limit the complexity of the simulation. A simpler simulation is preferable if it is still able to answer the study's questions and enable the user to judge competing designs. With an efficient simulation developed, focus must then shift to improving the optimization routine. Techniques discussed in the previous section provide promising strategies to reduce the time to reach an optimum while also potentially improving the solution quality. With these improvements in place, the hypothesis states that the optimization can be sped enough to make it usable for the development of time sensitive planning and scheduling problems.

With these sub-research questions addressed, the following hypothesis is formed in response to Research Question 2:

Research Question 2: Does a methodology that improves the interface between scheduling, simulation, and the human planners better address the needs of the planners?

Hypothesis 2: If the methodology requires a low amount of implementation effort and is shown to provide clear benefits with acceptable increases in computation time over traditional scheduling methods while effectively integrating the knowledge of the human planner, then the methodology can be successfully implemented to solve "real-world" problems.

This overarching hypothesis contends that, in addition to the improvements investigated through Sub-research Questions 2.1 and 2.2, properly integrating the knowledge of the human planner is needed to ensure deployability. This is intended to improve acceptance of the simulation results while also enabling the decision makers to utilize the simulation results without intimate knowledge of simulation.

The following section identifies the main gaps addressed by this research and concludes this chapter with a discussion of the overall research objective.

2.7 Gap Analysis

This research addresses multiple gaps identified within the literature. The first gap to be addressed is related to Research Question 1. The current scheduling tools have difficulty in modeling and optimizing the complex manufacturing systems that are commonly required for modern aerospace vehicles. These difficulties are the result of various limitations in commonly implemented methodologies.

The first limitation, as addressed by Sub-Research Question 1.1, is that the mathematical models typically employed to represent schedules have difficulty capturing the level of detail required to make well-informed decisions. Additionally, as addressed by Sub-Research Question 1.2, many traditional optimization paradigms (e.g. linear programming, branch & bound, etc.) are not suitable for application to simulation models. While applicable optimization paradigms have been developed, there is little research into how this interface can be improved for scheduling-specific optimization problems. The final limitation, from Sub-Research Question 1.3, is that many common practices do not explicitly account for uncertainty to improve the schedule's robustness. Uncertainty and disruptions will continue to increase with system complexity, so this consideration will continue to grow in importance.

The second gap identified in the literature is a lack of discussion about the perceived relevance of existing approaches and their implementation within the industry. It has been observed that many advances in scheduling, while well-formulated in academia, struggle to be transferred to industry. As such, Research Question 2 seeks to understand what makes a newly developed scheduling system deployable to industrial practitioners to encourage adoption. Usefulness is tied to reducing the time and effort required to both setup a system and use it to generate results. As such, opportunities to improve aspects of implementation throughout model development and execution have been identified in response to Sub-Research Questions 2.1 and 2.2. Finally, developing the method to effectively work with the human planner is key to encourage implementation.

Addressing these gaps called for the development of a new methodology that enables robust scheduling for increasingly complex manufacturing processes. This leads to the development of the research objective discussed in the following section.

2.8 Research Objective

From the gap analysis and assertions previously discussed, the following research objective is formulated:

Research Objective

To enable the integration of robust design principles with current, deterministic scheduling practices to efficiently schedule processes so as to reduce risk within increasingly complex production systems

To fulfill this research objective, a methodology is required that has the following characteristics. These characteristics are directly dictated by the hypotheses formulated in Sections 2.4.2 and 2.6.

Required Characteristics & Capabilities

1. Quickly and easily generate any required models from existing scheduling tools
2. Be mostly automated and integrated with current systems
3. Capability to evaluate schedule robustness
4. Integrate with an optimization routine that is suitable for the problem and is capable of improving the system's schedule robustness
5. Be scalable to manufacturing problems seen throughout the aerospace industry
6. Incorporate the knowledge and experience of the human planner

A methodology exhibiting these characteristics and capabilities can help reduce costs by supporting many levels of the organization. Example applications can be found in Table 5.

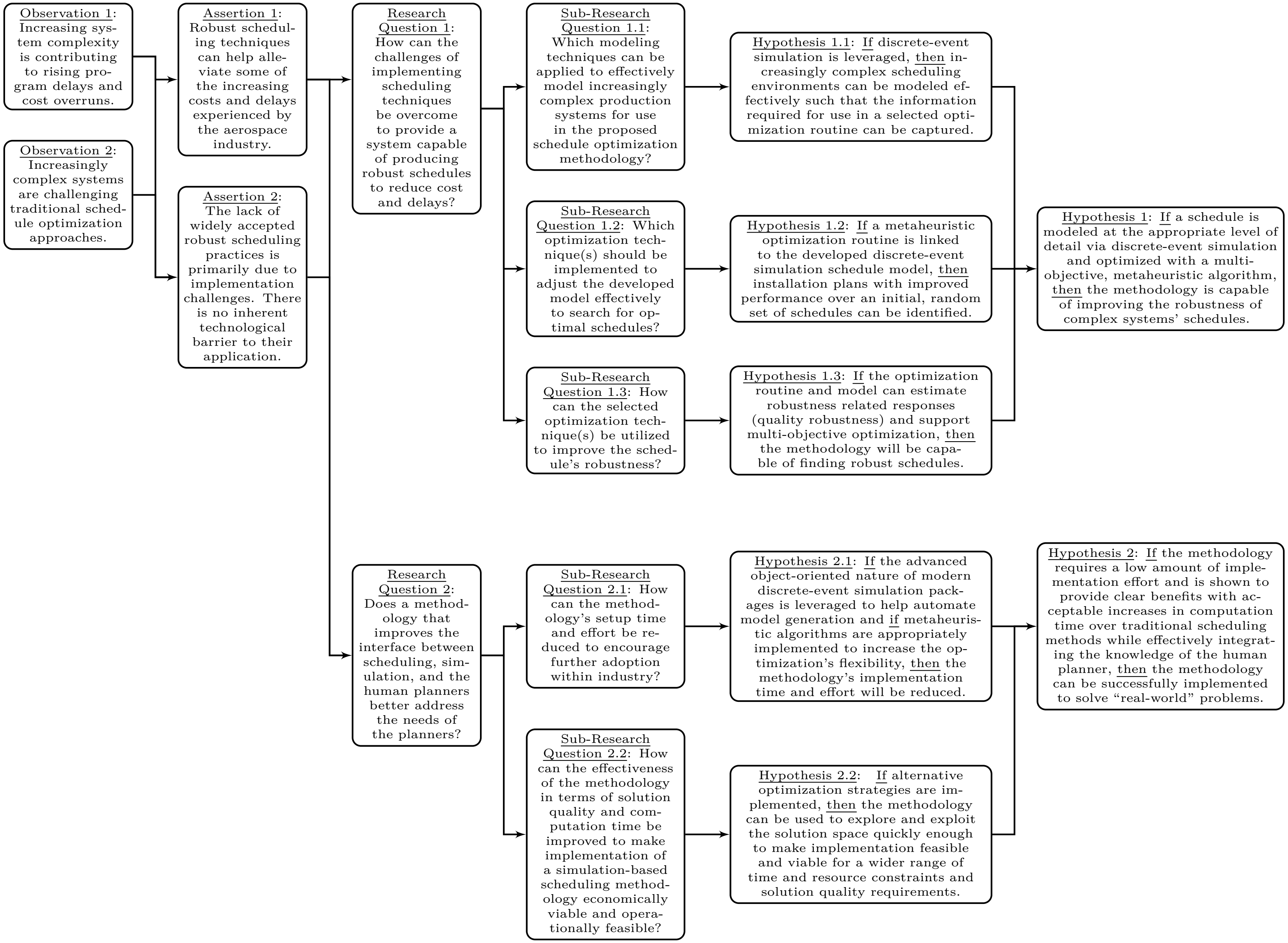
2.9 Chapter Summary

This chapter has reviewed common scheduling practices and has identified gaps that make it difficult to schedule processes for modern, complex systems. Through the review, research questions (Research Question 1 and its sub-research questions) designed to elicit characteristics of a successful robust scheduling system are developed.

Table 5: Examples of the Proposed Methodology's Benefit Throughout an Organization

Analysis Level	Management Level	Organizational Level
Support legacy tools to break down barriers to implementation	Identify overarching strategies for robust optimization	Reduce risk of schedule overruns

Following a further review of complex system modeling techniques, hypotheses are formulated that identify the necessary characteristics of a modeling and optimization framework to overcome the identified gaps. In addition, this chapter has identified the need to incorporate both technical feasibility and economic viability considerations into any developed methodology to ensure it can be deployed within an industrial setting. This leads to additional research questions (Research Question 2 and its sub-research questions). The hypotheses put forward then delineate additional requirements to reduce the barriers to implementation and increase the overall methodology's usefulness. This chapter then provides a summary of the identified research gaps and the research objective that are further addressed by the proposed methodology described in the following chapter. Figure 9 illustrates this dissertation's research structure. The following chapter discusses the methodology proposed to fulfill the research objective.



Observation 1: Increasing system complexity is contributing to rising program delays and cost overruns.

Observation 2: Increasingly complex systems are challenging traditional schedule optimization approaches.

Assertion 1: Robust scheduling techniques can help alleviate some of the increasing costs and delays experienced by the aerospace industry.

Assertion 2: The lack of widely accepted robust scheduling practices is primarily due to implementation challenges. There is no inherent technological barrier to their application.

Research Question 1: How can the challenges of implementing scheduling techniques be overcome to provide a system capable of producing robust schedules to reduce cost and delays?

Research Question 2: Does a methodology that improves the interface between scheduling, simulation, and the human planners better address the needs of the planners?

Sub-Research Question 1.1: Which modeling techniques can be applied to effectively model increasingly complex production systems for use in the proposed schedule optimization methodology?

Sub-Research Question 1.2: Which optimization technique(s) should be implemented to adjust the developed model effectively to search for optimal schedules?

Sub-Research Question 1.3: How can the selected optimization technique(s) be utilized to improve the schedule's robustness?

Sub-Research Question 2.1: How can the methodology's setup time and effort be reduced to encourage further adoption within industry?

Sub-Research Question 2.2: How can the effectiveness of the methodology in terms of solution quality and computation time be improved to make implementation of a simulation-based scheduling methodology economically viable and operationally feasible?

Hypothesis 1.1: If discrete-event simulation is leveraged, then increasingly complex scheduling environments can be modeled effectively such that the information required for use in a selected optimization routine can be captured.

Hypothesis 1.2: If a metaheuristic optimization routine is linked to the developed discrete-event simulation schedule model, then installation plans with improved performance over an initial, random set of schedules can be identified.

Hypothesis 1.3: If the optimization routine and model can estimate robustness related responses (quality robustness) and support multi-objective optimization, then the methodology will be capable of finding robust schedules.

Hypothesis 2.1: If the advanced object-oriented nature of modern discrete-event simulation packages is leveraged to help automate model generation and if metaheuristic algorithms are appropriately implemented to increase the optimization's flexibility, then the methodology's implementation time and effort will be reduced.

Hypothesis 2.2: If alternative optimization strategies are implemented, then the methodology can be used to explore and exploit the solution space quickly enough to make implementation feasible and viable for a wider range of time and resource constraints and solution quality requirements.

Hypothesis 1: If a schedule is modeled at the appropriate level of detail via discrete-event simulation and optimized with a multi-objective, metaheuristic algorithm, then the methodology is capable of improving the robustness of complex systems' schedules.

Hypothesis 2: If the methodology requires a low amount of implementation effort and is shown to provide clear benefits with acceptable increases in computation time over traditional scheduling methods while effectively integrating the knowledge of the human planner, then the methodology can be successfully implemented to solve "real-world" problems.

Figure 9: Summary of Research Structure

CHAPTER III

PROPOSED METHODOLOGY

The assertions and review of the current literature discussed throughout the previous chapters have identified gaps within current scheduling practices for low volume, complex production systems. The development of a new robust scheduling methodology (PORRTSS: Production Optimization to Reduce Risk Through Simulation-based Scheduling) with the characteristics established in the previous chapter is needed to fulfill the research objective formulated as a result of the identified gaps. The required characteristics are derived from the hypotheses put forward to answer research questions identified throughout the literature review. Therefore, by implementing the new methodology, experiments can be performed to test the hypotheses and, ultimately, evaluate the efficacy of the proposed methodology. An overview of the proposed methodology is provided in Figure 10. Each step of the methodology is discussed further through the remainder of this chapter.

3.1 PORRTSS Methodology Overview

3.1.1 Step 1: Model Generation

The goal of Step 1 is to generate a simulation model suitable for optimization using information readily available in a tactical level schedule model. To do so, a deterministic schedule similar to those currently used in industry is used as the input. This schedule at a minimum contains a list of tasks to be completed, the nominal time to complete each task or process, planned task start and end times, a set of resources (e.g. manpower, materials, etc.) required for each task, and precedence relations for each process. These pieces of information are contained in any scheduling program that is in use today, and would be generated during a pre-planning phase. With

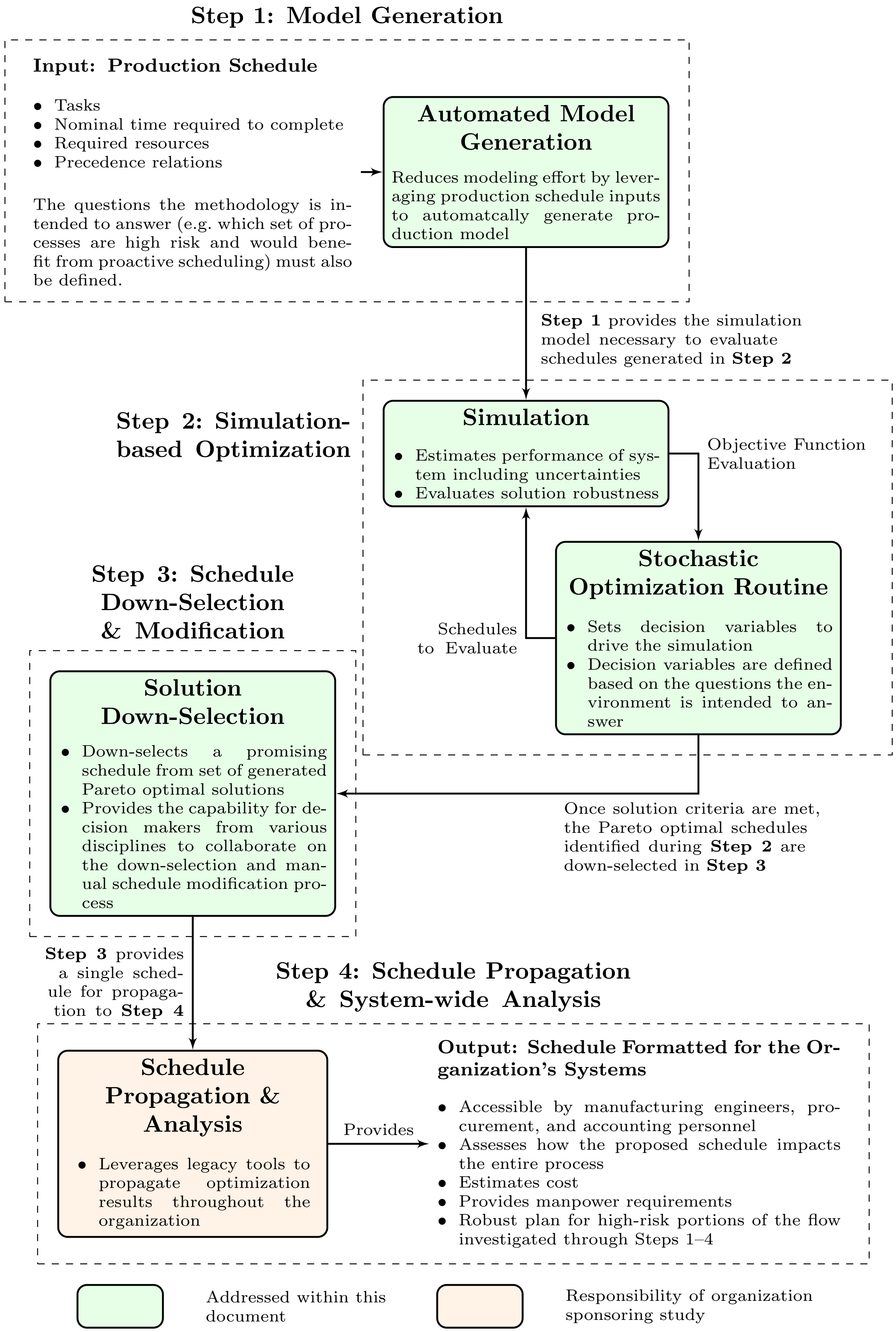

Step 1: Model Generation
Input: Production Schedule
• Tasks
• Nominal time required to complete
• Required resources
• Precedence relations
The questions the methodology is intended to answer (e.g. which set of processes are high risk and would benefit from proactive scheduling) must also be defined.
Automated Model Generation
Reduces modeling effort by leveraging production schedule inputs to automatcally generate production model
Step 1 provides the simulation model necessary to evaluate schedules generated in Step 2
Step 2: Simulation-based Optimization
Simulation
• Estimates performance of system including uncertainties
• Evaluates solution robustness
Objective Function Evaluation
Stochastic Optimization Routine
• Sets decision variables to drive the simulation
• Decision variables are defined based on the questions the environment is intended to answer
Schedules to Evaluate
Step 3: Schedule Down-Selection & Modification
Solution Down-Selection
• Down-selects a promising schedule from set of generated Pareto optimal solutions
• Provides the capability for decision makers from various disciplines to collaborate on the down-selection and manual schedule modification process
Once solution criteria are met, the Pareto optimal schedules identified during Step 2 are down-selected in Step 3
Step 3 provides a single schedule for propagation to Step 4
Step 4: Schedule Propagation & System-wide Analysis
Schedule Propagation & Analysis
• Leverages legacy tools to propagate optimization results throughout the organization
Provides
Output: Schedule Formatted for the Organization's Systems
• Accessible by manufacturing engineers, procurement, and accounting personnel
• Assesses how the proposed schedule impacts the entire process
• Estimates cost
• Provides manpower requirements
• Robust plan for high-risk portions of the flow investigated through Steps 1–4
Addressed within this document
Responsibility of organization sponsoring study

Figure 10: The PORRTSS: Production Optimization to Reduce Risk Through Simulation-based Scheduling Methodology

this information, an automated model generation procedure is developed and implemented to translate this information into servers, links, and resources required for the simulation model.

Simplifications to the simulation model can be made by leveraging decisions made during the original schedule generation. To illustrate, the original schedule optimization algorithm generates a optimal schedule (according to a defined objective) that accounts for all resource and precedence constraints. In this optimal schedule, tasks that have the same predecessors (and, hence, become available for completion at the same point) could require the same resources. In this case, the schedule optimization sequences these tasks to respect the resource constraints. By constraining the simulation to complete these tasks in the optimized order, even though the precedence relations do not specify this order, the simulation can approximate the resource constraints without directly modeling them. This helps to reduce the time to create and verify the simulation model while also limiting model run-time.

The outcome of this step is a simulation model that matches the performance of the input schedule but is also capable of incorporating more detailed logic and stochastic elements to better estimate the system's robustness.

3.1.2 Step 2: Simulation-based Optimization

The purpose of Step 2 is to incorporate the generated simulation, which is capable of estimating robustness measures, with a multi-objective, metaheuristic optimization routine to improve the robustness of the plan. With a sufficiently detailed simulation model in place, a stochastic optimization routine is then employed to intelligently set decision variables for the simulation. The objective function for the optimization includes an evaluation of system robustness. Robustness and risk are measured by the resulting schedule's quality robustness as well as additional heuristic metrics identified by the subject matter experts. Many options to improve the speed of the optimization for inclusion in the methodology are investigated. Once the solution criteria are met, the optimizer passes the proposed schedule(s) to a schedule down-selection decision support environment.

3.1.3 Step 3: Schedule Down-Selection & Modification

The goal of Step 3 is to support the experienced decision makers in the down-selection to a single schedule for implementation. Hence, this step provides data visualization and interaction capabilities to support collaborative decision making. This includes the capability to assess various levels of the plan: from high level, system-wide metrics to lower level, task-by-task interactions. Furthermore, the ability to modify the proposed plans and assess the results is provided to enable the planners to have ultimate control of the schedule. In doing so, this step helps to improve confidence in the methodology, incorporate expert knowledge from multiple backgrounds, and, finally, identify a schedule for system propagation during Step 4.

3.1.4 Step 4: Schedule Propagation & System-wide Analysis

As shown in Figure 10, Step 4 is not a new capability identified and tested within this work. Rather, Step 4 of the methodology is included as a means to “close the loop” and identify how the results from Steps 1–3 are incorporated back into the sponsoring organization’s overall planning system. As such, the capabilities discussed within this section are included for completeness but are neither developed nor tested throughout this work.

In most cases, the deterministic schedule models have the capability to perform some measure of Monte Carlo analysis to assess the schedule’s risk. While not as capable as the simulation, Monte Carlo analysis is able to interface more directly with the organization’s established systems and is likely to be more trusted than a new method. Therefore, a Monte Carlo analysis may be used to estimate how the proposed schedule’s risk propagates through the rest of the production plan.

After understanding how the proposed schedule impacts the overall process, the schedule is propagated through the system. This enables other members of the organization (from marketing and sales to the manufacturing engineers planning the step-by-step work orders) to utilize the more detailed and robust schedule in their work. As such, the methodology delivers a schedule formatted for use throughout the organization that estimates cost, provides manpower requirements, and assesses how the proposed schedule impacts the rest of the process.

Before detailing the specific implementation of the methodology in the next chapter, the next section discusses the case study that is used as a testbed for the methodology.

3.2 Case Study Description

The methodology is applied to plan the installation of sensors on an aerospace system. The major structural manufacturing and sub-system installation plan consisting of potentially over 1000 production steps is well defined. Planning sensor installations, however, is a more challenging exercise because each sensor has specific constraints on the times it can be installed during the primary production process. Some examples of the general constraints and requirements that may be important are:

- Some sensors require other sub-components to be integrated before they can be installed
- Access to the installation site could be blocked by other assemblies → the blocked sensors must be installed before access is blocked
- Installations must not occur during times when access is either unsafe or infeasible (e.g. during testing, inspections, moving operations, etc.)

The top-down design decision support process (Figure 11) [121] is commonly applied when addressing any engineering problem. As such, the following sections discuss how this process is implemented for the problem of interest:

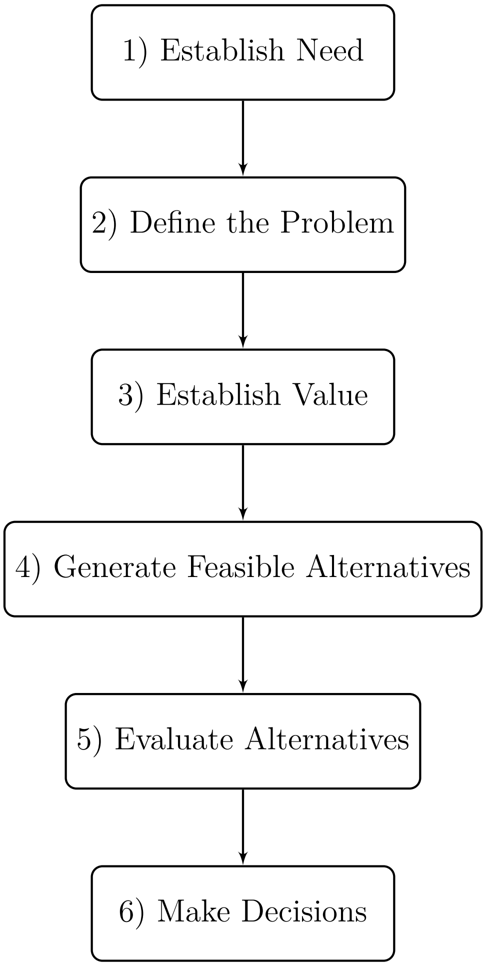


Figure 11: Top-Down Design Decision Support Process [121]

While the case study description provided in the following section briefly mentions the implementation of portions of the methodology, a full description of each step is contained in Chapter 4.

3.2.1 Establish Need

Sensor installations represent a significant portion of the manufacturing processes required to build the vehicle. Due to significant budget and schedule constraints, delays during production must be kept to a minimum. Being able to properly plan sensor installations as to minimize their impact on the primary production flow is thus critical to minimizing these potential delays.

The aerospace vehicle's production process is complex with high quality standards and many manual assembly and installation processes. This leads to a potentially highly variable production process. Adding sensor installations to this manufacturing system, with their limited installation windows and accessibility challenges, can only lead to increased variability and risk. Therefore, to help ensure that the overall schedule is met, controlling the uncertainty by developing a robust installation schedule is imperative.

3.2.2 Define the Problem

Sensor installations have traditionally been planned manually by leveraging manufacturing engineers' best judgment. In addition to being a very tedious and time consuming process that could delay other important planning activities, this traditional process is not able to quantify the impact on process time, risk, etc. of a chosen sensor installation plan. Because of this limitation, the developed plan may miss opportunities for system-level process improvement and risk reduction because of the low-level scope of the planning.

Furthermore, directly applying traditional scheduling methods to this problem is challenging. An objective function is difficult to formulate as a linear program or directed graph due to complexity related to the number of sensor installation opportunities available throughout the process flow. For instance, a sensor installation started in parallel with a primary process may not be completed during that process. In this case, depending on whether the next process is able to proceed alongside the sensor installation or not, this may or may not cause a delay. Additionally, dealing with specific rework scenarios (e.g. where a sensor may be damaged during installation) cannot be directly captured in common Monte Carlo implementations. As such, applying the PORRTSS methodology to this problem can help to reduce production risk by enabling the quantification of the impact of the sensor installations on the production flow. Such capability could motivate managers to replace the tedious, SME driven processes traditionally used to plan sensor installations by this more robust approach.

3.2.3 Establish Value

Implementing the PORRTSS methodology to this case study has two primary goals:

1. Improve the sensor installation plan for a major subassembly of a complex, low production rate aerospace vehicle by:

    - Minimizing the increase in process time from delays due to sensor installations
    - Improving the robustness of the proposed schedule by:
        - Accounting for process time uncertainty in the installation plan
        - Accounting for rework when optimizing the schedule
        - Providing options to install sensors early in the installation window to provide buffer for potential schedule changes or sensor damage
        - Planning for sensors that are physically close to be installed near each other in the schedule such that technicians may become familiar with the installation site
        - Quantifying the risk to the production schedule due to sensor installations
    - Minimize the manpower required to install sensors to free resources for other tasks
    - Maximize the number of sensors that are installed on the vehicle

2. Reduce the burden on planners by automating the scheduling process for sensors by:

    - Automatically building a model capable of quantifying the impact of sensor installation by adding details to information already contained within schedule models
    - Optimizing the schedule using the developed model and metaheuristic optimization routine
    - Providing decision support tools to support collaboration

3.2.4 Generate Feasible Alternatives

In order to identify feasible installation schedules, it is necessary to capture the constraints that impact or prohibit the installation of sensors. A wide variety of constraints exist for the problem: accessibility to the installation site, not allowing installations during unsafe or otherwise inhibiting primary production processes, etc. To facilitate this exercise, a compatibility matrix is defined that enumerates these constraints by matching each sensor installation with each primary production process during which it can be installed. A notional example can be found in Table 6.

Table 6: Notional Sensor and Process Compatibility Matrix

	Sensor 1	Sensor 2	Sensor 3	Sensor 4
Process 1	1	1	0	1
Process 2	1	0	0	0
Process 3	1	1	1	1
Process 4	0	0	0	0

The compatibility matrix maps each sensor to each primary production process to specify if the sensor can (1) or cannot (0) be installed in parallel with each process. This formal approach helps to organize the various constraints that could arise from multiple sources. With the constraints specified, feasible alternative installation schedules can be formulated. By specifying the constraints in this way, only feasible installation sequences are considered, which simplifies the optimization algorithm. Then, during model execution, the matrix is used again to check whether primary processes or sensor installations may continue based on process logic.

3.2.5 Evaluate Alternatives

With feasible alternatives defined, the model is then used to evaluate proposed installation schedules. Based on the requirements of the problem and by making conservative assumptions about the schedule provided, it was determined that simulating the process flow was sufficient to quantify the impact of the sensor installations (as opposed to a much more detailed physical simulation with actual locations modeled within the factory). With this model in place, each plan is replicated to provide an output distribution to evaluate the quality robustness of the proposed installation plan. The generation and evaluation of alternatives is guided by the implemented optimization routine.

The formal optimization problem statement is described in Equation 3:

$$\begin{aligned} \text{Minimize: } & \mathbf{F}\left(x_{labor}, x_{criticality}, x_i\right), \\ & i = 1 \ldots n_{sensors} \\ \text{Subject to: } & \textit{Compatibility Constraints} \end{aligned} \tag{3}$$

where: x_{labor} represents the maximum number of sensors that can be installed concurrently (due to a limited number of technicians) and $x_{criticality}$ denotes the criticality of a sensor that will not be re-installed if it is damaged during installation. For example, if a sensor that is included to gather data to validate a model is damaged, it may not be worth replacing. However, if a sensor critical for guidance or vehicle health monitoring is damaged, it must be re-installed regardless of the effort required. Next, x_i denotes the primary production process during which each sensor i is planned to be installed. Finally, the compatibility matrix accounts for all of the process constraints impacting the manual installations. All other constraints impacting the primary process tasks are accounted for by observing the process flow from the deterministic schedule. While the capability to include “criticality”-based scenarios is available, it is not currently used in the algorithm (because it is preferable to find scenarios where all sensors are installed). Therefore, the rest of this work assumes that all sensors, if they are damaged during installation, are re-installed.

The objective functions of interest are described in Table 7. Risk is reduced in multiple ways through the selection of the objectives:

1. The results of each replication are summarized by their median (to improve expected performance) and the 80th quantile (to reduce risk). In other words, it is possible that an installation plan has a good median performance, but because some of the sensor installations could cause significant delays, the 80th quantile could be significantly worse. The median and 80th quantiles of each objective are, therefore, included as separate objective functions for sorting. This accomplishes the general goal of robust scheduling by providing good average performance while also working to reduce risk (improving the quality robustness of the schedule).

2. Maximizing the slack time between a planned sensor installation and its final installation opportunity (as defined by the compatibility matrix) gives planners increased flexibility to re-plan an installation. This may be required if the sensor is damaged during the initial installation or the installation simply no longer works with the actual state of the factory during production. Increasing schedule slack time is an established practice to reduce schedule risk and helps to differentiate between installation plans that may have similar impacts on the process time.

3. In an effort to reduce the risk of major delays due to a critical sensor's installation error, additional slack time is allocated to these installations. This should provide more opportunities to discover a quality issue and re-install critical sensors.

4. Attempt to install physically close sensors around the same time within the schedule to reduce the time spent by the technicians getting familiar with the installation site. Furthermore, by bringing multiple sensors to a single installation site, the number of times the technicians must move in the vehicle is reduced, which could help to reduce the opportunities for damage.

By including these metrics, the risk to the production plan is *proactively* reduced by improving the plan's quality robustness and, by including slack time, rescheduling is eased. Finally, by closely scheduling neighboring sensor installations, the burden on the installation technicians should be reduced by increasing his or her familiarity with the installation location.

3.2.6 Make Decisions

Because the optimization routine returns a set of non-dominated solutions, a planner must make the final determination about which schedule is "best." This determination is made by considering the specific risk tolerance, available cost and time, and supplier information available at the time. Furthermore, these decisions require input from multiple stakeholders, industrial and manufacturing engineers along with the avionics team, so properly supporting collaborative decision making is essential. This is accomplished by providing a decision support environment.

Once a decision is reached, the chosen sensor installation schedule is fed back into the organization's overall production schedule. This will enable an evaluation of the developed plan on the overall schedule while also integrating the identified plan with the rest of the production schedule. By following this process, the organization

Table 7: Overview of Objective Functions

Objective Function	Description
Process Time	The time from the start of the first process to the completion of the final task during which work is completed. This excludes off-shift time (e.g. nights and weekends).
Slack Time Metric for All Sensors	The slack time metric for an individual sensor installation is the number of production processes between the planned installation and the final installation opportunity. The metric is used instead of actual simulation time to avoid encouraging delays in the process that artificially increase the time between a planned installation and its final opportunity. Mathematically: $\text{Slack Time} \equiv \sum_{i=i_{\text{Planned Installation}}}^{i_{\text{Final Installation Opportunity}}} \begin{cases} \text{Primary Process}_i \text{ is compatible,} & 1 \\ \text{Primary Process}_i \text{ is not compatible,} & 0 \end{cases}$ Slack time is a common heuristic designed to increase solution robustness. Increased slack time means that the sensor has more alternate opportunities to be installed if there is a delay or if the sensor is damaged during its planned installation. The *Slack Time for All Sensors* metric sums all of the slack times to provide an overarching measure of risk to compare between installation plans.
Slack Time Metric for Critical Sensors	This metric only sums the slack time for sensors defined as highly critical. In doing so, the mission critical sensors are favored with more slack to provide more opportunities for their installation without leading to production delays.
Closely Schedule Neighboring Installations	A significant portion of the time required to install sensors is spent locating and accessing the installation site. This metric favors scenarios where sensors that are physically close to each other are planned close together in the schedule. This metric is calculated by the following equation: $\sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{\lvert t_i - t_j \rvert}{max\left(\lvert \vec{x_i} - \vec{x_j} \rvert, 1\right)}$ (4) where: t represents the planned start time of each sensor installation and $\vec{x}$ represents the physical XYZ location of each sensor.

would create a robust installation schedule that mitigates risk and, ultimately, helps to reduce schedule delays and cost overruns.

The following section identifies how the PORRTSS methodology compares to the current system in place.

3.3 Current vs. Proposed Planning Process

The end goal of the planning process described throughout the previous section is to identify primary processes during which sensors can be installed without significantly impacting the baseline schedule. To better understand the potential benefits of implementing the PORRTSS methodology, an understanding of the current method common to project planning in complex manufacturing systems is provided.

Project and process planning and scheduling can be classified into multiple levels based on the level of detail and abstraction considered [60, 132, 136, 164, 165, 175]. Table 8 provides some common examples of this classification. The methodology is designed to help the transition from the middle/short-range (also referred to as Tactical phase [60]) to a more detailed scheduling (or Operational [60]) phase project plan.

The common method used to transition to the more detailed planning phases is to break down a project into manageable phases via a Work Breakdown Structure (WBS) [47,52,54,94]. A notional WBS demonstrating where the case study could fit alongside other common aircraft systems [154] is presented in Figure 12. Different levels of the WBS are leveraged during different levels of the planning process [47, 132]. For example, during the Tactical planning phase, the planners use macro level, aggregate work packages (e.g. Levels 2 and 3 in Figure 12) to perform “rough cut capacity planning [132].” Then, upon entering the more detailed, operational scheduling phase, lower level (e.g. Level 4 and possibly beyond in Figure 12) information is utilized to plan the actual work steps (e.g. drill holes at position X, locate sensor, install

Table 8: Classification of Scheduling Levels (Based on Reference [136] and Supplemented by References [47, 60, 132, 165, 175])

Level	Examples of Problem	Horizon
Long-range planning (Strategic)	Plant expansion, plant layout, plant design	2–5 years
Middle-range planning (Strategic)	Production smoothing, logistics	1–2 years
Short-range planning (Tactical)	Requirements plan, shop bidding, due date setting, macro process planning using aggregated work packages	3–6 months
Scheduling (Operational)	Job shop routing, assembly line balancing, process batch sizing, detailed work package planning	2–6 weeks
Reactive scheduling/control (Operational)	Hot jobs, down machines, late material	1–3 days

fasteners, etc.) required to complete the project or vehicle [47, 52, 132].

Various entities within the organization are responsible for executing each level of planning [95, 136]. Long and mid-range planning is primarily executed by Industrial Engineers (IEs) in concert with project management and finance offices [95, 157]. The IEs are primarily concerned with identifying system improvements that "serves the higher good or works to optimize the performance of the larger system [157]." When moving to the operational planning phase, Manufacturing Engineers (MEs), who are intimately familiar with the product's design and expected performance, work with the Industrial Engineers to identify "the machines, the equipment, the tooling, and the personnel to carry out the plan [95]." Hence, in the context of planning and scheduling, the MEs are concerned with developing a set of work orders and manufacturing steps to execute the higher level plan set forth by the IE department.

Because of the intimate knowledge and experience required to generate manufacturing plans (especially in low volume production systems) [95, 202, 215], detailed planning is primarily accomplished manually [79, 202, 215]. Especially as the complexity of the vehicle or system increases, this process puts a significant burden on planners who must develop instructions at very low-levels with limited ability to understand the system-level impact of their decision. Furthermore, because a significant amount of the planners time is spent identifying constraints, this lower-level planning and scheduling process is typically only concerned with identifying a feasible plan [59, 60, 75]. In the past, a simply feasible plan was sufficient because the number of sensors and small components to install was relatively small; however, with the increase in sensors and small components stemming increased complexity and sensing desires, the potential impact to the schedule may now be much greater. As such, a feasible plan that has poor performance can now lead to increased risk in the overall, system-level production plan.

In light of this discussion, Figure 13 compares the generic current and proposed

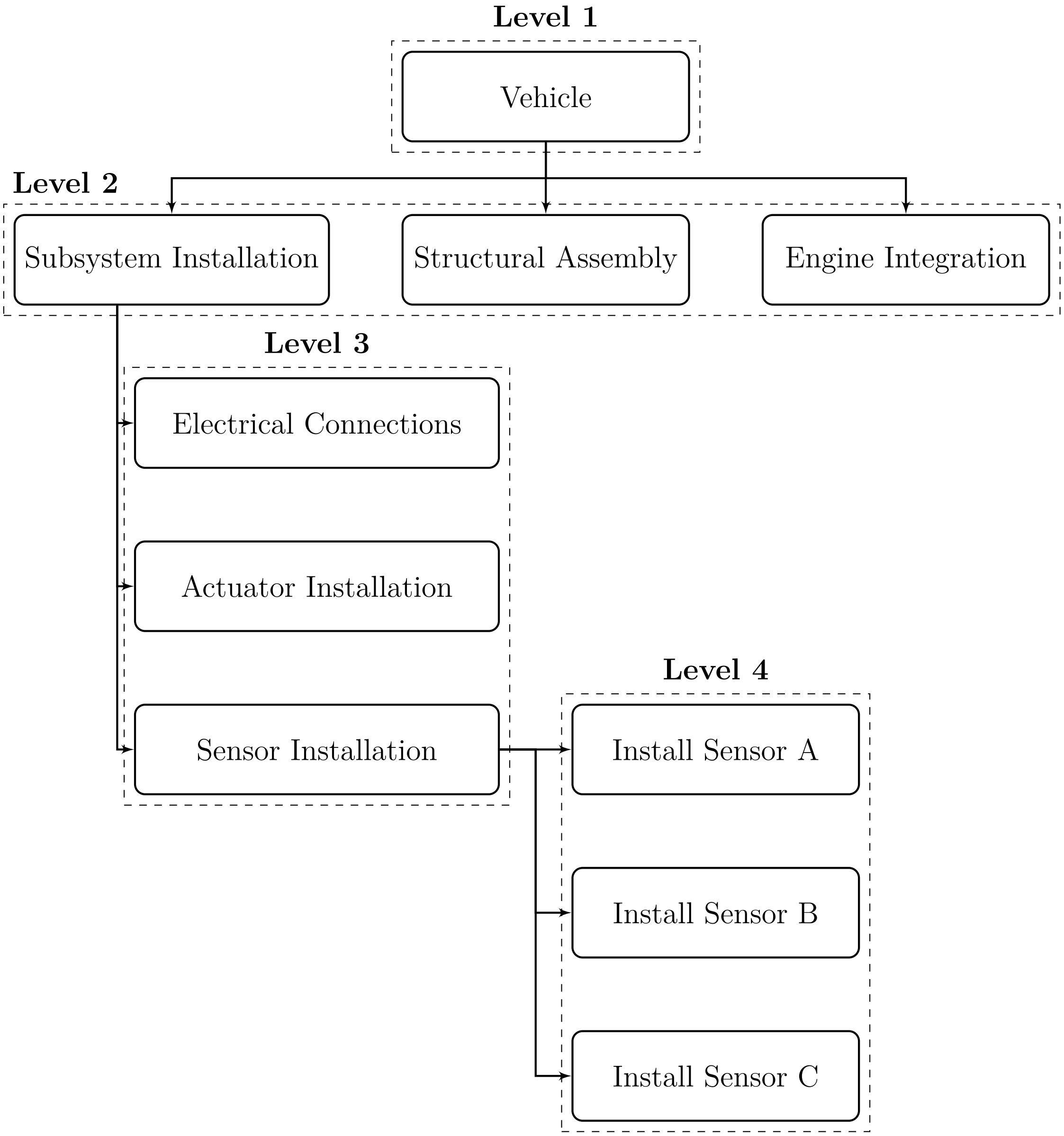
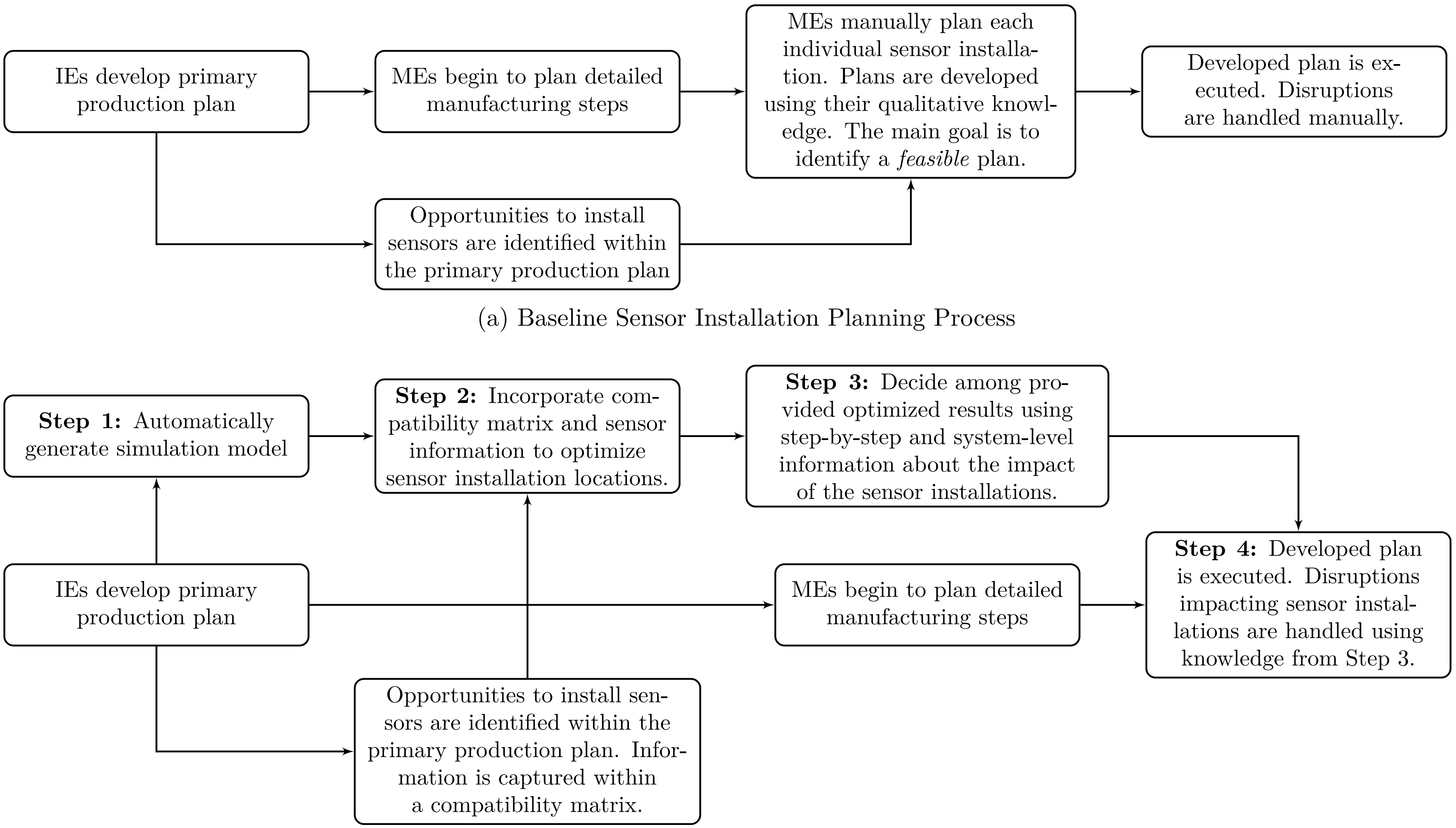


Figure 12: Notional Partial Work Breakdown Structure for the Case Study (Components of the WBS Inspired by Reference [154])

processes to plan the sensor installations. The currently implemented methodology (Figure 13a) begins with an mid-level primary production plan developed by the industrial engineering department. This plan contains an overall definition of the steps required to build the vehicle (at Levels 2 and 3 of the WBS in Figure 12). This mid-level process plan is then provided to the manufacturing engineering team to manually define specific, low-level steps and work orders to execute the plan. In parallel, the MEs also identify opportunities to install sensors and other sub-components within the schedule using their knowledge and experience with the system. The steps to install sensors (e.g. Level 4 and beyond in Figure 12) are typically too small to be directly considered in the tactical process plan; even when considered, the small subcomponents are commonly grouped into a large "Install all sensors" process step that is not specific to individual components. Hence, with the opportunities identified, the MEs manually identify primary processes during which to install sensors and subcomponents and produce specific work instructions to complete the installations.

Once a feasible plan has been defined and work orders issued, the plan is executed. If a disruption occurs during execution, the production manager or foreman decides on the best course of action to recover based on his or her best judgment or a set of heuristic rules [79].

In contrast to this traditional planning process, the methodology proposed in Section 3.1 is illustrated in Figure 13b. In the revised process, the baseline primary production plan is utilized by Step 1 of the PORRTSS methodology to generate a simulation model to quantify the impact of the sensor installations. The simulation model then incorporates the completed compatibility matrix and sensor information to optimize the installation plan within step 2. Step 3 then provides the optimization results back to the MEs and IEs within a visualization environment to select the plan to execute. Steps 2 and 3 serve to replace the manual identification of a feasible plan, which is time consuming and not a quantifiable process, from the baseline process.


IEs develop primary production plan
MEs begin to plan detailed manufacturing steps
MEs manually plan each individual sensor installation. Plans are developed using their qualitative knowledge. The main goal is to identify a *feasible* plan.
Developed plan is executed. Disruptions are handled manually.
Opportunities to install sensors are identified within the primary production plan

(a) Baseline Sensor Installation Planning Process

Step 1: Automatically generate simulation model
Step 2: Incorporate compatibility matrix and sensor information to optimize sensor installation locations.
Step 3: Decide among provided optimized results using step-by-step and system-level information about the impact of the sensor installations.
Step 4: Developed plan is executed. Disruptions impacting sensor installations are handled using knowledge from Step 3.
IEs develop primary production plan
MEs begin to plan detailed manufacturing steps
Opportunities to install sensors are identified within the primary production plan. Information is captured within a compatibility matrix.

(b) Proposed Sensor Installation Planning Process

Figure 13: Comparison of Current & Proposed Sensor Installation Planning Process

Now, by enabling the decision makers to select a plan based on its system level impact, the proposed process augments the experience of the MEs and IEs with quantifiable objectives. With a detailed plan outlined, it can then be propagated to the rest of the scheduling system for execution on the floor. Finally, because this process quantifies the impact of individual installations, this information can be used to help guide schedule recovery strategies after disruptions during execution.

There are two expected benefits from the implementation of the proposed planning process. First, by providing a quantifiable comparison between installation plans, the proposed process can identify optimal plans instead of plans that are simply feasible. Second, by formalizing the constraint identification with a compatibility matrix, the identification of feasible plans can be automated, which should greatly reduce the amount of time required to identify a plan. Both of these benefits are thoroughly examined through the experimental plan outlined in Chapter 4.

3.4 Chapter Summary

This chapter introduces a methodology that enables robust schedule modeling and optimization for complex manufacturing systems. The methodology leverages relatively recent advances in object-oriented simulation software to help ease simulation creation from pre-existing schedule models. The developed simulation models are able to more directly capture uncertainty within the production environment. Then, by replicating the simulation, the model can estimate the impact on the system's schedule quality robustness due to natural variation in process time, probability of damage to the sensors during installation, etc. This simulation is then linked to an optimization routine to search for well-performing and robust schedules. Because the problem is multi-objective, the optimization returns a set of Pareto optimal schedules that must then be further down-selected based on user-defined preferences. The selected schedule is then integrated with the overall schedule model for possible Monte

Carlo analysis to evaluate its impact on the larger, overall production schedule.

The case study that the PORRTSS methodology is applied to is also described. Planning the installation of sensors for a complex, low production rate aerospace vehicle's major subassembly is representative of the problems this methodology is designed to solve. Indeed:

- The production system contains complex logic and rules that are difficult to effectively model with traditional scheduling tools
- Aerospace vehicles generally have tight budgets and schedules with little cushion for disruptions
- Sensor installations can significantly delay the production process if not properly planned
- There is uncertainty in both the primary production tasks' and sensor installations' processing times
- The large scale (in terms of time, cost, and infrastructure) of aerospace vehicles and their production processes make the additional analysis required for this methodology worthwhile

The following chapter discusses in detail the implementation of the methodology for the aforementioned case study. Experiments aimed at testing the hypotheses developed in the previous chapter are then introduced.

CHAPTER IV

IMPLEMENTATION

This chapter discusses the implementation of the PORRTSS methodology in the context of the sensor integration case study. The development and testing of the methodology follows the engineering decision making process presented in the previous chapter. Chapter 1 *establishes the need* for a robust scheduling methodology to be applied to complex manufacturing systems. Chapter 2 *defines the problem* by exploring the shortcomings in common scheduling and simulation techniques. Chapter 2 also *establishes value* by developing motivating research questions and hypotheses to guide the research's experimentation. The previous chapter describes the PORRTSS methodology and the case study that is used to evaluate potential implementation alternatives. This chapter *identifies feasible alternative* options within the methodology. This chapter then concludes by discussing the experimental plan designed to *evaluate the alternatives*.

The following sections discuss the steps taken to implement the PORRTSS methodology to address the use case described in the previous chapter. In particular, links between the methodology steps and the research questions and hypotheses are made. This helps to inform the experimental plan, which is discussed in Section 4.6. An overview of the methodology's implementation specific to the case study can be seen in Figure 14. The following section describes the users and stakeholders that interact throughout the methodology's process. This discussion is included to frame the examination of the implementation steps.

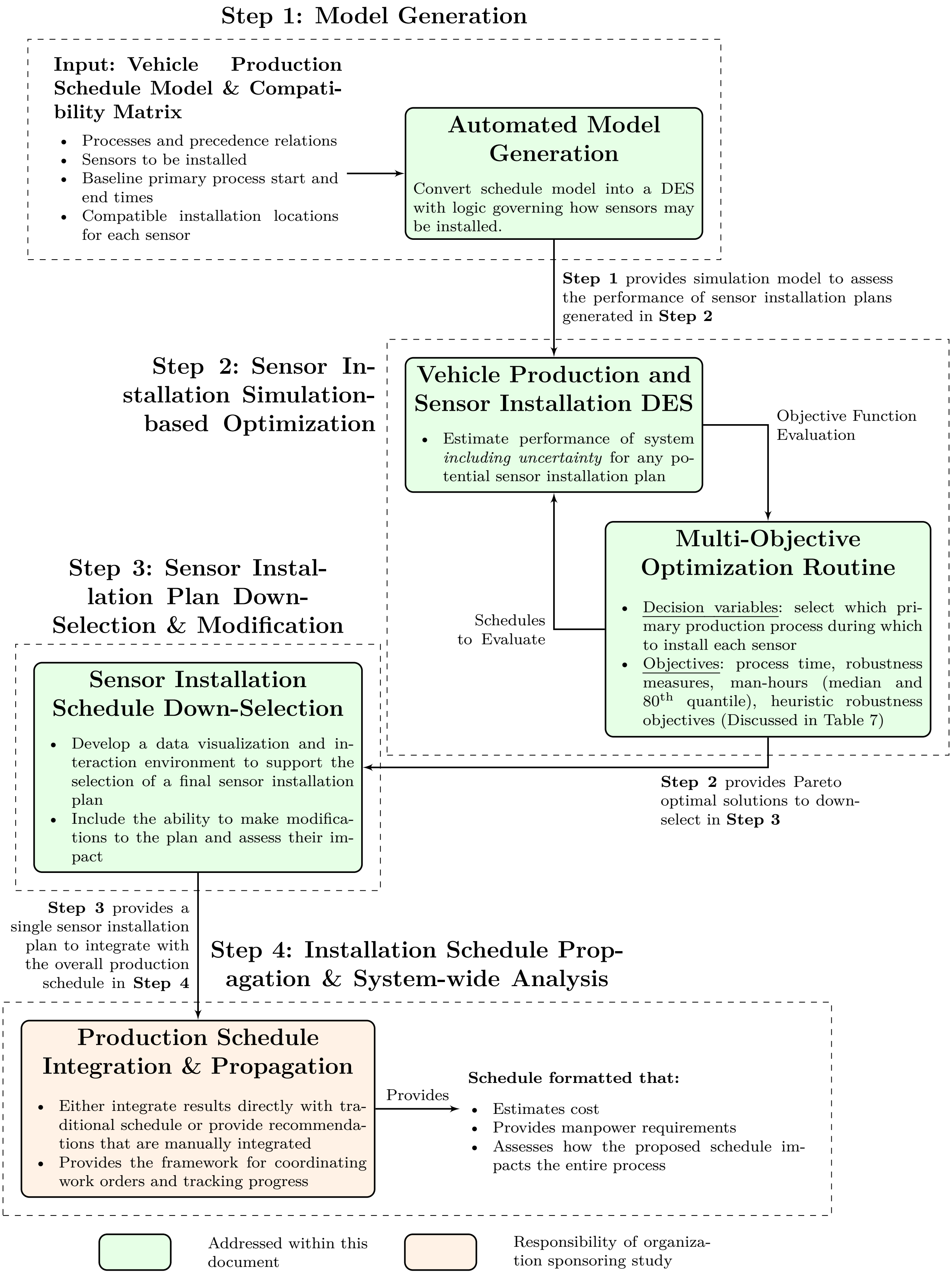

Step 1: Model Generation
Input: Vehicle Production Schedule Model & Compatibility Matrix
Processes and precedence relations
Sensors to be installed
Baseline primary process start and end times
Compatible installation locations for each sensor
Automated Model Generation
Convert schedule model into a DES with logic governing how sensors may be installed.
Step 1 provides simulation model to assess the performance of sensor installation plans generated in Step 2
Step 2: Sensor Installation Simulation-based Optimization
Vehicle Production and Sensor Installation DES
Estimate performance of system including uncertainty for any potential sensor installation plan
Objective Function Evaluation
Multi-Objective Optimization Routine
Decision variables: select which primary production process during which to install each sensor
Objectives: process time, robustness measures, man-hours (median and 80th quantile), heuristic robustness objectives (Discussed in Table 7)
Schedules to Evaluate
Step 2 provides Pareto optimal solutions to down-select in Step 3
Step 3: Sensor Installation Plan Down-Selection & Modification
Sensor Installation Schedule Down-Selection
Develop a data visualization and interaction environment to support the selection of a final sensor installation plan
Include the ability to make modifications to the plan and assess their impact
Step 3 provides a single sensor installation plan to integrate with the overall production schedule in Step 4
Step 4: Installation Schedule Propagation & System-wide Analysis
Production Schedule Integration & Propagation
Either integrate results directly with traditional schedule or provide recommendations that are manually integrated
Provides the framework for coordinating work orders and tracking progress
Provides
Schedule formatted that:
Estimates cost
Provides manpower requirements
Assesses how the proposed schedule impacts the entire process
Addressed within this document
Responsibility of organization sponsoring study

Figure 14: PORRTSS Methodology Applied to the Case Study

4.1 Implementation Stakeholders

Before examining the implementation of the methodology, a discussion of the users' and stakeholders' roles, responsibilities, and expectations is necessary. This discussion is based on the common division of responsibilities in a low-production environment [95, 157, 202, 215]. In the context of the case study, the four main users of the methodology are 1) Industrial Engineers (IEs), 2) Manufacturing Engineers (MEs), 3) Avionics Engineers, and 4) the person responsible for implementing the methodology. Each is discussed in the following paragraphs.

The industrial engineers are responsible for producing and maintaining the overall schedule at the system level [157]. As such, they define and optimize the primary process schedule based on precedence relations, resource constraints, and defined schedule objectives [59]; this is the schedule used for the input to Step 1 of the current methodology. A goal of the IEs is to quantify and reduce the risk in the production plan [157]. Hence, they expect the methodology to help quantify the impact of the sensor installations on the overall schedule, identify and justify the amount of resources that are required for the sensor installations, and conduct trades between schedule impact, risk, and resources (cost). To support the execution of the methodology, the IEs are responsible for providing the baseline primary process schedule, contributing to the final selection of the sensor installation plan to pursue, and utilizing the selected plan to schedule resources and update their risk assessments (constituting Step 4 of the methodology).

The next group to interact with the methodology are manufacturing engineers. The MEs are responsible for translating the schedule provided by the IEs into specific work orders and task instructions for the technicians on the factory floor [95]. They also work with the IEs to help define feasible process sequences; as such, MEs are familiar with the CAD model and manufacturing and assembly processes required to produce the system [95]. This familiarity means that the MEs are responsible for working with the avionics experts to identify feasible installation opportunities and complete the constraint matrix (as discussed in Section 4.3.2), which is the other key input for Step 1.

The MEs are the primary beneficiaries of the detailed sensor installation plan provided by this methodology. In the manual planning paradigm that the methodology is designed to improve upon, the MEs would be responsible for translating an overarching "sensor integration" block, which essentially says "install this block of 100 sensors at some point during this group of 200 production processes," provided by the IE department into a specific, operational level plan. This task is traditionally accomplished by manually exploring the CAD model and identifying portions of the flow where the MEs believe the installations could occur. It has been shown that human planners typically "spend 80-90% of their time determining the constraints that will affect the process [60]." This claim, further supported by Fox and Smith and Grant [59, 75], helps to illustrate that better automation in the constraint definition and alternative generation is needed. Furthermore, the time spent defining constraints and alternatives is likely exacerbated in this instance because of the system's complexity. Due to the difficulty in defining constraints, the MEs are primarily concerned with identifying a *feasible* plan with little time to identify a well-performing plan when manually scheduling sensor installations.

The methodology explored in this work provides many options for these detailed level plans. After the one-time compatibility matrix exercise, the estimated impact of each individual sensor installation, and a yet unseen estimate of the system level impact of a decision can be understood. This knowledge should help to improve the plan by better automating the alternative generation step and allowing the MEs to spend more time making decisions between production plans rather than identifying a single, feasible plan [59, 60, 75].

To ensure that customer specified performance requirements are met and that the manufacturing plan works with the product design, the planners (e.g. industrial and manufacturing engineers) are increasingly supported by personnel more familiar with the system or sub-system of interest [7]. For the case study, these personnel are avionics engineers. The avionics personnel have a significant amount of experience in designing and placing sensors, harnesses, and couplings on aerospace systems. This experience enables them to support the manufacturing engineers when determining the process constraints. Also, because these personnel interact directly with the customer on the avionics requirements, they are able to identify the most important installations (e.g. the most difficult or those sensors that are critical to mission success). With this knowledge, the avionics SMEs may choose to provide extra slack time or ensure easy access to the installation sites for these critical sensors to decrease the likelihood of damage.

The benefit that the avionics personnel anticipate from the application of this methodology is to hopefully make the avionics packages' integration, which is requested by the customer, feasible within the constrained manufacturing window. The manufacturing impact is commonly not considered when determining the subsystems to be installed on a vehicle [7, 86]. Hence, to help improve customer satisfaction, the avionics personnel are very invested in ensuring that all sensors and related equipment are given the best chance for installation without damage. This methodology supports this goal by identifying plans with less risk and larger windows to complete critical sensor installations.

The final user/stakeholder of the methodology is the methodology implementation expert. The implementor is responsible for integrating the schedule model provided by the Industrial engineers with the constraint matrix, which results from the manufacturing and avionics engineers' analysis. The schedule is converted to an appropriate simulation model by leveraging the automated model generation procedure within Step 1 (described further in Section 4.2). The model is then linked with the optimization routine through the process described in Step 2 (Section 4.3). Upon generating optimized cases, the implementor is responsible for providing the results in a suitable manner back to the IE, ME, and avionics engineers to support decision making. The resulting decision support environment is incorporated within Step 3 of the methodology and is discussed further in Section 4.4. Finally, once a final schedule is chosen for implementation, the IEs are responsible for integrating the plan into their overall schedule in Step 4 to estimate system-level risk.

With the roles and responsibilities of each stakeholder defined, the following sections discuss the methodology's implementation steps outlined above.

4.2 Step 1: Model Generation

The goal of this step is to produce a simulation model that properly captures the process constraints and can estimate the impact of parametrically defined sensor installations. The resulting simulation can then be used to drive the optimization in the following step to ultimately identify a schedule that is robust to the identified sources of variability. This should be accomplished with the minimal amount of data gathering effort and simulation complexity. The automated model generation strategy is implemented to reduce the effort required to gather information, develop, and build the model required for the methodology. Without this, the model building phase would quickly make the methodology infeasible as the problem size increases. Additionally, automating this process helps to standardize the constructed models to help ensure that the methodology can be applied to multiple problems as opposed to being a "one-off" analysis. A general overview of the schedule model generation strategy is provided in Figure 15.

The model generation strategy relies on the fact that data extracted from the schedule model corresponds to features (e.g. servers, links, entities, etc.) in a simulation model. As mentioned previously, simulations can be built to different levels

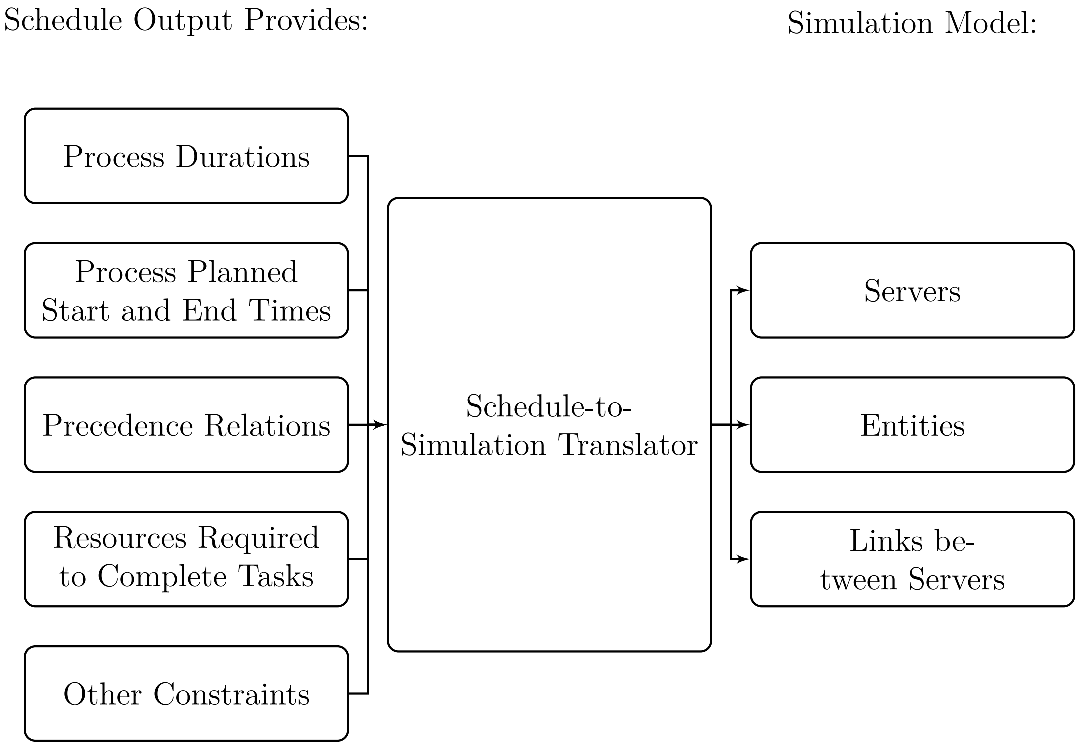

Schedule Output Provides:
Simulation Model:
Process Durations
Process Planned Start and End Times
Precedence Relations
Resources Required to Complete Tasks
Other Constraints
Schedule-to-Simulation Translator
Servers
Entities
Links between Servers

Figure 15: Schedule Model Generation Outline

of abstraction. Depending on the levels of abstraction and fidelity chosen different pieces of information from the schedule model are required. The end goal is to appropriately (i.e. to the level of detail required to make decisions in Step 2) define the major pieces of the discrete-event simulation, namely the servers, entities, and links between servers. Table 9 describes in detail how the schedule model information can be mapped to the various components of a simulation.

To accomplish this, the nature of the constraints accounted for within the schedule model must be understood. Reference [59] has identified 5 main categories of scheduling constraints that are commonly considered. Table 10 presents the identified constraints and examples of each from reference [59].

Each of these categories must be accounted for in the simulation. This commonly requires an extensive data gathering step during which the simulation builder must understand each physical, causal, and availability constraint [110]. Then, during

Table 9: Schedule Information Used to Generate Simulation Model

Schedule Model Information	Corresponding Simulation Information
	Servers
Process Durations	Time required to complete a process on the defined server
Resources Required to Complete Tasks	Resources (e.g. workers, material, etc.) that a server must seize to complete its process
	Entities
Process Durations	Entity specific process times (in the cases where multiple components are processed on a single server)
Process Constraints	Requirements to process an entity or move through the system
	Links between Servers
Precedence Relations	Develop links between processors to enforce precedence relationships
Planned Start and End Times	Create further links between processes to respect portions of the schedule that are already planned

model construction, the builder typically must directly account for each of these constraints (or assume them away) while providing the ability to account for preferences and evaluate the organizational goals. For the problem at hand, gathering information and then modeling the vast number of tooling availability and technician capacity constraints within the simulation would be extremely difficult. Furthermore, including this complexity within the simulation would negatively impact model execution time, which is a main concern for this study.

A promising source of information about the constraints identified in Table 10 that can help to simplify the simulation model is the schedule model itself. Scheduling, which is by definition “the allocation of tasks to resources over time in order to achieve optimality in one or more objective criteria in an efficient way [214],” necessarily identifies, defines, and incorporates relevant constraints into its optimization process. As simulation in manufacturing is mostly seen as a tool to evaluate “on-line” process changes (e.g. long term impact of adding a production line or changing production

Table 10: Scheduling Constraints (Reproduced from Reference [59])

Constraint	Example
Organizational goals	• Due date • Work-in-process • Shop stability • Shifts • Cost • Productivity goals • Quality
Physical constraints	• Machine physical constraints • Set-up times • Processing time • Quality
Causal restrictions	• Operation alternatives • Machine alternatives • Material requirements • Personnel requirements • Inter-operation transfer times
Availability constraints	• Resource reservations • Machine down time • Shifts
Preference constraints	• Operation preferences • Machine preferences • Sequencing preferences

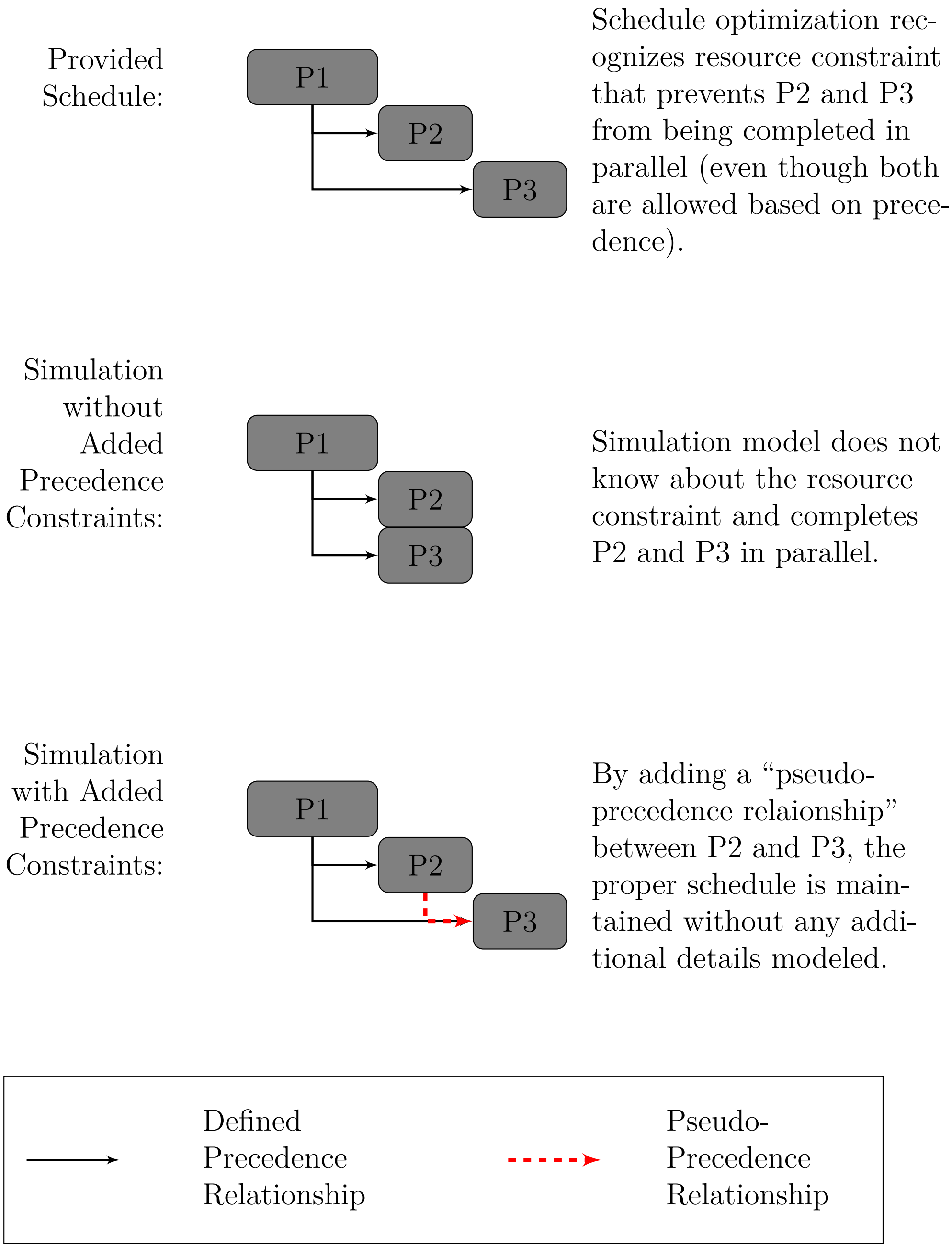
rules) [17] and the interaction between manufacturing software applications (e.g. simulation tools and schedule modeling and optimization software) are limited [88], most of the information used to design and generate the schedule is “re-discovered” when constructing the simulation model.

This methodology aims to leverage simulation during the initial planning phase, so this “re-discovery” and possible duplication of effort must be avoided. The traditional scheduling process attempts to optimize an objective by selecting, from all of the physically feasible schedules (e.g. those that observe precedence, material, capacity, etc. constraints), a single preferred process sequence [214]. Matching this preferred sequence in a simulation is traditionally difficult because even if the precedence and capacity constraints are modeled correctly, the simulation does not immediately produce the same results as the schedule. This is because, in cases where multiple jobs are available and competing for a resource, the simulation still does not know the order chosen by the schedule optimization. It is here that priority rules may be applied to attempt to capture the decisions made by the scheduling algorithm. This, however, again represents a duplication of effort as the schedule model has already defined an optimal process plan, so identifying a rule to match this result is not value-added.

Recall that the simulation is intended to use the provided optimal schedule to help identify the best sequence location during which to install sensors. Two aspects of this goal are critical to the implementation of the model generation strategy. First, the simulation is intended to accept the provided primary process schedule as truth and only make changes to the sensor installation plan. This means that the simulation does not require a means to modify the provided sequence of primary processes as would be possible if the model included detailed constraints and dispatching rules. Second, identifying the best sensor installation plan does not necessarily require the simulation to exactly model the sensor installation process; the optimization simply needs a comparison between potential plans.

With these requirements defined, a process to glean information from the schedule model to build a simulation capable of comparing sensor installation plans is implemented by incorporating strategies developed for process mining [130, 188, 189]. Process mining "aims at the automatic construction of models explaining the behavior in the event log [189]." These models, which are constructed primarily by observing the start and end times of processes, are easily transferred into discrete-event simulations. The discrete-event paradigm can very easily model precedence relationships generated from process mining studies using links to build a schedule graph. Simply including the precedence relationships explicitly defined in the schedule model is not sufficient to fully define the schedule because they do not account for resource constraints [59]; however, the start and end dates of the processes in the optimized schedule can be used to supplement the precedence relationships. Because the provided schedule is a planned schedule, the start and end dates may be mined to quickly determine the optimized primary process sequence. This is because the planned schedule starts jobs immediately after preceding ones without any delays that would actually be present in the executed plan. For example, in the planned schedule model, if process B follows process A, then the start time of process B is exactly equal to the end of process A. There are no discrepancies due to technicians not clocking into a job immediately after finishing a previous job or small changes that may be made on the floor. Hence, by adding these identified relationships to the already defined precedence relations, the baseline simulation (without sensor installations) can follow the provided schedule without the need for additional logic.

Figure 16 demonstrates the logic implemented through a notional Gantt chart. The top diagram illustrates the planned execution for processes P1, P2, and P3 from the schedule model. The arrows represent defined precedence relationships. Once P1 is completed, P2 and P3 can both be processed according to the precedence relationships. However, additional constraints exist that prevent parallel execution.



Provided Schedule:
P1
P2
P3
Simulation without Added Precedence Constraints:
P1
P2
P3
Simulation with Added Precedence Constraints:
P1
P2
P3
Defined Precedence Relationship
Pseudo-Precedence Relationship

Schedule optimization recognizes resource constraint that prevents P2 and P3 from being completed in parallel (even though both are allowed based on precedence).

Simulation model does not know about the resource constraint and completes P2 and P3 in parallel.

By adding a “pseudo-precedence relaionship” between P2 and P3, the proper schedule is maintained without any additional details modeled.

Figure 16: Demonstration of Basic Model Generation Logic Incorporating Choices Made by the Schedule Optimization Routine

Consequently, the optimization routine must determine in which order to execute the two available processes. In this case, the schedule model has determined that processing P2 before P3 is the best option. The middle diagram illustrates how a simulation would execute the schedule when only provided with the defined precedence information. In this instance, the simulation would ignore the additional constraints and process P2 and P3 in parallel. Even by directly modeling the additional constraints (e.g. personnel or work zone requirements), the simulation would need to apply a heuristic rule to determine is P2 or P3 should be completed first. This ignores the optimized schedule model and would likely lead to a primary process schedule that does not follow the provided schedule.

The bottom case in Figure 16 demonstrates how the implemented process addresses the problems encountered by the middle case. The bottom case, without directly modeling any additional constraints, is able to observe the optimized schedule through the addition of a pseudo precedence relationship between P2 and P3. This additional constraint is identified by the model generation algorithm by observing that P3's start time is equal to P2's ending time in the originally optimized schedule. Then, the simulation is built to include this pseudo constraint such that P1 *and* P2 must be completed before P3 can begin.

This method has the benefit of accounting for many of the availability, physical, and causal constraints identified in the schedule model without directly gathering information or modeling these constraints. Furthermore, these simplifications reduce the model's complexity, which helps to reduce the time spent verifying the model and, ultimately, improves runtime.

There are, however, drawbacks to this approach. Mainly, because the model is going to include uncertainty on the process times, it is conceivable that after a while, the simulation model could get far out of sequence and the pseudo precedences no longer fully capture the constraints in the model. For instance, if the processes occurring in parallel change significantly, the resource constraints that led to the additional pseudo constraints may no longer be applicable. While this is possible (or even likely), there are a number of "milestone" processes throughout the flow that require most if not all previous tasks to be completed before continuing. These would bring the model back to the original sequence and to improve the validity of this method.

Furthermore, the it is likely that this process leads to a more conservative estimate of the schedule completion time. This is because there are likely cases where pseudo precedence relationships are added to processes that simply happen to have an ending time that matches another starting time but are not actually related. Especially in the pre-production planning phase, the times assigned to tasks are usually only estimated down to the half hour, which can lead to many tasks ending or starting at the same time. This means that there can be instances where a process is waiting for a completely unrelated task to finish before starting.

Despite these limitations, it is proposed that this procedure is sufficient for the problem at hand because the resulting simulation can provide valid comparisons between installation plans. The nature of the optimization problem does not require an exact evaluation of the time to install the sensors, just a comparison between plans. Furthermore, these types of simplifications are required for this methodology to be feasible in terms of implementation time and runtime. The consequences of the choices made in the model generation strategy are examined in the experimentation plan.

The following section briefly describes the software implementation of the model generation strategy.

4.2.1 Software Implementation

The model generation strategy is implemented in C# to translate schedule model information into a format readable by a discrete-event simulation software package. For this research, the chosen simulation package is Simio [166]. Simio is chosen because it is a modern simulation language that supports object-oriented programming and provides access to its application program interface (API). Further justification for the selection of Simio is provided in the following section. Access to the API is necessary for integrating optimization routines and automating model building. The model generation code translates the schedule model information into a spreadsheet list of objects to be included within the Simio model following the logic described in the previous section. Information about the off-shift times in the schedule is considered such that a task that begins at the start of a shift can be identified as a predecessor to one that ends at the conclusion of the previous shift. Then, using Simio's API, the objects identified are placed into a shell model. The spreadsheet contains all of the properties for the objects (including process time, resources required, etc.) that are automatically placed within the model. The C# code finally defines the links between servers following the logic discussed in the previous section.

In all but the simplest cases, this model generation strategy is not immediately effective with a completely new simulation model and standard object library. The methodology is designed to be implemented in systems with potentially complex rules and interactions, so the logic available in the base classes is likely not capable of modeling the system. As such, before integrating with the schedule model, custom objects and logic must be defined within a "shell" simulation model. This is made easier by the object-oriented programming paradigm found within Simio. The logic is formulated within a smaller, testing model and generalized such that simple property definitions from the generated spreadsheets can fully define the model. Then, once the logic is verified, the model can be built automatically using a developed add-in to place model objects based on the spreadsheet created from the schedule model.

With the simulation developed, the simulation-based optimization routine contained within Step 2 of the methodology can search for an improved installation schedule.

4.3 Step 2: Simulation-based Optimization

The simulation-based optimization utilizes the developed simulation to identify a non-dominated set of robust sensor installation plans the reduce the risk of system-level schedule and cost overruns. This section first describes the specific requirements of the modeling environment to further justify the selection of Simio. It then discusses the process logic that was developed and implemented to: 1) Enable the parametric planning of manual tasks within the primary production flow. 2) Assess their impact on a variety of metrics of interest. With the modeling implementation described, the constraint definition strategy and integration of the optimization routine is described. Finally, the specific optimization algorithms that are explored through this research are presented.

The requirements for the production model(s) are driven by the nature of the problem to be addressed and the capabilities desired. The production planning problem to be modeled has the following characteristics:

Discrete: Production occurs in specific, discrete steps

Stochastic: Uncertainty exists in time to complete a specific task as well as passing quality assurance (e.g. amount of rework necessary, etc.)

Complex: Due to the accessibility constraints and potential for multiple, concurrent processes, manual installations cannot be identified as feasible at the start of a simulation run. Rather, compatibility between sensor installations and the primary production plan must be dynamically checked throughout the simulation run. This serves to determine if manual installations or primary processes are allowed to proceed based on the current state of the production system.

Therefore, in addition to supporting automated model generation (as discussed in Section 4.2), the modeling package must also be capable of dynamically applying production rules throughout the simulation to account for process compatibility constraints. Furthermore, to better accommodate the large number of cases required for the optimization, the chosen package should support distributed computing across a network. These additional requirements further justify the selection of Simio. The intelligent object paradigm contained within Simio allows for easy coding of the dynamic checks required. Furthermore, the software package can be setup to automatically distribute runs across a network without any specialized coding requirements.

Using Simio, a baseline model of the primary production processes is first built based on the provided production schedule models and value stream maps. The following section discusses how the base production logic contained within Simio is modified to support the parametric definition of sensor installations.

4.3.1 Production Logic

The primary planning decision of interest throughout this work is: during which primary processes should which sensors be installed to minimize the impact to the baseline production flow? To support this decision, logic is added to the model to enable the user to parametrically define scenarios to complete the manual installation tasks of interest.

This is accomplished by assigning manual installation tasks to primary production processes before the model run (e.g. complete the installation of sensor 24 during primary process 78). Then, when the assigned process is about to begin, logic is initiated to begin installation processes for the defined manual sensor installation task(s). The logic used to control the manual sensor installation processes obeys the following rules and assumptions:

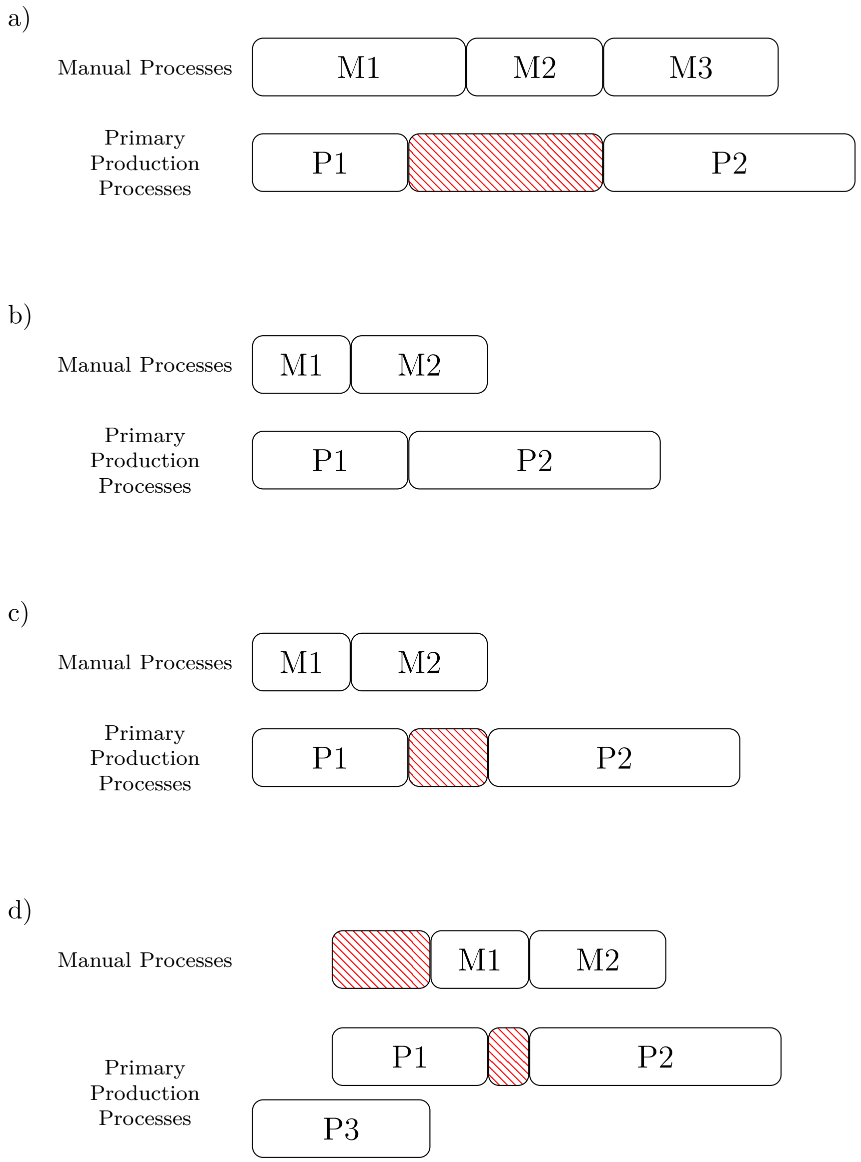
A1. All manual installation sequences scheduled to occur during a primary process must be started before the primary production flow can proceed. This is illustrated in Figure 17a. Once all installation sequences have *started*, the successive primary process is allowed to begin *if* it is compatible with the ongoing manual installations (Figure 17b). If it is *not* compatible, then the subsequent primary process must wait for the sensor installations to finish before beginning (Figure 17c).

A2. If an incompatible production process has started before a manual installation, the manual task must wait until the incompatible process has completed to begin processing (Figure 17d).

A3. Each manual installation requires a technician to be completed. Therefore, the processes must wait for a technician to be available and seized before beginning. The technicians respond to work requests based on a first come, first served basis. Hence, setting the number of technicians in the simulation controls the number of sensor installations tasks that can be completed in parallel.

A4. Manual installation processes can contain sub-processes that do not require a technician (e.g. time to allow an adhesive to cure before proceeding with the installation). In these cases, technicians are released to complete other touch labor tasks. Once the sub-process is completed, the next sub-process requiring a technician is added to the queue to request a technician.

The baseline model is generated incorporating this logic. The following section discusses the general process taken to generate the compatibility matrix based on subject matter expert opinions.


a)
Manual Processes
M1
M2
M3
Primary Production Processes
P1
P2
b)
Manual Processes
M1
M2
Primary Production Processes
P1
P2
c)
Manual Processes
M1
M2
Primary Production Processes
P1
P2
d)
Manual Processes
M1
M2
Primary Production Processes
P1
P2
P3

Primary process P2 must wait until M3, which is planned during P1, to begin before starting, even though P2 is compatible with M1, M2, and M3.

P2 is allowed to start immediately because it is *compatible* with M2 and all of P1's defined installations have begun.

P2 is not allowed to start immediately because it is *incompatible* with M2.

M1 and M2 (planned to occur in parallel with P1) are incompatible with P3, so they must wait for P3 to complete before starting. P2 must then again wait for M2 to finish before starting.

Figure 17: Examples of Manual Installation Logic

4.3.2 Compatibility Matrix Generation

As discussed in Section 3.2.4, the constraints for this methodology are captured within a compatibility matrix that maps each sensor to primary processes during which it can feasibly be installed. The constraints for this problem are identified during constraint definition workshops conducted with avionics experts, manufacturing engineers, and industrial engineers working on the vehicle of interest. To better solicit expert information for the generation of the compatibility matrix, a compatibility matrix tool is developed. The tool enables users to apply rules to each sensor and/or primary production process instead of directly manipulating the matrix, which would be overwhelming. The rules available and example applications of the rules are:

No sensor can be installed during a specific process: Due to safety concerns, no sensor installations can occur during certain overhead crane operations or electrical tests.

Sensor X must be installed *before* primary process A: The vehicles of interest are extremely cramped with components, subsystems, structure, etc. Accessibility issues quickly arise as general integration begins, so this constraint ensures that sensor installations are planned before another component blocks access to the installation site.

Sensor X must be installed *after* primary process A: Some sensors are installed onto other subsystems. This constraint ensures the sensors are not planned for installation before the installation site itself has been integrated.

Sensor X cannot be installed during primary process A: This constraint enables the expert to capture any accessibility, safety, or quality problems for individual sensors and processes. For example, if a process calls for a drilling operation above a sensor installation location, the sensor cannot be installed during that time to avoid the technicians interfering with each other and potential damage to the sensor from drilling debris.

During the workshops, the subject matter experts assign the above rules to each of the sensors that are to be installed within the current production flow. The resulting compatibility matrix is used throughout to drive the model operation and optimization.

While the present implementation requires SME input to define a compatibility matrix, opportunities to automate its development could be explored in the future. As the matrix is primarily concerned with interference between components, simulating the actual installation processes to identify interferences could be pursued. This would greatly simplify the implementation of the methodology and better encourage implementation. This is further discussed in the future work (Section 7.5).
The next section discusses the general implementation of the optimization routine.

4.3.3 Optimization Routine Requirements

The choice of the optimization strategy is dictated by the nature of the problem to be solved as well as the hypotheses formulated in the previous chapter. Hence, the optimization routine must:

Handle discrete variables: Scheduling problems are inherently discrete in nature. The purpose of scheduling is to properly plan discrete processes to minimize cost, time, risk, etc. Therefore, the optimization algorithm chosen to improve a schedule must be capable of solving problems with discrete decision variables.

Be multi-objective: As discussed throughout this thesis, scheduling problems are multi-objective in nature. Even when single-objective algorithms are utilized, the objective function is commonly a weighted cost function of multiple objectives. Additionally, the inclusion of robustness measures into the optimization increases the number of dimensions to analyze. Hence, to reduce the result's reliance on preset weightings and have the capability to add additional robustness measures to the optimization strategy, a multi-objective strategy is desired.

Limit the number of simulation runs required to find a solution: The simulations required to estimate the performance of the manufacturing systems of interest to this thesis are relatively complex. In addition, the number of simulation replications required to estimate the quality robustness of the system may be significant. These points lead to potentially long run-times for single optimization cases or generations. Hence, effort must be made to limit the number of cases required by implementing strategic optimization improvements that take advantage of specifics about the problem at hand. These improvements can involve limiting the replications required by intelligently controlling portions of the problem's uncertainty, using problem specific information to better guide the search, and improving the intensification process by supplementing the metaheuristic search with a local search algorithm. Potential optimization improvements are discussed further throughout the experimental plan.

Effectively explore the complex decision space: The scheduling problem for the case study has a complex design space with non-linear and competing objective functions. As such, effectively exploring the design space while avoiding becoming trapped in local optima is a must for the algorithm.

Incorporate robust design principles to reduce the impact of schedule disruptions: The algorithm must be able to incorporate and optimize for robustness measures (e.g. schedule quality robustness) because the case study represents high risk, high impact processes. This thesis aims to demonstrate that the incorporation of robust principles directly with the optimization routine helps to reduce the occurrence and impact of schedule delays.

Provide a general framework that can be applied to similar problems: The methodology is designed to be a general framework for working with complex manufacturing processes; therefore, the optimization should avoid (as much as possible) problem specific customizations. In this way, the methodology may be expanded to other problems with limited customization.

With these required characteristics defined, the following section details the implementation of the optimization algorithms investigated as a part of this work.

4.3.4 Simulation-based Optimization Implementation

This section describes the implementations of the various optimization routines investigated throughout this dissertation within MATLAB [122]. Each algorithm is coded using a base MATLAB code to evaluate cases within Simio. The algorithms are chosen in an attempt to explore trades between search result quality, search time required, and the uniformity of the search over the Pareto frontier of solutions. These are common metrics to evaluate multi-objective optimization algorithms [27,139,217], and are discussed further in Section 4.6.

The purpose of implementing multiple algorithms is to help identify a suite of optimization strategies capable of solving problems with a range of complexities, solution quality requirements, solution space exploration needs, and time horizons. There is always a trade-off between these requirements, so providing the decision maker with the ability to select a promising algorithm for the current problem can improve the "implementability" of the methodology. To explore each of these, a population-based metaheuristic (NSGA-II) is compared to single solution-based heuristic neighborhood search methods (Shifting Bottleneck-Inspired Local Search and Expanded Neighborhood Local Search) and single solution-based metaheuristic algorithms (Weighted Sum Simulated Annealing and Pareto Dominance-based Simulated Annealing). As is discussed during the experimental plan, each class of algorithm is expected to provide different strengths that can be mapped to the requirements of the problem. Further discussion about the algorithms chosen can be found in Section 4.6.6.

The following sections discuss each algorithm implemented for comparison during experimentation.

4.3.4.1 Non-dominated Sorting Genetic Algorithm-II Implementation

The NSGA-II algorithm is implemented within MATLAB and connected to the Simio discrete-event simulation through an API executable. An overview of the optimization strategy can be found in Figure 18. The algorithm first uses the compatibility matrix to determine how many bits are required to define the range of feasible installation locations within the primary process flow. Therefore, each sensor installation location is encoded by the compatible process it is installed during. This inherently removes infeasible cases from consideration, which helps to simplify the algorithm.

To start the optimization, the MATLAB code populates an initial generation with random, feasible sensor installation schedules. The optimizer is also capable of loading a pre-determined generation that was developed through some heuristic method to help seed the optimization with well-performing points. The entire generation is then evaluated by Simio via the API executable. The model only needs to be loaded once per generation and can distribute all of the replications required for a single generation. Therefore, with unlimited parallelization, the evaluation time is only subject to the time required to evaluate a single replication.

Once the population is evaluated, it is returned to MATLAB to determine the next set of population points to try. Common implementations of genetic algorithms usually evaluate the entire population at each generation, even if some members were evaluated in previous generations. When the objective function evaluation is almost instantaneous, this is acceptable; however because evaluating the simulation model could take minutes, this is highly inefficient for the problem at hand. Therefore, a cache system is implemented to save previous function evaluations and avoid wasting function calls.

The optimization is run until a specified stopping condition is reached. The stopping condition used varies depending on the time available to reach a good solution population and the purpose of running the optimization. If the goal is to compare algorithms, the optimization is stopped after a certain number of generations. If the optimization is being run to identify a high-performing set of solutions without time constraints, more detailed multi-objective optimization convergence criteria can be implemented. The two criteria implemented measure the relative improvement of each generation compared to the initial population and the number of Pareto points in a generation that survive to future generations. These criteria are discussed further in Section 4.6.6.2. Since the NSGA-II is used as the baseline optimization algorithm, the multi-objective convergence criteria are used to check the stopping condition.

As described previously, the main decision variables are defined to help select, for each sensor, the primary production process during which it is to be installed. An important response to take into account is the quality robustness of the solution. Multiple options for quantifying this metric are possible and are discussed further in Section 4.6. Finally, because the optimization is multi-objective, it produces a family of Pareto optimal solutions. These solutions are then down-selected by applying user preferences to weight the relative importance of the objective functions in Step 3 of the methodology before being propagated back to the overall production schedule in Step 4.

The NSGA-II is a very popular algorithm because it is extremely flexible; however, because it is a population-based strategy, it could require a significant amount of time and function evaluations to reach a "good" solution. As such, additional algorithm that operate on single point solutions are investigated in the following sections. These

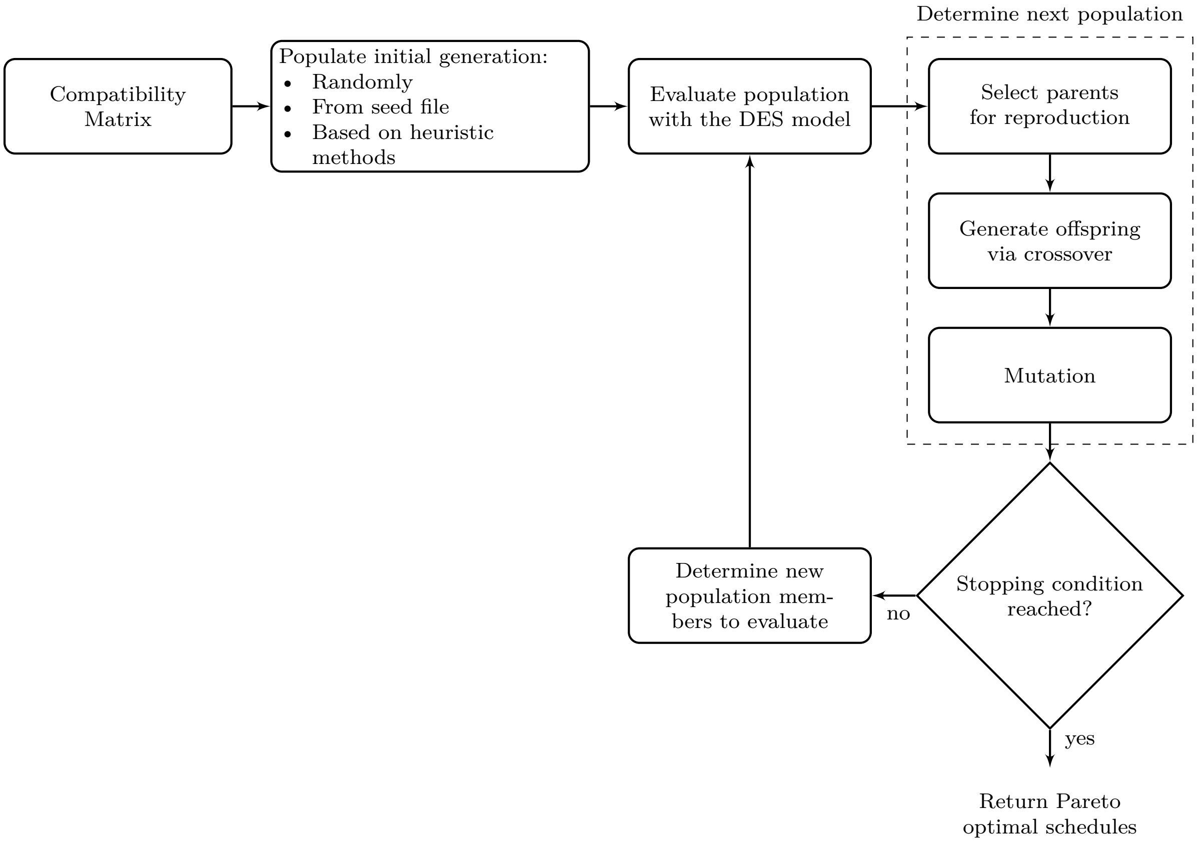

Determine next population
Compatibility Matrix
Populate initial generation:
• Randomly
• From seed file
• Based on heuristic methods
Evaluate population with the DES model
Select parents for reproduction
Generate offspring via crossover
Mutation
Stopping condition reached?
no
Determine new population members to evaluate
yes
Return Pareto optimal schedules

Figure 18: Optimization Strategy

point solution-based algorithms, while sacrificing the exploration of the design space, are likely able to converge to promising solutions more quickly than the NSGA-II. Furthermore, depending on the complexity of the design space, different neighborhood definitions and overall search strategies may be more or less effective. Section 4.6.6 further describes the purpose of implementing the following algorithms.

All of the following algorithms follow the general compatibility matrix → MATLAB → Simio formulation discussed in this section. The following section describes the implementation of heuristic shifting bottleneck-inspired local searches.

4.3.4.2 Shifting Bottleneck-Inspired Local Search Strategies

The commonly implemented shifting bottleneck strategy identifies the most constraining portion of the schedule (e.g. the bottleneck machine). With the bottleneck identified, a smaller problem is solved to alleviate the bottleneck (e.g. by planning processes from the bottleneck onto another machine). When the optimal solution to this subproblem is identified, a new bottleneck is found, and the algorithm iterates.

The classic implementation of the shifting bottleneck procedure only considers make-span as it objective function [136]. The sensor installation problem of interest, however, is multi-objective and requires a modification to the classic implementation. For the three primary objective function categories of interest (process time, slack time, and neighboring installation preference) described in Table 7, each sensor installation's contribution to each objective can be identified. In other words, for each sensor, the delay to the primary production process, its individual slack time, and its closeness (within the schedule) to neighboring installations can be calculated. This information can then be used to identify the "bottleneck" sensor (i.e. the sensor leading to the most degradation in schedule performance).

Identification of the "bottleneck" sensor is accomplished by first normalizing each of the three individual sensor installation performance metrics such that the worst performing installation per metric equals one. With the normalized values, a composite objective can be created by summing the three normalized values. Other options for combining the objectives are available, however the summation method has been shown to provide sufficient results in similar situations [40]. Once the composite values for each installation are determined, one of the sensor installations is selected for modification based on either a greedy strategy or weighted random draw. This sensor installation is then moved to a different compatible location in the schedule. At each iteration, multiple options for the new location can be evaluated, and then the best performing schedule is selected as the next point. Not all possibilities must be evaluated for each iteration because this could be extremely time consuming; however, if the same sensor is chosen for modification during the following iteration, previously explored options are excluded from consideration.

Two variations of this algorithm are investigated. The *greedy* alternative always selects the "worst" performing sensor installation (according to the composite objective function) to modify. This is akin to following the steepest descent direction in a typical optimization formulation. If moving the selected sensor to all other potential installation locations does not produce an improved result (based on a weighted objective value), then the algorithm has become stuck in a local minima and exits.

The *stochastic* alternative selects the sensor to modify based on a probability (instead of always selecting the worst as in the greedy alternative). Each installation is weighted based on the composite individual sensor installation performance metric, and then the installation to modify is selected from this weighted probability distribution. The algorithm then proceeds in the same manner as the greedy strategy. In general, greedy algorithms typically struggle to explore the space at all, so by adding a measure of stochasticity to the problem, this algorithm has a better chance to find an improved solution. However, if the initial schedule is well-performing (e.g. the algorithm only needs to find the local optima), then the stochastic algorithm may take longer to improve the solution than the greedy algorithm. An overview of the procedures for the shifting bottleneck inspired algorithms can be seen in Algorithm 1 in Appendix A. Appendix A additionally contains algorithms for each of the following optimization strategies discussed. The next section discusses the implementation of a stochastic expanded neighborhood search algorithm.

4.3.4.3 Stochastic Expanded Neighborhood Search

The stochastic expanded neighborhood search optimization strategy takes general ideas from the shifting bottleneck heuristic (e.g. identify the constraining steps and attempt to modify the schedule to improve the performance of those steps) but incorporates a more expanded search heuristic. In the bottleneck-inspired methods described previously, the search in each iteration of the algorithm is limited to moving a single sensor installation throughout the schedule. This expanded neighborhood search, however, is allowed to move multiple sensors during each iteration. As such, this algorithm can potentially explore the space more quickly.

As with the previous algorithms, the stochastic expanded neighborhood search algorithm is initialized with a random plan for sensor installations. The plan is evaluated and the performance of each individual sensor is calculated. Then, a specific number of sensors to move during this iteration are selected from a single sensor up to 5% of the total number of sensors to plan. Similar to the stochastic shifting bottleneck method, sensor installations to be modified are selected from a weighted probability distribution based on their individual performance. Each installation to be modified is then assigned a new, random, feasible installation location. The new schedule is then evaluated based on a weighted sum of the objectives such that each contributes approximately equally to the total objective function value. If the weighted sum objective is improved, the new schedule is accepted. Otherwise, the original schedule is kept, and the next iteration begins.

While somewhat similar to the shifting bottleneck heuristic methods described in the previous section, this method is less susceptible to becoming stuck in a local minima. This is because the "neighborhood" definition for this method is broader than the first algorithms described; hence, the possibility exists to jump out of a local minima. The general procedure for this algorithm is outlined in Algorithm 2.

While the stochastic neighborhood search algorithm presented in this section should provide enhanced exploration compared to the shifting bottleneck-inspired algorithms, it is still a heuristic method that could become trapped in a local minima. Furthermore, the algorithm may have difficulties making the small changes to converge at the end of an optimization run. However, the algorithm can also be used as an underlying heuristic for metaheuristic algorithms, which could help to avoid these challenges. Therefore, the following section discusses the implementation of a standard simulated annealing algorithm that can guide the neighborhood searches discussed in this section to enhance exploration and avoid becoming trapped in local minima [214].

4.3.4.4 Weighted Sum Simulated Annealing

The simulated annealing (SA) algorithm begins by evaluating an initial, randomly generated schedule. This schedule is then modified using either the stochastic shifting-bottleneck algorithm (Algorithm 1) or the stochastic neighborhood search algorithm (Algorithm 2). Then, following the common implementation of a simulated annealing algorithm, the new point is accepted if it is better than the current point. If it is worse, the point is accepted with the probability defined by Equation 2 (reproduced below) to attempt to escape local minima. After each iteration, the “temperature” (t_k) is reduced according to an annealing schedule. Reducing the temperature reduces the probability that a worse move is accepted, so as the algorithm progresses, it moves from an exploration mode to an exploitation mode. A summary of the algorithm can be found in Algorithm 3.

$$p\left(\Delta f\right) = \exp\left(\frac{-\Delta f}{T_k}\right) \tag{2}$$

As with the shifting bottleneck-inspired local search and expanded neighborhood search algorithms, this weighted sum simulated annealing algorithm uses a composite, weighted objective function. The three main categories of objective functions (process time, slack, and the neighboring installation metric) are weighted such that they contribute approximately equally to the overall objective evaluation [129].

The initial temperature is selected such that there is a high probability (about 80%) of accepting a worse move [22]. The procedure to select this initial temperature, as proposed by Ben-Ameur, first evaluates a set of transitions around the neighborhood of the initial point. The goal of this sampling procedure is to understand the range of the objective function values within the neighborhood, and the size and sampling strategy used is based on experience with the problem [22]. With a sample (S) of the positive transitions (i.e. those with increases in the cost function) around the neighborhood of the initial point evaluated, an initial guess of the temperature can be made. An estimation of the probability of the transition at $\hat{\chi}\left(T_n\right)$ can then be computed by the following equation with $n = 1$:

$$\hat{\chi}\left(T_n\right) = \frac{\sum_{t \in S} \exp -\frac{E_{\max_t}}{T_n}}{\sum_{t \in S} \exp -\frac{E_{\min_t}}{T_n}} \tag{5}$$

where $E_{\max_t}$ is the upper cost function value for each transition t in S. $E_{\min_t}$ is each corresponding lower cost function value. With the guess $\hat{\chi}\left(T_n\right)$ calculated, the temperature guess is updated by the following equation:

$$T_{n+1} = T_n \left[\frac{\ln \hat{\chi}\left(T_n\right)}{\ln \chi_0}\right]^{\frac{1}{p}} \tag{6}$$

where χ_0 is the desired initial acceptance probability and p is a real number ≥ 1. This process is iterated until the estimated transition probability is close to the desired probability. With the initial temperature defined, the simulated annealing algorithm can proceed. While many complicated annealing schedules are available, this thesis is focused on showing the utility of including additional algorithms, not identifying the best algorithm and settings for this particular optimization problem. As such, the simple annealing schedule defined in Equation 7 is utilized [30, 103, 208].

$$T(t) = \alpha T(t-1), \qquad \alpha < 1 \tag{7}$$

The main benefit expected from the SA algorithm is that it is better able to explore the design space and avoid becoming trapped in local minima better than the purely heuristic methods. Hence, the algorithm's performance should be less sensitive to the initial schedule than the stochastic local search. This could, however, come at the cost of increased computational time because SA algorithms can be inefficient at the start of the search by accepting too poor of points. Hence, a modified version of the simulated annealing algorithm that modifies the annealing schedule to improve early efficiency is discussed in the following section.

4.3.4.5 Weighted Fast Simulated Annealing

A cited drawback of a traditional simulated annealing algorithm is the excessive amount of computational effort required to converge to a solution [40]. One source of inefficiency in the classic implementation is that too many uphill moves are accepted in the early stages of the algorithm's run. Since the starting point for the algorithm is commonly randomly assigned, accepting worse moves early does not necessarily help the algorithm identify and move to a region of well-performing points [40].

In an effort to alleviate these issues, a modified annealing schedule is implemented as described in reference [40]. The modified schedule is composed of 3 phases:

1) **High temperature random search:** The initial temperature is set high such that the probability of accepting worse moves is close to 1. This early random search period works to ensure the algorithm is not initially trapped in a local minima. This period is relatively short lived, and the algorithm quickly moves to the next portion of the annealing schedule.

2) **Pseudo-greedy Local Search:** This is the period that attempts to reduce the function calls required to find the region of good solutions. In this period, the

initial temperature is quickly reduced and approaches zero. In this way, very few inferior solutions are chosen such that the algorithm quickly moves into a promising region.

3) **Hill-climbing Search:** After the Pseudo-greedy Local Search has identified a promising region of the search space, the temperature is raised again to facilitate exploration of the promising neighborhood. The temperature is then gradually reduced as in the classic simulated annealing algorithm to hopefully converge to a promising solution.

The algorithm itself is implemented similarly to the simulated annealing algorithm described in Section 4.3.4.4. The primary difference is that the annealing schedule follows Equation 8 [40].

$$T_n = \begin{cases} \frac{\Delta_{avg}}{\ln P}, & n = 1 \\ \frac{T_1 \Delta_{cost}}{nc}, & 2 \leq n \leq k \\ \frac{T_1 \Delta_{cost}}{nc}, & n > k \end{cases} \tag{8}$$

In this equation, n is the iteration number, Δ_{avg} is the average initial increase in objective function values, P is the initial desired probability to accept a worse point, Δ_{cost} is the cost change for the current temperature, and c and k are parameters. Using this annealing schedule, the implemented algorithm is described in Algorithm 4. One modification from the implementation by Chen [40] is that the initial temperature is identified following the improved process from Ben-Ameur [22] as described in Section 4.3.4.4.

Of the 4 objective functions of interest, the process time can be considered the primary function; however, there are many similar sensor installation plans that lead to a schedule with low process time. Therefore, the greedy phase of this algorithm, when combined with a weighting scheme that favors a reduction in process time, should be able to identify a well-performing region quickly. Then, when the algorithm enters the final phase, it is hoped that it can find a schedule with a similarly low process time but also works to explore and increase the slack time and the closeness of neighboring sensors within the schedule.

The two implementations of a simulated annealing algorithm discussed above represent a classic approach to dealing with multi-objective searches (e.g. using a weighted sum to condense the problem to a single objective) [139]. While this approach is useful and, if run multiple times, should help to quickly identify portions of the Pareto frontier [139], the results identified are subject to proper weightings, which is less then desirable. The following section describes an approach that does not require user input weights and instead uses the Pareto dominance principle.

4.3.4.6 Pareto Dominance-based Multi-objective Simulated Annealing

The desire to explore the Pareto frontier without biasing the search via user-defined weightings or requiring the long runtimes common to population-based methods motivates this multi-objective Simulated Annealing strategy. Instead of relying on weighting schemes and multiple runs of the algorithm to find Pareto optimal solutions, this algorithm is designed to explore the Pareto frontier in a single run. This is accomplished by incorporating the Pareto dominance principle into the objective function evaluation criteria. The algorithm described throughout this section is based on reference [139].

The Pareto Donimance-based, Multi-objective Simulated Annealing algorithm uses the same basic framework as Algorithm 3. The neighborhood definition and annealing schedule remain the same; however, the determination about whether or not to move to a new point is now based on Pareto dominance. Therefore, instead of using a weighted sum to determine a fitness function, the Pareto rank of the original and candidate solution are used to determine whether or not to move. If the trial point's rank is better, then the algorithm moves to the new point. If they are incomparable (i.e. the original and trial points' ranks are the same), then the algorithm still shifts to the new point to encourage exploration. Reference [139] observed that this strategy helped to find regions in the middle of the Pareto frontier. Finally, if the trial point is Pareto dominated by the original point, then the new point is accepted with a probability defined by Equation 2.

Reference [139] examined multiple ways to evaluate Δf for use in Equation 2. Assuming f is a vector of objective functions, they suggest evaluating Δf as the minimum, maximum, random weighting, sum, or average of the difference between $f_{old}(i)$ and $f_{new}(i)$, where $f(i)$ is the value of the i^{th} objective. Additionally, a fixed Δf value was used so that the probability to move was based solely on the temperature. The article reports that the random, average, and fixed strategies performed well across their test-bed problems; hence, the random strategy is chosen for implementation to potentially prevent one objective with a significant amount of potential improvement (e.g. the slack time objectives) from heavily biasing the search. An overview of the algorithm can be found in Algorithm 5.

This section has reviewed the various optimization algorithms and strategies that are investigated throughout this thesis. These investigations are intended to provide a suite of algorithms that can be selected based on the complexity of the problem, time available to identify a solution, and knowledge about the importance of the identified objectives. The following section briefly reviews the process taken to identify the number of replications required to estimate the objective functions when allowing for uncertainty.

4.3.5 Optimization Computing Budget Allocation

As previously discussed, the "large number of simulation replications [that] are often required to effectively distinguish between designs is a major challenge that often inhibits stochastic simulation optimization [38].” Hence, ensuring that the proper number of simulation replications are run such that the optimization is able to distinguish between competing plans while not wasting resources is essential to the success of this methodology.

While techniques have been identified that dynamically allocate computing resources to promising designs [38, 62], this is beyond the scope of this thesis. Enough replications should be run such that each case’s process time median and 80th quantile can be estimated to a specified certainty or indifference region. The process time is the only metric considered here because it is the primary metric impacted by the simulation’s stochasticity. The other metrics (e.g. slack time and the neighboring sensor installation metric) are heuristic measures that are not largely impacted by the stochastic nature of the simulation.

This leads to the need to properly identify an overarching number of replications required to properly rank each point to a certain probability. To accomplish this, a random set of 100 sensor installation scenarios (with 2–4 technician cases run for each scenario) are run for 100 replications each. The 100 replications are shown to provide relatively stable solutions for both the median and 80th quantile of the process time (Figures 19 and 20). Hence, the median and 80th quantiles of process time calculated from 100 replications are used to estimate the “true” values of the median and 80th quantiles. Figure 19 presents the calculated median value of the process time at each replication for the 100 installation scenarios examined. For example, a point at Replication 40 represents the median value of replications 1–40. Figure 20 presents the same information for the 80th quantile of the process time.

With the truth data identified, the estimations can be examined at each replication to determine their closeness to the “true” median and 80th quantile. Figure 21 summarizes the difference between the estimates and “true” values of process time as more replications are completed. The box in each box plot shows the interquartile

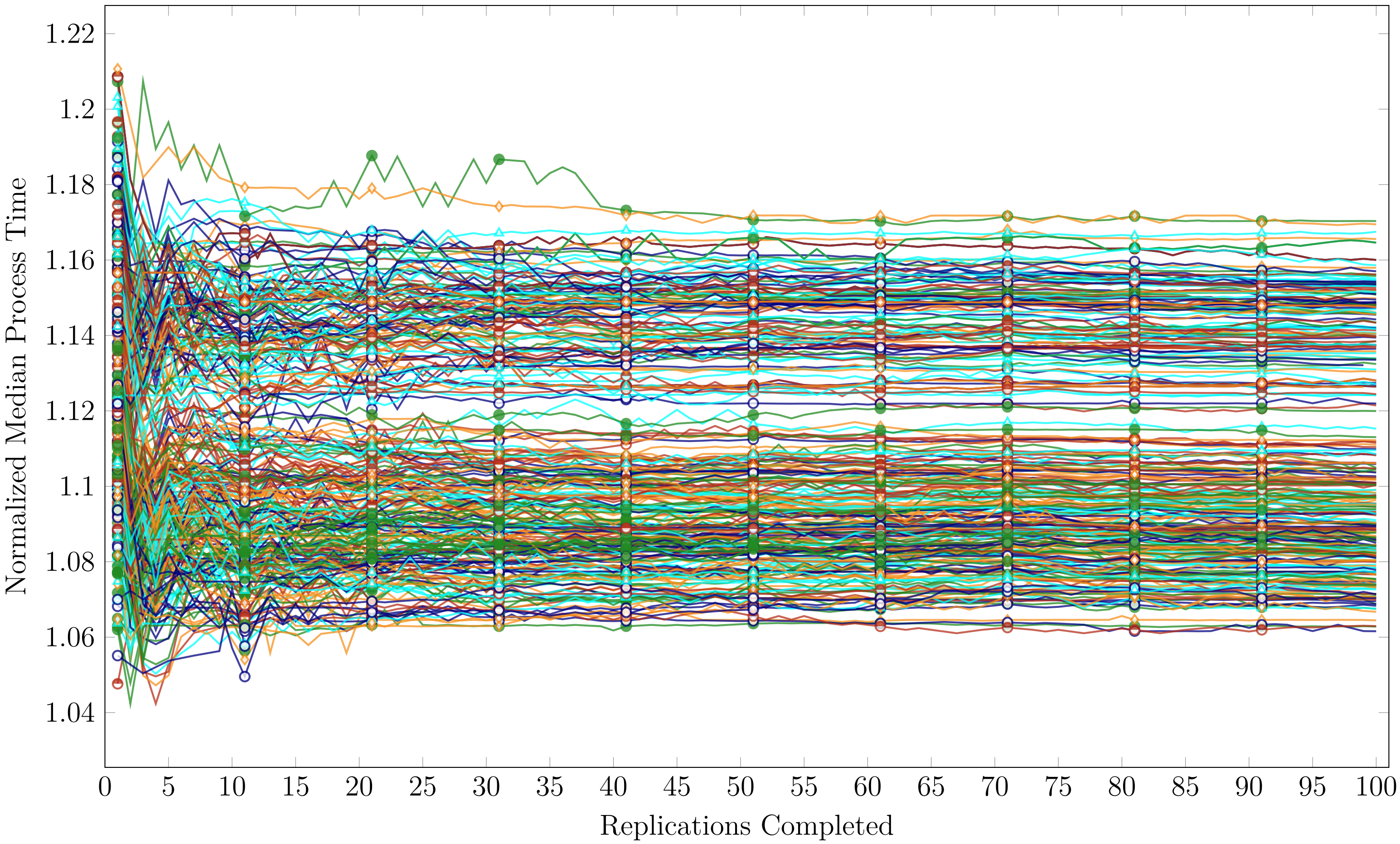


Figure 19: Median Value of 100 Random Sensor Installation Plans for Each Replication

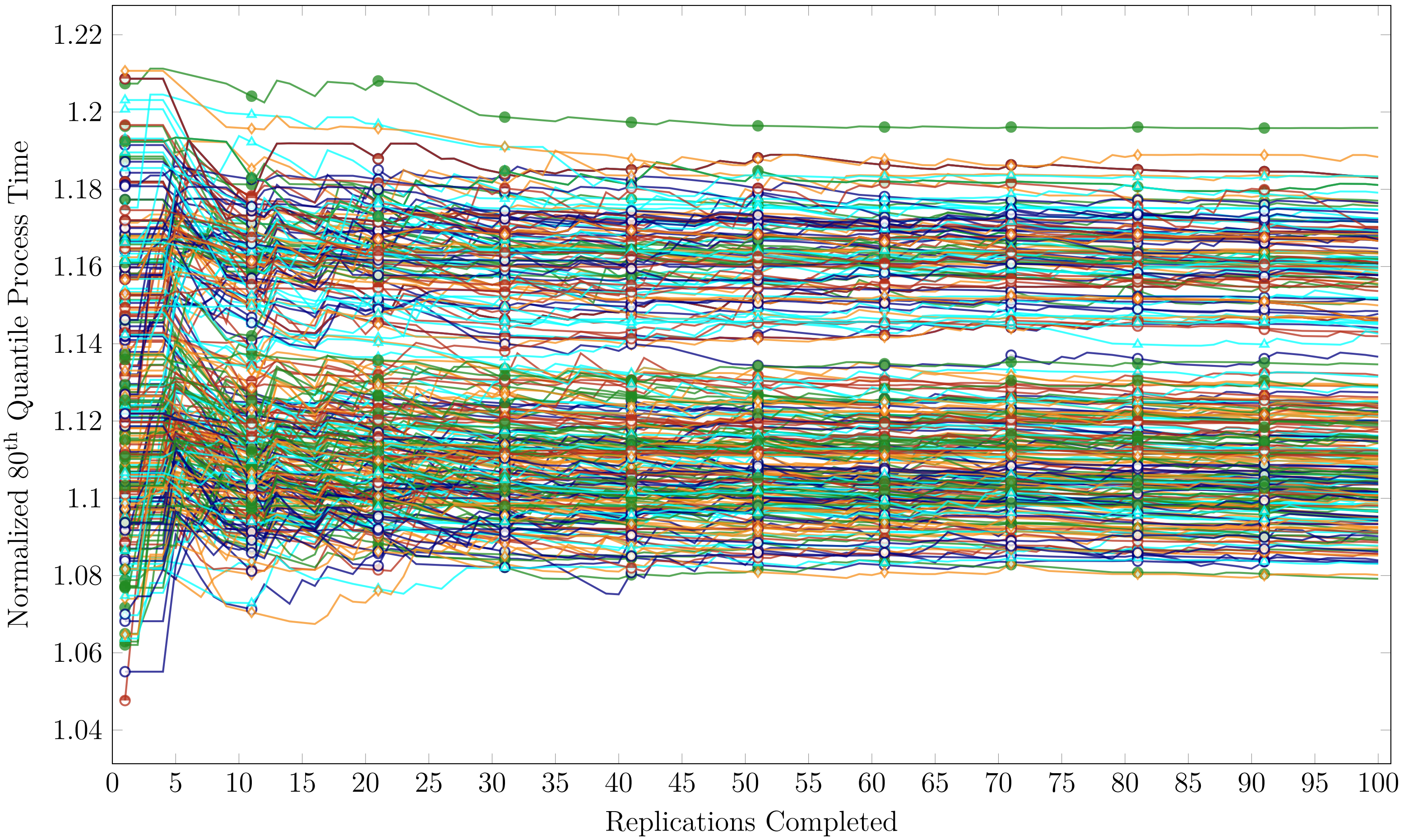


Figure 20: 80th Quantile Value of 100 Random Sensor Installation Plans for Each Replication

range, and the whiskers denote the 5th and 95th quantiles of the data. The figure shows that the estimation's improvement is very rapid until around replication 20. At replication 35, shown as a red box plot, the whiskers are within a reasonable indifference region (i.e. variation between the whiskers is not significant). Hence, it is concluded that 35 replications are sufficient to adequately distinguish between points for optimization.

This concludes the discussion of the major components required for Step 2 of the PORRTSS methodology. Therefore, the following section details the integration and stakeholder interaction required to successfully complete Step 2.

4.3.6 Simulation-based Optimization Component Integration

Figure 22 summarizes the flow of data from the provided schedule and sensor data through the optimization. The outcome of this step in the methodology is a Pareto optimal set of results that must be down-selected and re-integrated with the original planning and scheduling system during Steps 3 and 4, as described in the following section.

4.4 Step 3: Sensor Installation Schedule Down-Selection

With a Pareto efficient set of solutions identified during Step 2, this step of the methodology provides the means to down-select to a final robust schedule for actual implementation. To facilitate this final selection, a decision making environment must be created to compare the Pareto optimal results across the metrics of interest and provide the capability to manually modify the plan based on the planners' expertise. Furthermore, to increase the potential for the methodology to be implemented in the "real world", the visualization environment must contain visuals that are commonly employed by production planners and schedulers to facilitate collaboration across the multiple stakeholders.

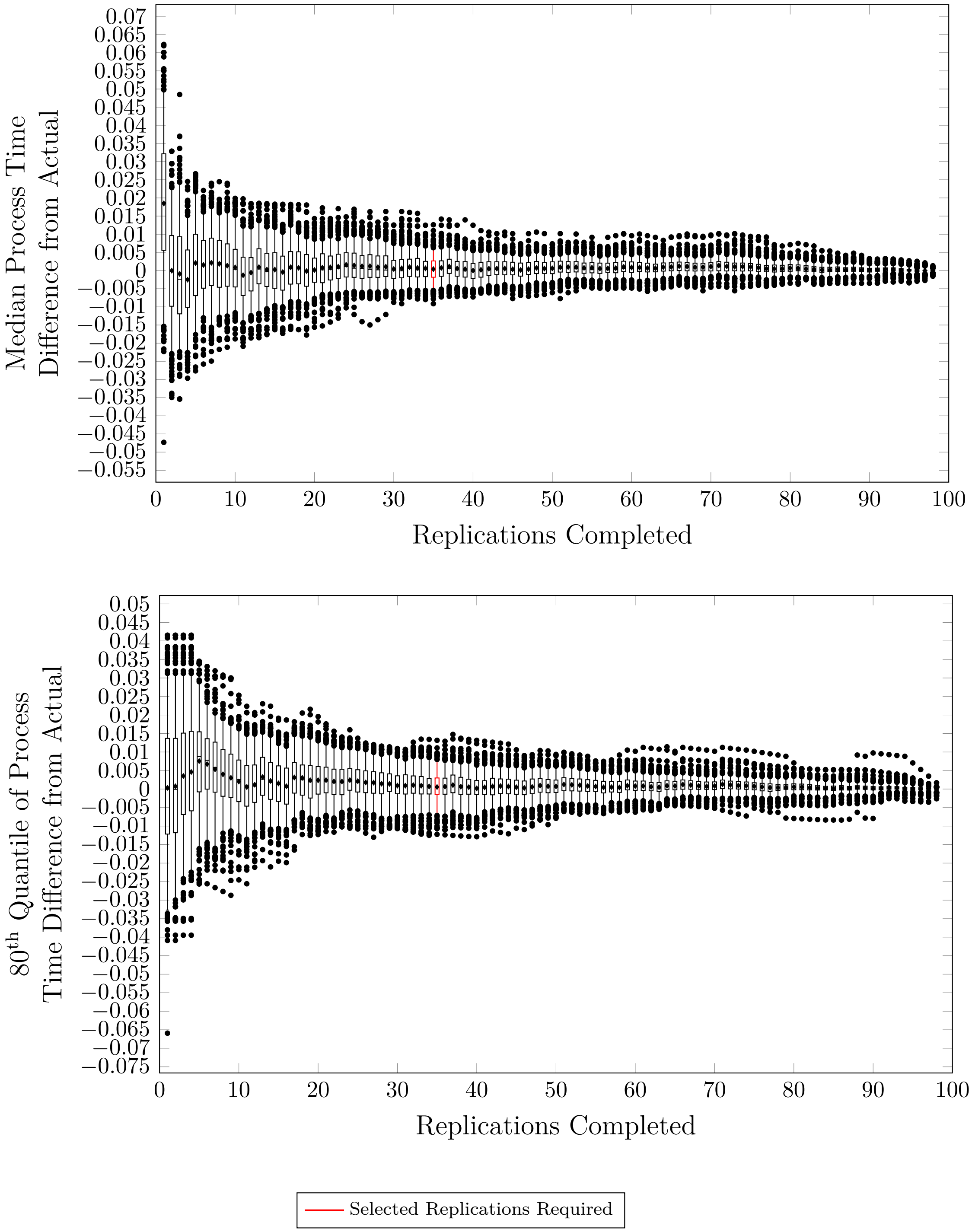


Figure 21: Box Plots Presenting the Difference Between the Current Estimation and True Value of Process Time as More Replications are Completed

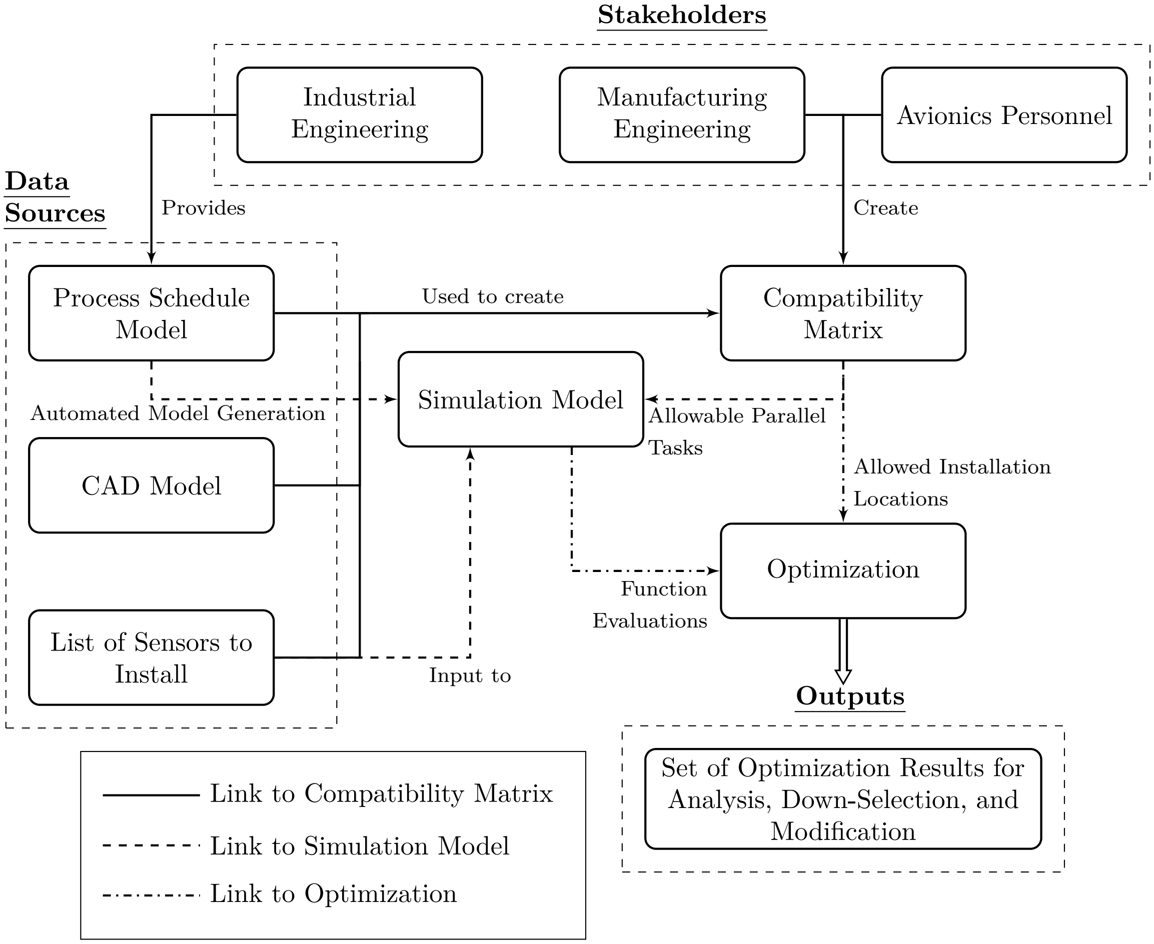


Figure 22: Overview of Data Flow From Step 1 to Step 3

The selection process must incorporate input from each of the three main methodology stakeholders: 1) Industrial Engineers (IEs), 2) Manufacturing Engineers (MEs), 3) Avionics Engineers[1]. Each stakeholder has a different, possibly competing, goal for the sensor installations. The IEs want a plan that has minimal impact and risk to the production plan. The manufacturing engineers are looking for a plan that makes the installations easier for the technicians to complete. Finally, the avionics engineers want to ensure that the sensors, especially the critical ones, are installed and tested correctly and with the highest quality. With these potentially competing goals, a decision support system that can facilitate collaboration is required. The following section discusses the selection and development of visualization techniques and views contained within the decision support environment developed for the purpose of this research.

4.4.1 Visualization Development

The decision support environment must be capable of visualizing and ranking potential solutions in multiple dimensions to support the effective *a posteriori* selection of a solution from the provided set of Pareto optimal solutions [114, 172]. As discussed in Table 7, this problem has 4 objectives of interest; however, due to the parameter settings considered (different number of sensor installation technicians) and the possible strong competition between objectives, the optimization could easily produce thousands of candidate points. Hence, identifying a proper way to visualize, filter, and down-select between these points is necessary [124].

An interactive scatterplot matrix is identified by reference [172] as a strong visualization tool for this type of data. The relevant identified strengths of scatterplot matrices (reproduced from reference [172]) are:

[1] While the Methodology implementor is a *user* of the methodology, he or she is not considered a *stakeholder* at this point. This is because while his or her role is essential, the implementor is not going to be making the final decisions on the plan.

- Ideal for design spaces with few dimensions but many data items
- Useful for the visualization of correlations and trade-offs
- Used as a region query by selecting interesting data points

Because of these strengths, an interactive scatterplot matrix that enables the user to filter and color points based on their preferences and select points of interest for more detailed analysis forms the backbone of the data visualization environment. To further support the stakeholders who may be unfamiliar with scatterplot matrices, additional multi-criteria decision making (MCDM) techniques are implemented.

From discussions with the methodology's stakeholders, the IEs and MEs are more comfortable making decisions with the help of a multi-criteria ranking algorithm. The goal of including the decision making algorithm is to help guide the analysis conducted using the scatterplot matrix by coloring points based on their rankings. As such, an algorithm that uses a user friendly weighting method is desired to help identify regions of points that are high performing according to the defined preferences.

A popular MCDM method that fits these requirements is the Technique for Order Preference by Similarity to Ideal Solution (TOPSIS) [196]. The underlying principle driving the TOPSIS algorithm is that the "best" solution should be geometrically closest to the ideal solution (e.g. the individually best values for each objective function) and farthest from the negative ideal solution [144]. Furthermore, this algorithm provides the ability to guide decision making by illustrating the effect of changing user preferences (e.g. by coloring points based on how their rankings change with modified preferences).

Figure 23 demonstrates the implementation of the TOPSIS algorithm in the decision support environment. The relative weighting of each objective can be modified, which updates the Ranked Solutions table. The table rows' colors indicate how a

Figure 23: TOPSIS Weighting Selection and Ranked Solutions

Response	**Weighting**
Median Flow Time	5.0
Slack Time for All Sensors	0.5
Slack Time for Critical Sensors	0.5
Technicians (Cost)	0.7
Neighboring Sensor Metric	1.0
80^{th} Quantile of the Flow Time	2.5

Index	Median Flow Time	80th Quantile Flow Time	All Sensor Slack Time	Critical Sensor Slack Time	Technicians	Neighboring Sensor Metric
5,864	1.014	1.012	0.842	0.863	3	1.019
977	1.012	1.012	0.845	0.846	3	1.06
2,480	1.019	1.033	0.903	0.933	3	1.047
758	1.013	1.012	0.851	0.846	3	1.064
2,807	1.012	1.013	0.817	0.833	3	1.029
1,511	1.012	1.013	0.816	0.833	3	1.029
3,770	1.014	1.03	0.897	0.899	3	1.079
623	1.014	1.012	0.851	0.845	3	1.065
3,257	1.013	1.012	0.851	0.846	3	1.068
4,382	1.014	1.027	0.833	0.853	3	1.017

solution's rank changed as the weightings change. For instance, if a rows rank improves, it is colored green; if it worsens, the row is colored red. Finally, if the rank remained constant, it is left white. This coloring helps the user to see how changing the weighting is impacting the type of solutions that come out on top.

The TOPSIS results are further used to color points on the scatterplot matrix (Figure 24). The points are colored from green to red based on their TOPSIS rank. This provides a visual cue as to the regions of well performing points. Furthermore, by seeing how the high performing regions change as the weightings are modified, the decision maker can quickly identify, at a macro level, those regions that are consistently well-performing.

While the macro level analysis provided by the scatterplot matrix and TOPSIS algorithm is very helpful to identify candidate solutions, the decision makers require more details to make a final selection and improve the transparency of the solution process. The primary metric to minimize is the delay to the primary processes due to

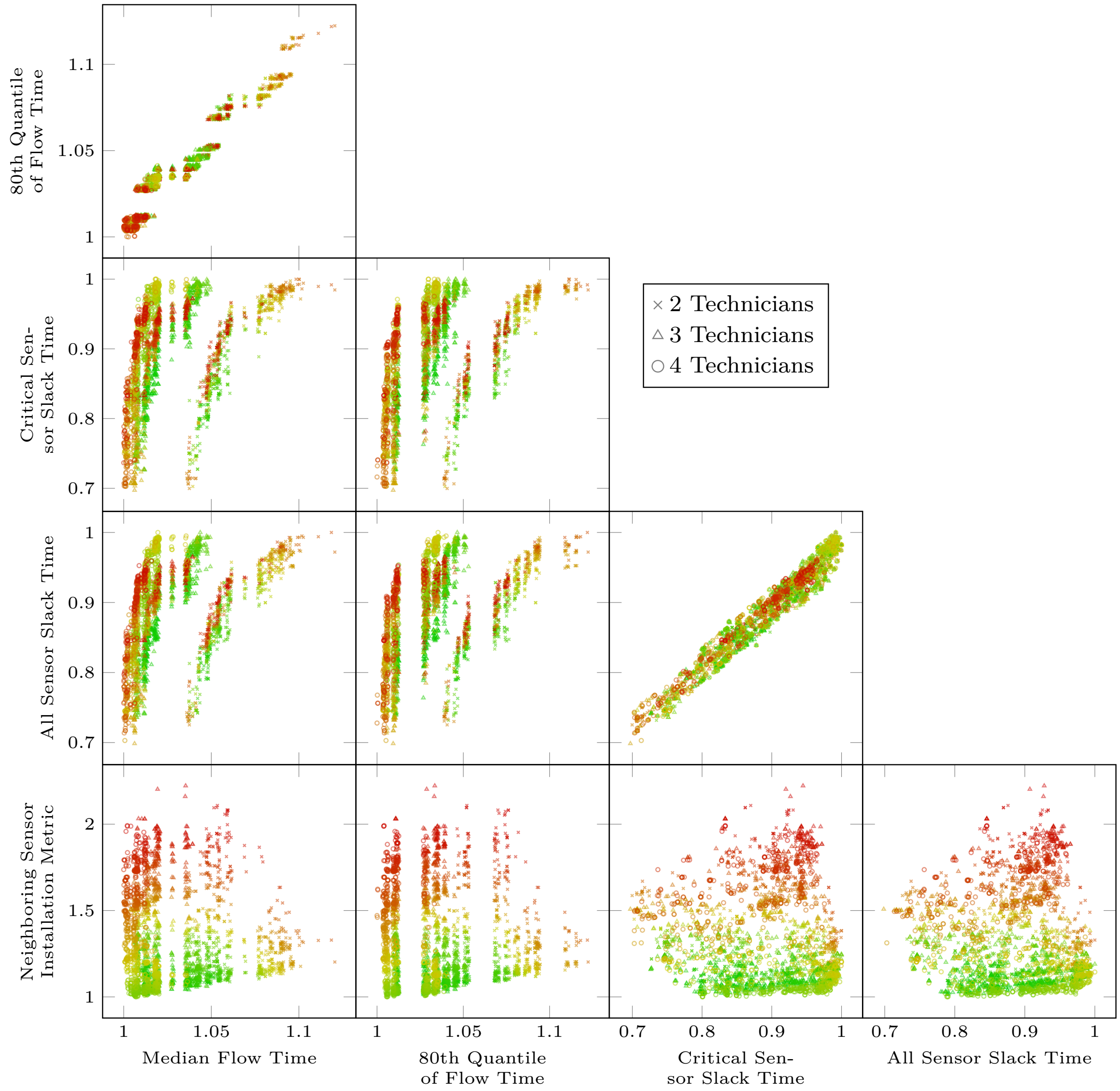


Figure 24: Example Decision Support Environment Scatterplot Matrix with Points Colored from Green to Red Based on their TOPSIS Rank with Weightings Defined in Figure 23

sensor installations. Therefore, providing the capability to visually compare scenarios, which are identified using TOPSIS and the scatterplot matrix, based on the delays to the individual processes is needed.

A promising visualization technique for these comparisons are parallel coordinate plots [9]. An example of the implemented parallel coordinate plot is displayed in Figure 25. The parallel plot presents the delay to the start of each process caused by sensor installations completed during previous processes (vertical axis) for each primary production process (horizontal axis). To further illustrate the distribution of the points (especially if a significant number of scenarios are chosen) [209], a box-and-whiskers plot for the delay to each process' start is drawn behind the parallel plot. A description of the box-and-whiskers plot is illustrated in Figure 26.

This new combination of the parallel and box-and-whiskers plots is motivated by the desire to identify scenarios that are unusual and processes that are largely affected by the differences in the chosen scenarios. The parallel plot provides an easy means to identify the selected scenarios that are much more poorly performing than others. However, as the number of scenarios for comparison begins to increase, it becomes more difficult to effectively identify points that have delays over the median or higher quantile of the data. In these cases, the box-and-whiskers plot can help by providing quick access to summary statistics about the selected scenarios. Furthermore, if the user is interested in identifying processes that have a very inconsistent amount of delay across the scenarios chosen, he or she may turn off the parallel plot. With only the box-and-whiskers plot visible, the user can immediately see those processes with inconsistent delays, which may require additional investigation using the Gantt chart.

Using the table shown in Figure 23, the user can select scenarios to be compared by checking the *Selected* box. Doing so populates the aforementioned parallel plot with information about the selected points. Then using the parallel plot, the user can understand, at a high level, where each plan contains delays. While this provides a

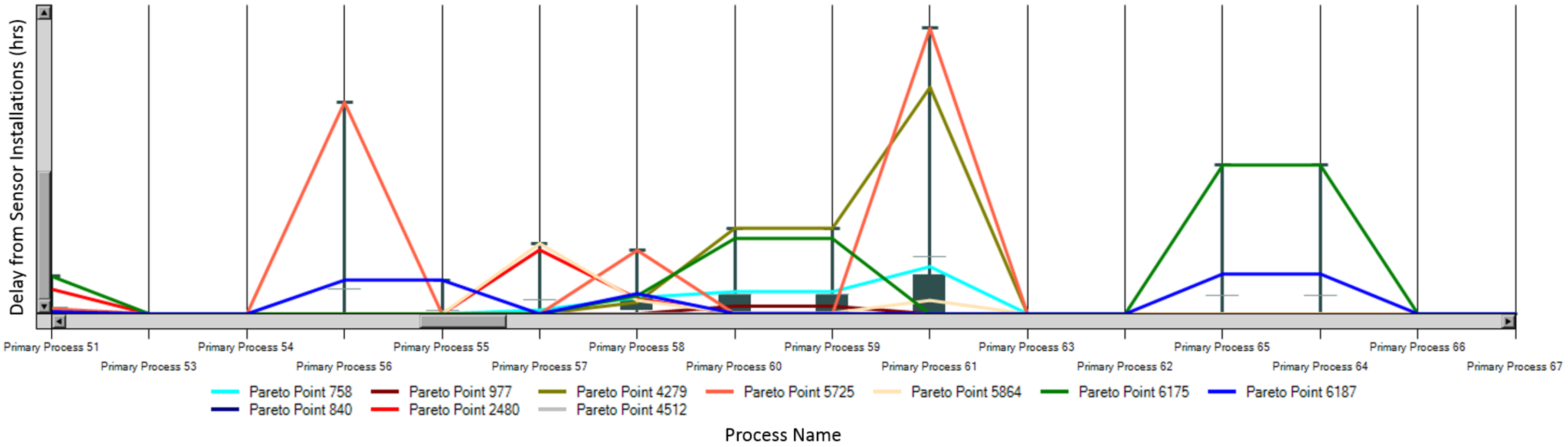


Figure 25: Example Decision Support Environment Scenario Comparisons - Parallel Coordinate Plot Displaying the Delay to the Start of Each Primary Process Due to Sensor Installations for Selected Scenarios

good starting point to make the final decision, the IE and ME stakeholders may also like to assess the impact that a specific the sensor installation has on the schedule. This schedule information should be provided in a convenient format with appropriate interactivity, as discussed in the following section.

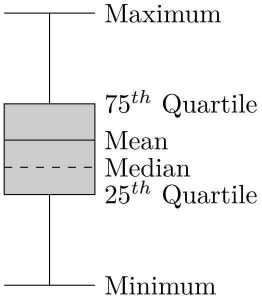
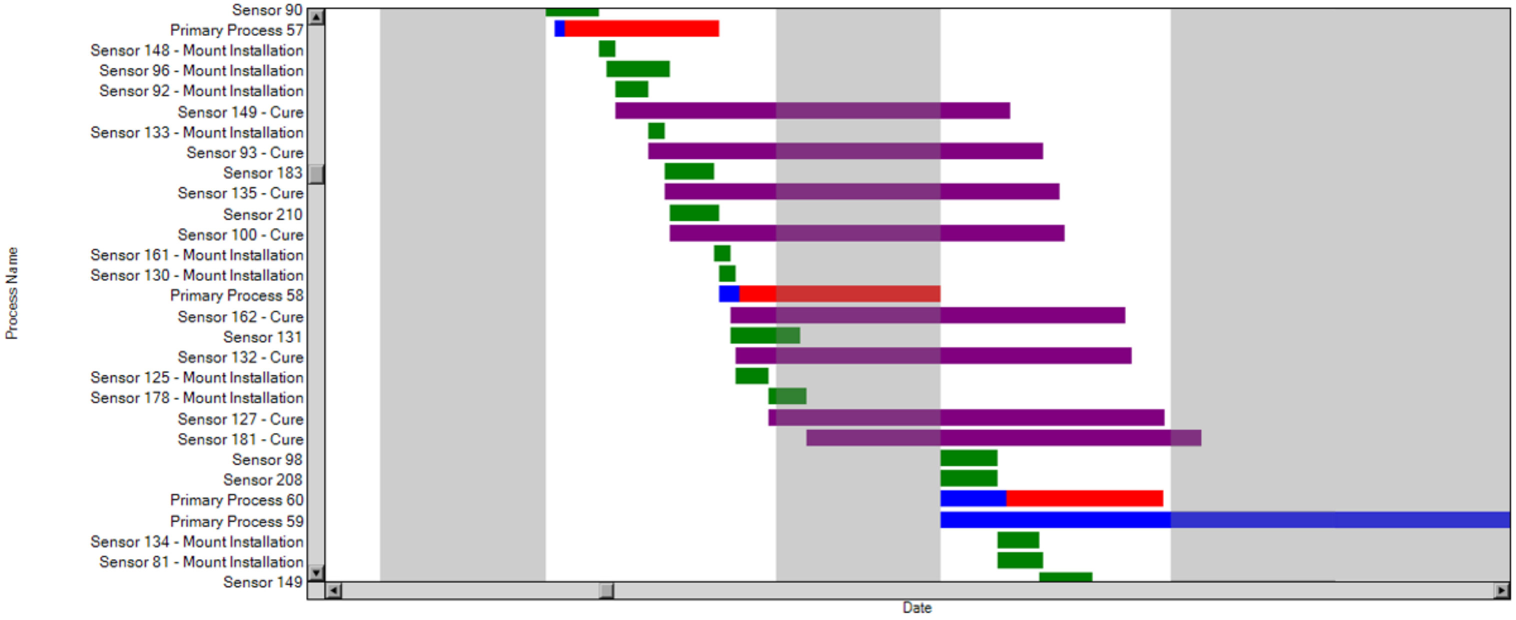


Figure 26: Description Box-and-Whiskers Plot Used in the Decision Making Environment

4.4.1.1 Schedule Visualization and Interaction

It is observed that "it seems necessary for viable scheduling systems to combine the best of historical human expertise, theoretical or mathematical knowledge, and the common sense of the current user [136]." Hence, an effectively implemented schedule decision making system necessarily requires an effective data visualization and user interface to support the human decision maker [45, 60, 79, 125, 198, 201, 205]. An immediately understandable and popular method to display this schedule information is a Gantt chart [79, 210]. Gantt charts are simple tools to provide information in a very comprehensible format [210] while also having the ability to identify reasons for inefficiencies [79]. The popularity of the Gantt chart has only increased with the advent of interactive scheduling applications due to their strength in both displaying and interacting with schedules [72, 210]. Therefore, an interactive Gantt chart is implemented into the decision support environment to display information about the current schedule, facilitate manual schedule modification, and increase transparency [210].



🟦 Primary Process 🟥 Delay to Primary Process 🟩 Subcomponent Installation Requiring Touch Labor 🟪 Subcomponent Installation not Requiring Touch Labor ▨ Off-Shift Time

Figure 27: Example Decision Support Environment Gantt Chart

Figure 27 shows an example of the implemented Gantt chart. The process bar colors help to illustrate the impact of the sensor installations on the schedule. Figure 27 also provides the legend for the Gantt charts to be displayed throughout the remainder of this thesis. In the Gantt chart, the red bars clearly indicate where delays in the process are observed. The causes for the delays can be understood by investigating the green and purple sensor installations planned during the primary process of interest. This chart provides the IE and ME decision makers with the ability to identify when, in the schedule, delays occur and ultimately helps them moving forward with a chosen plan.

Incorporating the planners' expertise is seen as an important aspect of successful scheduling systems [79, 84, 159, 207, 210]. Consequently, it is desired that the human scheduler has input into the schedule beyond choosing the "best" schedule provided by the optimization routine. This capability is included in a "re-planning" view that allows the user to make manual changes to the schedule and re-run the simulation to see their impact. This view contains an interactive Gantt chart that allows the user to select a specific process in the Gantt chart and move sensor installations to or away from the selected primary process. By then re-running the simulation, the planner can identify if the changes he or she made had an impact on the quantifiable objective functions. Then, using his or her knowledge, the planner can weigh this possible change against any other potential qualitative gains from the modification.

Once promising schedules have been identified, down-selected, and potentially modified, they are returned to the original production schedule. Specific formatting and naming conventions must be observed to ensure interoperability. By integrating the results with the original schedule, the planners can then propagate the results through the entire production system. This enables them to understand how the uncertainty and risk affects the overall project completion time to decide whether the proposed solution is acceptable. The following section discusses this integration.

4.5 Step 4: Schedule Propagations & System-wide Analysis

The final step is to propagate the overall schedule throughout the organization. While this is an important step in the methodology, propagating the schedule is the sole responsibility of the IEs and MEs and is, thus, not implemented by the methodology implementor in this thesis. If propagated, the schedule is used to estimate overall cost by the financial and accounting teams, generate work orders by the manufacturing engineers, schedule shifts and resources by production managers, etc. Therefore, ensuring that the results derived from Step 2 and finalized in Step 3 are correctly formatted for the systems in place is necessary to ease adoption.

With the specific implementation of the methodology to the representative case study described, the experimental plan can now be discussed. The case study is used to test each developed hypothesis. Ultimately, the experiments attempt to support the two overarching hypotheses: 1) the methodology is technically feasible for implementation in complex manufacturing environments, and 2) it is “implementable” in the context of complex, low volume, high impact production flows.

4.6 Experimental Plan

The experimental plan follows the structure of the hypotheses. Therefore, the experimental plan begins by addressing Sub-Hypothesis 1.1:

4.6.1 Experiment 1.1: Effectiveness of Discrete-event Simulation to Model the Production System of Interest

Hypothesis 1.1: If discrete-event simulation is leveraged, then increasingly complex scheduling environments can be modeled effectively such that the information required for use in a selected optimization routine can be captured.

The case study features a production environment with strict quality controls and many accessibility and safety issues that need to be considered when planning tasks. Therefore, planning sensor installations within this production system represents the level of complex manufacturing environments that the methodology is designed to solve. The following experiment is setup to test Sub-Hypothesis 1.1:

Experiment 1.1 Using the logic and automated model generation strategy described in Sections 4.2 and 4.3.1, build a discrete-event simulation of the vehicle's production flow with the capability to model sensor installation steps.

Evaluation Criteria The ability of the discrete-event simulation to assess the impact of varying the sensor installation schedule on the overall production flow.

Specifically:

Model Generation Logic is Appropriate: The resulting simulation should respect the choices made by the schedule optimization routine. Incorporation of sensor installations and uncertainty should not violate identified causal process constraints.

Appropriate Sensor Installation Process Logic: The assumptions described in Section 4.3.1 are investigated. The model must adhere to each assumption to ensure the logic is appropriate for integration with the optimization routine.

Impact of Concurrent Installations Allowed: Increasing the number of concurrent sensor installations (by allowing more technicians to install sensors in parallel) should reduce the impact of the installations. Furthermore, because additional resources are available, the risk in the schedule should also be reduced with a more technicians.

Impact of Compatibility Assumptions: Modifying the compatibility assumptions should impact the delay and risk incurred in the schedule. More stringent assumptions should increase delays.

4.6.2 Experiment 1.2: Determine Optimization Strategy's Ability to Improve System Performance

Once the discrete-event simulation is shown to effectively model the production systems of interest and evaluate the possible schedule alternatives, Sub-Hypothesis 1.2 is investigated to determine whether the implemented metaheuristic optimization routine is capable of improving the schedule's performance:

Hypothesis 1.2: If a metaheuristic optimization routine is linked to the developed discrete-event simulation schedule model, then installation plans with improved performance can be efficiently identified.

This sub-hypothesis proposes that a metaheuristic optimization strategy is suited to solve the problem at hand. A major benefit from the formulation of the case study is that the absolute optimum schedule (with respect to process time) is known: the best possible case is to install all sensors in parallel with the primary schedule to avoid any delays due to sensors. This provides a baseline to compare against in Experiment 1.2:

Experiment 1.2 Link a metaheuristic optimization algorithm to the schedule simulation and attempt to find a set of well-performing schedules. This experiment solely focuses on the link between simulation and optimization: no uncertainty is included in the simulation for this experiment

Evaluation Criteria Ability to efficiently identify schedules that are deterministically well-performing with regards to the multiple objectives of interest as described in Table 7. Performance is measured by comparing the improvement of the Pareto frontier of identified installation plans to the initial random population of schedules. Efficiency is measured by the number of generations and time required to identify well-performing points. In addition to qualitative investigations of the algorithm's convergence, Section 4.6.6.2 discusses a set of multi-objective algorithm performance metrics (e.g. the relative improvement and consolidation ratio) useful for quantitatively evaluating the improvement of the population of installation plans.

If successful, this experiment provides evidence that a metaheuristic optimization routine linked to the simulation model is capable of finding improvements to the tested schedule. To complete the investigation of Hypothesis 1, robustness-related measures (specifically quality robustness) must be included as discussed by Sub-Hypothesis 1.3.

4.6.3 Experiment 1.3: Impact of Robust Optimization Considerations

Hypothesis 1.3: If the optimization routine and model can estimate robustness related responses (quality robustness) and support multi-objective optimization, then the methodology will be capable of finding robust schedules.

Testing this sub-hypothesis requires including robustness criteria (e.g. quality robustness) in the optimization. This, in turn, requires that the model be able to estimate schedule robustness criteria and that the optimization algorithm be able to use these criteria to identify better schedules. This leads to Experiment 1.3:

Experiment 1.3 Expand the optimization objective functions to search for schedules with reduced risk. Include uncertainty within the simulation to enable the evaluation of schedule quality robustness. Evaluate the risk in the optimal points identified through the deterministic optimization and compare to those identified through the stochastic optimization.

Evaluation Criteria

Quality Robustness: Ability to identify schedules with reduced process time and risk.

Improvement to Other Risk Metrics: Ability to provide a family of solutions that work to reduce risk associated with sensor installations.

Comparison to Deterministic Results: Ability to differentiate between deterministically optimized schedules and schedules optimized that include quality robustness (e.g. the 80th quantile of the process time). Ability to answer the question: Do the results from the stochastic optimization routine have lower risk than those identified through the deterministic optimization runs?

Conducting these three experiments help to validate Hypothesis 1.

4.6.4 Experiment 1: Methodology's Capability to Reduce Risk in Production Schedules

Hypothesis 1: If a schedule is modeled at the appropriate level of detail via discrete-event simulation and optimized with a multi-objective, metaheuristic algorithm, then the methodology is capable of improving the robustness of complex systems' schedules.

Hypothesis 1 posits that a framework containing the attributes described by Sub-Hypotheses 1.1–1.3 is able to improve the robustness of a complex manufacturing system's schedule. This leads to Experiment 1:

Experiment 1 Integrate the capabilities discussed in Experiments 1.1, 1.2, and 1.3 into an integrated framework. Complete Steps 1 and 2 of the methodology for the use case discussed in Section 3.2.

Evaluation Criteria Investigate whether the integrated framework enables:

- Modeling of complex manufacturing systems
- Optimization of the developed model
- Assessment through the model and improvement through the optimization routine of the schedule's robustness

This first set of experiments (1.1–1.3) is designed to primarily test the technical feasibility of the PORRTSS methodology. The second set of experiments investigates ways that the traditional simulation-based optimization approach can be improved to promote its implementation in the "real-world." As previously discussed through the literature review, improving "implementability" is linked to reducing the time and effort required to setup the methodology (to save cost), shortening the time required to arrive at a solution (to make decisions within the planning horizon), and improving solution quality (to reduce cost and risk in the production environment). This investigation will begin by testing Hypothesis 2.1.

4.6.5 Experiment 2.1: Modeling Speed Improvements

Hypothesis 2.1: If the advanced object-oriented nature of modern discrete-event simulation packages is leveraged to help automate model generation and if metaheuristic algorithms are appropriately implemented to increase the optimization's flexibility, then the methodology's implementation time and effort will be reduced.

This sub-hypothesis focuses on reducing the time and effort required to proceed through the model design and development phases of a simulation project. This sub-hypothesis is tested through the following experiment:

Experiment 2.1 This experiment focuses on evaluating the ability to quickly apply the methodology to new sets of data. Therefore, this experiment will compare the time and effort required to implement the methodology for:

- Manual development of the model with a small number of processes
- Automated model development with a small number of processes
- Automated model development for full-scale problem with more complex logic

Evaluation Criteria

- Time and effort required to generate model for new data set
    - Comparison of assumptions required between the manual and automated strategies
- Methodology scalability → does modeling the larger number of processes require modifying the methodology?
- Changes required to apply the optimization algorithm to a model with a larger number of processes and more complex logic

With the identification of ways to reduce the time to implement the methodology, Hypothesis 2.2 investigates reducing the time to arrive at a set of high quality solutions.

4.6.6 Experiment 2.2: Improve Optimization Convergence Quality and Time

Hypothesis 2.2: If alternative optimization strategies are implemented, then the methodology can be used to explore and exploit the solution space quickly enough to make implementation feasible and viable for a wider range of time and resource constraints and solution quality requirements.

This sub-hypothesis deals with applying different optimization improvement strategies. The following experiment is designed to test this range of strategies:

Experiment 2.2 Investigate a range of optimization algorithms (specified in Table 11) and robustness responses to consider (Table 12) applied to a subset of the problem.

Evaluation Criteria

- Ability to converge to a well-performing schedule with low impact on process time and low risk
- Computing time required for the optimization algorithm to converge

Table 11: Optimization Algorithms to Investigate in Experiment 2.2

Optimization Algorithms
• Genetic Algorithm
• Shifting Bottleneck Inspired Heuristic
• Expanded Neighborhood Local Search
• Weighted-sum Simulated Annealing
• Pareto-based Simulated Annealing
• Weighted-sum Fast Simulated Annealing
• Pareto-based Fast Simulated Annealing

Table 12: Schedule Robustness Considerations to Investigate in Experiment 2.2

Robustness Considerations
• None (Optimize the deterministic model)
• Median of replications
• Higher quantile (80th)
• Multi-objective (median and variance and/or other risk metrics)

Table 13 describes the strategies identified throughout literature that are integrated to improve the optimization's capability. These improvements help to reduce the time required to arrive at quality solutions. The main goal of this experiment is to identify the combination of optimization algorithms and robustness considerations that effectively search the design space. The "effectiveness" can depend on a variety of factors including the size of the model (i.e. the time required to run the model), the time available for the optimization, and the weighting or preference of the decision maker on the various objective functions. Ultimately, this investigation is conducted to identify multiple optimization strategies to select from based on the individual problem's characteristics.

Table 13: Experiment 2.2 Integrated Strategies to Improve Optimization Performance

Desired Improvement	**Strategies**
Reduce Time to Evaluate a Simulation Replication	• Model at the correct level of detail • Observe modeling best practices (remove unnecessary logic, efficiency improvements, etc.)
Reduce Time to Evaluate Full Set of Replications Required to Estimate System Responses	• Maximize parallelization • Reduce replications required by appropriately controlling uncertainty • Minimize overhead time (time to load the models, calculations for the optimization, etc.)

4.6.6.1 Strategy Down-Selection

A challenge to consider when conducting this experiment is the time required to complete each run of the optimization algorithm. The major drawback of metaheuristic algorithms are the number of function calls required for convergence, so unfortunately, not every combination of algorithms, objectives, model sizes, convergence improvement strategies, and robustness considerations can be tested. Therefore, a down-selection process is followed to identify promising strategies for extended analysis. The steps in the down-selection process are outlined in Figure 28.

Implementing this down-selection process assumes that the performance of the algorithms is similar for both the deterministic and stochastic problems. The primary difference between the two problems is that the stochastic problem (e.g. one that runs multiple replications to estimate the 80^{th} quantile or standard deviation of the

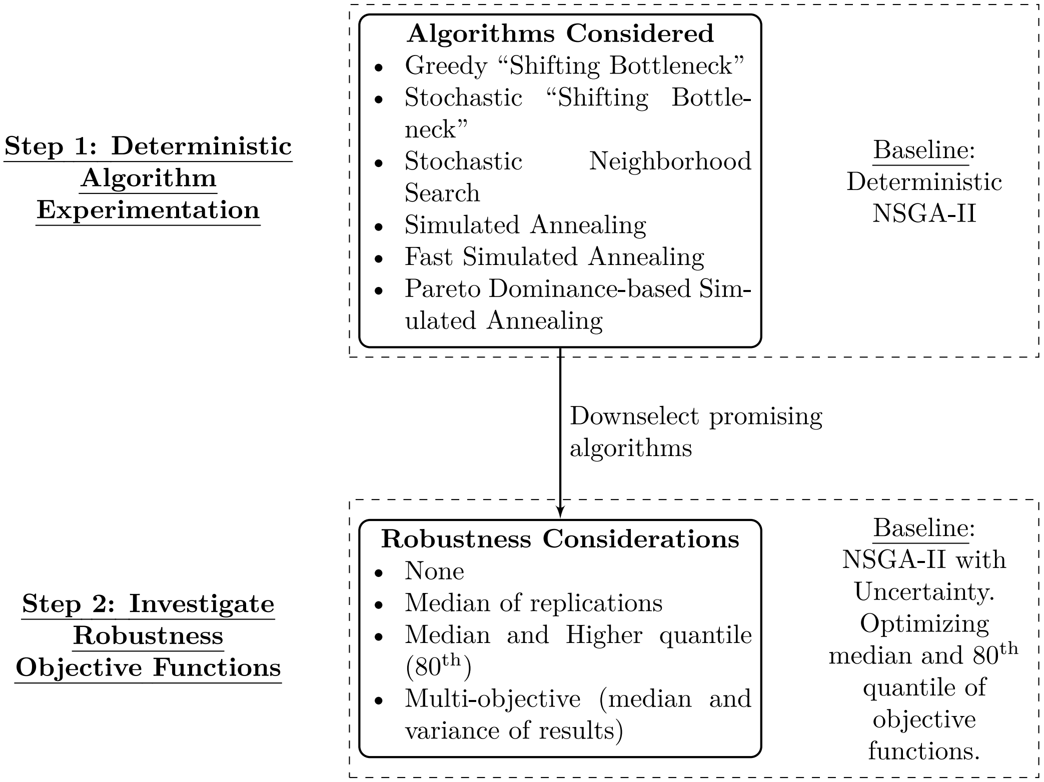

Step 1: Deterministic Algorithm Experimentation
Algorithms Considered
• Greedy "Shifting Bottleneck"
• Stochastic "Shifting Bottleneck"
• Stochastic Neighborhood Search
• Simulated Annealing
• Fast Simulated Annealing
• Pareto Dominance-based Simulated Annealing
Baseline: Deterministic NSGA-II
Downselect promising algorithms
Step 2: Investigate Robustness Objective Functions
Robustness Considerations
• None
• Median of replications
• Median and Higher quantile (80th)
• Multi-objective (median and variance of results)
Baseline: NSGA-II with Uncertainty. Optimizing median and 80th quantile of objective functions.

Figure 28: Experiment 2.2 Down-Selection Strategy

process time) includes an additional objective function to measure schedule robustness. While it is shown that increasing the dimensionality of the objective space can increase challenges related to finding the true Pareto frontier [51], the specifics of the problem at hand limit this effect. First, the median of the process time should behave similar to the deterministic process time because the stochastic process time is set as a triangular distribution centered around the nominal process time. As such, the median of the process time essentially replaces the deterministic process time in the optimization. Second, the added robustness measure, whether it is the 80th quantile of the process time or standard deviation, should also behave similarly to the median process time. For example, if the median is decreasing, the 80th quantile and standard deviation, because of the problem simplifications, should generally decrease as well. Finally, the optimization problem is formulated such that it is unconstrained. Dealing with constraints is difficult for metaheuristic algorithms and leads to challenges when adding dimensions [51], so by removing the need for constraint handling, the algorithm should scale better to higher dimensional problems. Therefore, the assumption that the deterministic algorithm's performance is translatable to the stochastic problem is reasonable.

Throughout the down-selection process, the NSGA-II serves as the baseline for comparison of solution quality, exploration, and convergence time. The NSGA-II is a suitable candidate to provide this baseline because, as discussed in Section 5.1.2, genetic algorithms are generally highly flexible algorithms that, if given enough time, thoroughly explore the Pareto frontier. The NSGA-II can also naturally accommodate multi-objective optimization, so its results are not influenced by objective function weightings. As such, the results found from an extended evolutionary search should provide a suitable approximation of the best quality of results that could be generated. Results from other algorithms and strategies can then be compared to the NSGA-II results. With the down-selection strategy for this experiment outlined, the following section briefly introduces the performance metrics that are used to compare the algorithms.

4.6.6.2 Algorithm Performance Metrics

Common metrics used to compare multi-objective optimization approaches are [27, 51, 139, 192, 193, 217]:

1) **Search Precision:** The algorithm should identify globally Pareto optimal solutions (or at least solutions close to the Pareto frontier).

2) **Computational Time/Effort:** The algorithm must be efficient (measured by actual time and/or number of function calls).

3) **Uniform Search Over Pareto Frontier:** The solutions should be widely and evenly spread over the Pareto frontier.

Comparing computational effort is relatively simple (usually either number of function calls or actual computation time [217]), however for multi-objective problems, comparing the solution quality (closeness to frontier and wide, even spread) is not straightforward [216]. In many studies for multi-objective optimization, algorithm comparisons are done qualitatively (e.g. by comparing plots of Pareto frontiers observed from various runs) [192, 197]. To supplement the subjective nature of this comparison, a more quantitative approach is desired.

Quantitatively assessing the performance of a population of points resulting from a multi-objective optimization run generally requires knowledge of the true Pareto frontier (i.e. the full set of non-dominated solutions that would be found if every possible combination is evaluated) [192,193,197]. Determining this true Pareto frontier is not feasible in this instance, so a baseline estimate is identified and discussed below. With the true frontier identified, a common metric used to evaluate an algorithm is the average normalized distance from each generated Pareto efficient solution in the current Pareto frontier to the closest member of the true Pareto frontier [197]. When the true Pareto frontier is not know, substituting a "combined pool of all generation-wise populations" is appropriate [49]. Equation 9 (reproduced from [197]) defines this metric. In the equation, n is the number of Pareto efficient points found in the current population, d_i is the normalized Euclidean distance from point i to the closest point on the true Pareto frontier, and $p = 2$.

$$G \equiv \frac{\left(\sum_{i=1}^{n} d_i^p\right)^{1/p}}{n} \tag{9}$$

Another common metric to compare population-based algorithms is the spacing between identified optimal points [197]. If the goal is to find Pareto optimal solutions, the algorithm ideally produces a population that is very close to the true Pareto frontier and equally spaced along the frontier to provide a diverse set of options to the decision maker [216]. Hence, a spacing metric can be defined as seen in Equation 10 (reproduced from [197]).

$$S \equiv \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}\left(\bar{d}-d_i\right)^2} \tag{10}$$

where n is again the number of Pareto efficient points found, d_i is the normalized distance from point i to the nearest point in the found set of Pareto points, and $\bar{d}$ is the average of d_i, where d_i is defined as:

$$d_i \equiv \min_j \{|\vec{p_i} - \vec{p_j}|\} \tag{11}$$

where $\vec{p}$ is the vector of values for each Pareto point. A value of 0 for the metric defined in Equation 10 indicates that the points are evenly spaced.

A final metric of interest to help identify the quality of an algorithm's results is the percentage of points identified that are Pareto optimal [197]. While this metric does not give a sense for the quality of solutions, it helps to illustrate how efficiently the algorithm is able to generate Pareto optimal solutions.

These metrics help to quantify the effectiveness of a population based metaheuristic, however there remain challenges to their use in the comparisons for this dissertation. One challenge is that the actual Pareto frontier must be known to calculate the metrics. Because the frontier for the problem studied in this thesis is unknown, this necessitates the need for a baseline algorithm. Secondly, the metrics are designed to study population-based, evolutionary algorithms, while the present study is interested in single points algorithms as well. Hence, the experiments must be conducted in such a way as to enable relevant comparisons using these metrics.

To address the first challenge, a suitable baseline Pareto frontier must be produced to enable comparisons between additional algorithms. The NSGA-II algorithm is selected to provide this baseline because it 1) inherently handles multi-objective optimization without requiring weighting schemes or other biases and 2) is one of the most popular evolutionary algorithms used for multi-objective optimization. Because evolutionary algorithms "have become *the* method at hand for exploring the Pareto-optimal front in multi-objective optimization [216]," the NSGA-II is, if given enough time, likely to be the most successful of the considered algorithms. Furthermore, because the problem at hand only contains discrete variables, there is no discretization errors usually associated with genetic algorithms. Hence, to provide the baseline for the other algorithms of interest, the NSGA-II is run until it converges. After the algorithm has converged, the Pareto optimal cases from every generation are gathered to serve as an estimate of the true Pareto frontier to enable comparisons between the other algorithms of interest.

Determining the convergence of a multi-objective algorithm is, however, also not straightforward [71, 192, 193]. One challenge is that many measures utilized to test the performance of a multi-objective algorithm, as discussed above, rely on knowledge of the true Pareto Frontier. To overcome this challenge, metrics that compare the relative improvements of the algorithm are proposed.

The first proposed metric investigates the relative improvement of each generation using the generational distance (i.e. closeness to the Pareto frontier) defined by Equation 9 [20,192,193]. Mathematically, the relative improvement is defined as [193]:

$$RP \equiv \ln \sqrt{\frac{G_1}{G_i}} \tag{12}$$

where G_1 is the generational distance of the initial population and G_i is the generational distance of the i^{th} generation. In each case, G is calculated using the Pareto frontier containing the Pareto points identified through all of the generations of the optimization run. By normalizing by the first generation, this metric shows how the current generation compares to the initial generation. Hence, this metric increases as the current generation gets closer to the found Pareto frontier. As progress slows, the relative improvement begins to flatten. This indicates that while new Pareto efficient points may be found, the overall quality of the population is not significantly increasing. Therefore, at the point when the relative improvement curve flattens, the algorithm can be considered converged.

The second metric to be used to help determine the convergence of the NSGA-II is a consolidation ratio [71]. The consolidation ratio is interested in the “survival” of Pareto points from each generation. The general idea is that if the Pareto efficient points from one generation are still non-dominated in a future generation (i.e. no improved points have been identified), then the algorithm has converged. For each generation, an archive of non-dominated points from that and all previous generations are stored. Then, starting from generation $i = j + 1$, where j is a specified generational offset to reduce noise, the archive from generation i is compared to that from generation $i - j$. For instance, if $j = 10$, then the number of Pareto efficient points from generation 1 that are still present in the archive from generation 11 are compared. The number of surviving points found is then divided by the total number of points in the archive from generation $i - j$ to provide a fair comparison. A value of 1 means that every point from generation $i - j$ is still present in generation i. As this metric approaches a specified limit (e.g. when 80% of the Pareto points survive j generations), then the algorithm can be considered converged because there is little improvement still occurring [71].

Both metrics are considered when determining the convergence of the NSGA-II. The relative closeness metric helps to show how much improvement, compared to the initial generation, the algorithm is achieving as it nears convergence. The consolidation metric indicates whether running additional generations are likely to identify a significant number of new Pareto points. Therefore, once the algorithm slows improvement and is not producing many new Pareto points, it is considered converged and, because the NSGA-II is shown to perform very well for this class of problems, can provide a baseline to compare the other algorithms.

With a suitable baseline identified, an approach to compare results from point solution algorithms to the population-based NSGA-II must be developed. By running the algorithm multiple times at various starting locations, the algorithm should arrive at different points on the Pareto frontier. Then, by comparing this "population" to the baseline frontier determined by the NSGA-II using Equation 9, the resulting solutions' closeness to the baseline Pareto frontier can be evaluated. The spacing metric (Equation 10) is only evaluated for the Pareto-based simulated annealing algorithms because the weighted sum algorithms ideally should converge to a single point.

With the experimental down-selection process discussed and detailed algorithm performance metrics identified, Experiment 2.2 is performed. Upon successful completion, Experiments 2.1 and 2.2 help to inform the experiment testing the overall Hypothesis 2.

4.6.7 Experiment 2: Methodology's Potential for Implementation to Handle "Real-World" Problems of Interest

Hypothesis 2: If the methodology requires a low amount of implementation effort and is shown to provide clear benefits with acceptable increases in computation time over traditional scheduling methods while effectively integrating the knowledge of the human planner, then the methodology can be successfully implemented to solve "real-world" problems.

This Hypothesis focuses on taking the results from the previous experiments and applying the methodology to a "real-world" problem. Therefore, Experiment 2 is developed:

> **Experiment 2** Apply the developed methodology to support the scheduling of sensor installations for a major subassembly within the case study vehicle.

Evaluation Criteria Investigate whether the needs identified in Hypothesis 2 have supported the implementation of the methodology. Specifically, investigate:

- Time required (methodology setup and optimization time) to find the set of potential solutions. Can this be accomplished within the required planning window?
- Optimization results and analyses are improved compared to the baseline's planning process
  - Are the proposed plans feasible?
  - Are the results of high enough quality and well spread across the Pareto frontier to facilitate decision making?
  - Did the model properly identify portions of the production flow impacted by sensor installations?
- Stakeholders can leverage the provided data visualization and decision support tool to down-select and modify the sensor installation plans

This final experiment is important to the overall evaluation of the methodology because it tests the "implementability" of the methodology. One of the major complaints about many scheduling methodologies is that they are difficult to use in the "real world." Hence, exploring how the decision makers utilize the provided results and decision making environment enables a final check on the quality of the methodology. In essence, this experiment is designed to confirm that the developments explored and implemented through the previous experiments combined with the presentation of the results in the decision making environment produce a scheduling system that can be applied by industrial users.

4.6.8 Experimental Plan Summary

The experimental plan is designed to continuously test and build up the capabilities required to integrate robust design principles to schedule production and assembly processes in manufacturing environments with significant constraints. Upon completion of the experimental plan, a determination can be made as to the capability of the developed methodology to model and optimize these systems to reduce risk. One area that is not directly examined is the methodology's extendability to other scheduling problems. While the experimental plan does not directly test the methodology's extendability by applying the methodology to multiple problems, the customizations and effort required to implement the methodology for the specific case study are reported. Hence, the effort required to implement specific portions of the methodology can inform possible users about its applicability to their problem.

The following chapter discusses the completion of the experimental plan and initial observations made. Experimental results are presented and discussed. The Hypotheses generated throughout Chapter 3 are then examined in the context of the experimental results to assess whether they are plausible responses to the identified research questions.

CHAPTER V

EXPERIMENTATION & RESULTS

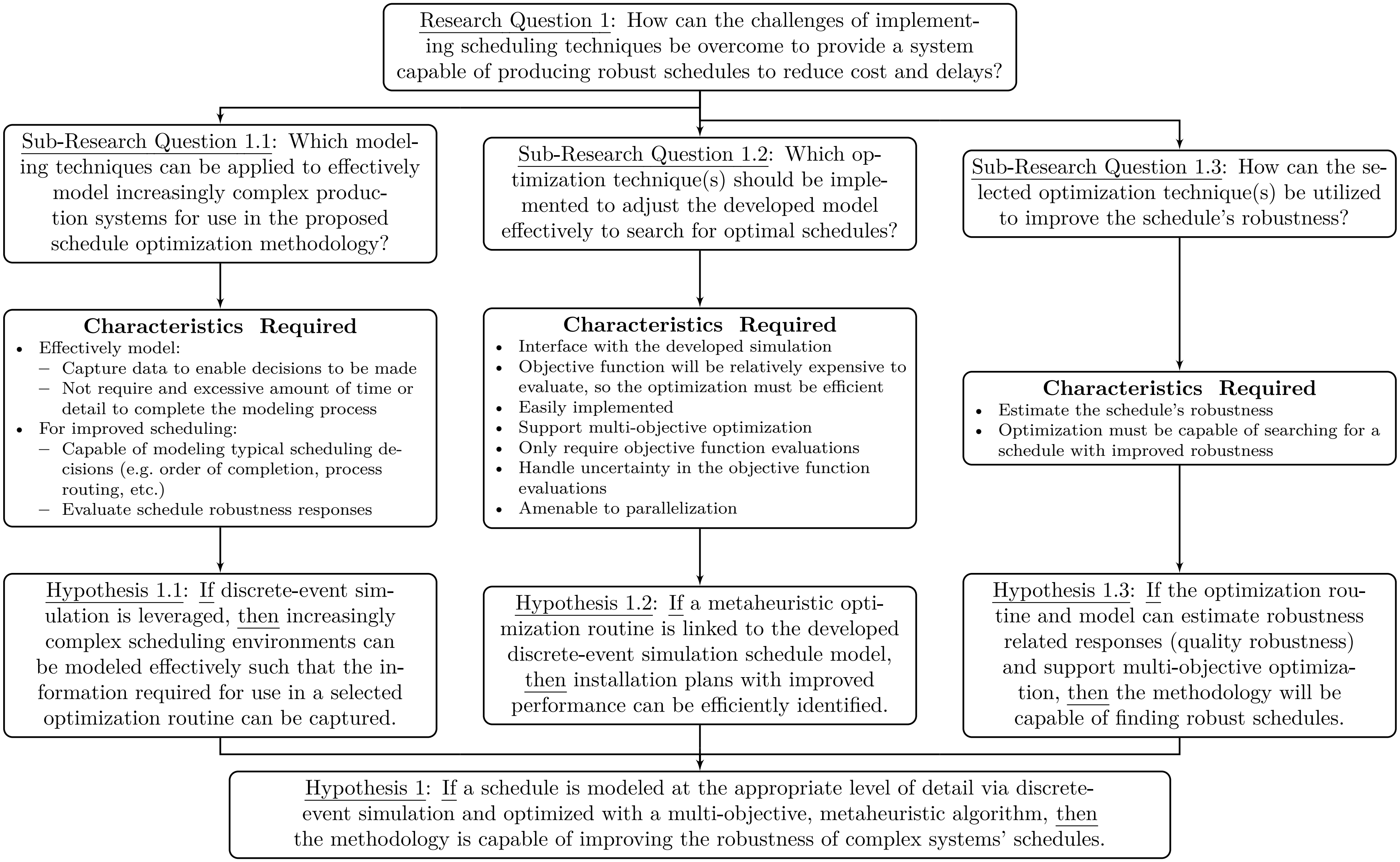
The experiments identified in Chapter 4 are completed to incrementally build up support for the hypotheses developed throughout this thesis. The ultimate goal of this chapter and Chapter 6 is to demonstrate that the PORRTSS methodology, by demonstrating the capabilities and characteristics identified throughout Chapters 3 and 4, successfully addresses the identified research gaps and improves on the current detailed planning process. Hence, results from Experiment 1 are addressed in the following section.

5.1 Experiment 1: Methodology's Capability to Reduce Risk in Production Schedules

This experiment is designed to investigate Hypothesis 1:

> **Hypothesis 1**: If a schedule is modeled at the appropriate level of detail via discrete-event simulation and optimized with a multi-objective, metaheuristic algorithm, then the methodology is capable of improving the robustness of complex systems' schedules.

The sub-hypotheses and associated experiments are developed in order to build up support for Hypothesis 1. Therefore, each sub-hypothesis is examined in detail in the following sections. The results are generated using a simulation of the structural assembly and sub-system integration process for a major sub-assembly of the case study's vehicle. The process flow is chosen because 1) it is representative of the complexity encountered in other sequences for the case study and 2) has a reasonable runtime to ensure many model runs can be evaluated.



Research Question 1: How can the challenges of implementing scheduling techniques be overcome to provide a system capable of producing robust schedules to reduce cost and delays?

Sub-Research Question 1.1: Which modeling techniques can be applied to effectively model increasingly complex production systems for use in the proposed schedule optimization methodology?

Sub-Research Question 1.2: Which optimization technique(s) should be implemented to adjust the developed model effectively to search for optimal schedules?

Sub-Research Question 1.3: How can the selected optimization technique(s) be utilized to improve the schedule's robustness?

Characteristics Required

- Effectively model:
  - Capture data to enable decisions to be made
  - Not require and excessive amount of time or detail to complete the modeling process
- For improved scheduling:
  - Capable of modeling typical scheduling decisions (e.g. order of completion, process routing, etc.)
  - Evaluate schedule robustness responses

Characteristics Required

- Interface with the developed simulation
- Objective function will be relatively expensive to evaluate, so the optimization must be efficient
- Easily implemented
- Support multi-objective optimization
- Only require objective function evaluations
- Handle uncertainty in the objective function evaluations
- Amenable to parallelization

Characteristics Required

- Estimate the schedule's robustness
- Optimization must be capable of searching for a schedule with improved robustness

Hypothesis 1.1: If discrete-event simulation is leveraged, then increasingly complex scheduling environments can be modeled effectively such that the information required for use in a selected optimization routine can be captured.

Hypothesis 1.2: If a metaheuristic optimization routine is linked to the developed discrete-event simulation schedule model, then installation plans with improved performance can be efficiently identified.

Hypothesis 1.3: If the optimization routine and model can estimate robustness related responses (quality robustness) and support multi-objective optimization, then the methodology will be capable of finding robust schedules.

Hypothesis 1: If a schedule is modeled at the appropriate level of detail via discrete-event simulation and optimized with a multi-objective, metaheuristic algorithm, then the methodology is capable of improving the robustness of complex systems' schedules.

Figure 29: Research Question and Hypothesis 1 Buildup

Figure 29 presents an overview of the research questions and hypotheses investigated throughout Experiment 1. To build up overarching support for Hypothesis 1, the following section investigates Experiment 1.1.

5.1.1 Experiment 1.1: Effectiveness of Discrete-event Simulation in Modeling Production Systems of Interest

Before investigating any optimization or schedule improvement approaches, this initial experiment is designed to demonstrate the capability to assess the impact of a defined installation plan on the overall production schedule using the model generation logic discussed in Section 4.2. With the simulation model and compatibility matrix in place, results demonstrating the effectiveness of the logic and its usefulness for decision-making are reported to investigate Hypothesis 1.1:

> **Hypothesis 1.1**: If discrete-event simulation is leveraged, then increasingly complex scheduling environments can be modeled effectively such that the information required for use in a selected optimization routine can be captured.

The effectiveness of the model logic is investigated using Experiment 1.1's evaluation criteria:

Appropriate Model Generation Logic: The resulting simulation should respect the choices made by the schedule optimization routine. Incorporation of sensor installations and uncertainty should not violate identified causal process constraints.

Appropriate Sensor Installation Process Logic: The production flow logic discussed in Section 4.3.1 is investigated. Each assumption is detailed and an example from the model run is presented in Section 5.1.1.2. The model must adhere to each assumption to ensure the logic is appropriate for integration with the optimization routine.

Impact of Concurrent Installations Allowed: Increasing the number of concurrent sensor installations (by allowing more technicians to install sensors in parallel) should reduce the impact of the installations. Furthermore, because additional resources are available, the risk in the schedule should also be reduced with more technicians.

Impact of Compatibility Assumptions: Modifying the compatibility assumptions should impact the delay and risk incurred in the schedule. More stringent assumptions should increase delays.

The following sections discuss how the model logic meets the aforementioned effectiveness criteria.

5.1.1.1 Model Generation Experimental Results

Evaluating the model logic begins by ensuring the baseline simulation without any uncertainty or sensor installations matches the schedule model. The Gantt charts for the schedule and simulation can be compared to ensure the simulation matches the schedule. Figures 30 and 31 display the Gantt charts for the baseline schedule and simulation model, respectively. As seen in the figures, the general structure of the schedule is very similar. Some slight discrepancies can be explained by the small number of processes that must be completed in a single shift (e.g. a moving operation requiring a crane). For these processes, the schedule does not start them near the end of a shift; however, the simulation model does not account for this constraint because it only affects a small number of processes and the overall impact is negligible (the simulation and schedule's process time match within an acceptable 0.1%).

The simulation must also observe the structure of the provided schedule even when sensors are incorporated. Figure 32 provides the Gantt chart for the primary production processes when sensors are installed. The figure is plotted on the same scale as Figures 30 and 31. The general structure of the Gantt chart is again similar

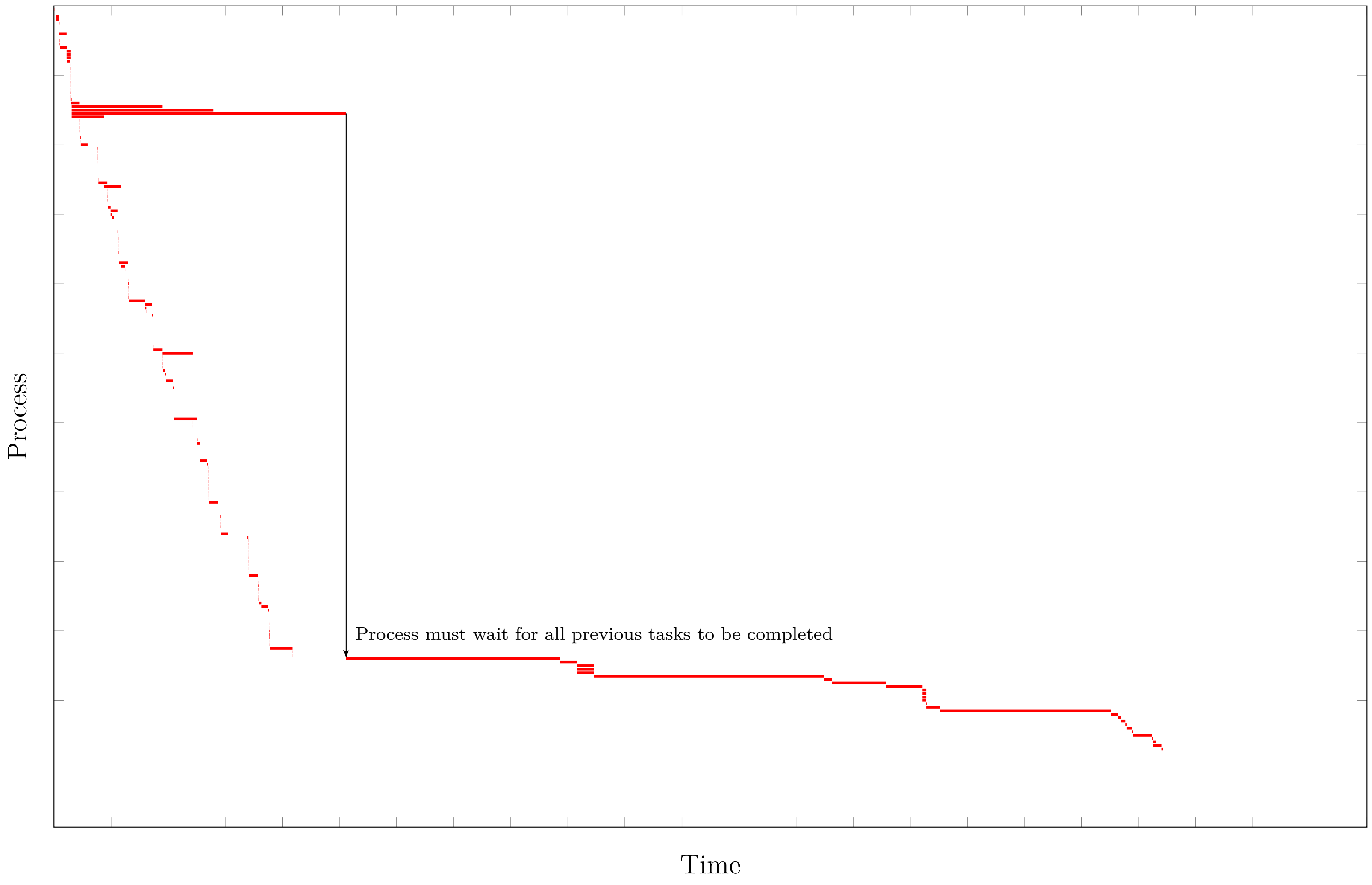


Figure 30: Gantt Chart for the Optimized Schedule Model

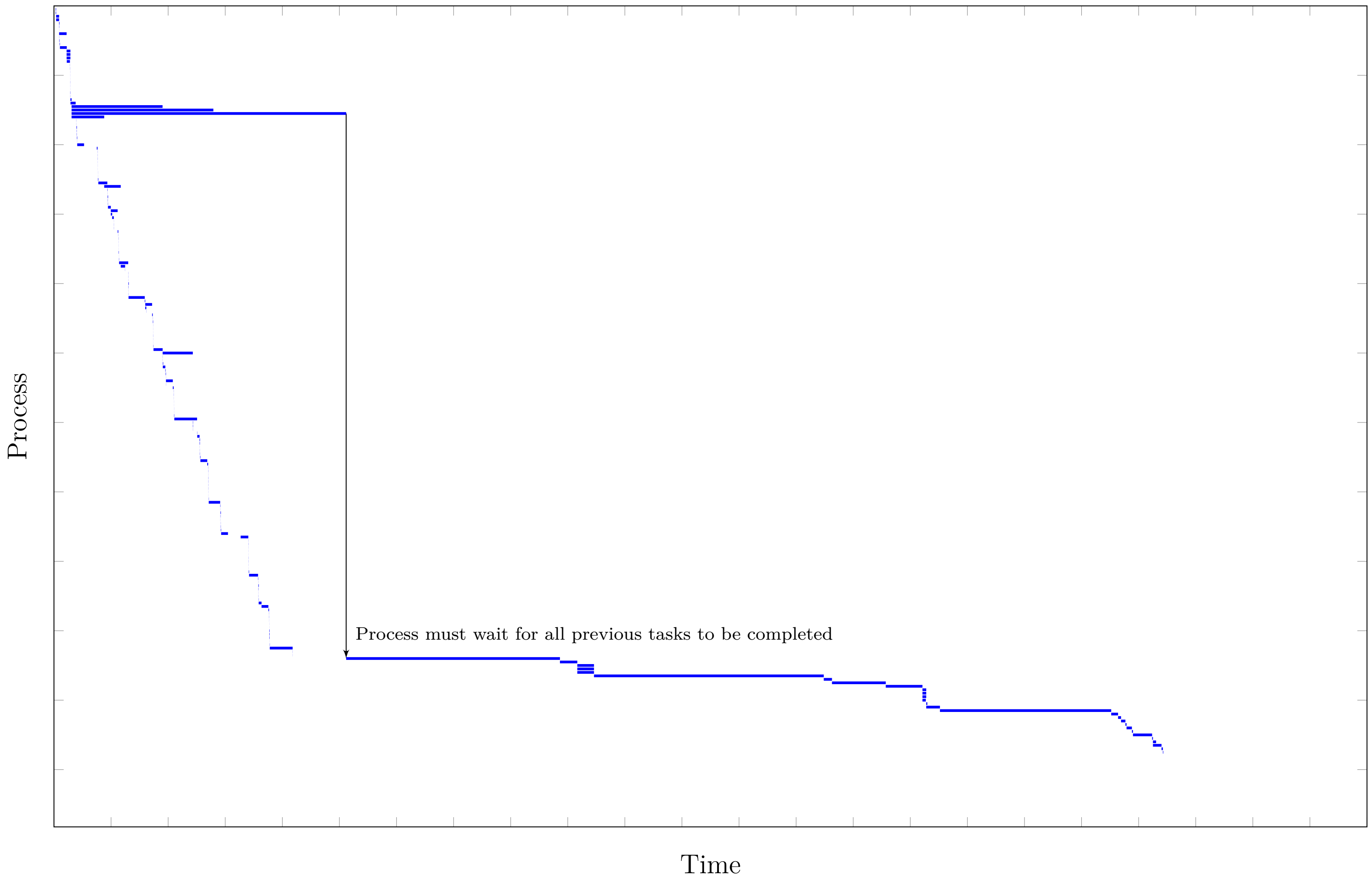


Figure 31: Gantt Chart for the Baseline Simulation Model without Sensor Installations

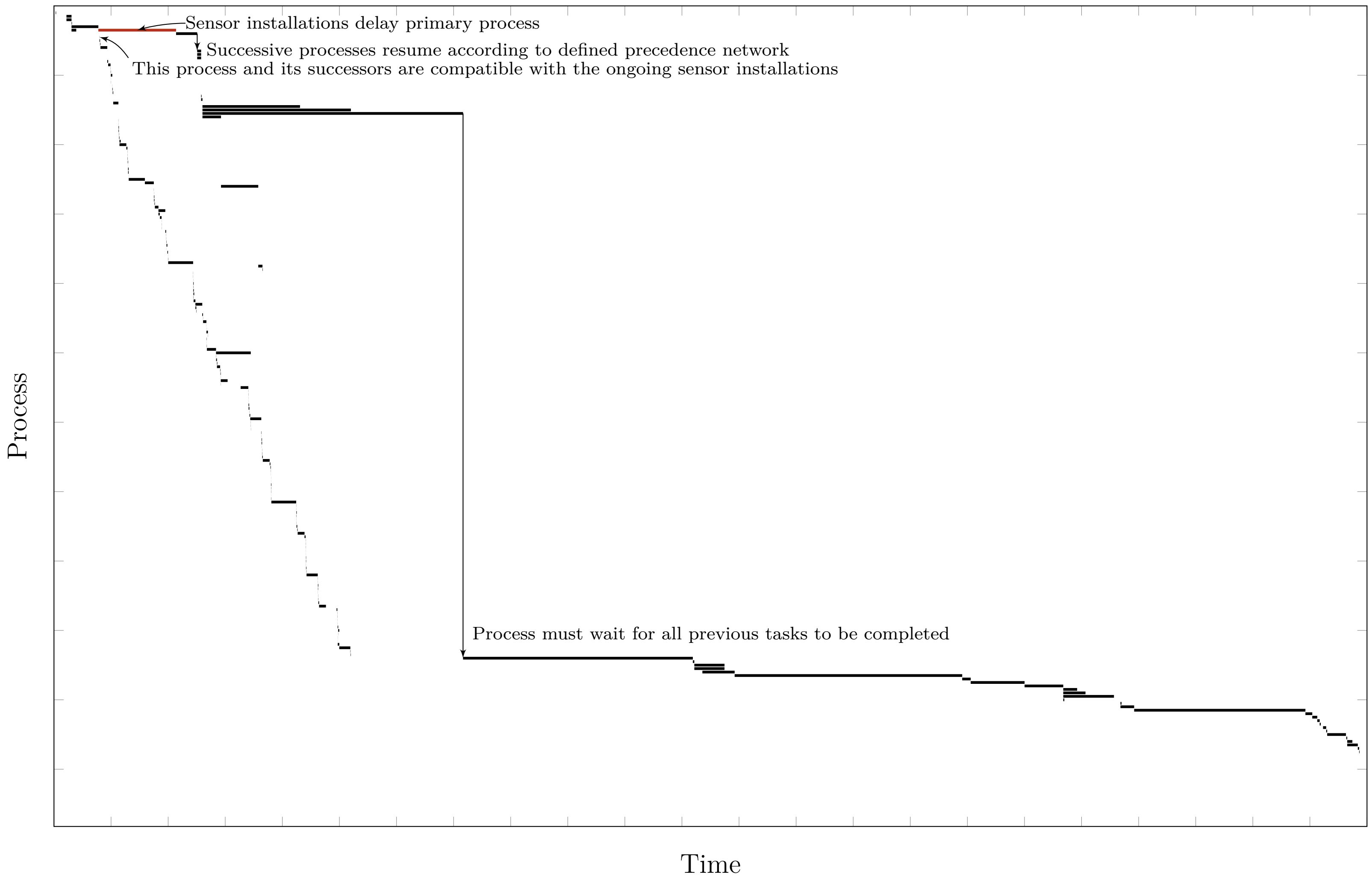


Figure 32: Gantt Chart for the Simulation Model with Sensor Installations

to the baseline shown in Figure 31. It can be seen that the primary difference between both figures is that adding sensor installations logically causes some of the primary processes to be started late (because sensor installations during previous tasks delayed the process) or be extended. Figure 32 illustrates, at a high level, how a delay early in the process cascades to linked processes further along in the flow. A more detailed look at the causes of these delays is presented in Section 5.1.1.2.

The full set of process constraints are not captured by identifying the pseudo-precedence relationships; this could lead to some violations of physical or causal constraints when sensor installations disturb the baseline flow. While this is an expected result of the assumptions made, this should not lead to an overly optimistic model that cannot discriminate between sensor installation plans. To investigate this, the total number of parallel processes throughout the Gantt chart can be identified. If many more parallel tasks are allowed in a simulation run with sensor installations, this would indicate that the model is allowing extra parallel work because it does not have a full picture of the constraints. Task_i is considered in parallel with task_j if the following condition is true: $\text{end}_i > \text{start}_j$ & $\text{start}_i < \text{end}_j$. The baseline simulation (Figure 31) for this case has 526 processes that occur in parallel while the simulation with sensor installations (Figure 32) has only 446. Therefore, there is actually less parallel work occurring, indicating that the simulation with sensor installations is actually more conservative in the completion of primary processes.

These somewhat qualitative investigations of the impact of the model generation strategy are lacking a complete quantitative justification. Unfortunately, without complete access to the schedule model and underlying constraints, complete justification is not possible (i.e. the full set of constraints are not known; only those that can be identified by analyzing the precedence network and schedule start and end times are available). However, because the model is designed to identify the best sensor installation plan (as opposed to providing a true evaluation of the schedule impact due to the installations), the simulation model is capable of providing sufficient information for the optimization routine.

5.1.1.2 Investigation of Sensor Installation Logic

With the simulation appropriately respecting the choices of the provided, optimized schedule, each of the assumptions described in Section 4.3.1 (reproduced below) are investigated to ensure the sensor installations are correctly modeled to support optimization. The results are investigated by examining Gantt charts that are output from a simulation run. This section provides examples from the verification process; a more thorough process that examined the logic in multiple models with a myriad of installation plans was conducted to fully verify the logic. The Gantt chart is color coded to help investigate the impact of the sensor installations. The process logic rules are:

A1. All manual installation sequences scheduled to occur during a primary process must be started before the primary production flow can proceed. Once all installation sequences have *started*, the successive primary process is allowed to begin *if* it is compatible with the ongoing manual installations. If it is *not* compatible, then the subsequent primary process must wait for the sensor installations to finish before beginning.

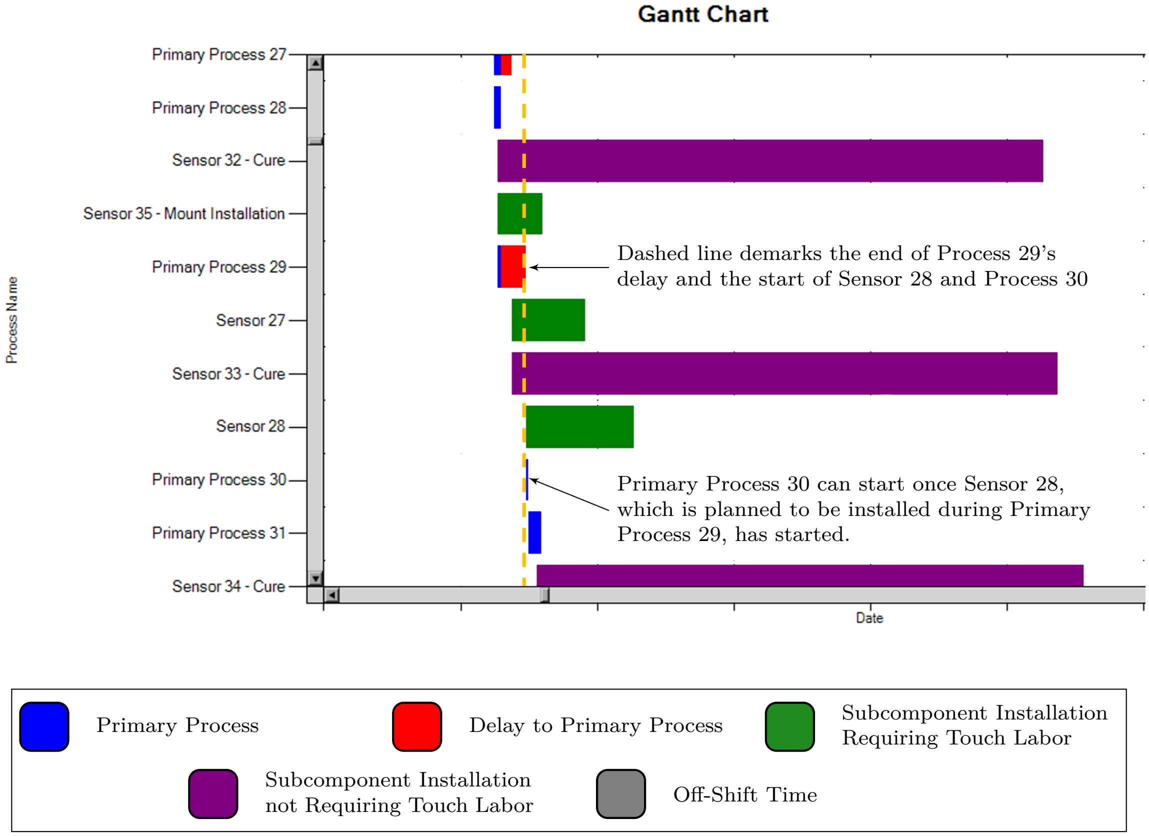
A2. If an incompatible production process has started before a manual installation, the manual task must wait until the incompatible process has completed to begin processing.

A3. Each manual installation requires a technician to be completed. Therefore, the processes must wait for a technician to be available and seized before beginning. The technicians respond to work requests based on a first come, first served basis. Hence, setting the number of technicians in the simulation controls the number of sensor installations tasks that can be completed in parallel.

A4. Manual installation processes can contain sub-processes that do not require a technician (e.g. time to allow an adhesive to cure before proceeding with the installation). In these cases, technicians are released to complete other touch labor tasks. Once the sub-process is completed, the next sub-process requiring a technician is added to the queue to request a technician.

The first assumption (A1) to investigate is that the model *starts* each sensor installation sequence defined to occur during a primary process before allowing the successive primary process to begin. This manifests itself as a delay in the current process, which ends when the last sensor installation has started. Figure 33 illustrates this assumption. In the figure, Sensor 27 and Sensor 28 are set to be installed during Primary Process 29. Due to the installation technicians being utilized to install other sensors, these 2 cannot start immediately. Once each are started, however, Primary Process 30 is started as well because it is compatible with each sensor installation. The dashed yellow line shows where Sensor 28 and Primary Process 30 are allowed to start, concluding the delay to Primary Process 29.

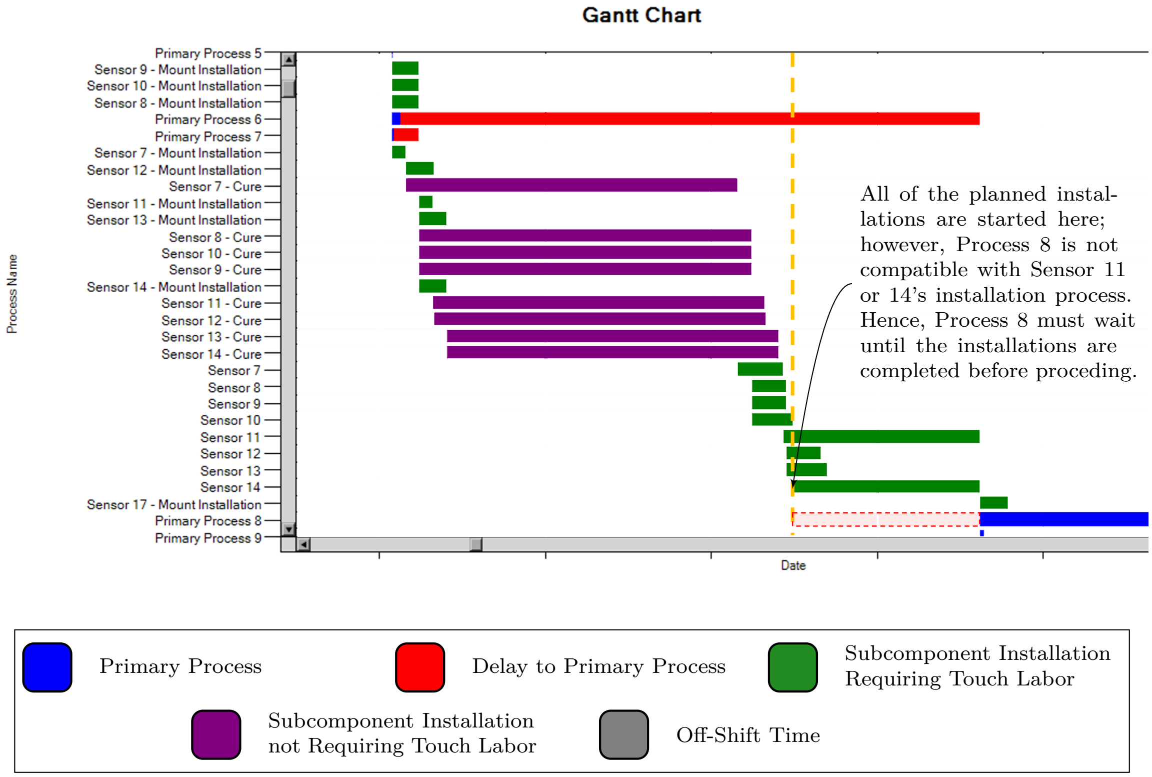
The second half of assumption A1 requires that, if a successive primary process is not compatible with the sensors currently being installed, it does not start until the installations are completed. An example of this can be seen in Figure 34. In this plan, Sensors 7–14 are planned to be installed during Primary Processes 6 and 7. Sensor 14 is the final sensor planned to be installed during Processes 6 and 7; therefore, if all of the installations were compatible with Process 8, then the process would have been allowed to begin upon starting Sensor 14's installation because all of the sequences from the previous process have started. However, because of the compatibility constraints, Process 8's start is delayed as indicated by the light red block. Hence, Figures 33 and 34 demonstrate the process taken to verify process assumption A1.



Dashed line demarks the end of Process 29's delay and the start of Sensor 28 and Process 30

Primary Process 30 can start once Sensor 28, which is planned to be installed during Primary Process 29, has started.

Figure 33: Gantt Chart to Illustrate Logic for Assumption A1



All of the planned installations are started here; however, Process 8 is not compatible with Sensor 11 or 14's installation process. Hence, Process 8 must wait until the installations are completed before proceding.

Figure 34: Gantt Chart Illustrating Logic to Delay the Start of Primary Processes due to Assumption A1

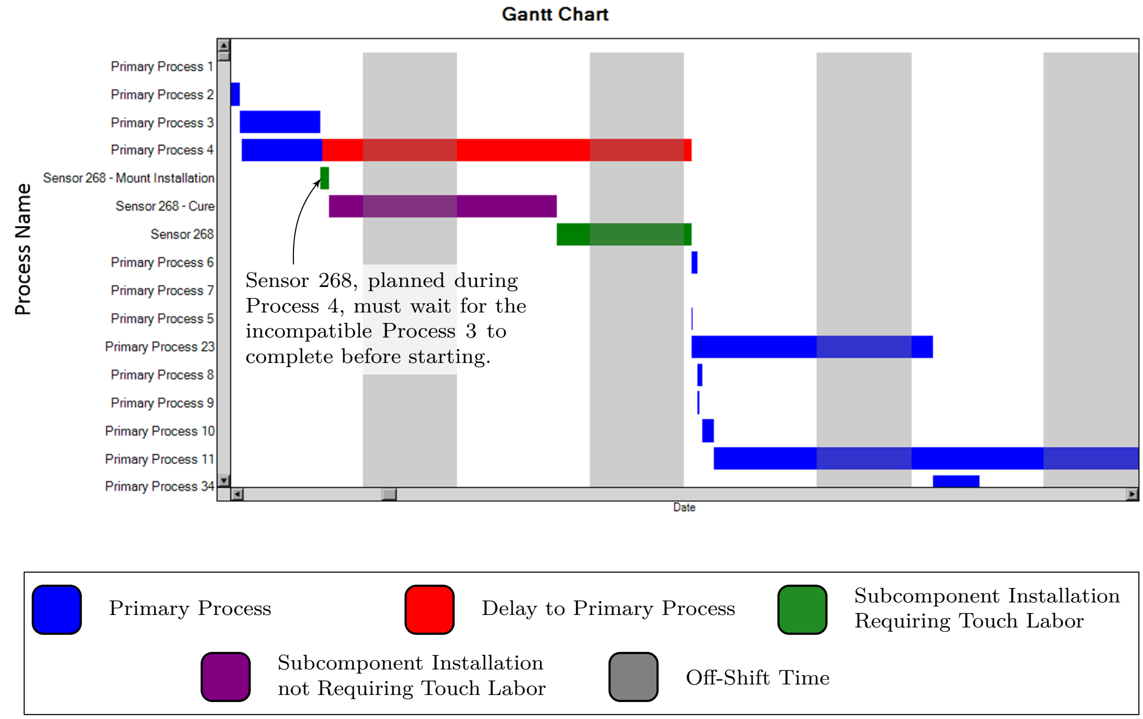


Figure 35: Gantt Chart Illustrating Logic to Delay the Start of Sensor Installations due to Assumption A2

Assumption A2 states that if an on-going primary production process is incompatible with a sensor installation attempting to begin, the installation must be delayed until the incompatible process is completed. Figure 35 demonstrates this logic. Sensor 268, which is planned during Primary Process 4, is incompatible with Primary Process 3. Hence, when Process 4 begins slightly after Process 3, Sensor 268 is forced to wait for Process 3 to end before beginning processing. Then, because Sensor 268 is also incompatible with Process 6, the primary process is required to wait until the entire Sensor 268 installation sequence is completed.

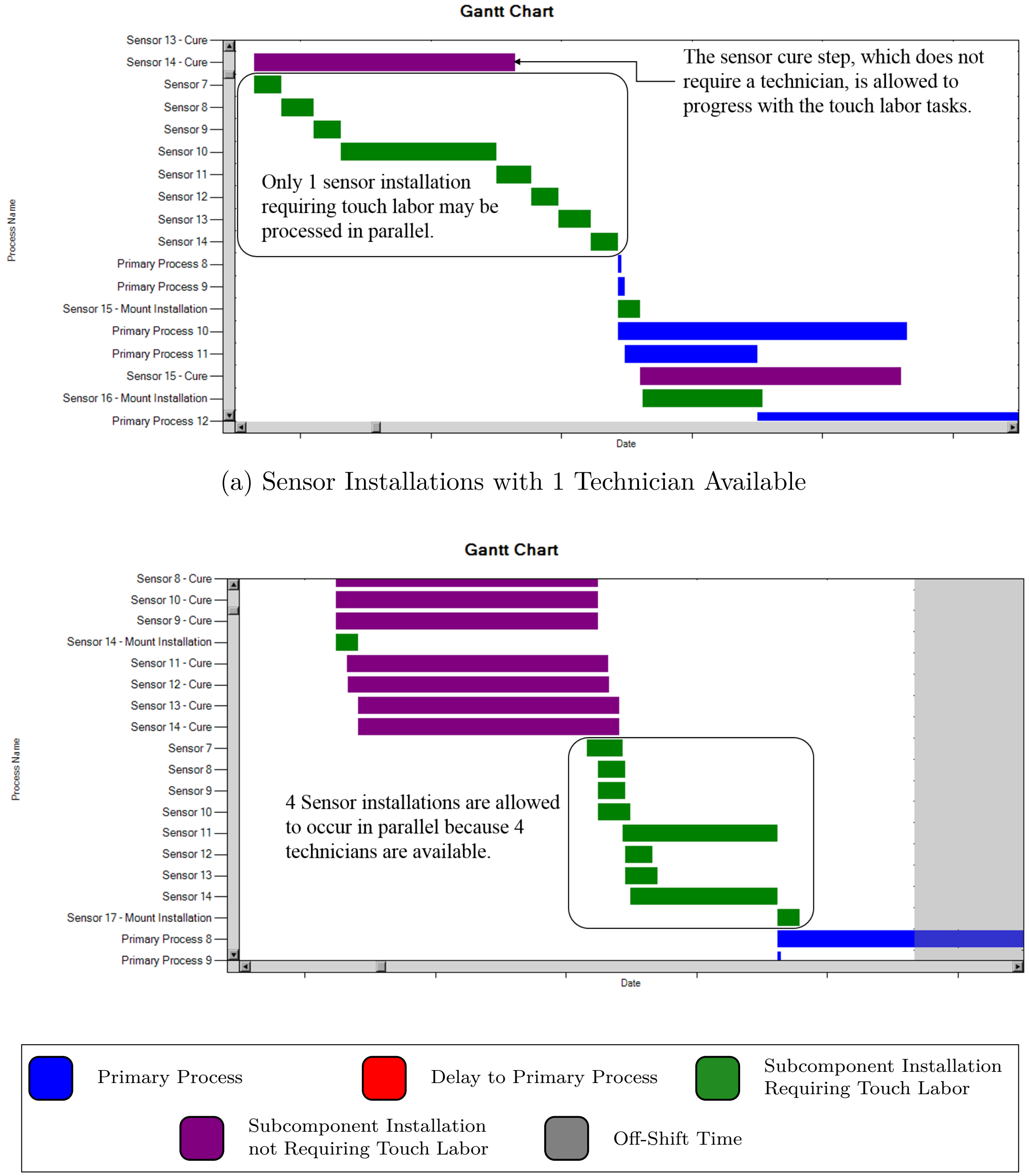
Assumption A3 details that the model must limit the number of possible parallel sensor installation tasks to the number of technicians made available. The results of this assumption can be seen in Figure 36. The two Gantt charts illustrated in this figure are generated using the same sensor installation plan. Figure 36a has only 1 installation technician available to install sensors, while the chart in Figure 36b has 4 technicians in the system. With the extra technicians, the plan illustrated in Figure 36b is able to complete 4 sensor installations that require touch labor (denoted by the green bars) in parallel. Figure 36 also illustrates that the processes that do not require touch labor (shown in purple) are not impacted by the number of technicians and do not affect the number of parallel touch labor processes allowed. Similar results are seen throughout the full verification process.

The final model assumption to be examined is A4, which states that the sensor installations requiring touch labor must follow the shift schedule of their assigned primary production process. This scenario is demonstrated in Figure 37. In the figure, off-shift time is represented by the shaded gray area. The processes to install Sensors 70, 71, and the mount for Sensor 73 are completed after the factory is brought back on shift. The cure process for Sensor 73 is allowed to complete during the off-shift time because it does not require any touch labor. Once the production system comes back on-line, the final installation step for Sensor 72 is allowed to be completed. Hence, assumption A4 is properly observed within the production model.

This section has demonstrated that the production logic is properly observed in the model. The following sections serve to verify that by following this logic, the expected overall trends are observed in the model.

5.1.1.3 Concurrent Installations Allowed - Experimental Results

The system-level impact of varying the number of concurrent sensor installations allowed (i.e. number of technicians available to perform installations) can be seen in Figure 38 and is summarized in Table 14. The histograms present the deterministic process time for the 1000 population members from generation 250 of the NSGA-II algorithm optimization run. The time is normalized by the baseline process time (the process time without sensor installations).



The sensor cure step, which does not require a technician, is allowed to progress with the touch labor tasks.

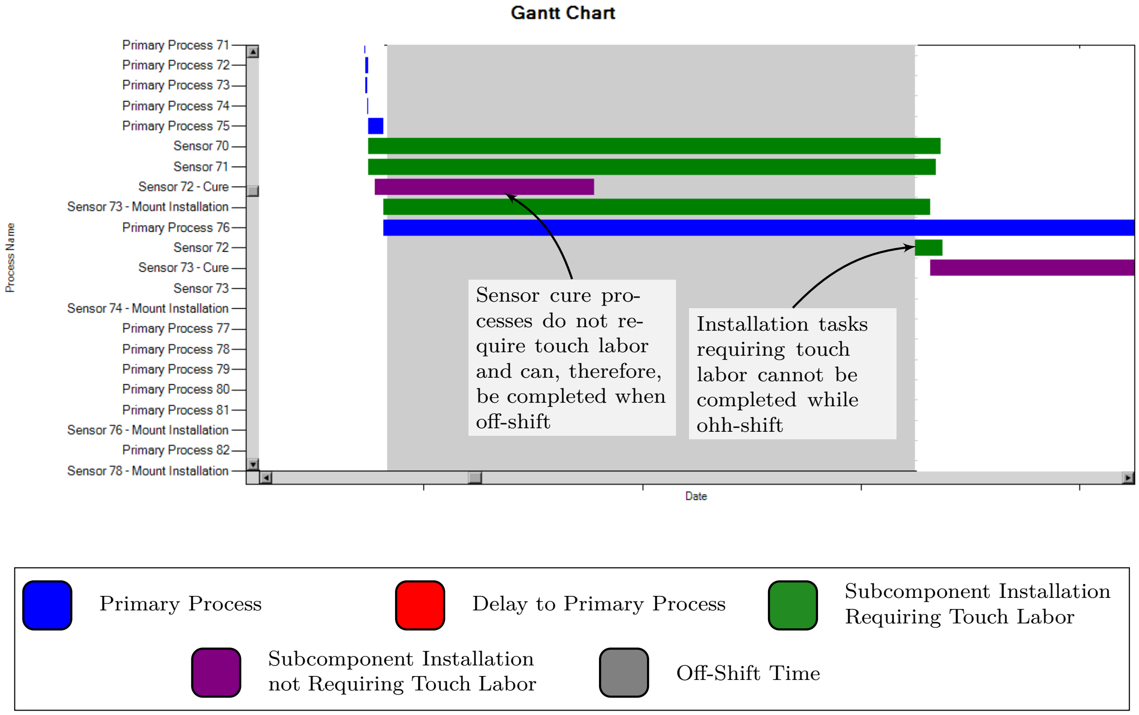
Only 1 sensor installation requiring touch labor may be processed in parallel.

(a) Sensor Installations with 1 Technician Available

4 Sensor installations are allowed to occur in parallel because 4 technicians are available.

(b) Sensor Installations with 4 Technician Available

Figure 36: Comparison of Parallel Sensor Installations with 1 and 4 Technicians Available



Sensor cure processes do not require touch labor and can, therefore, be completed when off-shift

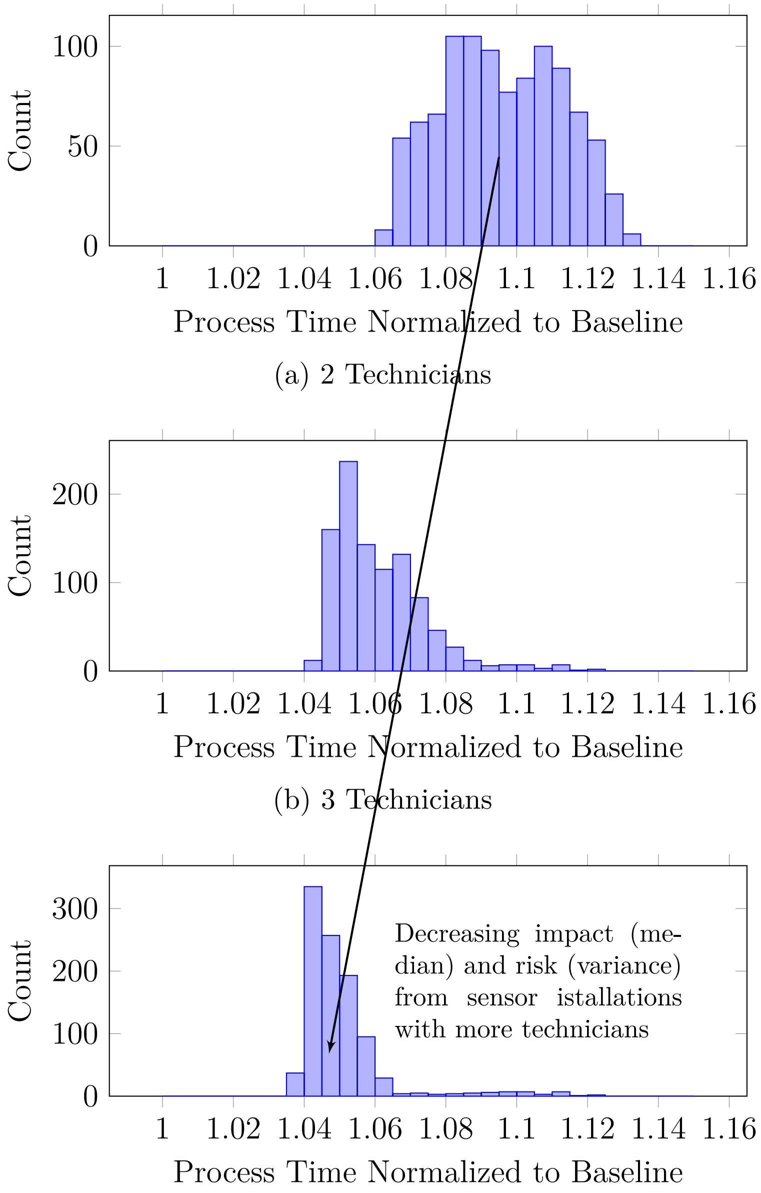
Installation tasks requiring touch labor cannot be completed while ohh-shift

Figure 37: Gantt Chart Illustrating Parallel Task Shift Schedule

The results presented in Figure 38 illustrate that, generally, the model is responding as expected to varying the number of concurrent installations allowed. Increased parallelization is expected to lead to fewer delays, which is seen by the median shifting to the left as more technicians are added. Additionally, an increase in parallelization adds flexibility to the schedule. This flexibility is expected to enable the simulation to perform well (in terms of process time) for a wider range of scenarios; therefore, the range of process times recorded should narrow with more technicians. This is also seen in Figure 38 as the standard deviation of the results decreases when more technicians are added to the system.

Table 14: Summary of Concurrent Sensor Installations' Impact on Process Time

Scenario	Median	Standard Deviation
2 Technicians	1.095	0.017
3 Technicians	1.058	0.013
4 Technicians	1.047	0.012



(a) 2 Technicians

(b) 3 Technicians

(c) 4 Technicians

Decreasing impact (median) and risk (variance) from sensor istallations with more technicians

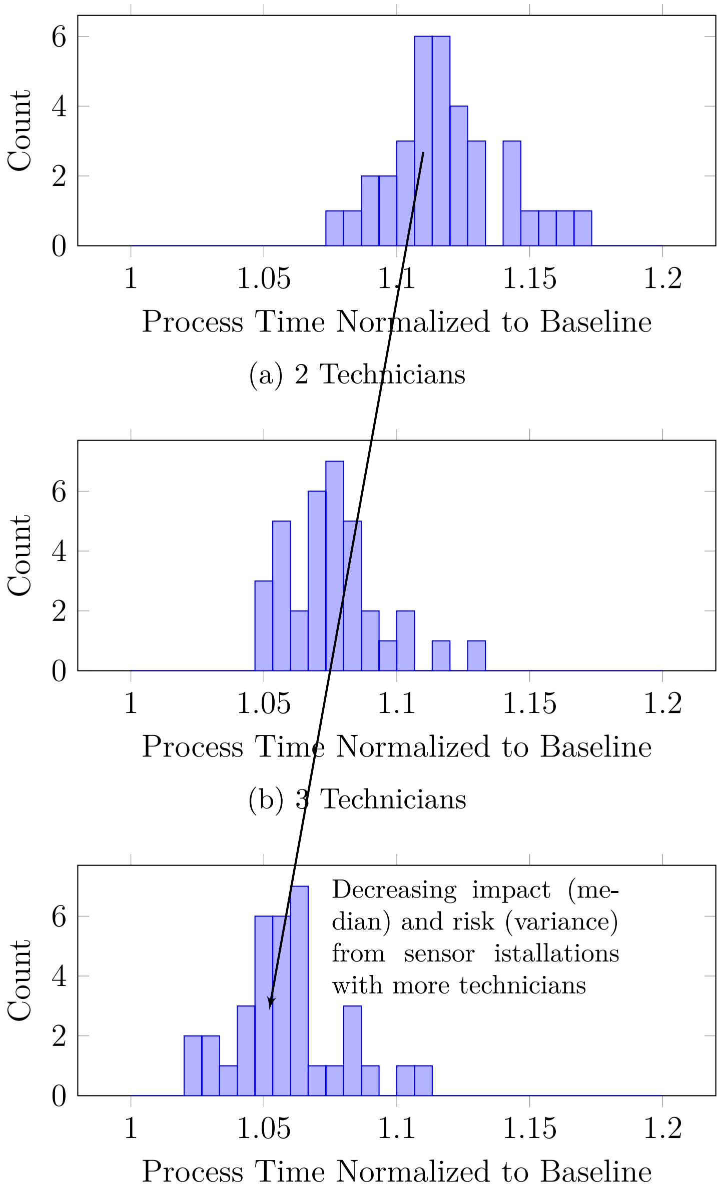
Figure 38: Comparison of Simulation Results for 2–4 Concurrent Sensor Installations Allowed

Similar trends are expected when investigating the process time distributions for a single case with uncertainty. Figure 39 shows the process times for a single sensor installation plan run through the stochastic simulation. As anticipated, the additional technicians shifts the distribution to the left, which indicates fewer delays from sensor installations. From these results, the model logic is shown to appropriately capture the impact of varying the number of sensor installation technicians available.

5.1.1.4 Compatibility Assumptions Impact - Experimental Results

The compatibility matrix provides the information for the sensor installation logic to appropriately guide the simulation's execution. Hence, making the compatibility assumptions more stringent (e.g. by adding additional incompatible process/sensor pairs) should increase the process time and help to verify the interaction between the constraints and the installation logic. To explore this behavior, half of the baseline compatibility matrix's *compatible* installations are changed to be *incompatible*. A sensor installation plan is then created for this more stringent compatibility matrix. The simulation is then run with the same sensor installation plan for both the updated, more stringent compatibility matrix and the baseline matrix. Results for scenarios with 2–4 sensor installation technicians are presented in Figure 40.

As shown in Figure 40, the scenarios using the less constraining, baseline compatibility matrix require significantly less time to complete the plan. Even though the installation plan was developed for the more constraining matrix (so that all sensor installations are planned to occur within a compatible process), the process time is still significantly impacted. This indicates that parallel processes, which may now be incompatible, are causing more delays in the more constrained cases. Furthermore, more compatible processes are likely followed by incompatible processes, which prevents the model from moving forward until the current sensor installations are completed. Overall, modifying the compatibility assumptions lead to the expected



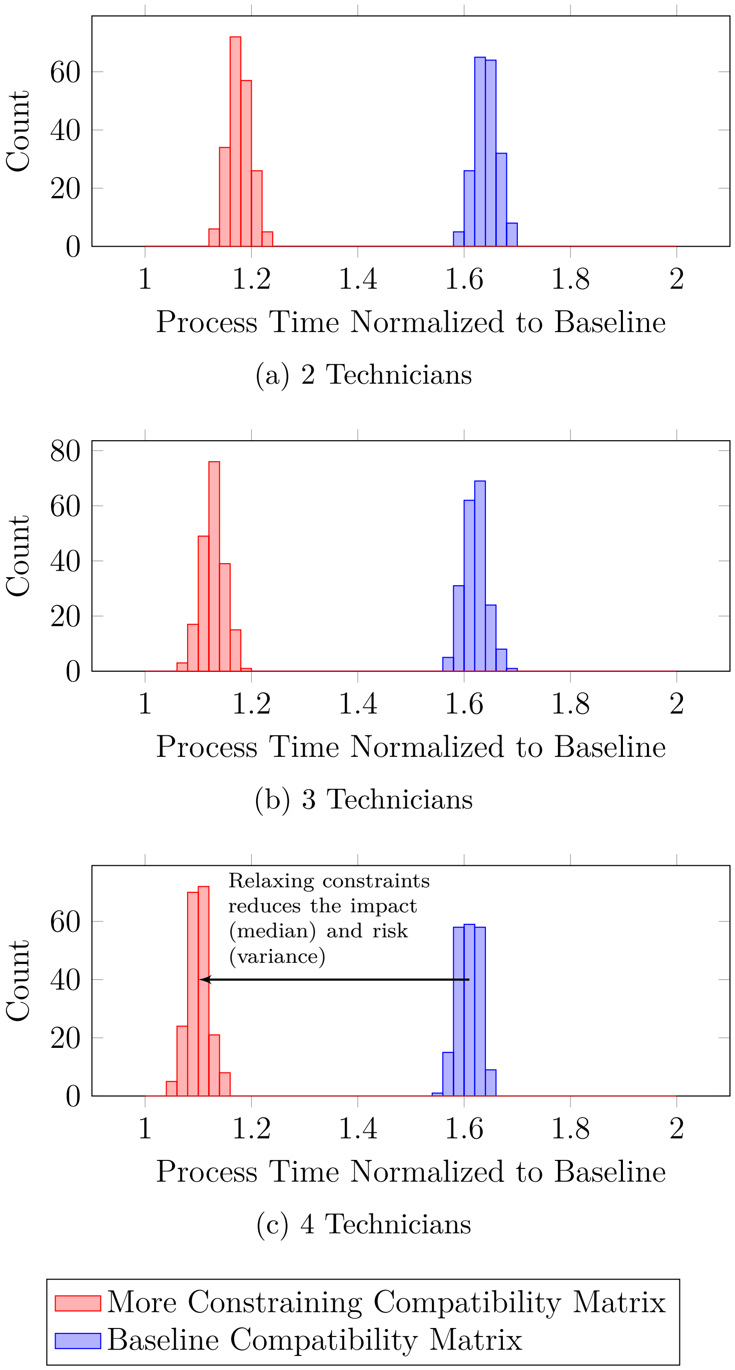
(a) 2 Technicians

(b) 3 Technicians

Decreasing impact (median) and risk (variance) from sensor istallations with more technicians

(c) 4 Technicians

Figure 39: Comparison of Process Time for a Single Stochastic Simulation Run with 2–4 Concurrent Sensor Installations Allowed



(a) 2 Technicians

(b) 3 Technicians

(c) 4 Technicians

Figure 40: Comparison of Simulation Results for the Baseline and More Stringent Compatibility Matrix

changes in the model behavior and help verify that the simulation is ready for integration with the optimization routine.

5.1.1.5 Results Discussion

The results presented throughout this section have demonstrated that the modeling approach is able to sufficiently represent and assess the manual sensor installations of interest. The model respects the assumptions developed to both effectively model the process to install the sensors and estimate their impact for integration with an optimization routine. Hence, Experiment 1.1 is successful and supports the hypothesis. The verified model is next utilized to carry out Experiment 1.2 as discussed in the following section.

5.1.2 Experiment 1.2: Optimization's Ability to Improve System Performance

With the modeling strategy shown to provide the capability to estimate the metrics of interest in an efficient manner, Experiment 1.2 is conducted to investigate the appropriateness of the optimization routine. Following the outline in Figure 29, this experiment is designed to solely test the link between a metaheuristic optimization routine and the simulation model investigated in the previous section to evaluate Hypothesis 1.2:

> **Hypothesis 1.2**: If a metaheuristic optimization routine is linked to the developed discrete-event simulation schedule model, then installation plans with improved performance can be efficiently identified.

The goal is to show that a metaheuristic algorithm is able to improve the production plan by utilizing information from the simulation model to improve the identified metrics of interest. As previously mentioned, this experiment focuses solely on finding a good deterministic solution to the problem and does not consider quality robustness. The impact of the uncertainty analysis is investigated in Experiment 1.3. This experiment also only investigates results from a single metaheuristic algorithm (the NSGA-II). The genetic algorithm is chosen for this initial experiment because GAs are shown to thoroughly explore Pareto frontiers and do not require knowledge of the inner workings of a problem to be effective [66]. Generally, the other metaheuristic algorithms of interest (e.g. Simulated annealing, Neighborhood searches, etc.) require lower level heuristics to be efficient (while a genetic algorithm typically does not); as such, their investigation is reserved for Experiment 2.2, which examines the impact of strategies designed to speed the optimization. Therefore, results for the NSGA-II, whose implementation is described in Section 4.3.4.1, are presented in the following section.

5.1.2.1 Experiment 1.2 Results

The results presented throughout this section are generated over 1686 generations of the NSGA-II, which is composed of approximately 198,000 individual function calls. After this number of generations, the non-dominated set of solutions has not seen meaningful improvement for a few hundred generations. Of the 198,000 points generated, ~7,500 are Pareto optimal across the 4 objective functions for 3 installation technician parameter settings. An initial investigation of the model showed that utilizing only a single installation technician is not sufficient to produce quality results. Hence, the case with 1 installation technician is ignored in the optimization.

Results for this experiment are shown in Figures 41 and 42 for selected generations. Figure 41 shows the algorithm's progression for the scenarios utilizing 2 technicians, while Figure 42 represents the cases with 4 technicians. The figures present results for the four responses of interest (process time, slack time for all sensors and only critical sensors, and the neighboring sensor installation metric) generated from deterministic runs of the model. Each population has 1000 members, but only the Pareto efficient members from each generation are plotted. The results are normalized; the slack time and neighboring installation metric are normalized to the best result identified throughout the optimization (largest slack time and smallest neighboring installation metric), while the process time is divided by the baseline time with no sensor installations (to illustrate the overall impact of the sensor installations).

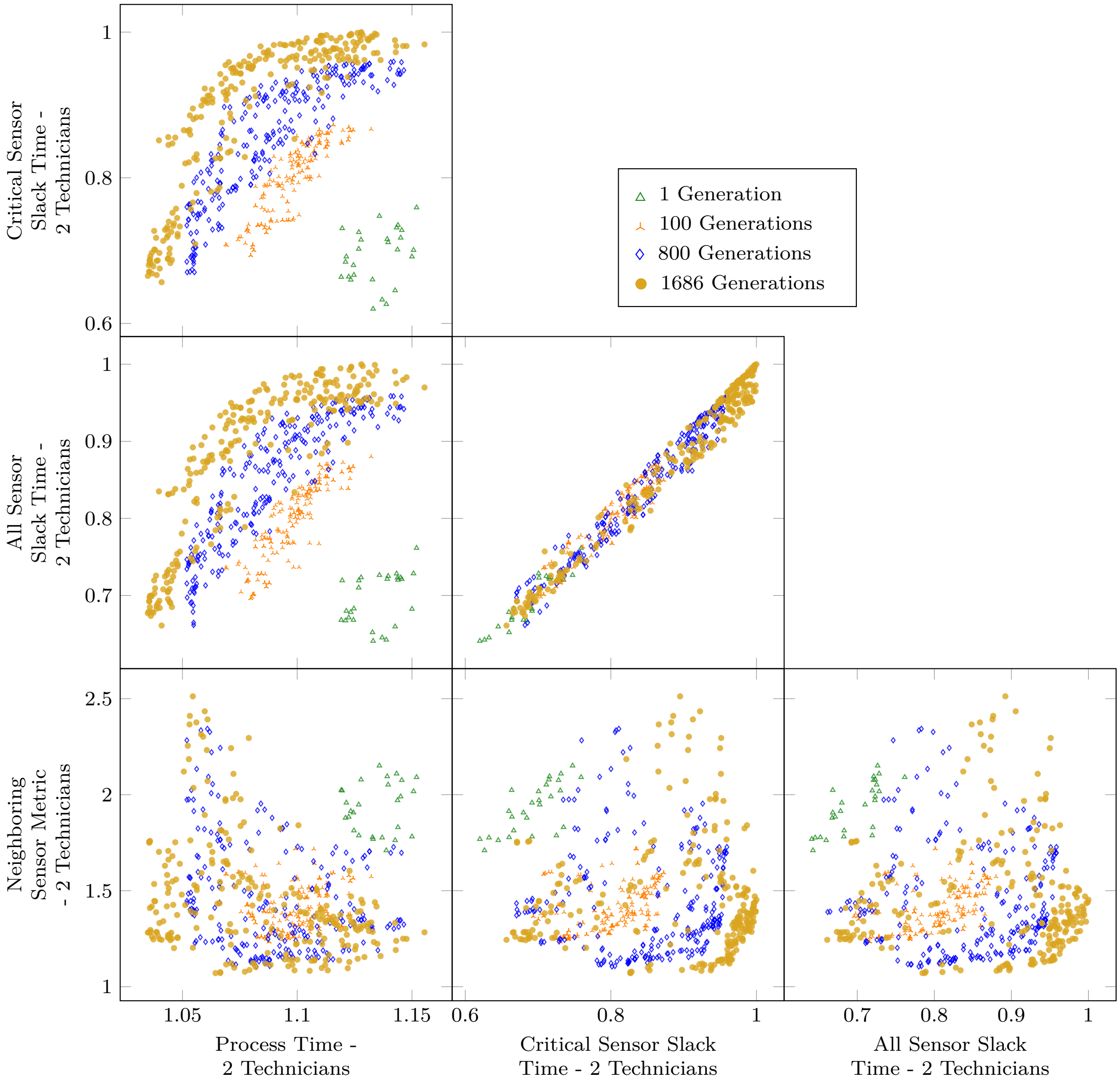


Figure 41: Normalized NSGA-II Progression for Deterministic Optimization with 2 Technicians

The results shown in Figures 41 and 42 demonstrate that the optimization algorithm is able to utilize the model to move towards the Pareto frontier of well-performing (according to the defined metrics) installation plans. For all technician

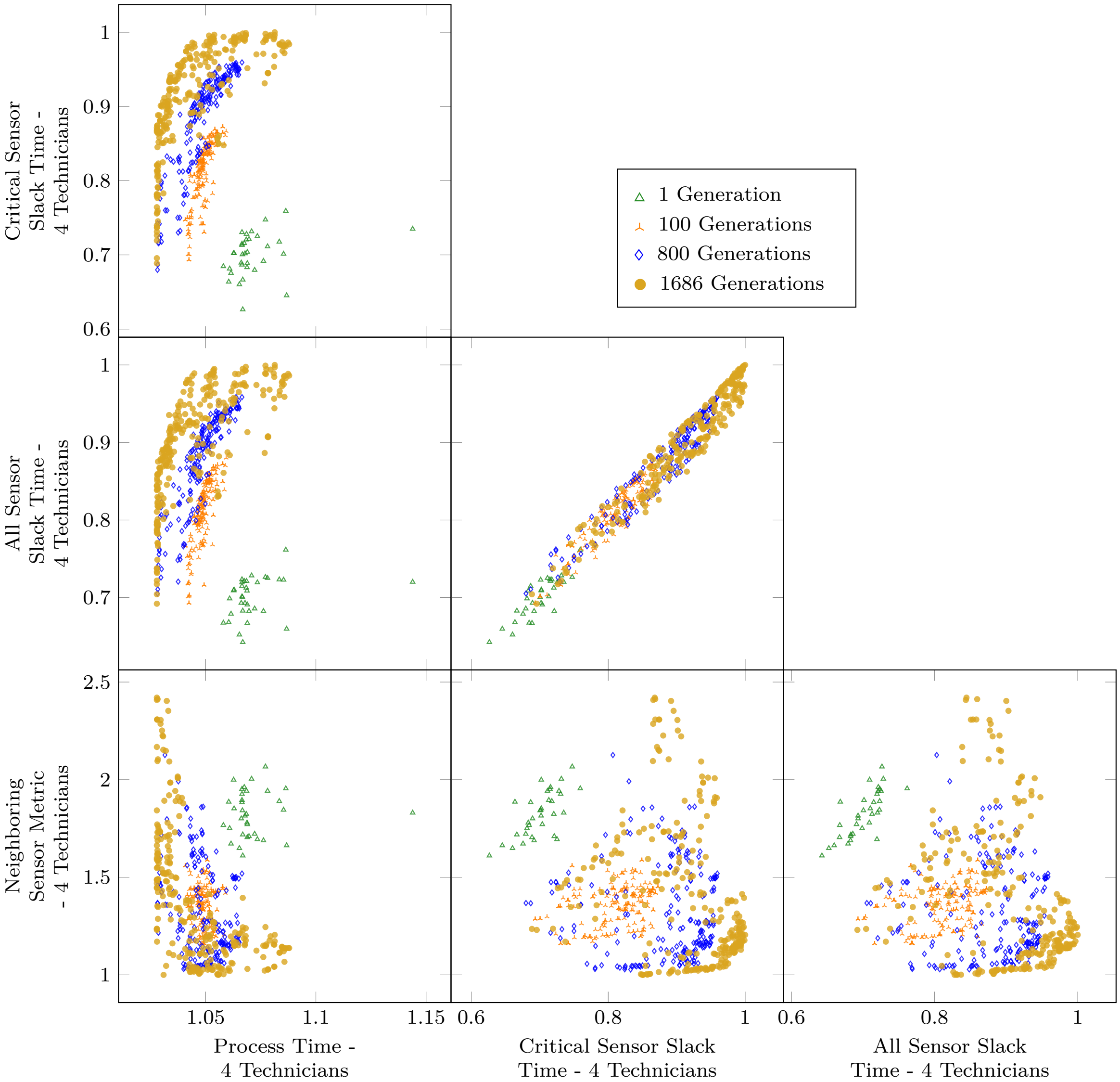


Figure 42: Normalized NSGA-II Progression for Deterministic Optimization with 4 Technicians

levels, the Pareto frontier is improved in all objective functions as the algorithm progresses. For the 2-technician scenario, the improvement is more pronounced than the 4-technician scenario. With the additional technicians, the 4-technician scenario can better handle additional sensor installations planned in parallel. This leads to an increased number of plans with low values for process time. However, when only using 2 technicians, the simulation could experience more significant delays when compared to scenarios using 3 or 4 technicians. For example, a strategy to increase the slack time of an installation plan could be to plan more sensors in parallel earlier in the process flow. The scenario with 3 or 4 technicians may be able to accommodate this plan without incurring delays; however, the 2-technician scenario may not be able to get all of the sensors installed within the required process time to avoid delaying the schedule.

Figure 43 shows the final set of Pareto optimal solutions for each of the 2–4 technician scenarios. Increasing the number of technicians available not only improves the process times, but it also leads to less compromise between slack time, process time, and the neighboring sensor metric. This is again because the additional technicians allow the optimization algorithm to plan more sensor installations closer together without encountering a delay. Therefore, the schedule can be front-loaded with more parallel sensor installations to increase their slack without causing delays. This further indicates that the NSGA-II effectively explored the design space.

Understanding the objective functions with the most room for improvement helps to improve the understanding of the problem at hand. To this end, Figure 44 shows the best values obtained from each generation of the genetic algorithm. Figure 45 presents the median values of each objective by generation to provide an indication of the population's performance. The values are normalized as discussed previously. The figures show that the majority of improvement to the process time occurs in the first 100–300 generations. This is expected as the problem is not overly constrained.

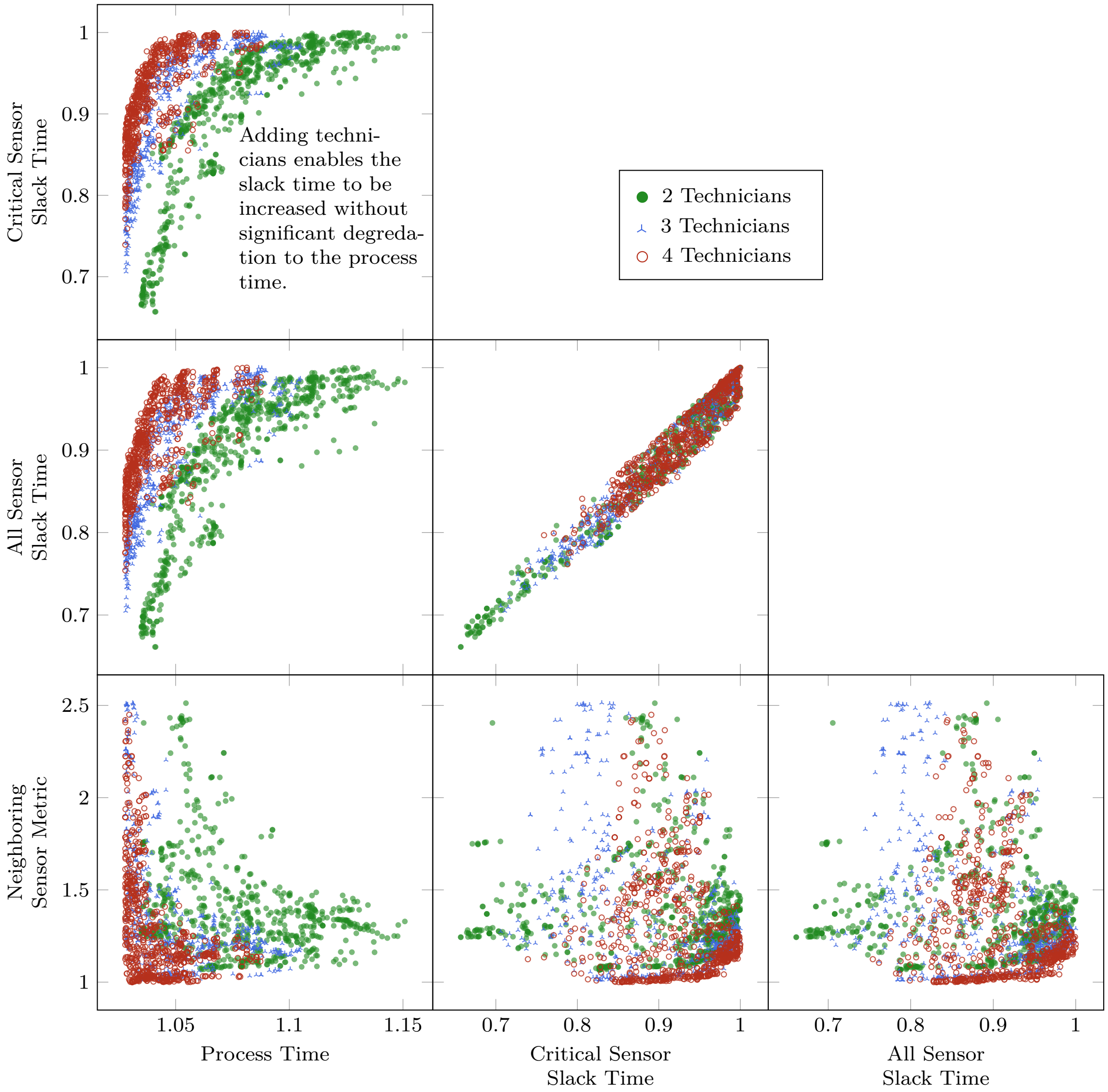


Figure 43: Normalized NSGA-II Pareto Efficient Population Comparison for 2, 3, and 4 Technician Scenarios

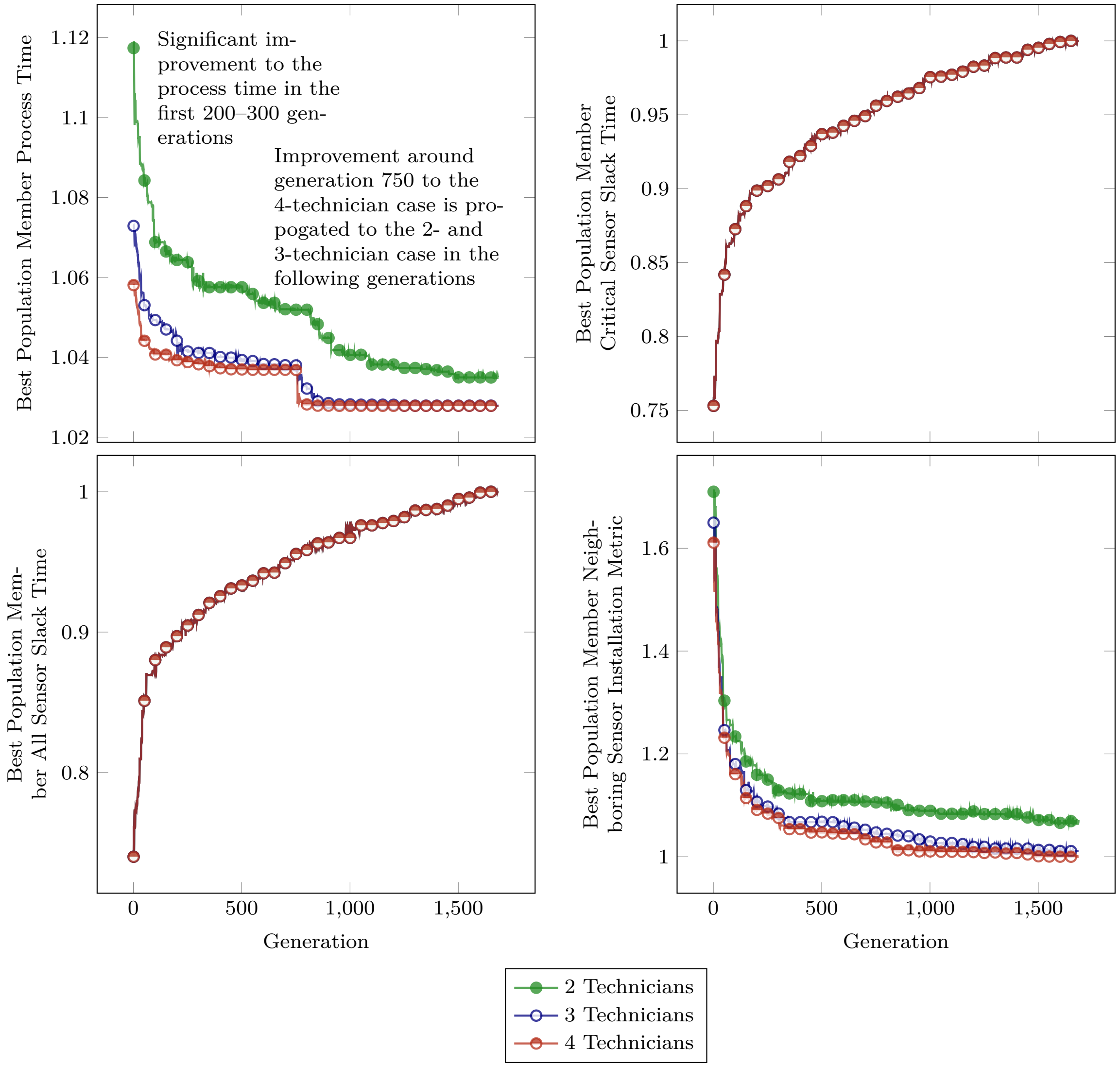


Figure 44: Best Values for the Four Objectives of Interest Found at Each Generation of the Deterministic NSGA-II

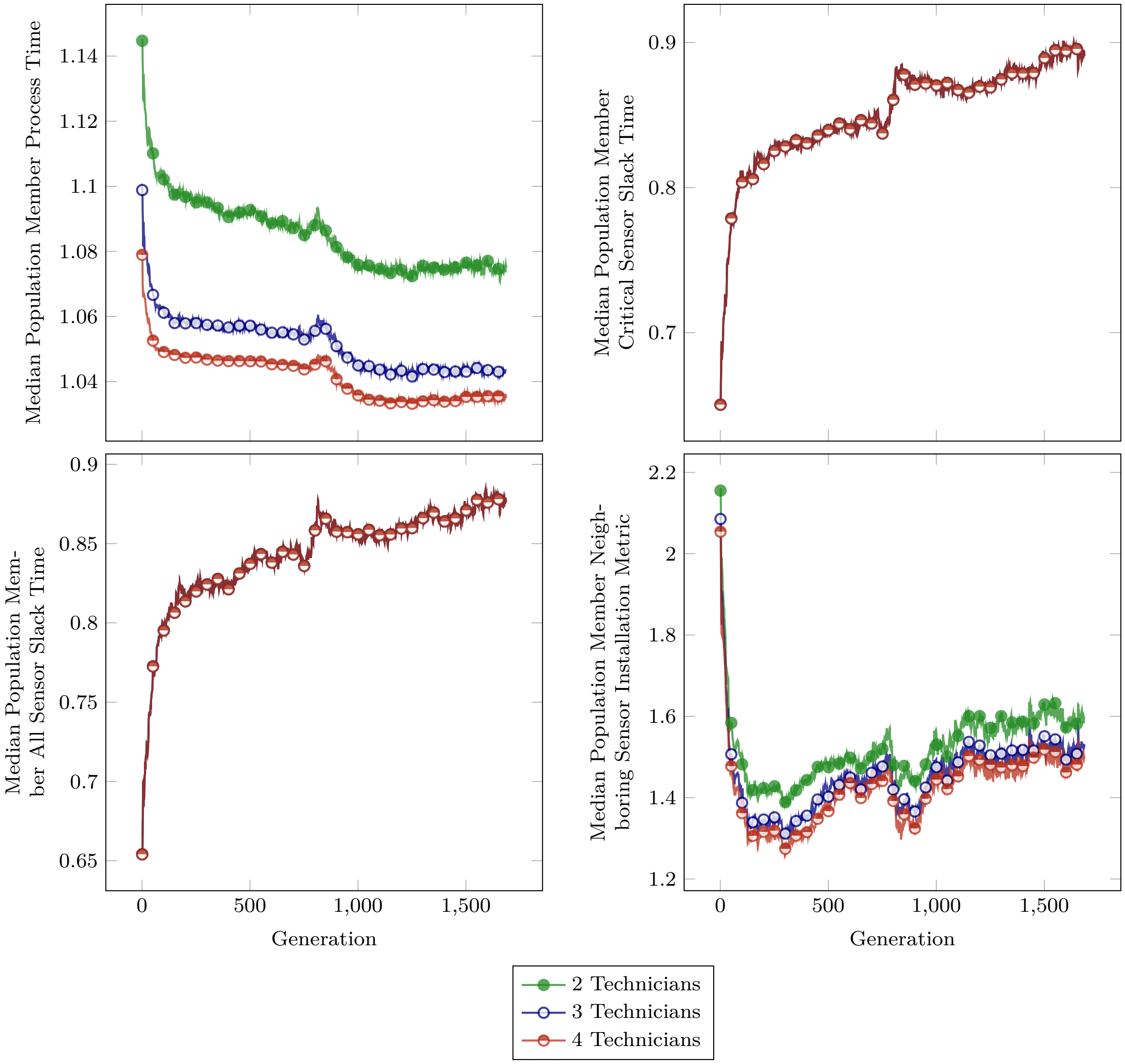


Figure 45: Median Values for the Four Objectives of Interest Found at Each Generation of the Deterministic NSGA-II

As seen in Figure 44, a sharp decrease in process time is observed in the 4-technician scenario near generation 750. After this initial decrease in the best values found, the median population value (Figure 45) begins to decrease again, which indicates that well-performing portions of the newly identified plan are being propagated through the population.

The other metrics show significant improvement to the best scenario as the algorithm progresses even after the process time stops improving around generation 1000. This indicates that, while the process time may not be improved by a significant percentage through the later stages of the optimization run, there are significant opportunities to reduce risk in the plan by better planning slack and ensuring that neighboring sensors are installed closer together. The median values for slack time also continue to increase as presented in Figure 45. As previously discussed, the slack time can be almost continuously improved by moving all sensor installations earlier in the flow; hence, the algorithm actually begins to sacrifice improvements to the neighboring sensor installation metric to explore points with more slack time.

In addition to the qualitative investigations of the optimization results, the quantitative convergence metrics discussed in Section 4.6.6.2 are also evaluated. Figure 46 presents the relative improvement, consolidation ratio, and ratio of Pareto points for the deterministic NSGA-II. The consolidation ratio was calculated with a separation of 10 generations (i.e. the results show the ratio of Pareto optimal points from generation $i - 10$ that are still Pareto efficient in generation i).

These 3 metrics all indicate that the NSGA-II drives the optimization towards an improved set of sensor installation plans. The increase of the relative improvement and consolidation ratio indicate that the algorithm quickly improves during the first 100–200 generations and then slowly proceeds towards convergence. The ratio of Pareto points also quickly improves from about 7% in the random initial population to 81% in generation 100. This shows that the algorithm is outperforming a random

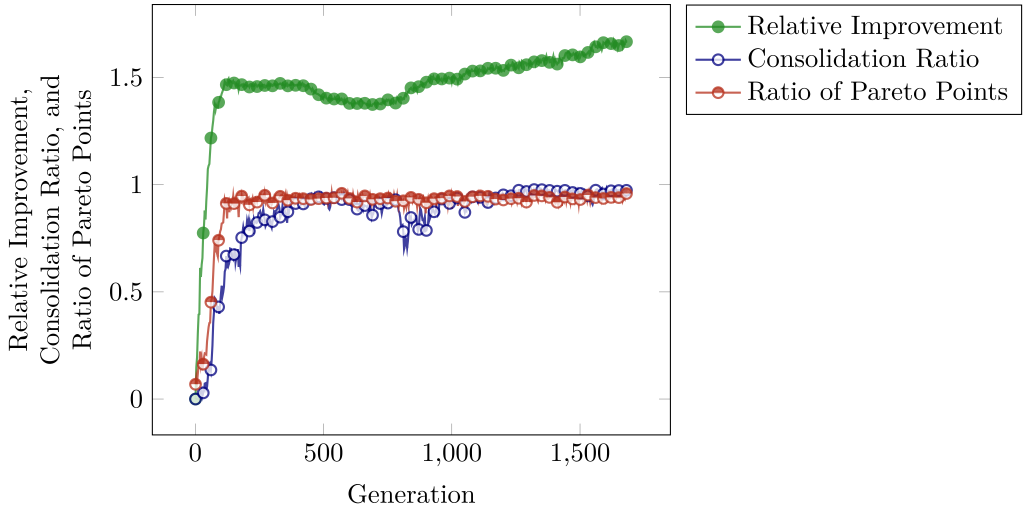


Figure 46: Optimization Algorithm Convergence Metrics for the Deterministic NSGA-II

search by more efficiently producing Pareto optimal points. The following section discusses the results' implications for Hypothesis 1.2.

5.1.2.2 Experiment 1.2 Discussion

The results presented throughout this section demonstrate that the genetic algorithm is able to simultaneously improve the 4 primary objective functions of interest. Furthermore, by including cases with 2–4 installation technicians as separate objective functions within the optimization routine, the algorithm is able to simultaneously identify well-performing plans for each staffing level. Finally, the qualitative multi-objective algorithm performance and convergence metrics demonstrate that the algorithm is purposefully improving the sensor installation plan to identify a family of well-performing solutions.

Hypothesis 1.2 is thus strongly supported by the results from the experiment. The DES model provided adequate objective function evaluations to allow the linked genetic algorithm to improve the schedule model. The optimized results not only reduced the total amount of process time required to complete the schedule, but also reduced risk by adding slack to the process and working to ensure that neighboring sensors are installed close together. The following section furthers this investigation by examining how the inclusion of quality robustness evaluations in the optimization routine informs the results to better reduce risk associated with the sensor installation tasks.

5.1.3 Experiment 1.3: Impact of Robust Optimization Considerations Results

Continuing along Experiment 1's buildup demonstrated in Figure 29, this section investigates the impact of considering quality robustness directly in the optimization routine. The optimization results presented throughout this section were generated using the stochastic version of the production model as discussed in Section 4.3.4.1. The purpose of this section is to investigate the benefits of directly considering these sources of uncertainty while optimizing the sensor installation plan to evaluate the validity of Hypothesis 1.3:

Hypothesis 1.3: If the optimization routine and model can estimate robustness related responses (quality robustness) and support multi-objective optimization, then the methodology will be capable of finding robust schedules.

Throughout, the following evaluation criteria are investigated:

Quality Robustness: Ability to identify schedules with reduced process time and risk (measured by the 80th quantile of the process time).

Improvement to Other Risk Metrics: Ability to provide a family of solutions that work to reduce risk associated with sensor installations.

Comparison to Deterministic Results: Ability to differentiate between deterministically optimized schedules and schedules optimized that include quality robustness (e.g. the 80th quantile of the process time). The question one tries to answer is: Do the results from the stochastic optimization routine have lower risk than those identified through the deterministic optimization runs?

The results are generated by running the simulation model for 35 replications (shown in Section 4.3.5 to provide a sufficiently accurate estimate of the objective functions). The median and 80$^{\text{th}}$ quantile for the process time is then calculated (the slack time and neighboring sensor installation metrics are not significantly impacted by the uncertainty). The 80$^{\text{th}}$ quantile is selected for the risk metric because risk-averse entities are known to pursue projects with an 80% probability of remaining within cost and schedule [87]. These calculations then replace the deterministic process time utilized in the deterministic formulation.

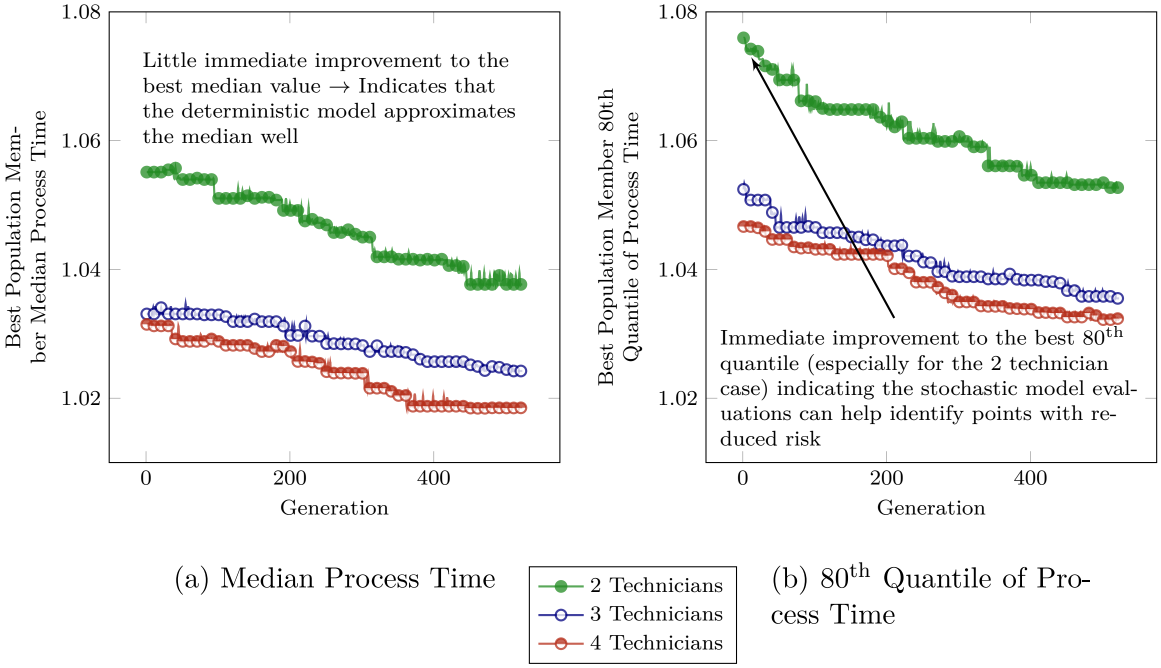
Without the availability of a large cluster of computing nodes, evaluating 35 replications for a single case significantly increases the computation time for each generation. Simio's Replication Runner is used to distribute evaluations across a network of desktop computers that support 64 concurrent replications. A single generation of the NSGA-II with the deterministic model can be evaluated in about 2 minutes with these resources; when evaluating 35 replications per case, this time increases to over 40 minutes per generation.

This computation time is too large to reasonably run anywhere close to the number of generations evaluated using the deterministic model. Therefore, generation 500 from the deterministic optimization is used to seed the optimization run with quality robustness. At this point in the deterministic run, the process time has not significantly improved for over 100 generations. The optimization with quality robustness is then run for an additional 450 generations. Results from this optimization run are discussed in the following sections.

5.1.3.1 Quality Robustness Improvement

The first evaluation criterion investigates the algorithm's ability to improve the robustness of the sensor installation plan. Figures 47 and 48 present the improvement of the median and 80$^{\text{th}}$ quantile of the process time over each generation of the stochastic

NSGA-II run. Figure 47 presents the best value found per generation, while Figure 48 presents the median value of each generation.

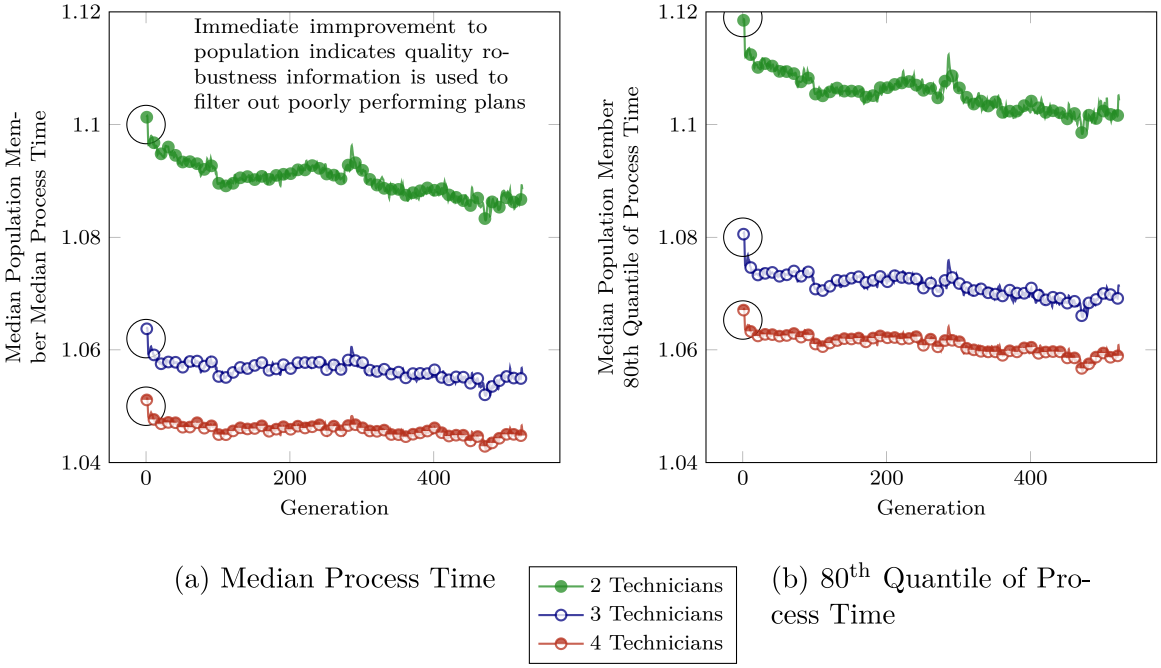


(a) Median Process Time

(b) 80th Quantile of Process Time

Figure 47: Best Values for the Process Time Observed for Each Generation of the Stochastic NSGA-II

Figure 47a shows that there is not an immediate improvement to the best median process time found by the optimization algorithm. This indicates that the deterministic objective functions provide a good indication of the median performance, at least for the highly performing scenarios. Figure 47b, however, shows a steady improvement to the best value of the 80th quantile of process time for the 2- and 3-technician scenarios through the first approximately 50 generations. The 4-technician scenario, however, does not see as much of an improvement possibly because the additional technicians work to increase the amount of parallel work possible. By increasing the number of parallel sensor installations, the 4-technician scenario can work to mitigate the impact of a sensor installation failing or having multiple installations take longer than the primary process during which they are planned. Hence, by including the quality robustness metrics, the algorithm appears to be able to quickly identify plans



(a) Median Process Time

(b) 80th Quantile of Process Time

Figure 48: Median Values for the Process Time Observed for Each Generation of the Stochastic NSGA-II

with reduced risk using fewer resources.

Figure 48 shows that the median of the population members' median and 80th quantile of process time is quickly improving across the board. This shows evidence that some population members identified through the deterministic optimization, which may have had strong deterministic performance, do not perform as well when considering stochasticity. Figure 49, which presents the normalized distribution created by subtracting the deterministic process time from the median process time for generation 500 of the NSGA-II, further supports this claim. While many instances see similar values for the median and deterministic process times, there are a number of cases with significant differences. Furthermore, as expected, the distribution is slightly shifted to the positive direction, which indicates that the deterministic process time is commonly overly optimistic. As such, these points that are shown to be dominated when evaluating quality robustness are quickly identified and removed from the population.

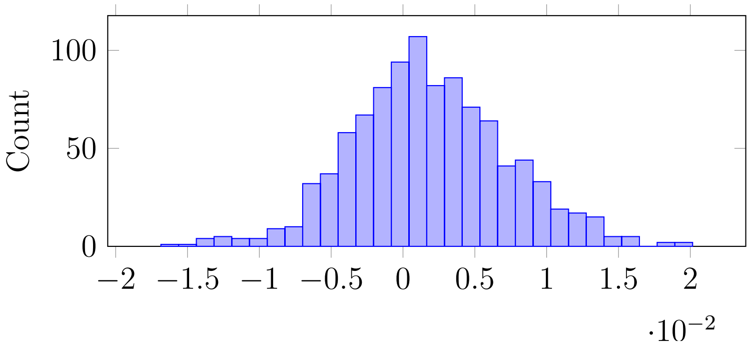


Figure 49: Normalized Difference between the Deterministic and Median Process Times for Generation 500 of the Deterministic Optimization (Positive Values Indicate that the Stochastic Median is Greater than the Deterministic Value of Process Time)

In both Figure 47 and 48, the optimization algorithm is shown to improve the quality robustness of the best population member and the population as a whole. Figure 47b shows a more significant decrease in the best 80th quantile of process time than the median, which indicates that the algorithm has successfully used the 80th quantile objective function to find plans that use fewer resources but still have reduced risk. While this result is encouraging, the remaining evaluation criteria must be investigated to fully test Hypothesis 1.3.

5.1.3.2 Varied Family of Plans to Increase Robustness

The second evaluation criterion requires that the optimization considering quality robustness provides a family of Pareto optimal solutions that concurrently improve the deterministic risk metrics (slack time and the neighboring sensor installation metric). In other words, the algorithm should explore regions with improved slack time along with higher robustness. Figure 50 presents the results from the optimization considering robustness. The scatterplot matrix displays information about both the median and 80th quantile of the process time in addition to the slack time and neighboring sensor installation metric.

Figure 50 demonstrates that the algorithm is able to identify a varied set of points

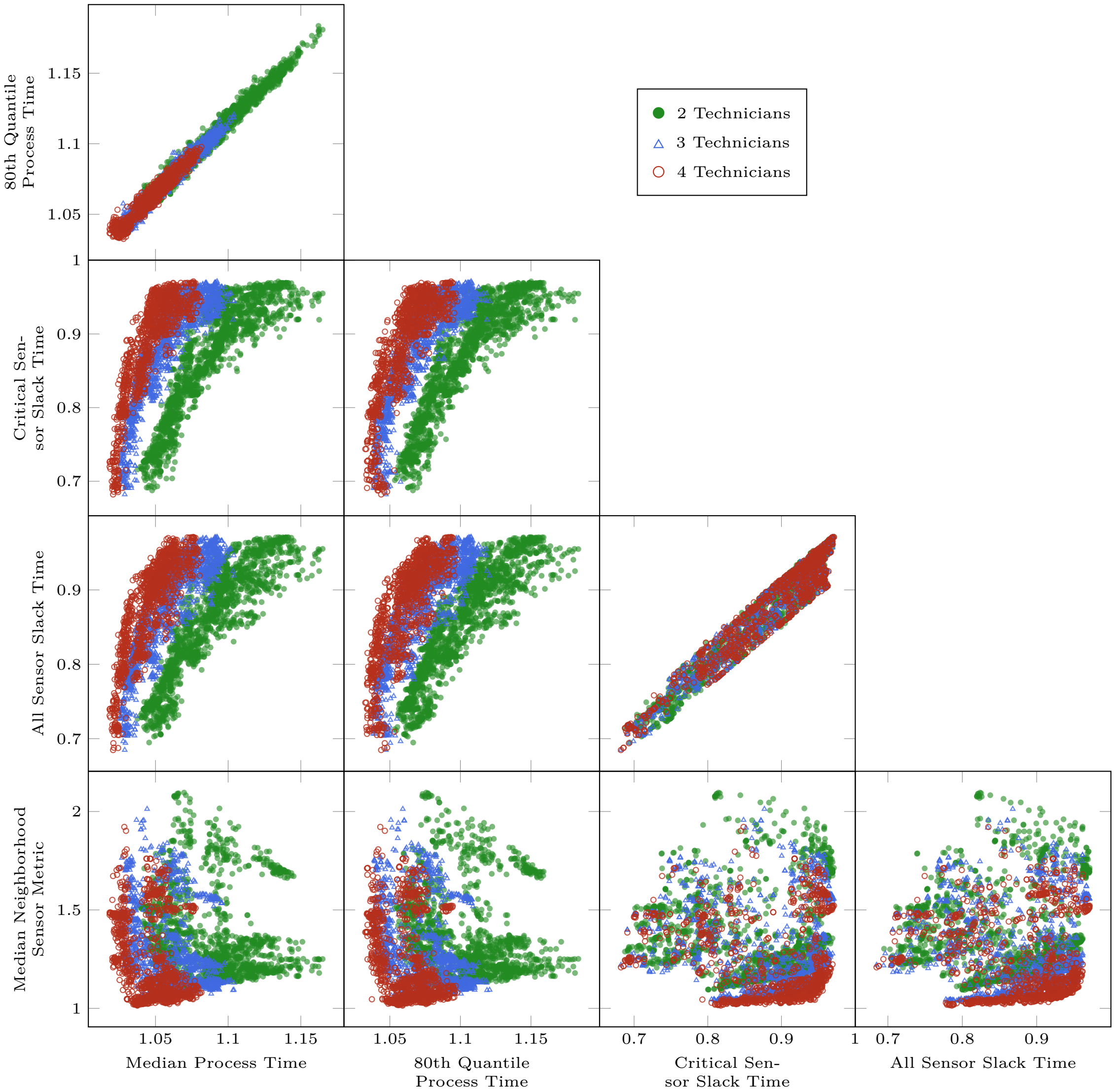


Figure 50: Normalized NSGA-II Pareto Efficient Population Comparison for 2-, 3-, and 4-Technician Scenarios

that enable trades between the quality robustness (median and 80$^{\text{th}}$ quantile of process time) and the deterministic risk metrics. The plot comparing the median and 80$^{\text{th}}$ quantile of the process time shows that cases with similar median process times can have different amounts of risk. As such, including the higher quantile information equips the decision maker with information to identify a plan with a good balance between quality robustness and the deterministic risk metrics. The following section compares the results presented so far to results obtained using the deterministic optimization strategy.

5.1.3.3 Comparison of Optimization Results Generated with and without Robustness Considerations

The inclusion of quality robustness in the optimization should provide information to the algorithm to better identify robust plans. Hence, results from the optimization formulations with and without robustness are compared. Recall that the optimization with uncertainty was seeded with the population from generation 500 of the deterministic run. Hence, to facilitate comparison, the populations from generations 600, 700, 800, 900, and 950 of the deterministic run are evaluated for 35 replications to estimate the median and 80$^{\text{th}}$ quantile of the process time.

Before discussing the results further, it must be noted that the following analyses cannot be generalized and are limited for 2 primary reasons: first, the amount of computing time/resources available are limited. While completing a single optimization run using the available network of computers is feasible, re-running the entire optimization enough times to identify statistically significant differences is not. Second, the use case does not see a very large impact from the stochasticity. While Figure 49 did demonstrate a difference between the deterministic and stochastic results, the majority of the deltas are within the indifference region. In other words, for the majority of the plans evaluated, the uncertainty does not make many of the plans identified deterministically significantly worse. Hence, the information added by considering the quality robustness is of limited value.

With this caveat, the remainder of this section discusses preliminary observations and trends identified by comparing optimization runs with and without considering robustness. Section 7.5 discusses future steps that can be completed to better support this study.

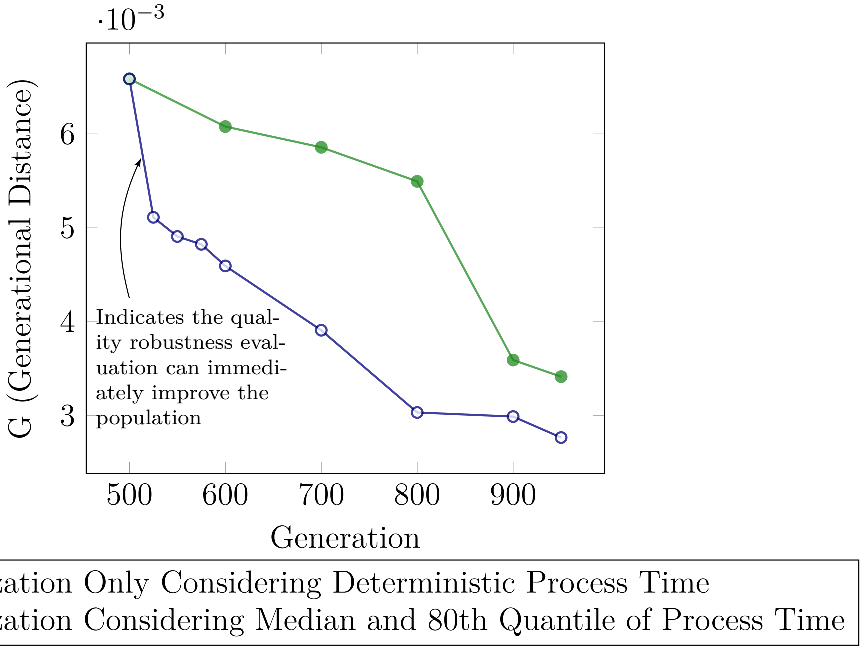


Figure 51: Comparison of the Generational Distance for the Optimization Runs with and without Quality Robustness Considerations

Figure 51 presents the generational distance (G), which is the average distance from each non-dominated point in a generation to the overall Pareto frontier as defined by Equation 9, for the optimization runs with and without considering quality robustness. The "true" Pareto frontier used in the calculation is generated by combining the results from each generation of the stochastic algorithm with results from the re-evaluated cases of the deterministic run. Furthermore, the stochastic algorithm is allowed to run for an additional 75 generations to generate an improved Pareto frontier with the goal of removing bias in the G calculations. This is needed because comparing the distance from the non-dominated points of the last generation to an overall Pareto frontier that contains those population members would not produce a fair result. For example, a large number of the plans that are Pareto efficient in the final generation would also be non-dominated across all of the evaluated generations. In these cases, their individual distance from the “true” Pareto frontier would be zero and would, hence, unfairly bias results. The final deterministic population (generation 1686) is also re-evaluated and added to the “true” Pareto frontier used in the evaluation of G.

The results presented in Figure 51 follow the trends observed in Figures 47 and 48. The optimization run considering quality robustness experiences a sharp decrease in the generational distance within the first 25 generations. The deterministic optimization scenario, conversely, does not see a similar improvement to its generational distance until somewhere between generation 800 and 900. It is between these generations that a significant improvement to the population’s deterministic process time is identified and propagated through the population (Figures 44 and 45). Hence, the stochastic optimization run does see an immediate improvement when compared to the deterministic algorithm; however, the overall performance is not significantly different by the time the final generation is evaluated (Generation 950).

To further investigate this initial improvement, Figures 52 and 53 present the Pareto frontiers from generation 1 and 10 of the optimization runs with quality robustness considerations for the 2- and 4-technician scenarios, respectively. In both scenarios, the algorithm has pushed the frontier outward by reducing the process time metrics while improving slack time and the neighboring sensor installation metric. Figure 54 focuses on the Pareto frontier for the 80$^{\text{th}}$ quantile of process time and the slack time. As shown in the figure, the initial 10 generations of the stochastic NSGA-II have identified population members with, in some cases, significant (larger than the indifference region) reductions in process time with similar amounts of slack.

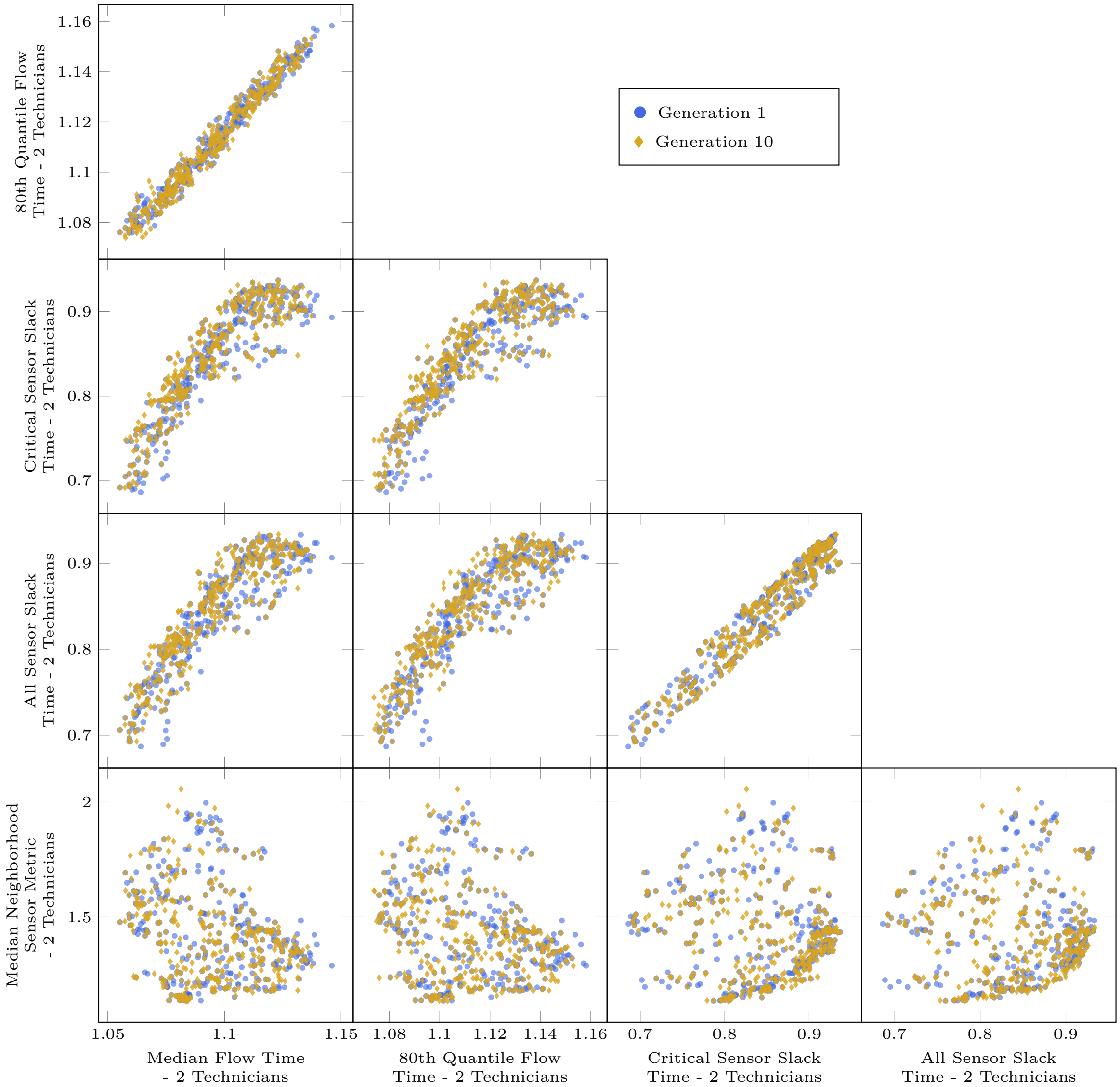


Figure 52: Initial Improvement When Considering Quality Robustness in the Optimization - 2 Technicians

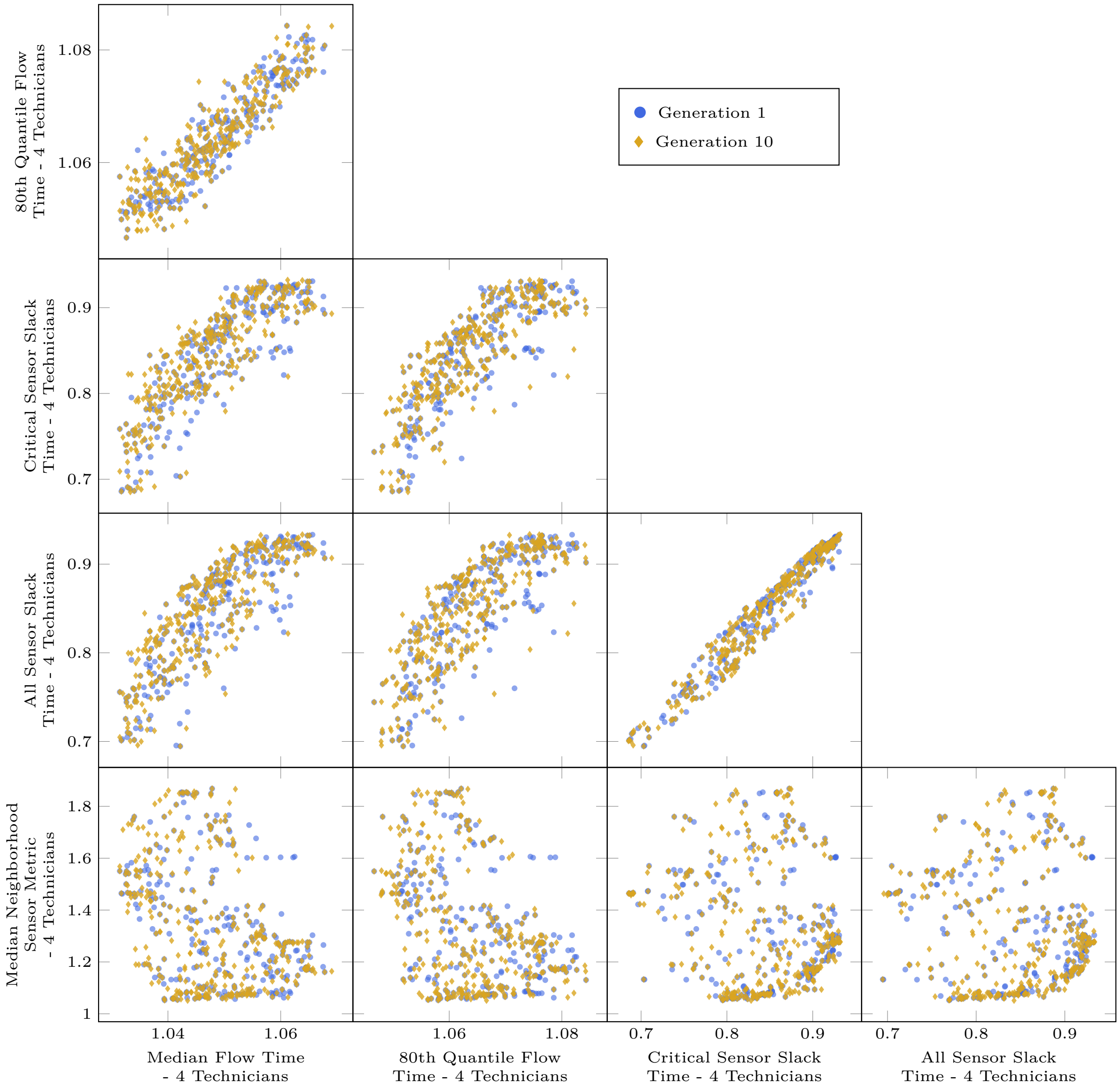


Figure 53: Initial Improvement When Considering Quality Robustness in the Optimization - 4 Technicians

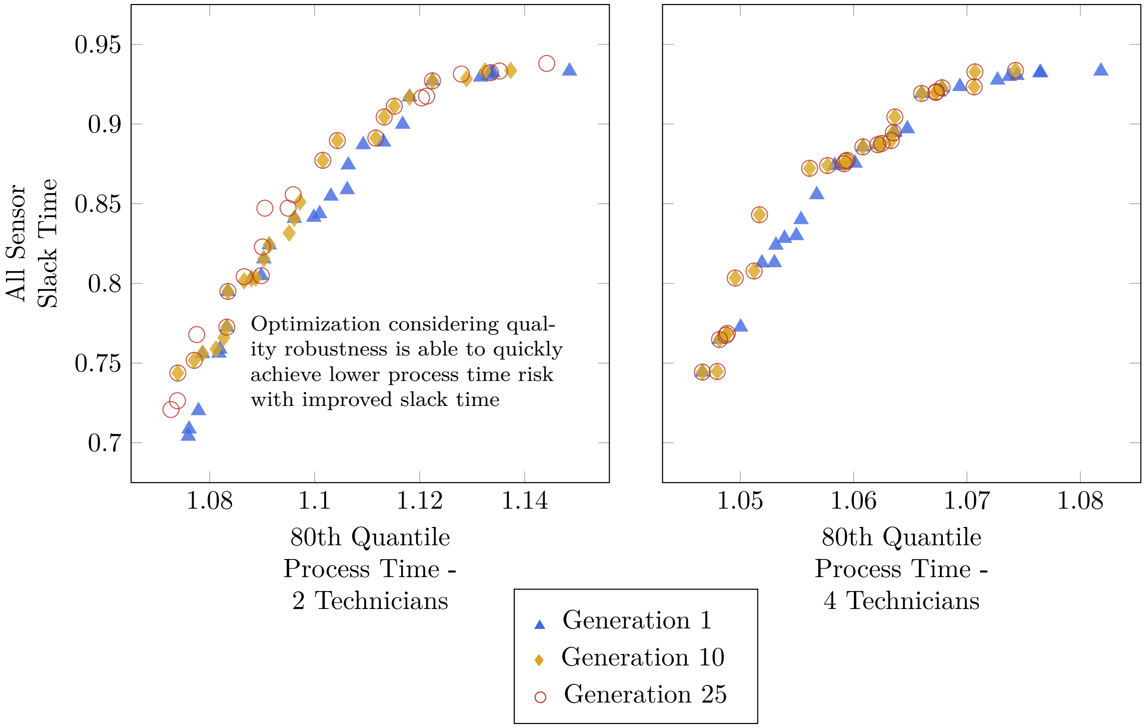


Figure 54: Initial Improvement to 80$^{\text{th}}$ Quantile of Process Time and Slack Time When Considering Uncertainty in the Optimization

This further shows that the stochastic optimization run is capable of improving a population identified deterministically with a limited number of generations.

Figure 55 demonstrates how the algorithm with quality robustness considerations helps to improve convergence. The percentage of non-dominated points initially drops in the run with stochasticity because random points were added to the seed population; however, the algorithm is able to reach above 90% non-dominated points by generation 10. In contrast, the algorithm without robustness considerations remains in the 65–75% range for the entire optimization run. Hence, by having access to the quality robustness information, the algorithm is able to more efficiently generate new Pareto optimal points.

The performance of the two optimization runs is further compared by exploring the process times observed in the Pareto optimal set as the optimization run progresses. Figure 57 presents the distribution of the median process time in the Pareto optimal

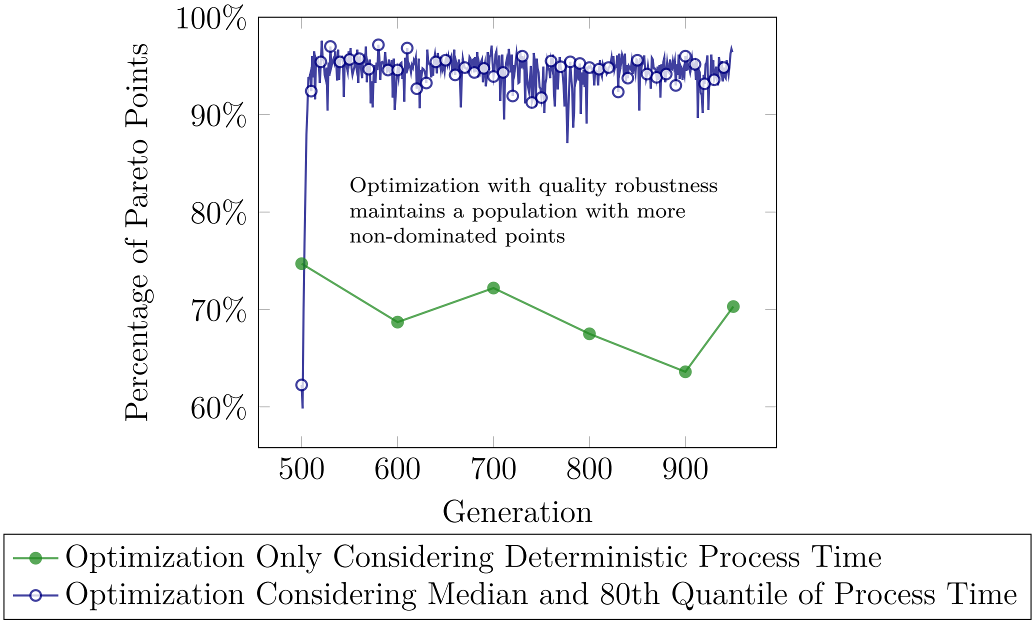


Figure 55: Percentage of Pareto Points Identified by the Optimization Runs with and without Quality Robustness Considerations

set found by the optimization considering quality robustness (blue dashed box plots) and without (black solid box plots). Figure 58 presents the distributions of the 80th quantile of the process time. The presented box-and-whiskers plots follow the convention shown in Figure 56.

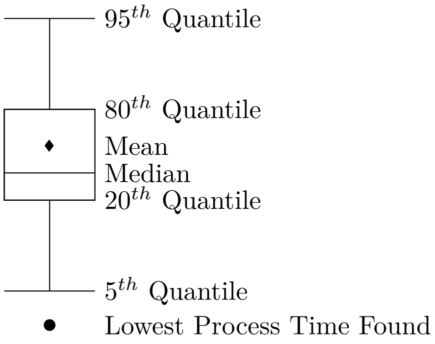
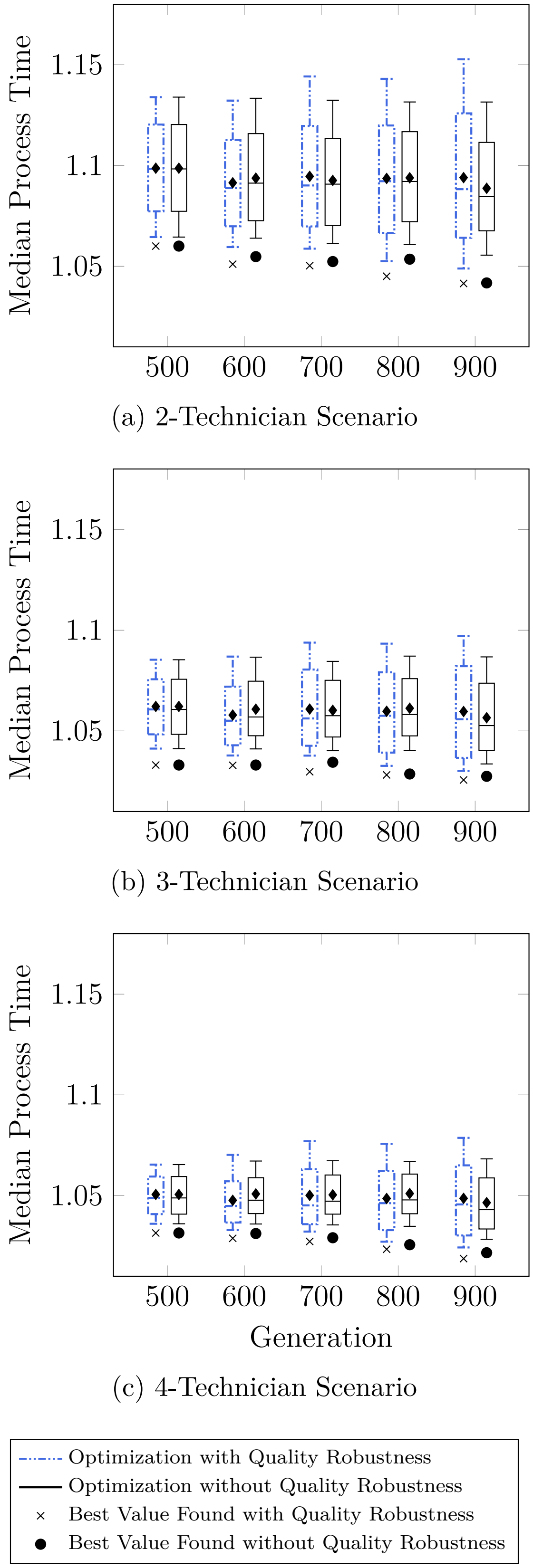


Figure 56: Description of the Box-and-Whiskers Plot Presented in Figures 57 and 58

These figures indicate that the optimization run considering quality robustness generally identifies points with lower process times than the deterministic optimization. In Figure 57, the best process times when considering robustness (shown as the x marker) are always lower or approximately similar to the best value found by the deterministic algorithm. Furthermore, the lower end of the box plots (the 5th and 20th quantile) show that more sensor installation plans with lower ranges of process time are maintained in the population that is evaluated with stochasticity.

Figure 58 exhibits more pronounced differences between the 80th quantile of process times observed in the populations of the two optimization runs. In many cases, the 5th quantile value for the cases from the stochastic optimization run are close to, or better, than the best value from the deterministic optimization. This is expected since the median and deterministic process time should be generally similar; however, the 80th quantile represents new information that is provided to the optimization with quality robustness. Hence, these results indicate that including the 80th quantile of the process time provides valuable information that can lead to solutions with better robustness than those identified with only the deterministic model.

While process time is specifically the objective that is expected to see the most improvement when including robustness in the optimization's formulation, the impact to the overall Pareto frontier is also important. Hence, Figures 59 and 60 present the Pareto frontier from generation 600 of the optimization runs with and without robustness considerations. The optimization with the robustness considerations can be seen to have improved values for process time for comparable values of slack and the neighboring sensor installation metrics. This impact is more pronounced for the 80th quantile of the 4-technician scenario. This could indicate that the information about the plan's risk enables the algorithm to identify scenarios that utilize the additional man-power to better reduce risk. Ultimately, this indicates that the quality robustness calculations add information to the optimization algorithm beyond that provided by the deterministic process time evaluations. The following section summarizes the results from Experiment 1.3 and discusses the validity of Hypothesis 1.3.

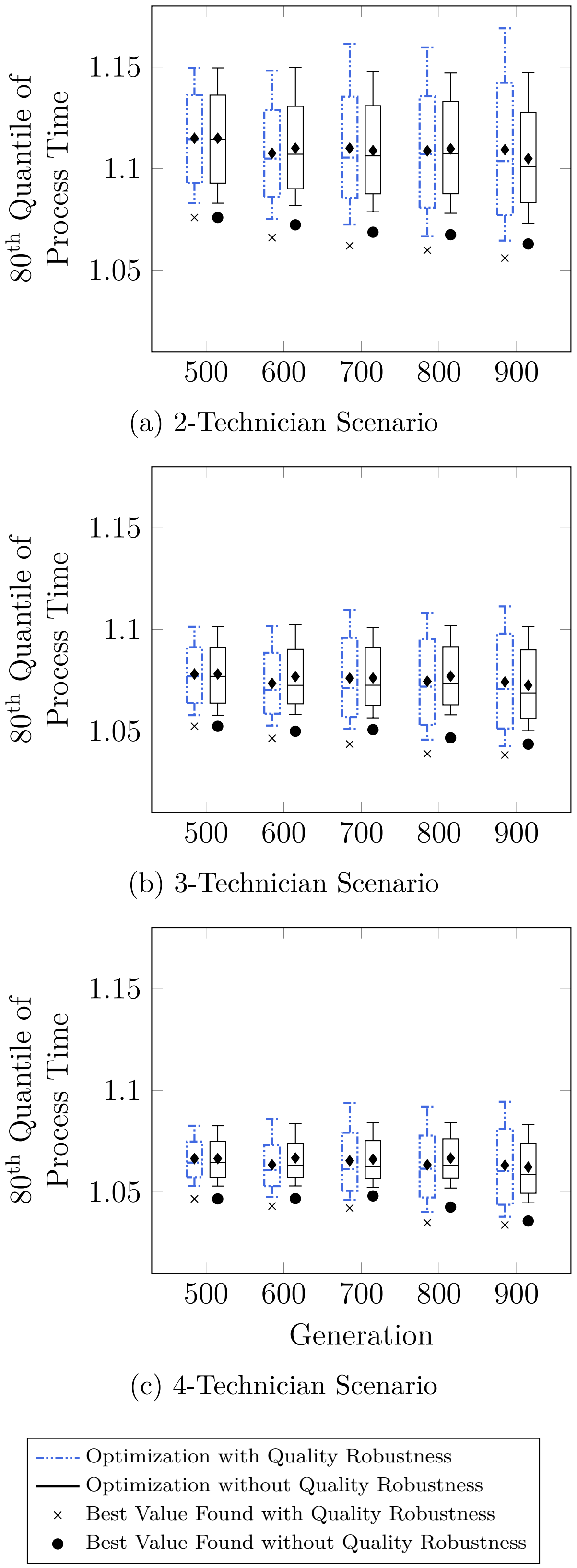


(a) 2-Technician Scenario

(b) 3-Technician Scenario

(c) 4-Technician Scenario

Figure 57: Distribution of the Pareto Optimal Median Process Time Found by the NSGA-II Optimization with and without Considering Quality Robustness

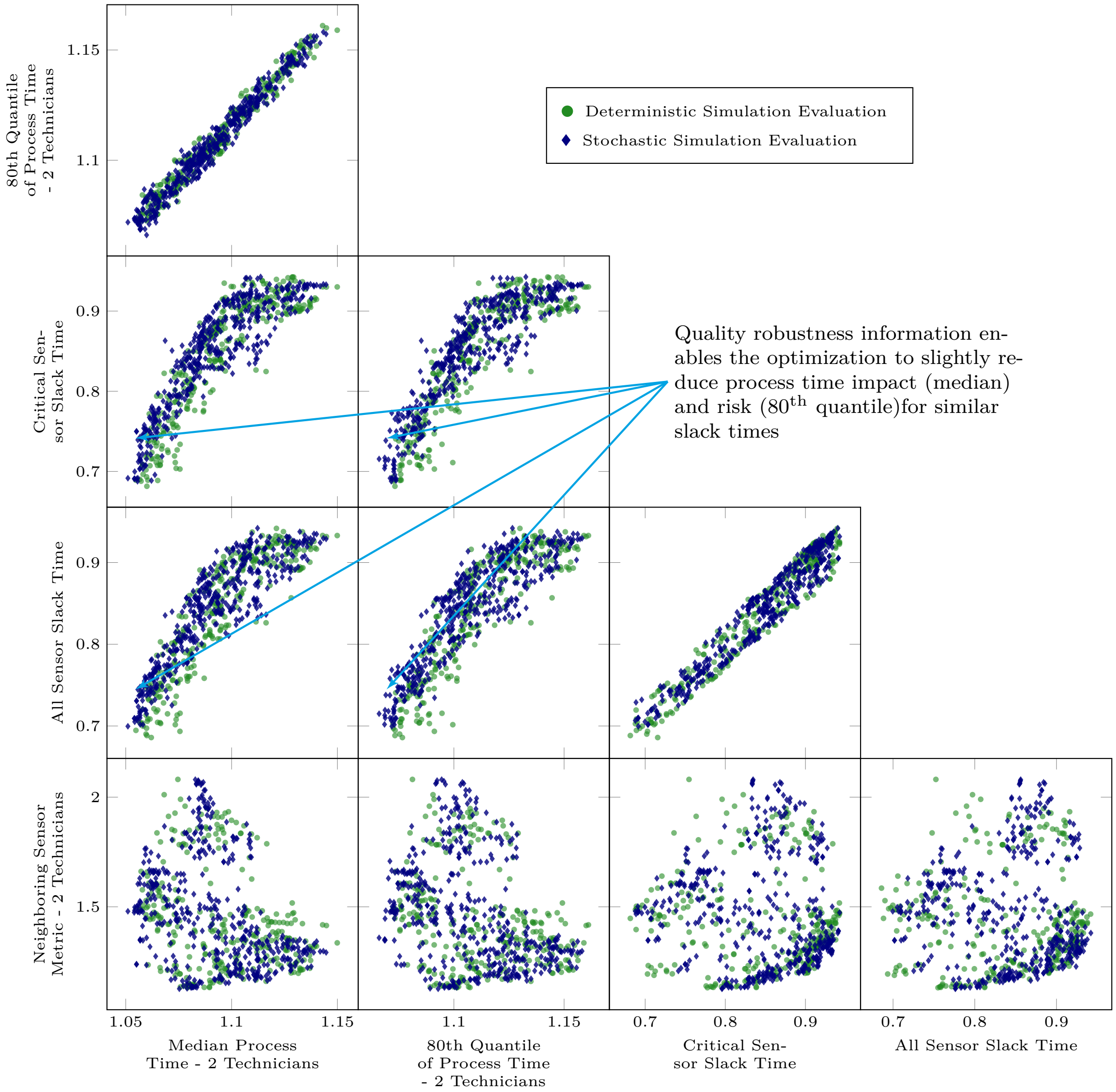


(a) 2-Technician Scenario

(b) 3-Technician Scenario

(c) 4-Technician Scenario

Figure 58: Distribution of the Pareto Optimal 80th Quantile of Process Time Found by the NSGA-II Optimization with and without Considering Quality Robustness



Quality robustness information enables the optimization to slightly reduce process time impact (median) and risk (80th quantile)for similar slack times

Figure 59: Comparison of the Pareto Frontiers Found at Generation 600 from the Optimization with and without Considering Quality Robustness with 2 Technicians

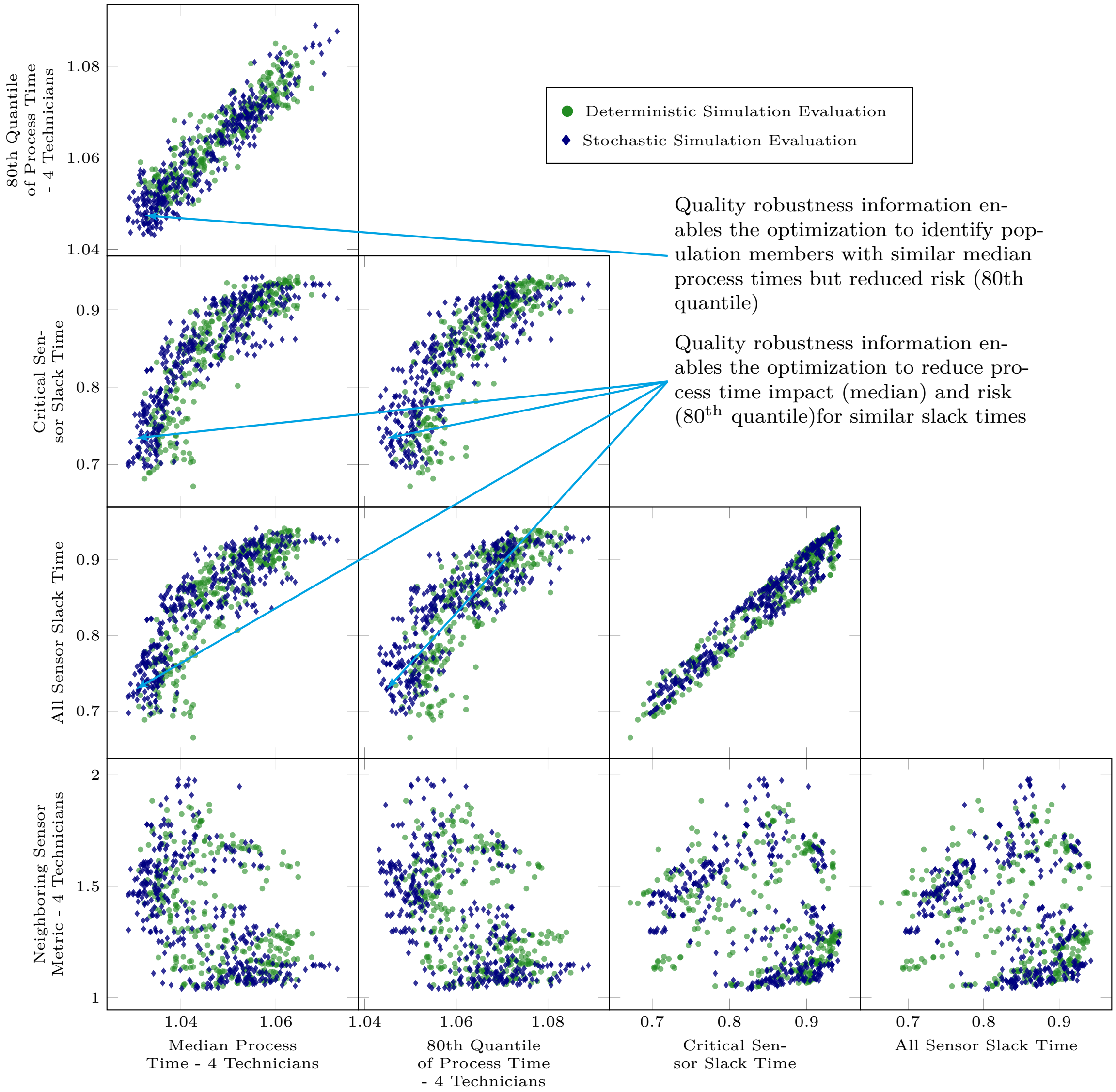


Figure 60: Comparison of the Pareto Frontiers Found at Generation 600 from the Optimization with and without Considering Quality Robustness with 4 Technicians

5.1.3.4 Section Summary

The results presented throughout this section support two conclusions. First, utilizing only the deterministic process time in the optimization routine enables the algorithm to identify plans with improved quality robustness. This shows that, for the selected problem, the deterministic results provide a good indication of the performance of the installation plan under uncertainty. While this dissuades the need to consider the expensive quality robustness evaluation in the optimization, Figure 51 also shows that the optimization including robustness is able to make significant progress in a relatively few number of generations. Hence, for problems similar to the test case problem, running a small number of generations with the quality robustness calculations at the end of a deterministic optimization run could provide quick refinement to the Pareto frontier of solutions.

With regards to the experimental evaluation criteria, the optimization with quality robustness considerations is shown to improve the robustness of the identified plans. The additional information made available to the optimization algorithm by evaluating the quality robustness is shown to increase the percentage of Pareto optimal points identified at each generation. This shows that the new information increases the ability of the algorithm to find Pareto efficient points, which likely leads to the improved performance in fewer generations.

The algorithm with robustness is also shown to provide a set of non-dominated solutions that cover a range of objective function values to enable the decision maker to trade between quality robustness and the additional risk metrics. Furthermore, differences between the 80^{th} quantile of process time for plans with similar median process times make an additional trade available to the decision maker.

The final criteria investigated provides a comparison between the deterministic and stochastic optimization results. The investigation of the generational distance metric (Figure 51) shows that the optimization with robustness quickly improves towards the Pareto frontier; however, the deterministic optimization does eventually achieve similar performance between generations 800 and 900. Therefore, as discussed previously, these results can not be used to draw generalized conclusions and could potentially just be due to luck.

Similar trends are seen when specifically investigating the process times identified by the deterministic and stochastic optimization runs (Figures 57 and 58). While considering robustness does lead to populations with members having reduced medians and 80th quantiles of process time, the results are not vastly improved over the long term by the optimization considering robustness. Finally, the investigation of the Pareto frontiers from generation 600 of the deterministic and stochastic optimization runs illustrates that the stochastic results are slightly better. Additionally, the improvements are mostly seen in the process time metrics; the points identified on the Pareto frontier of the slack and neighboring sensor installation metrics are generally similar. Overall, these findings exhibit trends that indicate that the robustness considerations are utilized by the algorithm to identify better populations.

The results from this experiment, therefore, support Hypothesis 1.3. The multi-objective nature of the optimization algorithm combined with the evaluation of quality robustness appear to lead to the identification of more robust schedules. However, as discussed throughout, the results are not significant enough to draw clear conclusions about the benefit of including robustness. Indeed, the slight benefits observed likely do not justify the significantly more time/computing resources required. This aspect is further addressed during the discussion of Experiment 2.2.

The following section discusses the implications of Experiments 1.1–1.3 to the overarching Experiment 1.

5.2 Experiment 1 Discussion

Experiment 1 is carried out by utilizing the capabilities built through the evaluation of Experiments 1.1–1.3 (as shown in Figure 29) to complete Steps 1–2 of the methodology. When completing these steps, the following capabilities are investigated to determine the appropriateness of Hypothesis 1:

- Modeling of complex manufacturing systems
- Optimization of the developed model
- Assessment through the model and improvement through the optimization routine of the schedule's robustness

As shown in Experiment 1.1, the optimized schedule model and compatibility matrix are utilized to generate a simulation model. The experiment has demonstrated the capability of the model to assess the impact of parametrically-defined sensor installations to the level of detail required for the optimization routine. Hence, the modeling strategy is capable of modeling the complex production system rules required to drive the implemented schedule optimization strategy.

Then, as investigated through Experiments 1.2–1.3, the optimization strategy developed is capable of improving the multiple objectives of interest. The multi-objective formulation enables the identification of a family of solutions that can be explored by the decision maker to best reduce risk in the installation plan.

Finally, Experiment 1.1 demonstrates that the model is capable of assessing the robustness of the installation plan by evaluating the simulation for multiple replications. Furthermore, Experiment 1.3 demonstrates that the inclusion of quality robustness in the optimization does help to improve the robustness of the identified population of installation plans.

Hypothesis 1 is thus supported by the results generated by completing Steps 1 and 2 of the PORRTSS methodology. The simulation is quickly generated at a level of detail sufficient to delineate between installation plans. Then, through the multi-objective optimization routine, a variety of installation plans are generated that improve the robustness of the plan.

The following section discusses the results of experiments designed to test potential improvements to the deployability of the detailed scheduling methodology.

5.3 Experiment 2: Methodology's Potential for Deployment to "Real-World" Problems of Interest

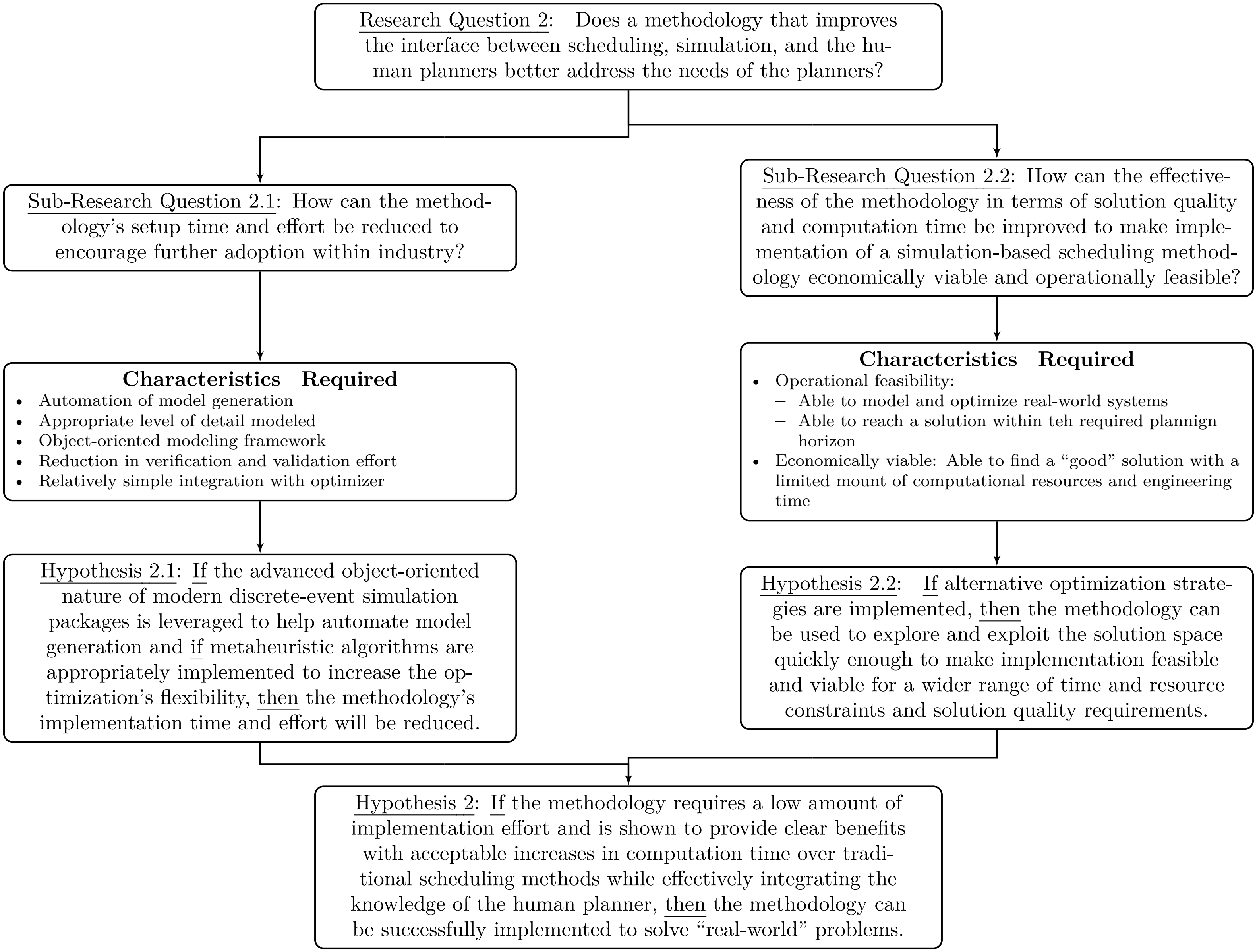
Experiment 2 is setup in order to investigate Hypothesis 2:

> **Hypothesis 2**: If the methodology requires a low amount of implementation effort and is shown to provide clear benefits with acceptable increases in computation time over traditional scheduling methods while effectively integrating the knowledge of the human planner, then the methodology can be successfully implemented to solve "real-world" problems.

As with Hypothesis 1, this hypothesis contains sub-hypotheses that build up support for the overarching hypothesis. Hence, this section begins by reporting results from Experiment 2.1 and Experiment 2.2. With those experiments completed, the overall hypothesis is then discussed in Chapter 6. Figure 61 presents an overview of the research questions and hypotheses investigated through the following experiments. The following section discusses the results from Experiment 2.1.

5.3.1 Experiment 2.1: Methodology Implementation Time Reduction

The goal of this experiment is to test how the PORRTSS methodology's implementation time can be decreased as described in Hypothesis 2.1 (reproduced below). As previously discussed, many of the mid-term, detailed planning and scheduling problems of interest may not have a long enough time horizon to justify a complete, manual



Research Question 2: Does a methodology that improves the interface between scheduling, simulation, and the human planners better address the needs of the planners?

Sub-Research Question 2.1: How can the methodology's setup time and effort be reduced to encourage further adoption within industry?

Characteristics Required

- Automation of model generation
- Appropriate level of detail modeled
- Object-oriented modeling framework
- Reduction in verification and validation effort
- Relatively simple integration with optimizer

Hypothesis 2.1: If the advanced object-oriented nature of modern discrete-event simulation packages is leveraged to help automate model generation and if metaheuristic algorithms are appropriately implemented to increase the optimization's flexibility, then the methodology's implementation time and effort will be reduced.

Sub-Research Question 2.2: How can the effectiveness of the methodology in terms of solution quality and computation time be improved to make implementation of a simulation-based scheduling methodology economically viable and operationally feasible?

Characteristics Required

- Operational feasibility:
  - Able to model and optimize real-world systems
  - Able to reach a solution within teh required plannign horizon
- Economically viable: Able to find a "good" solution with a limited mount of computational resources and engineering time

Hypothesis 2.2: If alternative optimization strategies are implemented, then the methodology can be used to explore and exploit the solution space quickly enough to make implementation feasible and viable for a wider range of time and resource constraints and solution quality requirements.

Hypothesis 2: If the methodology requires a low amount of implementation effort and is shown to provide clear benefits with acceptable increases in computation time over traditional scheduling methods while effectively integrating the knowledge of the human planner, then the methodology can be successfully implemented to solve "real-world" problems.

Figure 61: Research Question and Hypothesis 2 Buildup

simulation study. As such, improvements to the model construction and optimization integration strategy are identified and implemented into this thesis's methodology.

Hypothesis 2.1: If the advanced object-oriented nature of modern discrete-event simulation packages is leveraged to help automate model generation and if metaheuristic algorithms are appropriately implemented to increase the optimization's flexibility, then the methodology's implementation time and effort will be reduced.

Implementing the methodology encompasses two primary steps:

1. Constructing the model
2. Integrating the metaheuristic optimization routine

First, the strategies and assumptions used to construct the models are investigated in the following section.

5.3.1.1 Discussion of Model Constructions Strategies and Assumptions

Throughout this research, a diversified set of models are constructed to represent various portions of the case study's production flow at different levels of detail. A summary of the model sizes and assumptions are presented in Table 15.

The first model is an initial proof-of-concept focused on a small subset of the vehicle's sensors. To help verify the method and model logic, the extremely detailed sequence is simplified to enable manual model construction. The provided schedule for this initial investigation contains over 500 primary production processes. A significant number of these tasks are duplicate processes that occur in parallel at different locations on the vehicle. For modeling purposes, these parallel processes are combined into a single overall process to be completed. For example, if identical components are to be installed in parallel in multiple places throughout the vehicle, then these are condensed into a single process. This strategy reduces the modeling effort down to

Table 15: Overview of Model Sizes and Complexity Modeled Throughout this Research

Description	**Primary Processes**	**Sensors to Install**	**Model Assumptions**
Small model with simplifications	~500 processes combined and simplified to ~200	~50	• Simplified processes do not exactly replicate the provided schedule model • Many parallel processes were combined, which reduces the ability to understand the interactions between parallel processes • Multiple processes combined into a single server limits the information that can be easily obtained from the model
Medium-sized model with single shift schedule	~200	~200	• Processes are not combined → the baseline model without sensor installations very closely matches the provided schedule model • Each process has a separate server → full data collection is possible • Single-shift schedule enables simple definition of process time
Large-sized model with multiple shift schedules	> 1000	~500	• Processes are not combined → the baseline model without sensor installations very closely matches the provided schedule model • Each process has a separate server → full data collection is possible • Multiple-shift schedule does not allow for a good definition of process time → Flow time is used as the completion time metric

approximately 200 processes. While an improvement, this is still too many servers to manually place and link in Simio without taking an excessive amount of time. This manual approach is also prone to errors. Because many processes occur in series, the 200 processes can be further simplified by identifying progressions of serial processes and condensing them into a single discrete-event server. For example, instead of requiring 10 servers connected in series for a set of 10 serial processes, one server can be made to model the completion of multiple serial processes. This strategy produced a manageable number of servers (~30) to place manually and complete the modeling proof-of-concept.

Upon proper data preparation, constructing the first model prototype required about 15 hours of effort (excluding the significant time required to build the generic background logic). The construction effort was primarily dedicated to manually placing each server, linking them based on the schedule precedence relationships, and linking the input data tables to their proper servers. After building a prototype model, the verification effort required a significant amount of additional effort. Verification required identification of positions in the flow where the simplifications caused a large disparity between the simulation results and the provided schedule model. Throughout this effort, modeling mistakes were corrected and assumptions were modified, which increased the effort required to complete the model.

The small proof-of-concept model performed adequately, but does not contain the resolution and accuracy required by the decision makers to identify an actual production schedule. Combining multiple processes into a single server limits the information that can be retrieved from the model for decision making. There are also significant limitations when considering uncertainty. By collapsing multiple parallel processes into a single, overall process, potential discrepancies between the completion times of each process cannot be accounted for. Finally, due to the simplifications, some process constraints and relationships are ignored, which leads to an inability to match the simulation to the provided schedule model.

The limitations stemming from the simplifications and assumptions in the small model illustrate the need to increase the modeling detail. Increased fidelity enables better planning of processes and an improved understanding about the impact of uncertainty on the schedule. Unfortunately, for a real problem of interest, which is denoted as the Medium sized model in Table 15, manually modeling the flow without simplifications is unrealistic. Constructing a model with ~200 servers and ~500 links would require an excessive amount of time and effort. Furthermore, verifying the model logic for every link and server would require at least as much time as building the model. This demonstrates the lead to the need to automate the creation of the simulation model.

The automated process described in Section 4.2 helps to reduce the time required to construct the simulation from weeks to about a day of effort. In addition to automating the actual model object definition, utilizing the model generation strategy significantly reduces the need for model verification by eliminating human error. Therefore, for the medium sized model, the automated generation technique greatly reduced the modeling time, while also producing a more detailed and easily verifiable model. The resulting model's process time matches the original schedule within 0.1%, which is sufficiently accurate.

The final test for the model generation's ability to reduce the modeling effort is to examine the time to build the large model described in Table 15. The large model uses the schedule from the small model without simplifications and extends it to include additional assembly steps. The model generation strategy used for the medium sized model is applied to the large-scale model. The time to translate the schedule model to a format compatible with Simio is not significantly affected by the larger scale; however, the time and reliability of the process to generate the objects within Simio is significantly worsened. The capability of Simio to handle the larger number of servers and links is strained. While the medium model required approximately 10 minutes to translate the object spreadsheet to a working Simio model, the large model required multiple days along with using the 64-bit version of Simio to enable access to more memory. Hence, it seems that there are practical limitations to model size that are approached by the large scale model. Despite this, the strategy is successful in creating a model that replicates the schedule (within 0.1% of process time) with the required granularity.

5.3.1.2 Model Generation Results Discussion

The results presented above illustrate the success of the implemented model generation strategy. For “medium-sized” models, automating the creation of the models significantly reduces construction and verification time. Furthermore, the automatically generated models provide more detailed statistics than the simplified, “small” model, while also being capable of better matching the deterministic schedule’s results.

The “large-scale” model approaches the practical software limitations. Including the larger number of objects strains the software’s capabilities. Despite these drawbacks, the model can be feasibly generated. Compared to a manual approach, which requires significant assumptions and simplifications, this strategy provides increased knowledge about the system-level impact of a decision. The large model represents the most complex assembly sequence that would be considered; therefore, the model generation strategy reduces the implementation time of the methodology for a range of applicable problems.

More recent advances in the Simio language could potentially help better describe the “large-scale” model. When the PORRTSS methodology was in development, the precedence network had to be defined using linked servers; however, task sequences can now be defined by linking a sequence table to a single server. This would eliminate the need to define multiple servers and links in the model, which would help to resolve the identified capability gap. Furthermore, implementing this new feature would reduce the time to run a single replication. An investigation of these new capabilities could provide a wider range of applicability for this methodology. This is, however, beyond the scope of the current work.

The following section investigates the other contributor to the methodology's development time: integration of the optimization routine.

5.3.1.3 Optimization Integration

A major advantage of the general metaheuristic optimization framework chosen for this research is its extreme flexibility. Because these algorithms operate primarily on the objective function evaluations, once a framework is constructed, the algorithm should be immediately adaptable to additional models utilizing the same input and output formats. Furthermore, the code's modularity should enable quick implementation of new metaheuristics with little to no modification of the underlying function evaluator. Hence, this section investigates the general applicability of the optimization algorithm to the various models discussed in the previous section. Additionally, the implementation of new overarching, metaheuristic algorithms and objective functions is discussed.

The optimization framework, as previously discussed, utilizes the same inputs as the Simio model (e.g. compatibility matrix, sensor installation times, and primary process information) to setup the optimization problem. Hence, if a Simio model is successfully generated and run, the optimization algorithm requires very little modification to accommodate the new model. Indeed, other than changing the model name and potentially adding or subtracting objectives from consideration, the algorithm requires no further modification to begin optimizing new models. Both the medium and large scale models were successfully implemented with the optimization. Hence, while model size may impact the algorithm and strategy chosen for the optimization (discussed throughout Section 5.3), it does not affect the general applicability of the developed framework.

Another benefit of the metaheuristic paradigm is that the code-base should allow for easy swapping and implementation of search strategies. After an initial amount of effort devoted to creating the underlying function evaluation code-base, incorporating new metaheuristic strategies is relatively simple. With the Simio model setup as a black box function, implementing a new strategy simply requires changing how to select new input values based on the simulation's results. With the code-base to run the NSGA-II previously created in MATLAB, implementing a basic Simulated Annealing algorithm required only a day of effort. The following section summarizes the results related to Experiment 2.1 and discusses the implications for Hypothesis 2.1.

5.3.1.4 Experiment 2.1 Discussion

The purpose of this experiment is to investigate Hypothesis 2.1 by addressing improvements to the model creation and optimization implementation phases of the research. The time to create a model is significantly reduced through the implementation of an automated model generation scheme. The automated approach relies on the object-oriented nature of the simulation framework to create modular objects capable of representing a schedule. Then, using these objects as a base, the strategy is able to successfully assign properties and link the objects to effectively model the provided schedule. This approach was successfully applied to multiple schedules without requiring modification, which demonstrates its scalability and flexibility.

The metaheuristic optimization approach is also shown to be flexible in both its applicability to a range of problems and the ease of integrating new algorithms. Models created with the model generation strategy could immediately be integrated with the optimization routine without modification. Furthermore, because of the nature of the metaheuristic algorithm's structure, additional algorithms are easy to implement and swap with the first NSGA-II. Therefore, from these results, Hypothesis 2.1 is supported. The following section discusses the results from Experiment 2.2 aimed at investigating a range of optimization strategies for application to problems of various complexity with diverse solution time and computational resources available.

5.3.2 Experiment 2.2: Improve Optimization Convergence Quality and Time Results

Following Figure 61, Experiment 2.2 investigates a range of optimization algorithms and robustness considerations in an effort to identify a well-performing set of strategies that can be implemented for a range of problems. The best optimization strategy to apply is always highly problem-dependent, so investigating and discussing the benefits of a range of methods is important to examining the applicability of this methodology. The investigation is conducted via a two-stage down-selection process that identifies promising optimization strategies and quality robustness considerations. In doing so, this experiment investigates Hypothesis 2.2:

Hypothesis 2.2: If alternative optimization strategies are implemented, then the methodology can be used to explore and exploit the solution space quickly enough to make implementation feasible and viable for a wider range of time and resource constraints and solution quality requirements.

The first round in the down-selection process is to investigate a variety of algorithms (both heuristic and metaheuristic) without considering quality robustness (e.g. using the deterministic model). Results from each algorithm/strategy identified are presented and compared to results from the NSGA-II, which serves as the baseline. The strategies to carry forward, where the identified robustness measurement techniques are investigated, are then selected. The second round investigates how each robustness measure integrates with the strategies selected during the initial round. Ultimately, the goal is to provide an indication for how the various algorithms perform for problems of different complexities, time and resource constraints, and solution quality requirements.

The following section presents the results from each algorithm discussed in Section 4.3.4.

5.3.2.1 Deterministic Optimization Algorithm Down-Selection

This section presents the results from the first down-selection utilizing deterministic evaluations of the schedule. As described previously, results generated from the NSGA-II algorithm are used to provide the baseline Pareto frontier. As demonstrated through Experiment 1.2, the deterministic NSGA-II has converged sufficiently to provide a baseline to compare the performance of the deterministic algorithms.

With the converged baseline, the techniques and metrics discussed in Section 4.6.6.2 are used to compare the proposed algorithms and strategies. Furthermore, because process time is an important metric, the best process time found is also examined. To help ensure fairness across the experiments, a set of 5 initial points are created and used to start each algorithm. The process time is weighted such that it is 4 times as important as the slack time and neighboring sensor installation metrics. While the weightings impact the region of the design space investigated by the algorithms, the weights should not impact the comparison of the algorithms. Each algorithm is run for 1500 generations with the goal of investigating how well it can converge to a solution in a limited amount of time.

Results from each examined optimization strategy are displayed in Figures 62 and

63. The results do not contain points from the greedy shifting bottleneck inspired search or the Pareto optimal based simulated annealing strategies. The greedy algorithm, which always moves the “worst” performing sensor installation, consistently stalled after only 1–2 improving moves. The Pareto simulated annealing algorithm also did not perform well. For many iterations of the algorithm, the candidate points were of the same Pareto rank. This caused the algorithm to essentially devolve into a random search, which did not begin to approach the baseline frontier. In an effort to improve the Pareto simulated annealing's performance, only a single sensor installation technician scenario was run at a time to reduce the dimensionality. This, however, did not produce any meaningful improvement, so the following down-selection process is focused on the heuristic and metaheuristic, weighted-sum algorithms.

Figures 62 and 63 show that the underlying heuristic appears to drive the optimization towards different regions of the Pareto frontier. This is explicitly noted on Figure 62. The algorithms employing the shifting bottleneck heuristic tend to identify regions with lower process time than the algorithms using the expanded neighborhood formulation. However, the expanded neighborhood algorithms more consistently identify points with better values for the neighboring installation metric.

This could indicate that by limiting the neighborhood definition, the shifting bottleneck heuristic is able to better refine the process time. The process time can be significantly impacted by moving a single sensor installation; however, the neighboring sensor installation metric, as it is related to the interaction of all sensor installations, may require more significant moves to see a major change. Hence, the shifting bottleneck heuristic algorithms appear to drive down the overall objective function by steadily driving down the process time. In contrast, the expanded neighborhood algorithms, by modifying a larger portion of the plan at each iteration, are able to better drive down the weighted objective through identifying changes that improve

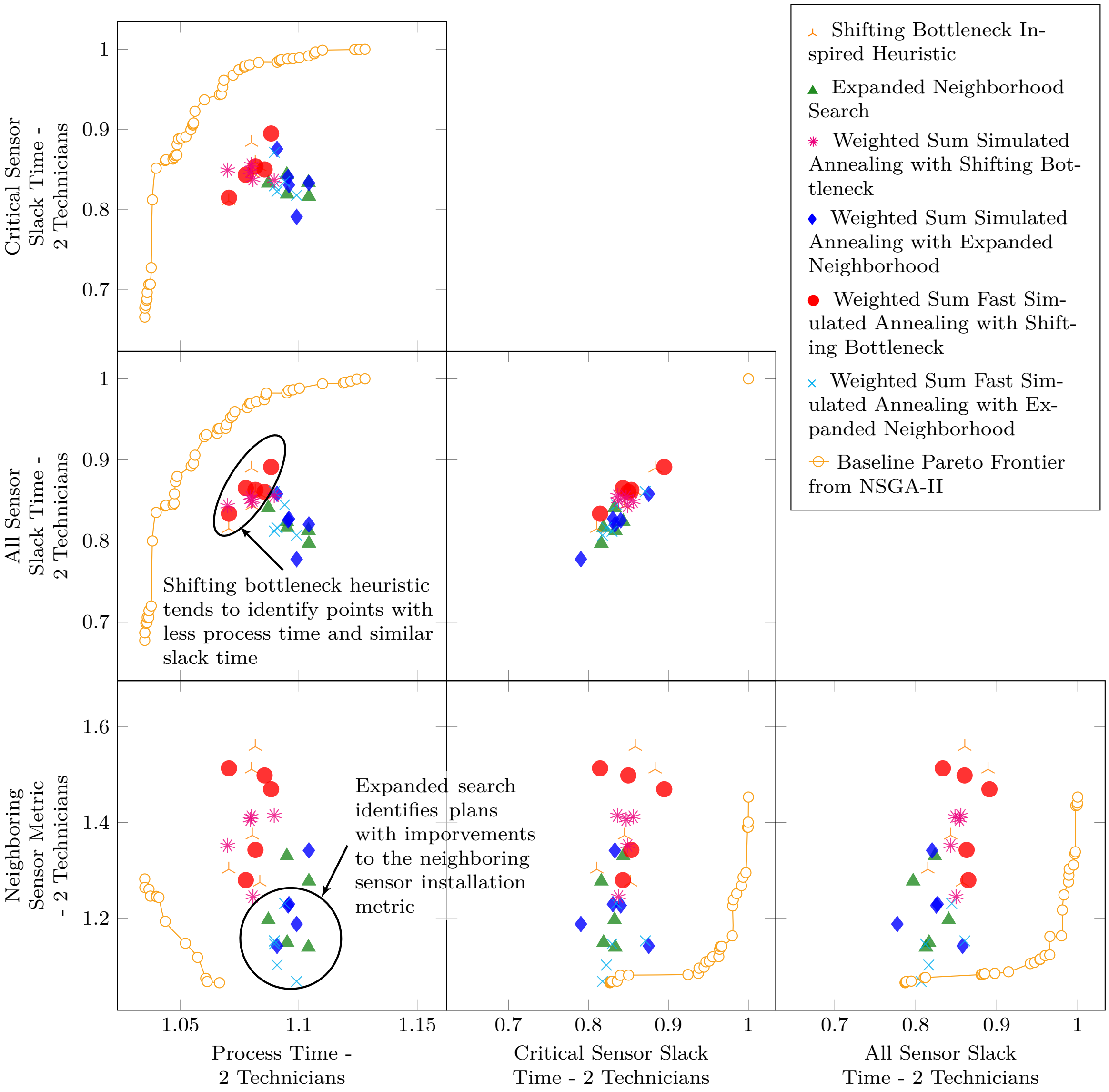


Figure 62: Weighted Sum Optimization Comparison with 2 Technicians

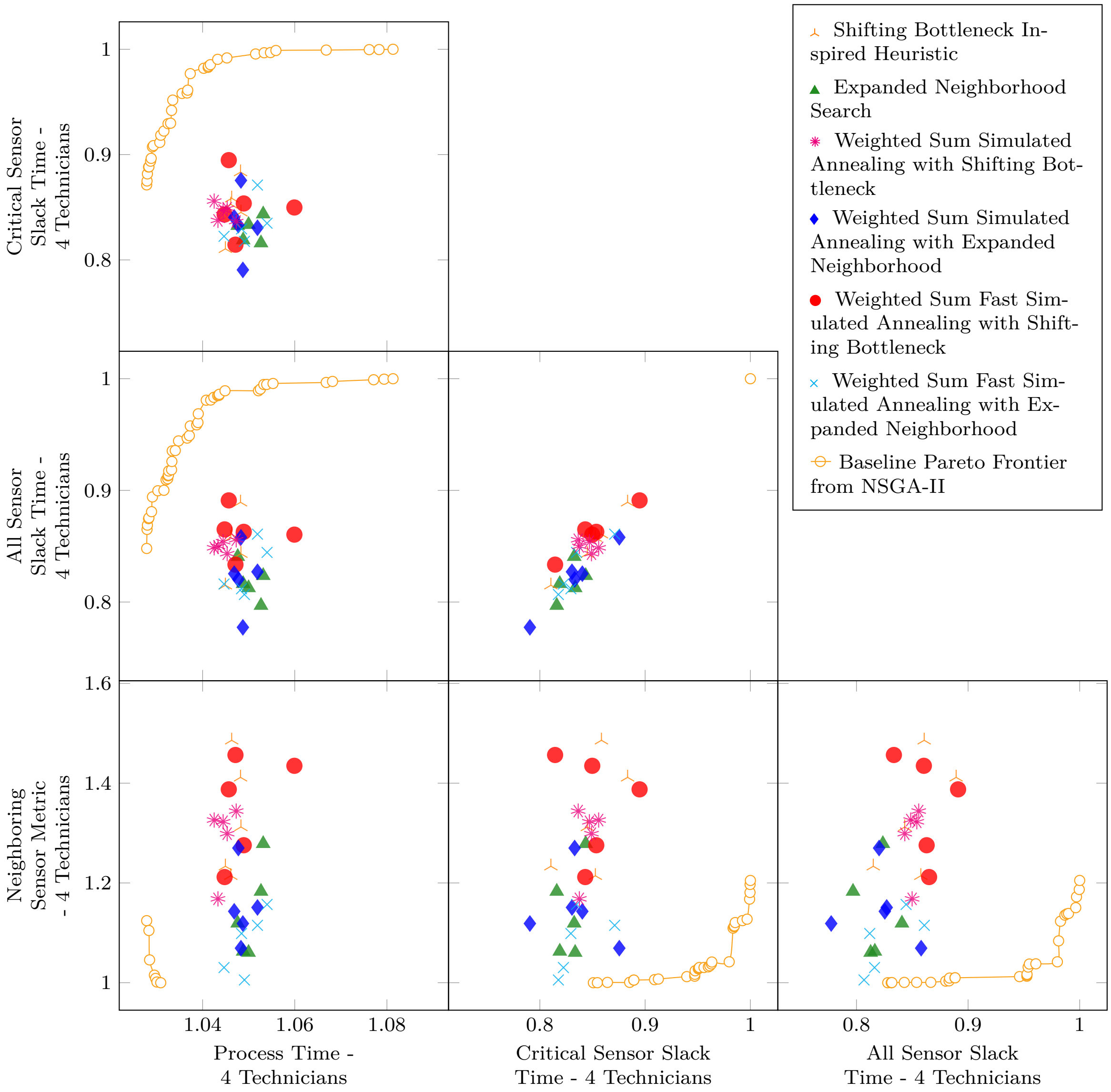


Figure 63: Weighted Sum Optimization Comparison with 4 Technicians

the neighboring sensor installation metric. However, the expanded neighborhood algorithms seem to not be capable of making the small changes to the installation plan that would help incrementally reduce the process time to the levels achievable by the shifting bottleneck algorithms.

Table 16 presents the summarized results from the deterministic optimization runs. As discussed in Section 4.6.6.2, the closeness to the Pareto frontier (G) is calculated as follows. Each objective function is normalized between 0–1, where 0 represents the worst value found for the objective in the baseline Pareto frontier. A value of 1 corresponds to the best value found in the Pareto frontier. This normalization ensures that the closeness calculation does not favor one metric over another. With the frontier and points to examine normalized, the closest point in the baseline Pareto frontier to the current point is identified. The Euclidean distance between these points is then averaged across each trial for a single algorithm to identify the average closeness to the Pareto frontier. As such, (G) can be conceptually thought of as the average (across all objectives and trials) percent degradation from the baseline Pareto frontier to the trial points.

Based on the average distance to the Pareto frontier, the weighted sum fast simulated annealing with the expanded neighborhood algorithm performed the best. Many points identified by this algorithm approach the Pareto fronter in the neighboring sensor installation metric dimension. The next best algorithm (according to the average distance metric) is the regular weighted sum simulated annealing algorithm with the shifting bottleneck heuristic. This algorithm consistently identified points with improved process times though worsened values for the neighboring installation metric. Hence, these two algorithms are carried forward to investigate whether their performance is severely changed when different robustness measures are added to the formulation.

Table 16: Summary of Deterministic Down-Selection Results

Algorithm	**G** (Closeness to the Baseline Pareto Frontier)	Best Process Time - 2 Technicians	Best Process Time - 4 Technicians
Baseline: NSGA-II	N/A	1.052	1.037
Weighted Sum Fast Simulated Annealing with Expanded Neighborhood Heuristic	0.041	1.090	1.045
Weighted-Sum Simulated Annealing with Shifting Bottleneck Inspired Heuristic	0.049	1.070	1.043
Stochastic Shifting Bottleneck Inspired Local Search	0.058	1.070	1.045
Expanded Neighborhood Local Search	0.062	1.087	1.048
Weighted Sum Fast Simulated Annealing with Shifting Bottleneck Inspired Heuristic	0.067	1.071	1.045
Weighted Sum Simulated Annealing with Expanded Neighborhood Heuristic	0.067	1.091	1.047

5.3.2.2 Algorithm Performance with Robustness Considerations

The two selected metaheuristic strategies are used to optimize the system with robustness considerations. Three set of robustness measures are evaluated:

1. Median Process Time
2. Median and 80th Quantile of Process Time
3. Median and the Standard Deviation of Process Time

Each robustness measure is used in conjunction with the two selected metaheuristics. These combinations are then used in two optimization trials each; the starting points chosen are the same as the first two from the deterministic cases. This facilitates comparisons between the deterministic and stochastic results. Again, the process time is weighted 4 times as important as the other metrics. When the standard deviation is included, it is weighted such that it is equally as important as the neighboring sensor installation and slack time metrics. Finally, each optimization case is allowed to run for 1000 iterations (instead of 1500) to reduce the time required to complete the experiment. The relative performance of the algorithms can still be compared with the lower number of iterations.

Figures 64 and 65 present results from this experiment with 2 and 4 technicians respectively. Continuing the trend from Experiment 1.3, the inclusion of the robustness measures do produce meaningful improvements to the results. Some cases that optimize to reduce the standard deviation do produce the lowest standard deviations; however, this comes at the cost of higher process time. Additionally, the optimization utilizing the 80th quantile of process time does identify solutions with slightly lower 80th quantiles compared to those optimized with only the median. Ultimately, results from the deterministic optimization runs have very similar median and 80th quantile values for process time to some of the stochastic results (with improved slack time).

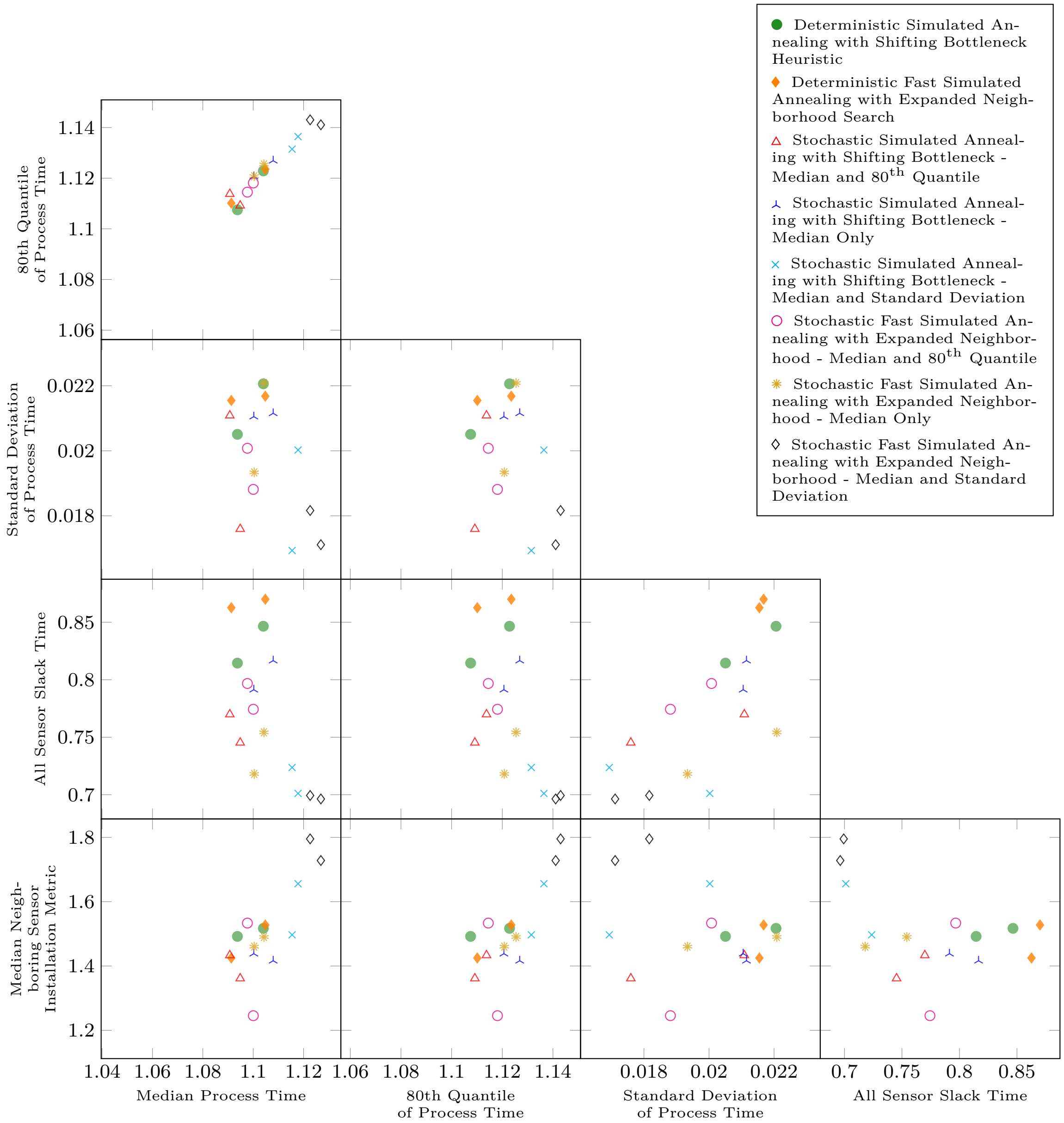


Figure 64: Final Points Found Using Various Metaheuristic Algorithms and Robustness Measures - 2 Technicians

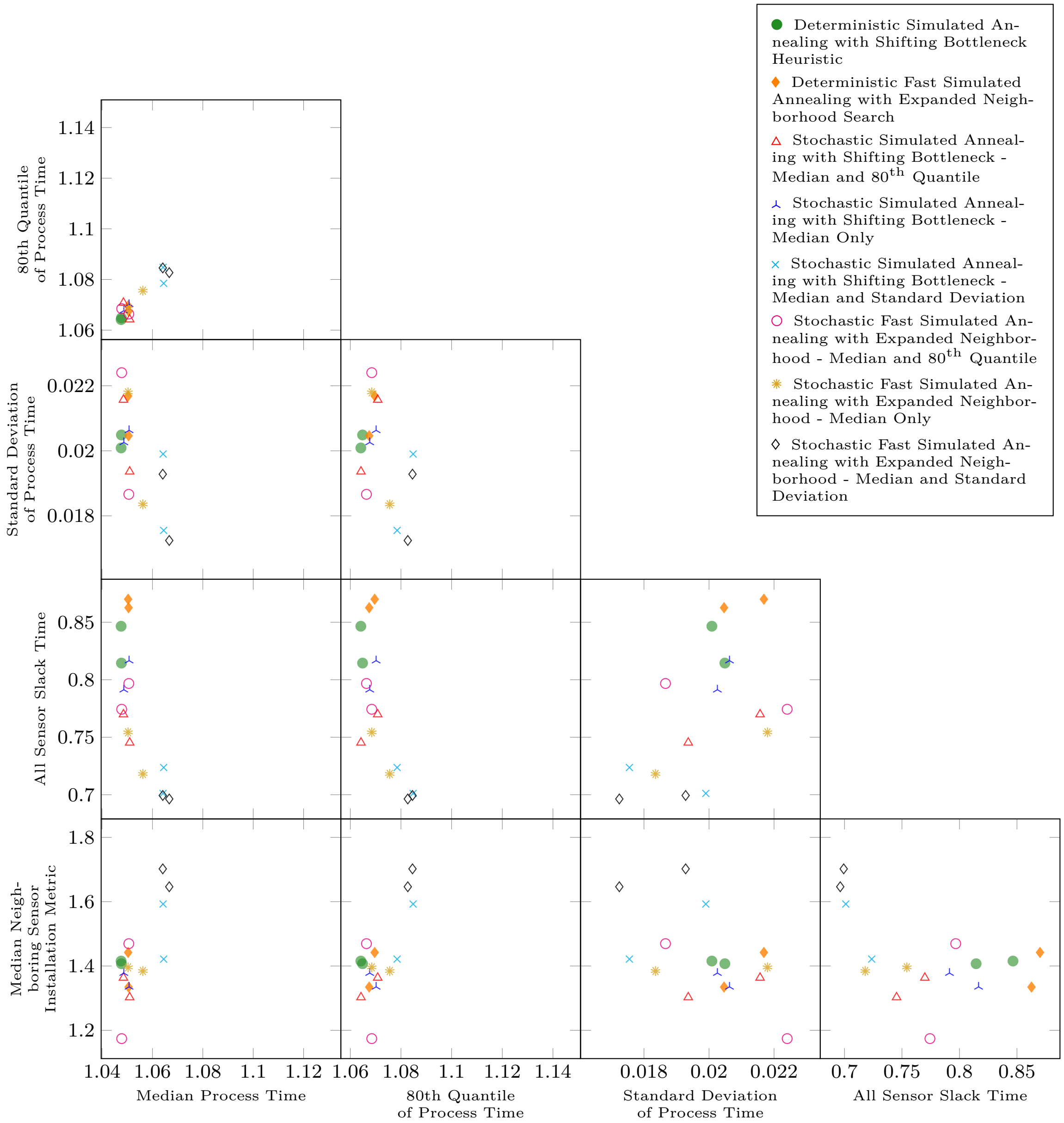


Figure 65: Final Points Found Using Various Metaheuristic Algorithms and Robustness Measures - 4 Technicians

This indicates that the observed trends may be due to the stochastic nature of the optimization algorithms and not the changes to the objective functions.

Similar to the challenges discussed in Section 5.1.3.3, the limited computational resources available preclude completing the number of optimization runs required to produce significant results. Also, as indicated by the initial results and observations throughout this work, the selected, real-world use case is not likely to receive benefits from including the robustness criteria. Some of the expected trends are observed (e.g. including the standard deviation can lead to solutions with a lower standard deviation), but the results are not conclusive. As discussed in Section 7.5, follow-up studies should run a larger number of cases with models of varying levels of uncertainty. This could serve to better justify the inclusion of quality robustness measures based on the nature of the problem and solution requirements. The following sections reviews the experimental results to draw conclusions about Hypothesis 2.2.

5.3.2.3 Experiment 2.2 Discussion

This experiment has been completed to test Hypothesis 2.2:

> **Hypothesis 2.2**: If alternative optimization strategies are implemented, then the methodology can be used to explore and exploit the solution space quickly enough to make implementation feasible and viable for a wider range of time and resource constraints and solution quality requirements.

To test this hypothesis, various optimization strategies and objective functions have been applied to the use case. Optimization techniques are primarily judged on the solution quality achieved and the time required to achieve these results; however, in any comparison of optimization techniques, it is recognized that the best strategy to implement is highly dependent on the specific problem at hand [211]. Even when considering the project scheduling problem of interest in this thesis, a range of problem specific characteristics can impact the choice of optimization strategy. Hence, the following characteristics are presented to frame the generated results:

Time Available/Computational Resources Available: The time and computational resources available to solve the optimization problem dictates the number of function evaluations available to reach a solution or set of solutions.

Complexity of the Solution Space: The complexity of the solution space impacts how easily an algorithm can identify better performing solutions and could influence the class of optimization algorithms that is likely to succeed.

Knowledge of the Desired Solution: Previous experience with similar problems may enable the decision maker to set constraints on the solution space or identify a weighted objective function that can identify good solution(s). This could lead to better performance over a purely unconstrained, non-dominated search [32].

Convergence Requirements: The required closeness of the final point(s) to the true optimum can also dictate the choice of optimization strategy.

These results work to marginally support Hypothesis 2.2. The results presented throughout this section indicate that a point-based algorithm applied to the deterministic model can identify solutions that are of decent quality when compared to the NSGA-II. These solutions, however, can be found using significantly less time and/or computational resources than the NSGA-II. Hence, if a proper weighting scheme can be identified from previous experience with similar problems, then the point-based strategies can work to reduce the turn-around time with limited losses in solution quality.

If additional time and/or resources are available, it may be better to use a population-based evolutionary algorithm (such as the NSGA-II). With a problem containing limited uncertainty, optimizing a deterministic model may be sufficient. However, as indicated by results from Experiment 1.3, running a limited number of generations with the stochastic model could help to refine the deterministic solutions.

While not explicitly observed in this work, systems with increased uncertainty have been shown to benefit from including robustness measures in the optimization [93, 218]. Hence, for a more complex or constrained system, the optimization with robustness measures has the potential to improve performance. The methodology's ability to quickly incorporate these additional objectives helps to broaden its applicability.

The experimental results are inconclusive; the use case model does not exhibit significant benefits from optimizing with robustness, and an exorbitant amount of time would be required to run more optimization cases with the resources available. This means that the results cannot confirm that a *wide range* of model characteristics can be supported by the various optimization strategies.

The following section introduces the results from Experiment 2, which are discussed in depth within Chapter 6.

5.4 Experiment 2: Methodology Implementation

Experiment 2 is designed to serve as a final review of the PORRTSS methodology's capability to support planners. As such, the experiment requires a thorough review of the actual steps taken to implement the methodology to solve a "real-world" problem. This includes describing how the methods and improvements developed and tested through Experiments 1 and 2.1–2.2 support the implementation. Hence, Chapter 6 provides a description of the methodology as applied to the use case with focus on how the developed capabilities have supported implementation. Additionally, Chapter 6 discusses the developed decision support system and provides examples of how it supports planners to make better decisions.

The following section summarizes the results from Experiments 1 and 2.1–2.2.

5.5 Chapter Summary

This chapter has presented the results from Experiments 1 and 2.1–2.2. Experiment 1 primarily tested whether Steps 1 & 2 of the PORRTSS methodology provide the capability to identify a detailed installation schedule that improves the robustness of the overall production plan. Experiment 1's sub-experiments are performed to systematically explore the modeling, optimization, and risk reduction capabilities implemented in Steps 1 & 2.

Experiment 1.1 is performed to test whether the discrete-event simulation paradigm, when combined with the automated model generation strategy, is capable of modeling the process flow to sufficient detail to enable optimization. The results demonstrate that the causal constraints and choices made within the schedule are properly accounted for by the simulation. The sensor installation logic, which is developed to quantify the impact that the sensor installations have on the overall production flow, is verified. Finally, the varying the number of parallel sensor installations allows and the compatibility assumptions produce the expected system-level changes. Increasing the number of parallel sensor installations allowed generally reduces the average and standard deviation of the process time. Furthermore, making the compatibility assumptions artificially constrained increases the impact of sensor installations. Furthermore, more constraining compatibility assumptions increases the impact stemming from sensor installations. These trends help to verify that the installation logic leads to the expected system-wide performance. These results, therefore, support Hypothesis 1.1 and demonstrate that the simulation is ready for integration with the optimization routine.

The ability of the implemented optimization strategy to improve the deterministic installation plan is examined in Experiment 1.2. The Non-dominated Sorting Genetic Algorithm-II is utilized to optimize the primary production process during which each sensor is to be installed. It is observed that the NSGA-II is able to quickly improve the population of sensor installation plans for all objectives of interest. The simulation-based optimization problem, therefore, is appropriately formulated. Hence, the observed results from this experiment support Hypothesis 1.2.

With the optimization strategy shown to successfully optimize the deterministic model, Experiment 1.3 is conducted to test whether directly evaluating and considering the quality robustness of the candidate schedules in the optimization strategy works to improve the general robustness of the plans. The optimization routine is shown to provide plans with improved robustness (as measured by the 80th quantile of the process time). The impact is especially seen in the 2-technician scenarios, which indicates that the inclusion is able to reduce risk with fewer resources. In addition to the improvement of the process time, the robust optimization strategy is still shown to provide a well-spread family of solutions for the decision makers to trade against.

When compared to the results from the deterministic optimization strategy, the robust optimization is shown to quickly make improvements to the Pareto frontier. Starting from the deterministic seed population, the robust optimization strategy identifies populations that are closer to the Pareto frontier in just 25 generations than the deterministic population in 300 generations. This indicates that the inclusion of the quality robustness metrics adds some useful information to the optimization routine; however, as the original strategy without robustness does eventually become close in performance to the strategy with robustness, the utility of including quality robustness is potentially limited. For the test problem at hand, it appears more economical to optimize deterministically. Then use the converged population to seed a small number of robust optimization cases.

Despite this caveat, Hypothesis 1.3 is supported by the experimental results. The information provided by the quality robustness evaluation is shown to help guide the optimization to improved plans with reduced risk. It is also shown to help the algorithm maintain a population with an increased percentage of Pareto optimal points.

From these results, the capabilities implemented in Steps 1 & 2 of the PORRTSS methodology are shown to work to improve the robustness of the production schedule. Discrete-event simulation is shown to be capable of modeling the system at the required level of detail to support optimization. The multi-objective, metaheuristic optimization routine is then shown to be capable of providing improved, robust installation plans in the objectives of interest. Thus, the results from Experiment 1 support the overarching Hypothesis 1.

With the initial capabilities developed for Steps 1 & 2 of the PORRTSS methodology investigated, Experiment 2 and its sub-experiments are focused on identifying promising areas to improve the deployability of the methodology. Experiment 2.1 investigates the impact of including the automated model generation strategy and flexible, metaheuristic optimization routine on the methodology setup time. The automated strategy, which is enabled by leveraging the object-oriented nature of the simulation language, is shown to more quickly and accurately build the simulation when compared to a manual process. Furthermore, automating the model generation helps to improve the scalability of the methodology: as demonstrated in the case of the largest production model considered in this research. Finally, the automation reduces the assumptions required to build the model and increases the information available from the simulation.

With the various models automatically generated, each are then implemented with the optimization routine. A generated model can begin optimization immediately, and changing objective functions is quite simple. The code-base used to evaluate the simulation model is also very modular, which enabled new metaheuristic algorithms to be incorporated in less than a day's worth of effort. Therefore, Experiment 2.1's results support the hypothesis that the object-oriented simulation, which enables automated model construction, combined with a metaheuristic algorithm helps to significantly reduce the method's implementation time.

The final Experiment discussed throughout this chapter is conducted to evaluate Hypothesis 2.2. Recognizing that this methodology may be applied to problems with wide-ranging complexities, time requirements, computational resources available, and solution quality requirements, it is desired to identify a number of optimization strategies that can support the deployment of the methodology in a larger number of instances and contexts. As objective function evaluations are relatively expensive, a number of point-based algorithms are explored with the deterministic algorithms. Observing that the 2 underlying heuristics led to identifying points in different regions of the solutions space, 2 algorithms are selected for further exploration with direct consideration for quality robustness. These algorithms are then used to evaluate the performance of different quality robustness quantifications. Following the trend observed in Experiment 1.3, the comparison of robustness measures is inconclusive. Hence, Hypothesis 2.2 cannot be confirmed because the benefit from including robustness in the optimization for a range of problems cannot be properly assessed.

This chapter has thus reviewed the capabilities developed to support the deployment of Steps 1 & 2 of the methodology. To better encourage adoption within industry, Chapter 6 discusses the entire use case. In doing so, the importance of the decision-maker to increasing the deployability of the methodology is investigated.

CHAPTER VI

USE CASE & DECISION SUPPORT

The purpose of this chapter is to discuss the results of Experiment 2. This is accomplished by implementing the PORRTSS methodology in support of a “real-world” problem. Throughout, the capabilities identified in Hypothesis 2 (reproduced below) that are hypothesized to lead to the successful implementation of the methodology are discussed. As such, the chapter begins with a discussion on how the various stakeholders, described in Section 4.1, work together to implement the methodology. Then, the decision support tools developed to support the down-selection and manual modification of the plan in Step 3 of the methodology are discussed. The new trades enabled by the methodology are then presented in the context of the case study outlined in Section 3.2. Finally, results from Experiment 2 are examined to assess the appropriateness of the hypothesis.

> Hypothesis 2: If the methodology requires a low amount of implementation effort and is shown to provide clear benefits with acceptable increases in computation time over traditional scheduling methods while effectively integrating the knowledge of the human planner, then the methodology can be successfully implemented to solve “real-world” problems.

Experiment 2 strays from the purely quantitative evaluations of the previous experiments to a more qualitative discussion and evaluation of the PORRTSS methodology with subject matter experts. While this removes some objectivity from the experimental evaluation, the evaluation is based on comments and feedback from avionics, manufacturing, and industrial engineers who are working with the vehicle and have used the methodology to plan sensor installations for portions of the vehicle.

Therefore, in lieu of an objective measure for implementability, the feedback received from discussions with the set of decision makers who have used the methodology is the next best data source for this experiment.

Because this experiment is designed to examine the actual use of the PORRTSS methodology, a description of the entire methodology's process is provided in this section. Most components have been covered in the discussion of previous experiments; hence, while the entire methodology is presented, previously examined details are summarized. Finally, updates to the methodology and decision support environment have been included following feedback received after its implementation. Hence, the updated version of the decision support tool is presented. Notes on feedback received from the SMEs are included throughout the experimental discussion.

The following section describes the steps taken by the various stakeholders to support Step 1 of the PORRTSS methodology.

6.1 Compatibility Matrix Development

One of the major steps in the PORRTSS methodology is the development of a compatibility matrix for the sensors and primary processes of interest. The matrix is created by the relevant subject matter experts (the manufacturing engineers and avionics experts for the case study) using the compatibility matrix tool described in Section 4.3.2. This follows a constraint identification process whereby the system experts (the avionics experts) work with the manufacturing engineers to understand how the vehicle must be assembled to obtain the required system performance. Using the compatibility matrix generation tool, the avionics and manufacturing engineers were able to create a compatibility matrix for a major subassembly of the vehicle of interest by exploring the sensor installations within the CAD model. This results in a traceable set of constraints to be used to drive the simulation model and optimization routine. The generation of the simulation model utilizing the compatibility matrix and schedule information provided by the IEs is discussed in the following section.

6.2 Automated Model Generation Deployment

The methodology implementor is first provided with the required data from the IE and avionics personnel to begin the model generation process. The industrial engineers provide optimized results from their schedule model to form the basis of the production model. The implementor then runs the schedule-to-Simio translation code described in Section 4.2.1. This code generates an input spreadsheet to place objects in the previously developed shell Simio model. Furthermore, input data tables for the simulation are created for the scheduled tasks, sensor installations, and compatibility matrix.

The input data sets required to generate the model are readily available to the stakeholders. The schedule model requires a simple export from the industrial engineers' constraint-based scheduling models. The sensor information is kept within a large database that can be filtered based on the portion of the vehicle currently under investigation. Then, because there are a relatively few number of sensor types, generic installation sequences and times for each type are identified. Therefore, through automation, this portion of the methodology requires little input from the primary stakeholders.

With the simulation model and constraint matrix developed, the next section describes the deployment of the optimization routine to identify a Pareto efficient set of solutions during Step 2 of the PORRTSS methodology.

6.3 Simulation-based Optimization Deployment

The simulation-based optimization routine is developed such that it can be immediately deployed once a simulation model and compatibility matrix are created. As such, there is not a significant amount of effort that is required for the optimization to be deployed for a new problem. Thus, the optimization routine is implemented with the objectives described in Table 7 (process time, critical and all sensors slack time, and the neighboring sensor installation metric). The statistics chosen to summarize each trial for the case study are the median and 80th quantile of the replications. These are chosen because the stakeholders are interested in minimizing the impact (the median process time) and risk (80th quantile of process time) resulting from the installation of sensors. Risk is further reduced by including the slack time (to allow for a larger window for potential re-installations) and the neighboring sensor metric (to make installations easier for technicians). The neighboring installation metric was identified by the manufacturing engineers as a way to make the installation processes easier for the technicians and was, therefore, added after analyzing an initial set of optimization results.

With the optimization routine implemented, the problem is optimized utilizing a 2-staged approach. Because the objective evaluations are expensive, an initial set of generations are run without considering the uncertainty in the problem. The goal is to generate a set of solutions that are nominally well-performing to be used to seed the stochastic optimization runs. After the deterministic optimization stop significantly improving, which happens at around generation 500 (after 3 days of computation time on a quad core i7 desktop with 16GB of memory), the stochasticity is re-enabled to evaluate the median and 80th quantile of process time. The entire set of inputs from the final deterministic runs are re-evaluated by running the 35 required replications (as discussed in Section 4.3.5). In an effort to enhance the genetic diversity, an additional 200 random population members are added to the 1000 original members.

The stochastic optimization formulation is then run until it again stops improving or reaches a maximum generation/time limit. The optimization runs are distributed over a network of computing nodes to reduce the time required to compute the objective function values. Results from Experiments 1.2 and 1.3 demonstrate that the deterministic model's process time is similar to the median process time in the stochastic model. Hence, following this strategy to first identify deterministically well-performing solutions to then refine with the quality robustness evaluations helps to improve efficiency.

Upon completing the optimization runs, the resulting Pareto efficient points are propagated to the decision support environment for down-selection and modification in Step 3 of the PORRTSS methodology. For the use case implemented by the stakeholders, the stochastic portion of the methodology was not included. This is because there was a small amount of time available to obtain a set of results. Therefore, while the following sections discussing the down-selection process include stochastic elements for illustration of the trades made available, these were not used in the actual implementation.

6.4 Sensor Installation Scenario Down-Selection and Manual Modification

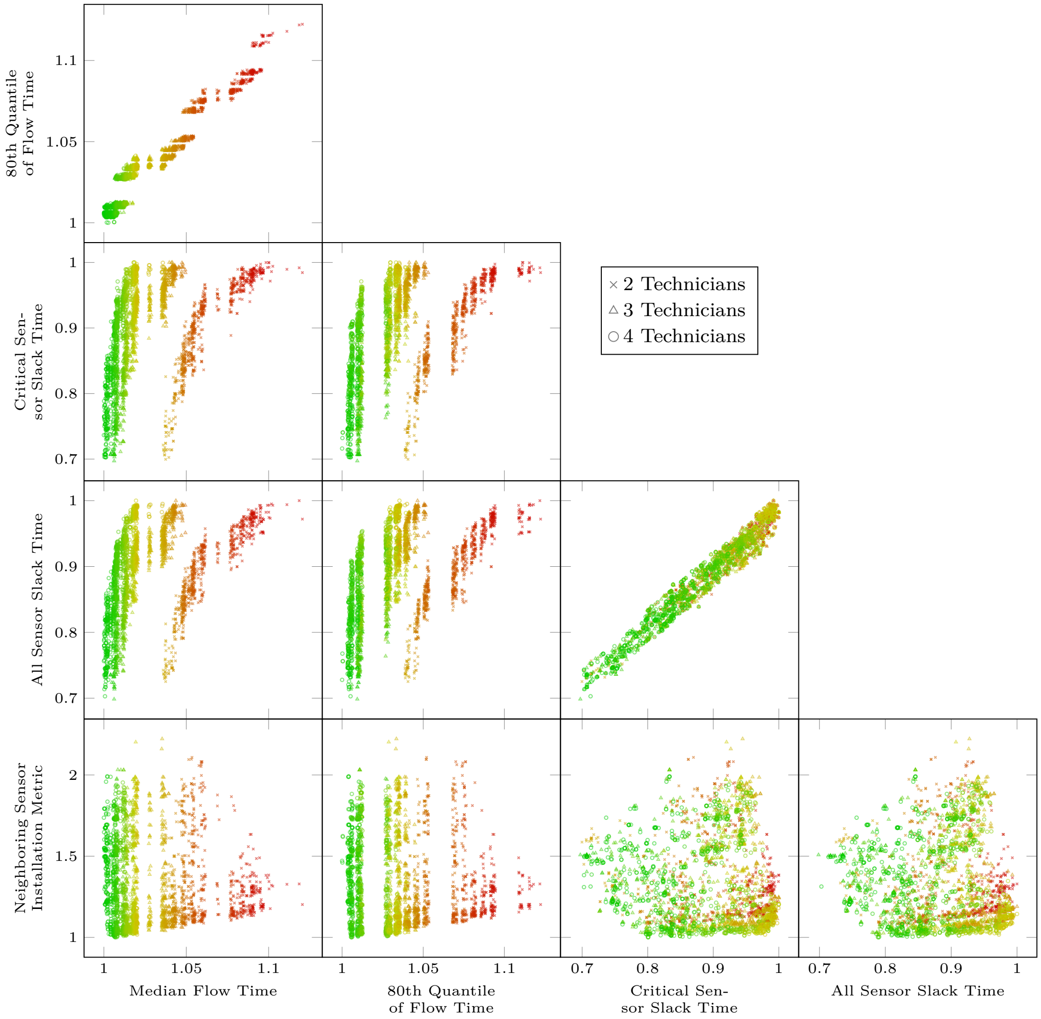
This section walks through the down-selection process (Step 3 of the methodology) as followed by the methodology's stakeholders when planning for sensor installations on the "real-world" vehicle. As such, a step-by-step progression through the process is presented with accompanying visuals. Step 3 begins with an initial down-selection from the set of potentially thousands of Pareto optimal points to a smaller subset to be carried through for further analysis.

6.4.1 Initial Down-Selection

As discussed in Section 4.4, the first step in the down-selection process is to explore the solution space to identify objectives of importance and regions of well-performing points. At each stage of this initial down-selection, top-ranking points can be selected for further, detailed scenario comparison. This detailed scenario comparison is

(a) Ten Best Potential Installation Plans Identified by TOPSIS Based Solely on Flow Time

Index	Median Flow Time	80th Quantile Flow Time	All Sensor Slack Time	Critical Sensor Slack Time	Technicians	Neighboring Sensor Metric
4,512	1	1.006	0.756	0.737	4	1.737
3,951	1	1.005	0.787	0.789	4	1.648
840	1	1.005	0.735	0.707	4	1.498
1,638	1	1.005	0.735	0.707	4	1.498
3,462	1	1.006	0.78	0.765	4	1.549
5,274	1	1.006	0.797	0.779	4	1.793
4,302	1	1.006	0.78	0.765	4	1.549
6,102	1	1.01	0.789	0.763	4	1.602
4,995	1	1.006	0.779	0.765	4	1.55
3,225	1	1.004	0.714	0.703	4	1.495



(b) Initial Scatterplot Matrix with Points Colored Based on Flow Time

Figure 66: Initial Down-Selection Visuals

discussed in Section 6.4.3.

Figure 66 shows the initial table of ranked solutions and corresponding scatterplot matrix. The values presented in Figure 66a are normalized by the best values found for each objective. The points are initially ranked, using a TOPSIS algorithm, solely by the median flow time. Flow time, which is the process time including breaks, nights, and weekends, is used to measure the impact of the sensor installations as it is a more common metric in planning. Process time is used to drive the optimization because it is a continuous function (flow time has discontinuities) and, therefore, provides a better objective for the optimizer. All of the values are normalized by the best value found during the optimization run. Hence, the ideal point has a value of 1 for each objective. Finally, because the median and 80th quantile of the flow time are normalized by their respective best values, it is possible to have an 80th quantile with a lower normalized value than the median's normalized value.

The initial weighting that solely focuses on the flow time metric is not extremely helpful to drive the down-selection process. It does not help to identify any regions that are well-performing in a multi-objective sense; it only shows the user that if cost and the other risk metrics (slack time and the neighboring sensor distance metric) are not a concern, then the user should select the solution with the lowest flow time regardless of any additional metrics. However, for comparison purposes, a couple of the points with the minimum flow time are selected for later scenario comparison.

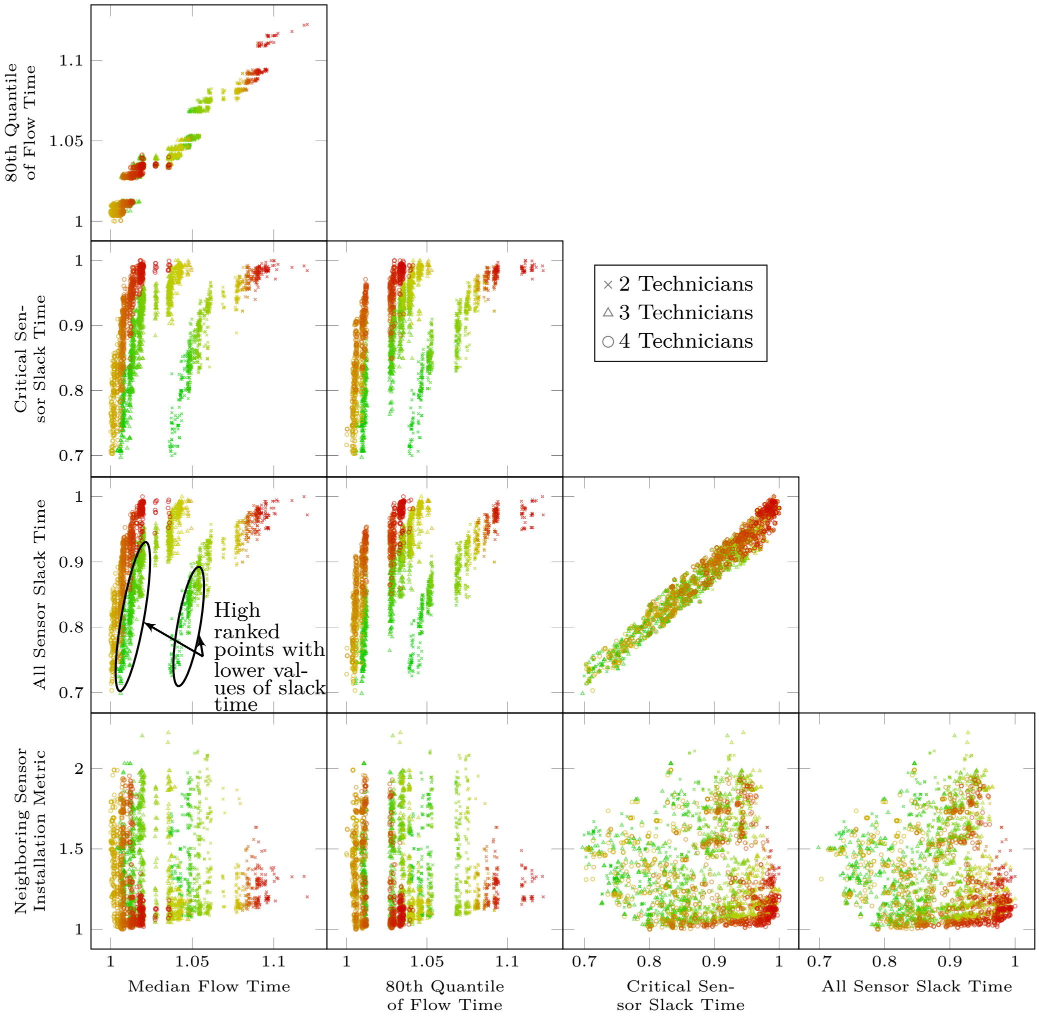
To further the exploration and begin to identify a potentially better set of points,

Table 17: TOPSIS Algorithm Weightings when Considering Flow Time and Cost

Response	**Weighting**
Median Flow Time	5.0
Slack Time for All Sensors	0
Slack Time for Critical Sensors	0
Technicians (Cost)	0.7
Neighboring Sensor Metric	0
80th Quantile of the Flow Time	0

(a) Ten Best Potential Installation Plans Identified by TOPSIS Considering Flow Time and Cost (Technicians) Based on Weightings Defined in Table 17

Index	Median Flow Time	80th Quantile Flow Time	All Sensor Slack Time	Critical Sensor Slack Time	Technicians	Neighboring Sensor Metric
6,187	1.036	1.04	0.743	0.721	2	1.609
6,175	1.036	1.039	0.737	0.715	2	1.596
2,581	1.036	1.039	0.737	0.713	2	1.605
5,920	1.037	1.045	0.73	0.706	2	1.593
6,178	1.037	1.04	0.735	0.711	2	1.591
1,924	1.037	1.039	0.728	0.703	2	1.608
2,755	1.037	1.039	0.74	0.718	2	1.633
5,416	1.037	1.039	0.731	0.706	2	1.602
5,893	1.037	1.041	0.781	0.758	2	1.719
3,142	1.037	1.041	0.758	0.75	2	1.385



(b) Scatterplot Matrix with Points Colored from Green to Red Based on their TOPSIS Rank From Weightings Defined in Table 17

Figure 67: Down-Selection Visuals Considering Flow Time and Cost (Number of Technicians)

cost should be brought into the equation. This is accomplished by adding importance to the *Technicians* metric for the TOPSIS algorithm. Table 17 shows the weightings chosen for this investigation. The technicians metric represents the number of sensor installation technicians available to install sensors, and it can be seen as a surrogate for the labor costs required to install the sensors.

The resulting TOPSIS rankings and scatterplot matrix can be seen in Figure 67. Comparing Figure 67a to 66a, adding the slight weighting to the technicians metric leads to ranking solutions that utilize only 2 technicians scoring better than ones requiring 4 technicians. This is because 2 technicians are able to complete some of the installation plans with only ~3.5% more flow time than the best scenarios found utilizing 4 technicians. Hence, based on the chosen weightings, sacrificing a small amount from the median flow time is worth halving the number of technicians. Some of these top-ranked points are again selected for future comparison.

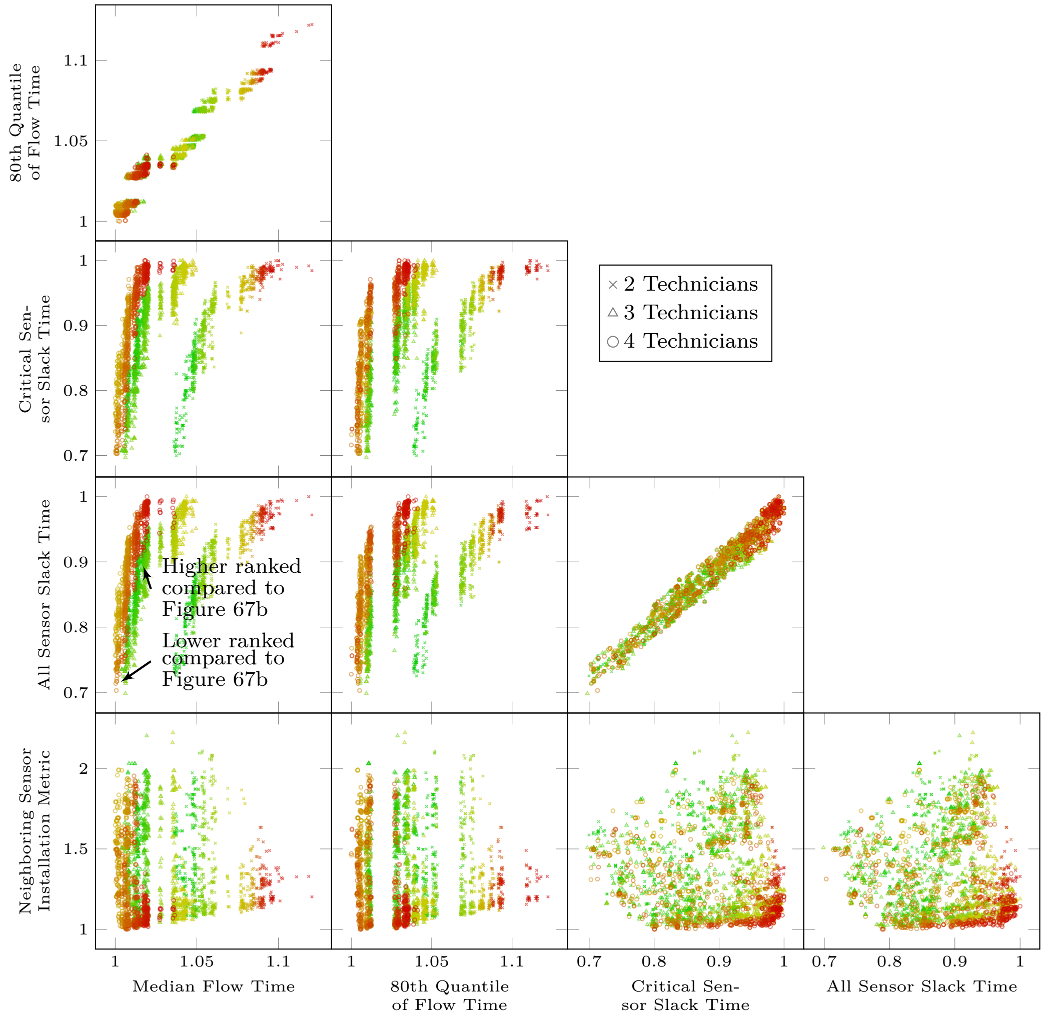
One interesting trend identified in Figure 67b is that many of the highest ranking points have a relatively low amount of slack time. This indicates that there are positions later in the process that nominally have good opportunities to install sensors, but this must be balanced with the desire to have a larger amount of slack time to provide more opportunities for re-installations. Therefore, it is interesting to add importance to these heuristic risk-reduction metrics to see how the distribution of highly ranked points changes.

Table 18: TOPSIS Algorithm Weightings when Considering Flow Time, Cost, and the Slack Time for All Sensors

Response	**Weighting**
Median Flow Time	5.0
Slack Time for All Sensors	0.5
Slack Time for Critical Sensors	0
Technicians (Cost)	0.7
Neighboring Sensor Metric	0
80th Quantile of the Flow Time	0

(a) Ten Best Potential Installation Plans Identified by TOPSIS Considering Flow Time, Cost (Technicians), and the Slack Time for All Sensors Based on Weightings Defined in Table 18

Index	Median Flow Time	80th Quantile Flow Time	All Sensor Slack Time	Critical Sensor Slack Time	Technicians	Neighboring Sensor Metric
4,279	1.037	1.045	0.813	0.826	2	1.36
5,776	1.037	1.042	0.788	0.766	2	1.701
5,893	1.037	1.041	0.781	0.758	2	1.719
1,990	1.038	1.04	0.786	0.773	2	1.628
5,725	1.04	1.045	0.812	0.825	2	1.363
6,097	1.038	1.041	0.772	0.751	2	1.343
6,187	1.036	1.04	0.743	0.721	2	1.609
3,142	1.037	1.041	0.758	0.75	2	1.385
2,422	1.037	1.041	0.758	0.75	2	1.385
3,202	1.037	1.042	0.759	0.75	2	1.383



(b) Scatterplot Matrix with Points Colored from Green to Red Based on their TOPSIS Rank From Weightings Defined in Table 18

Figure 68: Down-Selection Visuals Considering Flow Time, Cost (Number of Technicians), and the Slack Time for All Sensors

To begin considering the additional risk metrics, weight is added to the *Slack Time for All Sensors*. The updated weightings are provided in Table 18. By adding a relatively low weighting to the slack time when compared to the flow time, the TOPSIS algorithm is identifying points that have a larger amount of slack time with low flow times.

The resulting TOPSIS rankings based on the weightings defined in Table 18 are presented in Figure 68. The general regions of well-performing points identified in Figure 68b are very similar to the regions in Figure 67b. The plots depicting the slack time vs. the flow time show a slight shift towards favoring points with slightly higher flow time but larger amounts of slack.

Comparing the tables with the top 10 installation plans (Figure 68a vs. Figure 67a), one can see that all of the rows in Figure 68a except point 6197 are colored green. This shows that most points in the list moved up in its ranking based on the change in weighting scheme. Indeed, the best points identified in Figure 68a have about the same amount of flow time and use only 2 sensor installation technicians when compared to the previous weighting scheme (Figure 67a), but generally have an increased amount of slack time. Hence, no true trade has been made thus far: by allowing for a ~4% increase in the flow time, the number of technicians required can be halved while the slack time for all sensors is about 85% of the best value found.

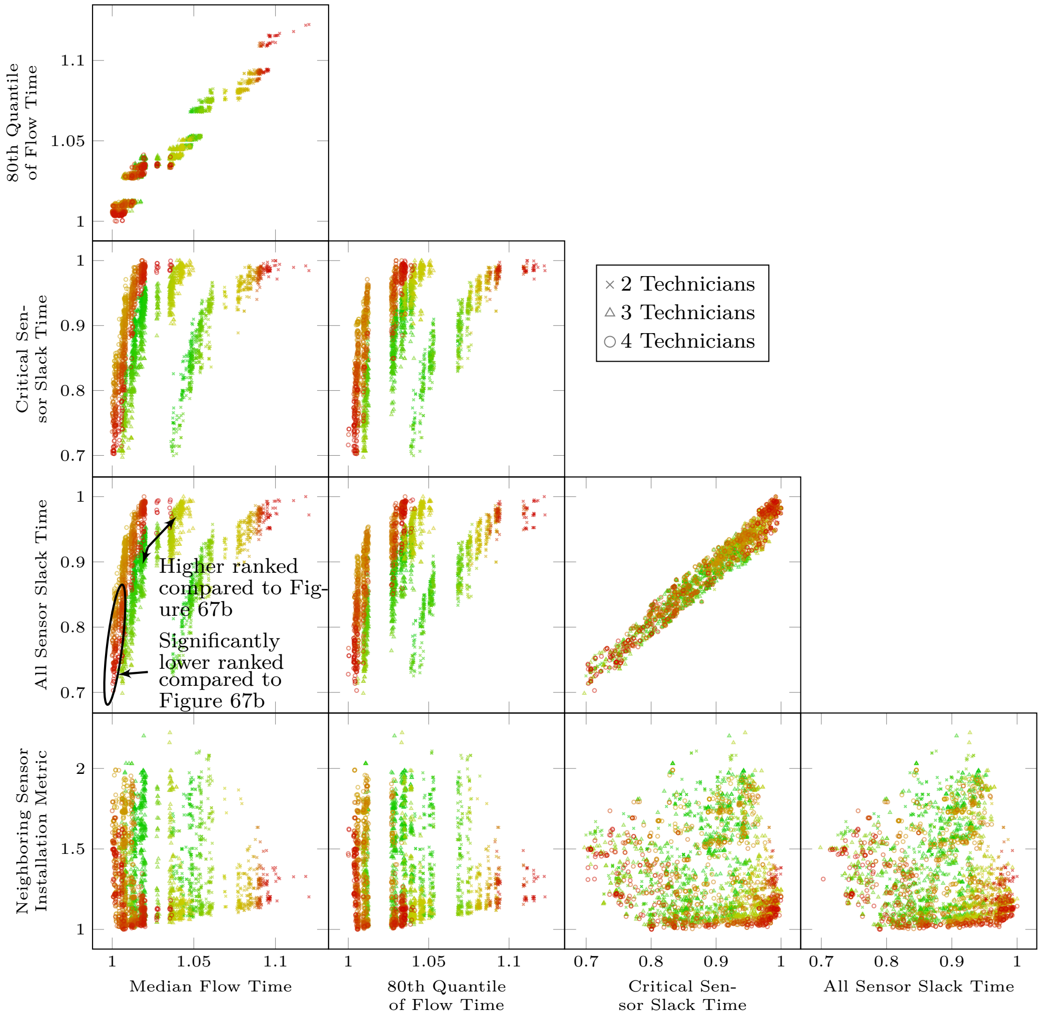
The down-selection process can continue by adding importance to the *Slack Time*

Table 19: TOPSIS Algorithm Weightings when Considering Flow Time, Cost, and the Slack Time for All and Critical Sensors

Response	**Weighting**
Median Flow Time	5.0
Slack Time for All Sensors	0.5
Slack Time for Critical Sensors	0.5
Technicians (Cost)	0.7
Neighboring Sensor Metric	0
80th Quantile of the Flow Time	0

(a) Ten Best Potential Installation Plans Identified by TOPSIS Considering Flow Time, Cost (Technicians), and the Slack Time for All and Critical Sensors Based on Weightings Defined in Table 19

Index	Median Flow Time	80th Quantile Flow Time	All Sensor Slack Time	Critical Sensor Slack Time	Technicians	Neighboring Sensor Metric
4,279	1.037	1.045	0.813	0.826	2	1.36
5,725	1.04	1.045	0.812	0.825	2	1.363
2,734	1.044	1.05	0.856	0.845	2	1.84
698	1.014	1.034	0.928	0.94	3	1.651
5,792	1.014	1.035	0.919	0.922	3	1.759
6,103	1.043	1.045	0.828	0.844	2	1.322
3,079	1.042	1.047	0.82	0.825	2	1.388
1,657	1.044	1.047	0.837	0.852	2	1.711
662	1.016	1.035	0.929	0.942	3	1.657
1,990	1.038	1.04	0.786	0.773	2	1.628



(b) Scatterplot Matrix with Points Colored from Green to Red Based on their TOPSIS Rank From Weightings Defined in Table 19

Figure 69: Down-Selection Visuals Considering Flow Time, Cost (Number of Technicians), and the Slack Time for All and Critical Sensors

for Critical Sensors metric. This serves to add importance to the overall slack metric (because the critical sensor slack contributes to a significant portion of the all sensor slack) while also prioritizing the additional slack time for the critical sensors. Table 19 presents the weightings for this scenario.

The resulting ranked table and scatterplot matrix based on the weightings defined in Table 19 are presented in Figure 69. Figure 69b shows that scenarios with slack times near the top of the range of values observed are becoming more favored. This can be seen further in Figure 69a where scenarios utilizing 3 technicians are beginning to enter the top 10 ranked solutions. By using 3 technicians, these scenarios are able to complete more sensor installations early in the flow (to improve the slack time) while not impacting the flow time. Therefore, based on the preferences defined to this point, the selection between 2 and 3 technicians available to the flow is important and requires additional attention. As such, top ranked points from this table that utilize both 2 and 3 technicians are selected for further analysis.

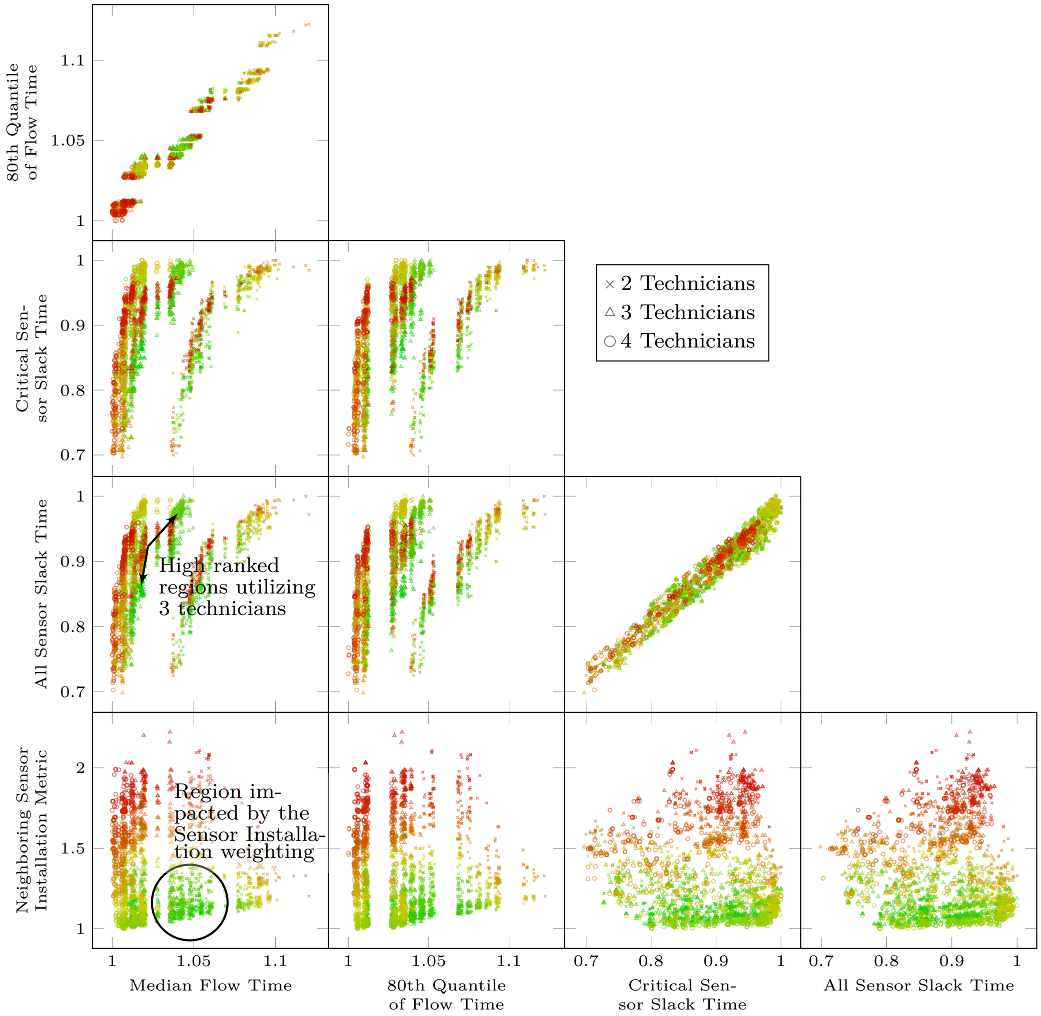
The results shown in Figure 69a demonstrate that, notably for 2 technician scenarios, the neighboring sensor installation metric is fairly high. Hence, considering this metric in the TOPSIS algorithm could help identify points that make the actual installations easier for the technicians. Therefore, the weightings are updated to those presented in Table 20. A weighting of 1.0 is selected to put the neighboring sensor installation metric on an approximately equal footing with the slack time metrics.

Table 20: TOPSIS Algorithm Weightings when Considering Flow Time, Cost, the Neighboring Sensor Installation Metric, and the Slack Time for All and Critical Sensors

Response	**Weighting**
Median Flow Time	5.0
Slack Time for All Sensors	0.5
Slack Time for Critical Sensors	0.5
Technicians (Cost)	0.7
Neighboring Sensor Metric	1.0
80^{th} Quantile of the Flow Time	0

(a) Ten Best Potential Installation Plans Identified by TOPSIS Considering Flow Time, Cost (Technicians), the Slack Time for All and Critical Sensors, and the Neighboring Sensor Installation Metric Based on Weightings Defined in Table 20

Index	Median Flow Time	80th Quantile Flow Time	All Sensor Slack Time	Critical Sensor Slack Time	Technicians	Neighboring Sensor Metric
2,480	1.019	1.033	0.903	0.933	3	1.047
5,864	1.014	1.012	0.842	0.863	3	1.019
3,770	1.014	1.03	0.897	0.899	3	1.079
4,598	1.018	1.039	0.881	0.909	3	1.052
5,897	1.019	1.035	0.893	0.915	3	1.055
3,068	1.019	1.033	0.902	0.931	3	1.07
5,666	1.016	1.03	0.898	0.901	3	1.078
1,985	1.018	1.029	0.895	0.874	3	1.046
5,540	1.02	1.035	0.895	0.889	3	1.037
3,347	1.016	1.035	0.862	0.888	3	1.051



(b) Scatterplot Matrix with Points Colored from Green to Red Based on their TOPSIS Rank From Weightings Defined in Table 20

Figure 70: Down-Selection Visuals Considering Flow Time, Cost (Number of Technicians), the Slack Time for All and Critical Sensors, and the Neighboring Sensor Installation Metric

The resulting ranked solutions table and scatterplot matrix are presented in Figure 70. By including the neighboring sensor installation metric, the TOPSIS algorithm has identified more solutions utilizing 3 technicians that are able to provide enough benefit to be worth the additional costs. Therefore, by allowing for the additional costs, solutions are identified that have low flow times and slack times with low values for the neighboring sensor metric. Figure 70b shows a distinct shift towards favoring points using 3 technicians that have low flow times, low neighboring sensor metrics, and medium to high slack times. This sharply contrasts the more compromised solutions that utilized 2 technicians shown in Figure 69b.

With promising points identified through an investigation of the various metrics related to impact, cost, and heuristic risk, the direct impact of the schedule's quality robustness on the selection of promising points is investigated. The quality robustness for the case study is quantified as the 80th quantile of the flow time identified by summarizing results from multiple simulation replications. The weightings used for this portion of the investigation are presented in Table 21.

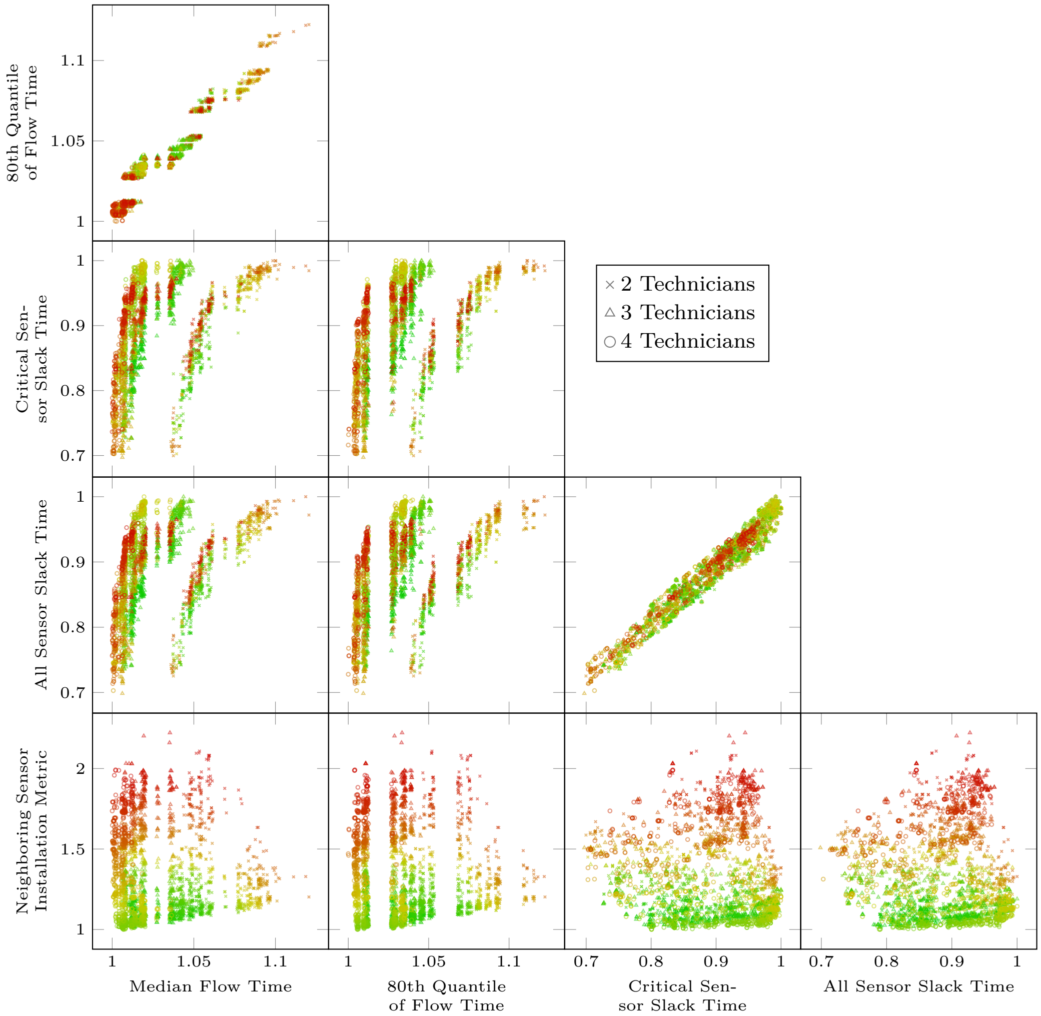
Figure 71 shows the results for the weightings in Table 21. As seen in Figure 71b, increasing the importance of the 80th quantile of the flow time did not significantly impact the regions of well-performing points. New properties can be seen in the ranked table (Figure 71a). As seen in the table, the new weightings have identified points that sacrifice slack time for more quality robustness (e.g. solutions in which

Table 21: TOPSIS Algorithm Weightings when Considering the Median and 80th Quantile of Flow Time, Cost, the Neighboring Sensor Installation Metric, and the Slack Time for All and Critical Sensors

Response	**Weighting**
Median Flow Time	5.0
Slack Time for All Sensors	0.5
Slack Time for Critical Sensors	0.5
Technicians (Cost)	0.7
Neighboring Sensor Metric	1.0
80th Quantile of the Flow Time	2.5

(a) Ten Best Potential Installation Plans Identified by TOPSIS Considering the Median and 80th Quantile of Flow Time, Cost (Technicians), the Slack Time for All and Critical Sensors, and the Neighboring Sensor Installation Metric Based on Weightings Defined in Table 21

Index	Median Flow Time	80th Quantile Flow Time	All Sensor Slack Time	Critical Sensor Slack Time	Technicians	Neighboring Sensor Metric
5,864	1.014	1.012	0.842	0.863	3	1.019
977	1.012	1.012	0.845	0.846	3	1.06
2,480	1.019	1.033	0.903	0.933	3	1.047
758	1.013	1.012	0.851	0.846	3	1.064
2,807	1.012	1.013	0.817	0.833	3	1.029
1,511	1.012	1.013	0.816	0.833	3	1.029
3,770	1.014	1.03	0.897	0.899	3	1.079
623	1.014	1.012	0.851	0.845	3	1.065
3,257	1.013	1.012	0.851	0.846	3	1.068
4,382	1.014	1.027	0.833	0.853	3	1.017



(b) Scatterplot Matrix with Points Colored from Green to Red Based on their TOPSIS Rank From Weightings Defined in Table 21

Figure 71: Down-Selection Visuals Considering the Median and 80th Quantile of Flow Time, Cost (Number of Technicians), the Slack Time for All and Critical Sensors, and the Neighboring Sensor Installation Metric

the median and 80$^{\text{th}}$ quantile of the flow time are similar).

This presents an interesting trade for the decision maker. The modeled uncertainty within the simulation identifies a point with a low 80$^{\text{th}}$ quantile of process time has a low amount of risk. The simulation is, however, limited to the types of risk and uncertainty modeled. This led to the inclusion of the slack time and neighboring sensor metric as heuristic alternatives, which help to capture some risk indicators without directly modeling their impact. Therefore, the decision maker must decide if he or she should accept a slightly increased amount of potential delays in the hope that the slack time can contribute to reduced risk. These decisions require further investigation of the selected points, which is discussed in Section 6.4.3 following the section summary.

6.4.2 Initial Down-Selection Summary

The initial set of trades presented above provides a good starting point for the down-selection process. By exploring a variety of weighting scenarios, the stakeholder is able to identify regions of the design space that contain promising solutions and select solutions for more detailed analysis.

While the scatterplot matrix and TOPSIS algorithm provide a solid platform to select a variety of potential points for implementation, the goal of the down-selection process is to identify a single scenario for implementation. The capabilities presented throughout this section lack the detailed information required to comfortably make this decision. As such, the goal of the initial down-selection is to identify a set of 10 or so points that are representative of the various promising regions identified by varying the TOPSIS weightings. The points selected by following the down-selection process discussed in this section are presented in Table 22. These representative points are then passed for more detailed analysis, as presented in the following section.

Table 22: Installation Plans Selected for Further Comparison (All Dimensions Normalized)

Index	**Median Flow Time**	**80^{th} Quantile Flow Time**	**All Sensor Slack Time**	**Critical Sensors Slack Time**	**Technicians**	**Neighboring Sensor Metric**
4,512	1	1.006	0.756	0.737	4	1.737
840	1	1.005	0.735	0.707	4	1.498
6,187	1.036	1.04	0.743	0.721	2	1.609
6,175	1.036	1.039	0.737	0.715	2	1.596
4,279	1.037	1.045	0.813	0.826	2	1.36
5,725	1.04	1.045	0.812	0.825	2	1.363
2,480	1.019	1.033	0.903	0.933	3	1.047
5,864	1.014	1.012	0.842	0.863	3	1.019
977	1.012	1.012	0.845	0.846	3	1.06
758	1.013	1.012	0.851	0.846	3	1.064

6.4.3 Scenario Comparison

The primary purpose of the scenario comparisons are to identify where and why delays are occurring in the selected points of interest. This transition from overarching values for the objective (flow time, slack time, etc.) to more detailed information pertaining to the delays at individual processes is important to increase traceability and transparency. For instance, with the information provided by the scenario comparisons, the stakeholder can better determine whether an observed increase in flow time can be mitigated (potentially by deploying more technicians for a specific set of processes) or is acceptable (because many sensors are installed during the delay). Using this information, he or she can better delineate between the promising scenarios to select a single scenario to carry forward.

The two visuals useful for the scenario comparison are a parallel coordinate plot and Gantt chart. When scenarios of interest are selected during the initial down-selection process, they are added to the installation delay parallel plot. Figure 25 presents a parallel plot for a segment of installation processes for the scenarios selected through the initial down-selection process. As a note, the process names are set before the analysis and are not fully indicative of the order of the processes; the location along the X axis does, however, give an indication of the sequence.

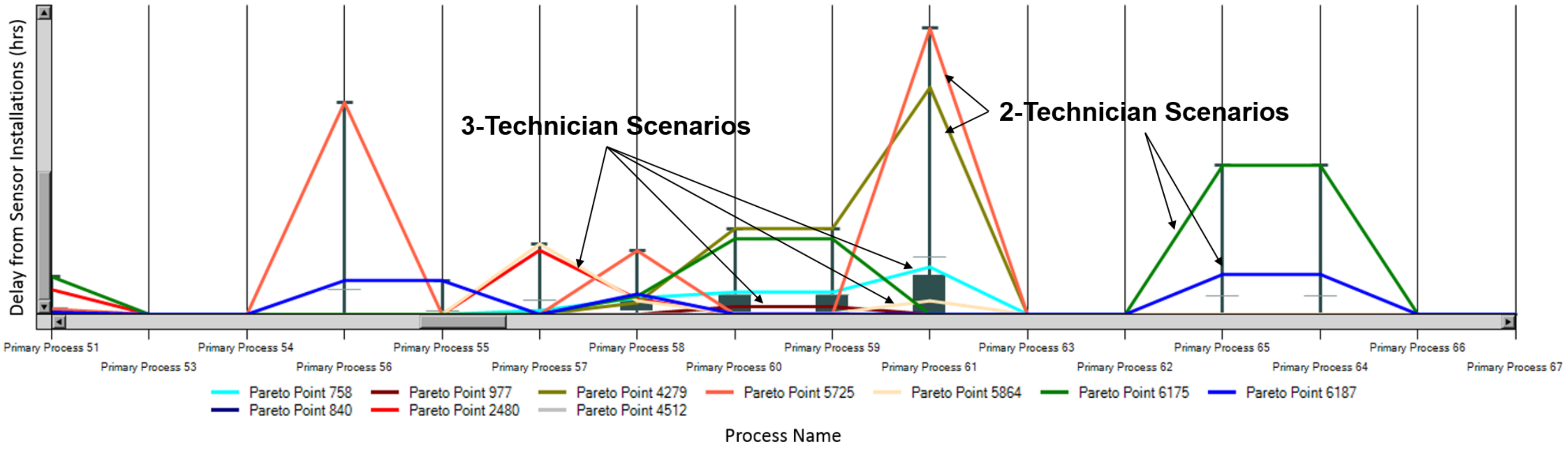


Figure 25: Parallel Plot Showing Delays to Each Primary Process Due to Sensor Installations for the Points Selected through the Initial Down-Selection (Reproduced from page 142)

The parallel plot allows decision makers to identify and assess, on a process-by-process basis, the delays caused by each sensor installation scenario. Ideally, a plan would be found that leads to no delays in the primary production flow; however, this is likely not possible due to the compatibility constraints. Instead, the optimization algorithm should identify plans that best distribute the sensor installations to minimize the overall amount of delay while also considering the heuristic risk metrics (slack time and the neighboring sensor installation metric).

The parallel coordinate plot presented in Figure 25 supports the application of human intuition and increases transparency in the planning process. These capabilities are identified in literature as essential to a deployable scheduling methodology [60, 206]. By showing how the sensor installations and resulting delays are distributed amongst the primary production processes, the decision maker is able to better understand how the optimizer identified the installation plans. By being provided with information about each sensor installed at each primary process, the user can then leverage his or her experience to better understand the severity of the delay or identify potential mitigation strategies.

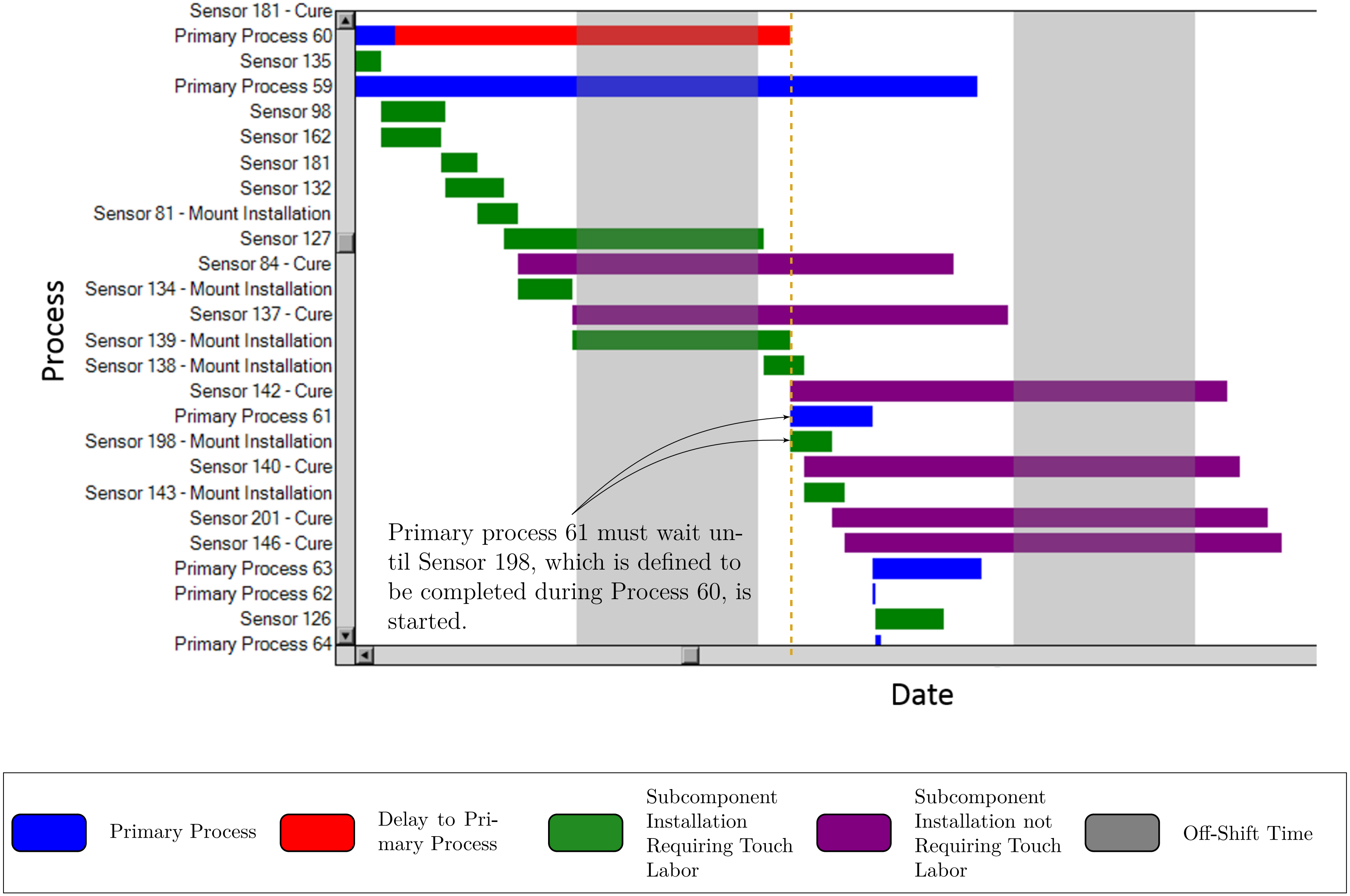
The parallel plot allows the user to scroll through the entire process to compare the scenarios of interest. From Figure 25, the decision maker can quickly see that Pareto points 5725, 4279, and 6175, which all utilize only 2 technicians, incur large delays in the presented processes. Using knowledge of the delayed process, the decision maker can begin to understand where and how the additional technicians reduce the impact of sensor installations on the flow. To further this understanding, the linked Gantt chart is utilized to identify further details about the selected points for comparison.

A Gantt chart specifically tailored to understand the impact of sensor installations is thus incorporated into the decision support environment. An example of the Gantt chart is presented in Figure 73. The Gantt chart is linked to the parallel plot such that when the user clicks on one of the parallel plot's lines, the Gantt chart automatically updates to display this scenario's information and zooms onto the specific process selected. For example, the Gantt chart in Figure 73 is displayed immediately when the line for Pareto point 4279 is selected in the parallel plot at Primary Process 61. This brushing capability enables the user to quickly zoom in on areas of interest or concern that he or she identifies in the parallel plot.

Using the Gantt chart displayed in Figure 73, the user is able to obtain a very detailed picture of why Primary Process 61 is delayed. Process 61, which is a successor of Process 60, must wait until the installation sequences for all of the sensors planned to occur during Process 60 are started. Once the last sensor sequence planned for Process 60 is started (Sensor 198), Primary Process 58 is allowed to begin because it is compatible with all of the ongoing sensor installations.

In using the parallel plot and linked Gantt charts, the decision makers are thus able to better understand the choices made by the optimizer to increase transparency. This information, when combined with the overall metrics provided by the scatterplot matrix and response table, allows the planners to down-select a strategy that forms the basis of the plan for implementation.

As demonstrated, the parallel plot and Gantt charts provide an overview of the strategies employed by each scenario to minimize the process delays while also improving the other metrics of interest. Additionally, the Gantt chart enables the user to better see why a delay occurs and to potentially make a judgment as to whether the delay is acceptable (e.g. if a large number of sensors are installed, it could be acceptable to delay the primary process for some time). Finally, in the implemented version of the decision support environment, the parallel plot is supplemented with information about each sensor installed during the process. The SMEs can then utilize this information to gain further insights about how the sensor installations impact the heuristic risk metrics and identify any additional problems or constraints that may not have been captured by the model.



Primary process 61 must wait until Sensor 198, which is defined to be completed during Process 60, is started.

Figure 73: Gantt Chart Displaying Results From Pareto Point 4279

At this point in the selection process, the user has identified, from the provided set of solutions, the single sensor installation plan that he or she prefers. The user has also contributed to the solution process by applying his or her preferences to make this selection. In order to add further freedom to the down-selection process and not limit the user to the solutions identified by the optimization routine, an opportunity to explore manual modifications to the plan is also provided, as discussed in the following section.

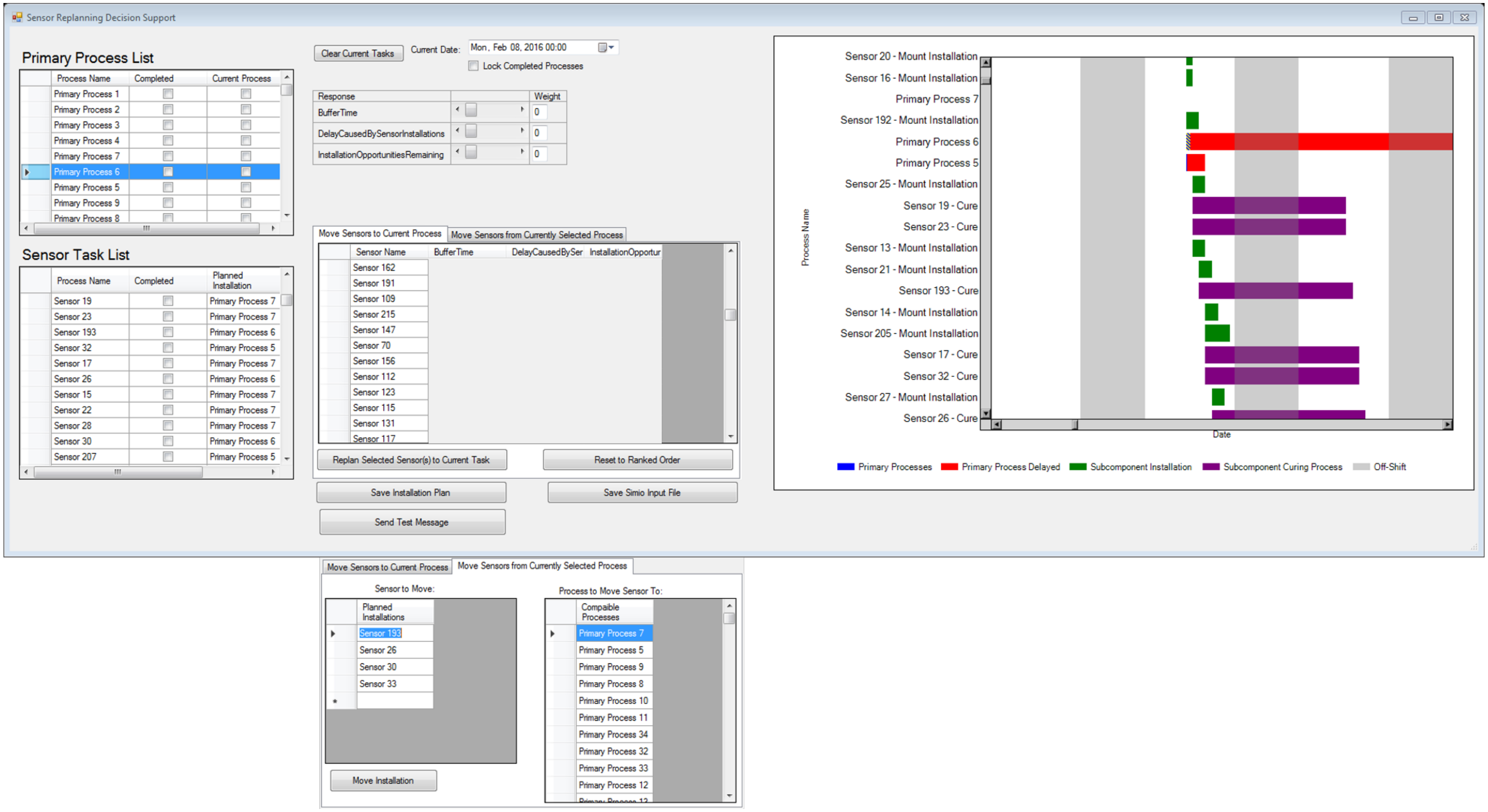
6.4.4 Scenario Modification

The constraints (compatibility matrix) and set of objective functions are implemented to guide the optimization algorithm. Despite the best efforts of the implementors and SMEs, these constraints and objectives can never fully capture the preferences and experience of the SMEs. As such, after the SMEs have narrowed the set of solutions down to a single solution of interest, the capability to easily modify the sensor installation plan to accommodate any additional knowledge from the human planners is provided.

This capability is provided in a re-planning view in the decision support environment. While the views discussed in the previous sections focused on ranking overall sensor installation scenarios, the re-planning view helps the user to sort through each sensor installation in a single sensor installation plan. An overall view of the re-planning view is shown in Figure 74.

The primary interfaces available in the re-planning view are: 1) lists of the primary processes and sensor installation processes, 2) an interactive Gantt chart that is linked to the process and sensor lists, 3) and a re-planning recommendation table. Each of these interfaces are discussed below.

The primary process and sensor installation task lists help the user navigate the Gantt chart. Clicking on either a sensor or primary process zooms to that area of



Sensor Replanning Decision Support

Primary Process List

Process Name	Completed	Current Process
Primary Process 1	☐	☐
Primary Process 2	☐	☐
Primary Process 3	☐	☐
Primary Process 4	☐	☐
Primary Process 7	☐	☐
Primary Process 6	☐	☐
Primary Process 5	☐	☐
Primary Process 9	☐	☐
Primary Process 8	☐	☐

Sensor Task List

Process Name	Completed	Planned Installation
Sensor 19	☐	Primary Process 7
Sensor 23	☐	Primary Process 7
Sensor 193	☐	Primary Process 6
Sensor 32	☐	Primary Process 5
Sensor 17	☐	Primary Process 7
Sensor 26	☐	Primary Process 6
Sensor 15	☐	Primary Process 7
Sensor 22	☐	Primary Process 7
Sensor 28	☐	Primary Process 7
Sensor 30	☐	Primary Process 6
Sensor 207	☐	Primary Process 5

Clear Current Tasks

Current Date: Mon, Feb 08, 2016 00:00

☐ Lock Completed Processes

Response		Weight
BufferTime		0
DelayCausedBySensorInstallations		0
InstallationOpportunitiesRemaining		0

Move Sensors to Current Process | Move Sensors from Currently Selected Process

Sensor Name	BufferTime	DelayCausedBySer	InstallationOpportur
Sensor 162			
Sensor 191			
Sensor 109			
Sensor 215			
Sensor 147			
Sensor 70			
Sensor 156			
Sensor 112			
Sensor 123			
Sensor 115			
Sensor 131			
Sensor 117			

Replan Selected Sensor(s) to Current Task

Reset to Ranked Order

Save Installation Plan

Save Simio Input File

Send Test Message

Sensor 20 - Mount Installation
Sensor 16 - Mount Installation
Primary Process 7
Sensor 192 - Mount Installation
Primary Process 6
Primary Process 5
Sensor 25 - Mount Installation
Sensor 19 - Cure
Sensor 23 - Cure
Sensor 13 - Mount Installation
Sensor 21 - Mount Installation
Sensor 193 - Cure
Sensor 14 - Mount Installation
Sensor 205 - Mount Installation
Sensor 17 - Cure
Sensor 32 - Cure
Sensor 27 - Mount Installation
Sensor 26 - Cure
Process Name
Date
Primary Processes
Primary Process Delayed
Subcomponent Installation
Subcomponent Curing Process
Off-Shift

Move Sensors to Current Process | Move Sensors from Currently Selected Process

Sensor to Move:

Planned Installations
Sensor 193
Sensor 26
Sensor 30
Sensor 33

Move Installation

Process to Move Sensor To:

Compaible Processes
Primary Process 7
Primary Process 5
Primary Process 9
Primary Process 8
Primary Process 10
Primary Process 11
Primary Process 34
Primary Process 32
Primary Process 33
Primary Process 12
Primary Process 13

Figure 74: Re-planning View

the Gantt chart. Furthermore, selecting a sensor task highlights the primary process during which it is planned for installation in the process list and Gantt chart. The planned process is also listed in the sensor task list. The interface also works in the opposite direction, so that when the user clicks on a primary or sensor installation process in the Gantt chart, the corresponding row in a table is highlighted. This allows the user to quickly gather information about each task to help him or her decide whether a modification to the plan is required.

Making modifications to the installation plan is accomplished by using the tables provided in the middle of the re-planning view. There are two options provided to re-plan sensor installations. First, the user may have identified a primary process that could support more sensor installations. In this case, the *Move Sensors to Current Process* tab provides a list of sensors that are compatible with the selected primary process. In addition to providing the compatible sensors, this table also provides metrics related to each individual sensor's current installation (not illustrated because of data restrictions). The provided metrics are:

Buffer Time (Slack Time): The total time between the currently planned installation and its final installation opportunity. This metric is summed across all of the sensor installations to determine the overall All Sensor Slack and Critical Sensor Slack Time metrics used to drive the original optimization algorithm.

Delay Caused By Sensor Installation: This is the total delay due to sensor installations to the primary process during which the sensor is currently planned. The user may use this metric to identify processes that have too many sensor installations planned and move the installation to the current process.

Installation Opportunities Remaining: The number of compatible primary processes remaining for the sensor installation. The user may chose to move sensors with few compatible installations remaining after their planned opportunity earlier in the process to help guard against disruptions. For instance, if a sensor has many opportunities available, then it might be acceptable to plan it later in the process. However, if it is planned near the end of its installation window, this option may not be available to the production manager.

The user may choose to manually explore the sensors that can be moved to a selected process. The table can be sorted by each metric, and the Gantt chart zooms to a selected sensor when highlighted. This allows the user to gain a fuller picture of how the installation is performing. A multi-criteria ranking algorithm (TOPSIS) is also provided to help the user better identify sensors to move to the current process. The user may set the importance of the three individual sensor metrics, and the table provides ranked options for re-planning.

The user may also choose to work in the opposite direction when making modifications to the installation plan. In this case, he or she would go to the *Move Sensors from Currently Selected Process* tab. This tab, which is shown below the Move Sensors to Current Process tab in Figure 74, provides the sensor installations that are planned to occur during the currently selected primary process. When a sensor is selected in the Sensor to Move column, the Process to Move Sensor To column is populated with every compatible process that could accept the currently selected sensor installation. Selecting a sensor or primary process in either column highlights the respective process in the Gantt chart. Once the user has selected the sensor to move and the destination process, clicking the *Move Installation* button propagates the move.

Upon completing some modifications to the installation plan, the user can save the input file and re-run the simulation model. Once the simulation run is completed, the results are pulled back into the decision support tool. Without considering quality robustness (i.e. only running a single case of the simulation model), this evaluation takes ~30 seconds on a quad core desktop. Running 35 replications of each 2–4 sensor installation technician scenarios to estimate the quality robustness of a new solution requires approximately 2 ½ minutes. This allows the user to see the impact on the overarching metrics of interest, the parallel plot, and the Gantt chart. The modified plan can also be propagated back to the re-planning view for further modification. This capability enables the user to make incremental changes to the installation plan to avoid any major, unexpected detriments to performance.

After the user is satisfied with the modified installation plan, the plan can be output to a list that matches each sensor installation with their planned primary production process. At this point, one final trade can be investigated utilizing the decision support environment. Throughout the entirety of this use case, it has been assumed that the number of technicians utilized for sensor installations remains constant throughout the execution of the schedule. In most cases, this is not required as a plan calling for 4 technicians may only need all 4 technicians during a few specific processes.

The parallel plot can be used to examine where the additional technicians are needed to reduce delays within the schedule. Recall that the sensor installation plans are evaluated for each of the 2–4 sensor installation technician scenarios. Because the logic is implemented such that each sensor installation sequence must be started during its defined primary process, a similar amount of work must be accomplished during each primary production task. The primary difference is that with more technicians, more parallel installations are allowed. Therefore, using the parallel plot, the user can determine if the additional parallel work supported by utilizing more technicians is required to avoid delays in each process. For example, if the 2 and 4 technician case both lead to no delays in a set of processes, the user can explore tasking only 2 technicians for those processes. However, if a large delay is seen when only 2 technicians are used, then the decision makers may decide to assign

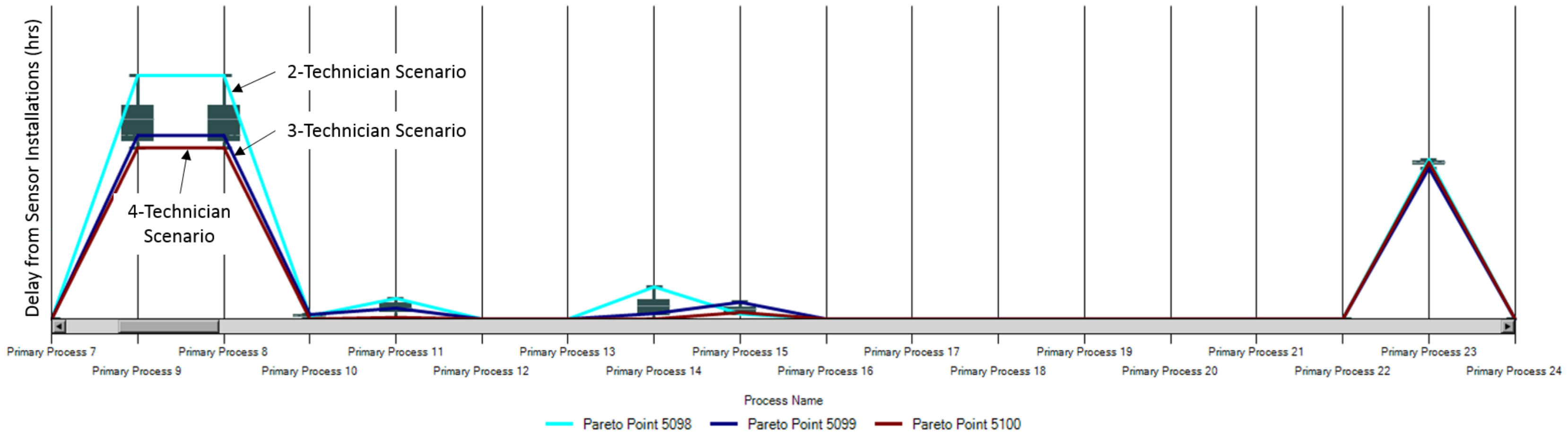


Figure 75: Parallel Plot Displaying Process Delays for the Same Sensor Installation Plan from 2 (Pareto Point 5098), 3 (Pareto Point 5099), and 4 (Pareto Point 5100) Sensor Installation Technician Scenarios

4 technicians during that set of processes.

To illustrate this trade, Figure 75 compares results for the 2–4 technician scenarios for the same sensor installation plan. The figure shows that Primary Processes 8 and 9 are delayed due to sensor installations occurring in previous processes. The amount of this delay is reduced significantly when moving from 2 (Pareto Point 5098) to 3 (Pareto Point 5099) technicians and by a lesser extent when moving from 3 to 4 (Pareto Point 5100). The user may then conclude that tasking a third technician to install sensors during the preceding processes is worth the extra cost. Looking forward in Figure 75, the delays caused by sensor installations are about the same for the 2–4 technician scenarios from processes 11–24. Hence, during these processes, it may be better to utilize only 2 technicians to save cost during this sequence. Providing this basic understanding of the number of technicians required during different points of the process flow can help the planners translate the provided schedule into an implementable plan during Step 4 of the PORRTSS methodology.

This culminates Step 3 of the methodology, which is the last step of the methodology developed throughout this thesis. With the installation plan output, it is the responsibility of the stakeholders to incorporate it into their overall plan. This can be accomplished by integrating the sensor installation plan with the rest of the production schedule; however, this could require a large amount of effort that may not be worthwhile for the type of low-volume project that this is methodology intended for. Hence, the sponsors may take a more manual approach and simply provide the list to the manufacturing engineering team to guide the creation of work orders and daily production plans. The determination of the integration approach to take, however, can be decided by the sponsoring stakeholders based on the scope and needs of their production schedule.

This concludes the discussion of the revised implementation of the methodology.

The following section shortly discusses the actual trades performed when the PORRTSS methodology was implemented and utilized in an actual industrial setting.

6.5 Methodology Implementation in the Industrial Setting

The major changes between the original implementation (the one used to plan processes in an actual, industrial setting) and the PORRTSS methodology are:

1. The original implementation did not consider the neighboring sensor installation metric

2. The automated re-planning capabilities were not implemented

Both capabilities were added in response to feedback received following the planning process. The feedback received is discussed in the following paragraphs.

The neighboring sensor installation metric was added to the optimization formulation following discussions with the SMEs after going through the planning process. While the avionics and manufacturing engineers who were investigating the proposed sensor installation plan were generally satisfied with the feasibility of the plan, they had identified multiple instances in which sensors that were located very close to each other were planned for installation very far apart in the schedule. Recognizing the opportunity to ease the burden on the technicians by allowing them to bring multiple sensors to the same installation site at one time, the users manually grouped some installations together after they had identified a set of promising plans.

Because the re-planning capabilities were not implemented, the users made the modifications to the installation plans manually. To do this, the users selected a few schedules that had similar performance based on their weightings and experience-based judgment. With these plans selected, they explored each schedule using the Gantt charts. Then, this information, the users created a composite sensor installation plan using the selected plans as a guide.

While the SMEs are highly capable of making local improvements to the plan, understanding the potential system-level impact of a change is extremely important. As discussed throughout this thesis, a major limitation of manual, human-driven planning is that it is difficult for a planner to fully understand the long-term interactions and repercussions of a decision [60]. The re-planning capabilities discussed in the previous section can help this by enabling the human planners to make small modifications to the plan, which can then be evaluated to identify the system-level impact.

With the discussion of the methodology's implementation complete, the following section summarizes the implementation of the methodology to the use case and highlights the new trades provided. This is included to identify specific elements of the methodology that contribute to its deployability in support of Hypothesis 2.

> Hypothesis 2: If the methodology requires a low amount of implementation effort and is shown to provide clear benefits with acceptable increases in computation time over traditional scheduling methods while effectively integrating the knowledge of the human planner, then the methodology can be successfully implemented to solve "real-world" problems.

6.6 Use Case Summary

By implementing the PORRTSS methodology, multiple new trades and capabilities are available to decision makers when compared to the baseline method discussed in Section 3.3. The trades and capabilities provided by the new methodology are:

1. Multi-objective, system-level impact assessment with detailed zooming capabilities

2. System-level assessment of local plan modifications

3. Manpower requirements analysis

Each capability is discussed beginning with the multi-objective, system-level impact assessment.

6.6.1 Multi-objective, System-level Impact Assessment

The main goal of the manual, baseline planning process (as discussed in Section 3.3) is to identify a feasible installation plan. Manual planning processes commonly require significant effort to identify feasible plans, which leaves little time to find plans with strong, system-level performance.

The PORRTSS methodology, by providing a multi-objective, system-level assessment of multiple potential installation plans, allows the users to perform more informed, cooperative decision making to find a well-performing plan. By formalizing the process to identify constraints through the constraint matrix definition tool and providing quantitative assessments for a large number of possible plans, the planners are now able to dedicate more time to finding better performing plans. Furthermore, the decision support environment helps to increase the transparency of the planning process, which is essential to increasing confidence in the selected installation plan. Because the planning system enables the users to rank the responses of interest, the extensive knowledge of the human planner can be incorporated into the final decision. Finally, by including a quantified assessment of the plan's quality robustness (in the form of the process/flow time's 80th quantile) and the heuristic risk metrics (slack time and the neighboring sensor installation metric), the methodology allows the planners to consider the risk of a plan instead of solely relying on its deterministic performance. The consideration of risk becomes more important as the scope, cost, and complexity of the project increases.

6.6.2 Manual Plan Modification and Assessment

Building upon an initial plan selected using the system-level assessment, the provided decision support environment also supports manual plan modifications and assessments. Recognizing that the scheduling system should work with the human planner, the decision support tool provides capabilities to guide the manual modification of the selected baseline plan. Then, when the planner has made changes, he or she can re-evaluate the plan by running it through the simulation. By bringing the simulation results back into the decision support environment, the planner is able to quantitatively assess the impact of his or her changes and identify any potential major, unintended changes to the system-level metrics of interest. In doing so, the planner is able to incorporate some preferences that may not have been captured by the objective functions or model while ensuring that the plan still performs well.

6.6.3 Manpower Requirements Analysis

Upon the selection and modification of a plan, the decision support tool enables the planner to identify portions of the process flow that require more or less manpower than specified by the selected scenario. By providing a guideline for when to plan for additional personnel, the decision support tool works to ensure additional personnel are available when needed while not continuously having un-needed personnel on hand.

These three main capabilities supported by the methodology and decision support environment have provided a system capable of planning sensor installations with reduced risk in a constrained production environment. With this overview completed, the following section expands on the implications for Hypothesis 2.

6.7 Experiment 2 Results Discussion

The discussion presented throughout this chapter is included to investigate Hypothesis 2. This hypothesis prescribes 4 needs that must be met to enable the PORRTSS methodology to be successfully implemented in a "real-world" scenario. The 4 needs are:

1. Require a low amount of implementation effort

2. Provide clear benefits over the baseline planning process
3. Require an "acceptable" (i.e. within the planning horizon) amount of computation time
4. Effectively integrate the knowledge and experience of the human planner

The evaluation criteria from Experiment 2 (reproduced below) are discussed to identify if 1) the needs identified in Hypothesis 2 are met and 2) whether this led to an implementable methodology. The evaluation criteria investigated are:

- Time required (methodology setup and optimization time) to find the set of potential solutions. Can this be accomplished within the required planning window?
- Optimization results and analyses are improved compared to the baselines planning process
  - Are the proposed plans feasible?
  - Are the results of high enough quality and well spread across the Pareto frontier to facilitate decision making?
  - Did the model properly identify portions of the production flow impacted by sensor installations?
- Stakeholders can leverage the provided data visualization and decision support tool to down-select and modify the sensor installation plans

Each criterion is evaluated in the following sections. Section 6.7.1 investigates how the flexibility in the methodology's implementation allows for implementation within a range of time constraints. Section 6.7.2 discusses how the optimization process and results compare to the baseline planning process. Finally, Section 6.7.3 discusses how the capabilities provided in the decision support tool work to increase implementability by increasing transparency and incorporating the planner's knowledge into the scheduling system.

6.7.1 Methodology Completion within the Required Time Horizon

The planning process for the vehicles and projects of interest can occur months to years in advance. In these cases, many months could be available to implement the PORRTSS methodology. This extended amount of time, however, was not available during the completion of the case study. Due to a set of planned meetings between the stakeholders, only 3 weeks were available to complete the first 2 steps of the methodology after receiving the required inputs. The automated model generation strategy enabled the implementor to construct and verify the simulation model over the course of only 2 days. Then, because the optimization routine is designed to work directly with the simulation inputs, no modification was required to start the optimization run. Due to the time constraints and limitations on the distribution of replications, the optimization, at the time, was completed without considering stochasticity and run on a quad core desktop.

By ignoring the stochastic nature of the model, the results are likely optimistic and do not directly consider the schedule robustness of the solution. However, by providing the ability to quickly change the objectives considered in the analysis, the methodology was able to provide a set of optimized solutions within the required time period. In this case, providing options to explore within the shortened time window was more important than fully exploring the solution's robustness. Through this implementation, the model generation and optimization speed improvements combined with the flexibility to complete the optimization at different levels of detail served to increase the implementability of the methodology.

When compared to the baseline planning process, the process proposed throughout this work provides much greater flexibility in terms of the time required to implement. The systematic process for defining constraints has been completed within a few days of meetings and analysis sessions. Then, generating the model and optimizing the sensor installation locations within the schedule has been completed within a couple weeks. Compared to the manual, judgment-based process, the new methodology is capable of producing potential solutions in a much more automated fashion. By automating a traditionally manual process, this methodology helps to free employees to complete other analysis tasks. This, in turn, can encourage implementation by contributing to cost reductions.

The following section investigates how the results and analyses produced from Step 2 support its implementation.

6.7.2 Optimization Results to Support Implementation

This evaluation is difficult because there is no direct comparison available between results from the proposed and baseline planning process (as described in Section 3.3). With this limitation acknowledged, the remainder of this section discusses how the results and analysis provided improve upon the expected results from the purely manual planning process. The main benefit expected by implementing the PORRTSS methodology is to automate the feasible plan generation procedure. By automatically providing a large set of Pareto optimal installation plans, planners can quantitatively evaluate the alternatives at a system and detailed level. Before investigating the quality of the results, the feasibility of the generated alternative installation plans must be investigated.

The provided Pareto optimal installation plans are necessarily feasible according to the provided compatibility matrix. The optimization formulation guarantees that only feasible plans are evaluated by the simulation model. Then, as shown in

Experiment 1.1, the model follows the defined production logic that ensures the compatibility matrix is properly observed during the simulation run. While the plans are feasible according to the defined constraints, a final check by the SMEs is needed to ensure that no major undefined constraints are missing from the formulation that would negate the usefulness of the results.

When the SMEs and decision makers evaluated the results produced from Step 2 of the methodology, no issues with the feasibility of the provided scenarios were discovered. While the true test of the feasibility will occur when the plan is executed, having the SME's assurance that the generated schedules could be completed similarly to the provided schedule is the best confirmation that could be expected. With the feasibility of the provided scenarios confirmed to the best possible certainty, the results must also provide the information required to make informed decisions about which scenario to select for execution.

Two classes of information are necessary to help the decision makers select between competing scenarios. First, the provided results must be of good quality and spread along the Pareto frontier to ensure that the decision makers have the ability to explore trades among the Pareto efficient solutions. Second, the results should indicate portions of the production flow that are most impacted by sensor installations. This information can be used by the decision makers to plan for mitigation strategies that are outside of the simulation's scope.

When reviewing the planning process completed by the SMEs, they agreed that the results provided a range of solutions that enabled them to explore compromises. Additionally, the information about the locations in the primary production flow that are impacted by sensor installations enabled them to better compare the provided plans. Therefore, in the estimation of the decision makers, the methodology is able to produce results that are helpful to the planning process.

The following section discusses the importance of the decision support environment in enabling the SMEs to better interpret the provided schedules to support improved implementation.

6.7.3 Decision Support Environment's Importance to Successful Implementation

The decision support environment is essential to the successful implementation of the methodology. As discussed throughout this chapter, the high level scatterplot matrix and multi-criteria ranking algorithm enable the decision makers to effectively explore the range of results provided from Step 2. Then, by selecting scenarios for further comparison, the parallel plot and Gantt chart help the decision makers to better understand how the production flow is impacted by sensor installations. Upon selecting a scenario to propagate forward, the re-planning view supports decision makers, who again may have little to no experience with the simulation, in making manual modifications to the plan. Therefore, the incorporation of the decision-support environment enables the decision makers to better understand the optimized plans and to directly incorporate their knowledge into the planning process. This helps to increase trust in the analysis and support the methodology's implementation.

During the down-selection process conducted by the SMEs, they used the scatterplot matrix and TOPSIS algorithm extensively to explore the design space and select points of interest. With points of interest selected, the parallel plot was used to compare solutions. It was seen that many solutions experienced delays in similar locations within the production flow. Furthermore, when exploring scenarios with different numbers of sensor installation technicians, the SMEs were able to identify points in the production flow where more or less technicians were needed to successfully complete the schedule. Therefore, when reviewing the planning process with the SMEs, the provided visualization and analysis capabilities provided sufficient information to decide between the Pareto efficient scenarios.

The re-planning capabilities were not fully implemented during the planning process completed by the SMEs. Instead, the SMEs selected a few promising scenarios and manually investigated modifications and combinations of the plans to identify the final schedule to implement. The majority of the re-planning capabilities were added in response to this shortcoming, and the resulting capabilities were well-received. Therefore, because the re-planning capabilities incorporate components identified by the literature as being essential to a schedule decision making system and were based on feedback from the SMEs, the re-planning capabilities are also shown to further the implementability of the methodology. The following section reviews the results from this experiment and explores its implications for Hypothesis 2.

6.8 Experiment 2 Results Review

The results from Experiment 2 have shown that the PORRTSS methodology is able to be successfully implemented in a "real-world" environment. While this is an important result, the goal of Experiment 2 is to show that the methodology was successful because it successfully incorporates the capabilities put forward by Hypothesis 2 (as discussed in the beginning of Section 6.7).

The importance of each of the components to the implementation of the PORRTSS methodology are discussed. The ease of implementation and flexibility in the optimization algorithm and objective functions enables the methodology to accommodate very short implementation times. The optimization results and data provided about each optimized point are shown to provide feasible alternatives with sufficient information to support down-selection. Finally, the decision support system allows the decision-makers to effectively sort through the results, understand how the sensor installations are impacting the production flow, and incorporate their knowledge to improve the plan. This demonstrates that the capabilities developed and integrated throughout this dissertation ultimately support the implementation of the methodology. Therefore, Hypothesis 2 is supported by the results from Experiment 2.

The following section presents a summary of this chapter.

6.9 Chapter Summary

This chapter presents a case study that details the completion of Steps 1–3 of the PORRTSS methodology to evaluate Experiment 2. The steps taken by the stakeholders to develop a compatibility matrix and simulation model are first discussed. It is shown that the automated model generation strategy is successful in translating the provided schedule model and compatibility matrix into a simulation model usable in Step 2. The optimization routine can then immediately take the generated schedule model and begin to generate optimized sensor installation plans.

Once the optimization routine has reached its stopping criteria (either maximum number of generations or convergence metrics), the results are propagated to the developed decision support environment. A use case is presented that illustrates how the decision makers were able to use the decision support environment's scatterplot matrix and TOPSIS algorithm to down-select to a few promising scenarios. With these solutions, the parallel plot and Gantt chart were used to further examine the potential schedules and select one to take forward into the re-planning view. Finally, the re-planning view enabled the decision makers, with little knowledge of the underlying simulation, to make changes to the selected plan and evaluate the results before finalizing the schedule to send to Step 4 of the methodology.

The use case description provided the basis for evaluation of Experiment 2. The 4 capabilities that Hypothesis 2 identifies as contributing to the implementability of the methodology are:

1. Require a low amount of implementation effort

2. Provide clear benefits over the baseline planning process
3. Require an "acceptable" (i.e. within the planning horizon) amount of computation time
4. Effectively integrate the knowledge and experience of the human planner

Each capability is discussed in the context of the case study, and its contribution to the implementation of the methodology is reviewed. The low implementation effort and optimization flexibility enabled the methodology to be implemented in the required amount of time. The generated results then enabled the decision makers to identify a promising scenario by utilizing the decision support environment. These results from Experiment 2 therefore support the claims of the Hypothesis and generally show that the capabilities developed throughout this work have led to an implementable methodology.

This chapter concludes the discussion of the experimental results and use case. The following chapters provide an overview of this dissertation and identifies the contributions provided by this work.

CHAPTER VII

SUMMARY & CONCLUSIONS

This chapter provides a summary of this dissertation and discusses conclusions, contributions, and potential avenues for future work.

7.1 Summary of Thesis Objectives

The knowledge and value gained from collecting data and being able to monitor vehicles' performance, safety, reliability, etc. have resulted in a sharp increase in the number of sensors being installed on modern aerospace vehicles. Integrating those sensors requires that additional production steps be dedicated to their installation. For complex aerospace vehicles, such as launch vehicles, satellites, or commercial aircraft, sensor installations are usually performed manually and present many challenges in terms of accessibility and precedence constraints. With manual installations also come increased risk for installation errors and quality issues, all of which contribute to production disruptions. Hence, while integrating sensors onto a vehicle provides valuable data, it also contributes to the increasing complexity of the newer generations of aerospace vehicles. This, in turn, contributes to the program cost overruns, increased risk, and production delays seen throughout the industry. As such, reducing the risk and impact of manual installation tasks on aerospace production flows is becoming increasingly important for such highly schedule- and cost-constrained vehicles.

Robust design methodology has the potential to meet the requirements of these scheduling problems. The goal of robust design is to design a system that is insensitive to noise factors that are difficult or impossible to affordably control. Robust scheduling aims to build schedules that minimize the impact of disruptions and reduce differences between the plan and execution. Real processes never go completely according to plan, and by not accounting for situations when processes take longer than planned, resources go off-line, or quality problems arise, detailed, optimized schedules can quickly become impractical or infeasible to implement.

Despite the benefit to be gained by implementing robust, detailed project scheduling methodologies, deterministic scheduling strategies still tend to dominate the industry. The limitations to a better deployment of robust scheduling techniques in an industrial setting lead to the two research needs that are addressed throughout this work:

1. The project scheduling methodologies in use today struggle to model and optimize real-world systems. The increasing complexity of modern aerospace vehicles is only going to exacerbate these difficulties and require improved plans to reduce system-level risk. Hence, a methodology that can better plan production and installation processes (e.g. subsystem or sensor integration) in more complex systems to reduce risk is needed.

2. The transition of new planning and scheduling practices from academia to an industrial setting is commonly challenging. Moreover, this transition is not generally discussed alongside the development of new methods. Therefore, capabilities to encourage adoption must be identified and implemented.

These research gaps and requirements lead to the development of the overall research objective:

Research Objective

To enable the integration of robust design principles with current, deterministic scheduling practices to efficiently schedule processes so as to reduce risk within increasingly complex production systems

A new planning methodology (PORRTSS: Production Optimization to Reduce Risk Through Simulation-based Scheduling) that leverages strengths from traditional scheduling methods, discrete-event simulation, and metaheuristic optimization is thus proposed. The methodology has the following characteristics:

- Incorporates discrete-event simulation with traditional schedule optimization
- Supports simulation-based optimization by:
    1. Leveraging decisions previously made by the schedule optimization to reduce the model fidelity and optimization decision space
    2. Effectively parallelizing the simulation replications
    3. Formulating the problem such that it is unconstrained
- Improves solution robustness by directly optimizing the modeled schedule's quality robustness and heuristic risk-reduction metrics
- Effectively includes the expertise of the human decision maker by:
    1. Increasing transparency in the constraint definition, modeling, and optimization process
    2. Providing decision support from the system-level down to a detailed, task-by-task view
    3. Enabling the assessment of manual plan modifications

The following section concludes on the proposed approach's ability to meet the aforementioned research objective by summarizing the research questions, hypotheses, and experimental results presented throughout this work.

7.2 Research Summary

An overview of the research structure is presented in Figure 9 (reproduced below). In Chapter 1, it is observed that increasing complexity is contributing to rising costs and delays that are challenging traditional schedule optimization methods. Then, two assertions are made that drive the research: First, it is asserted that robust scheduling techniques have the potential to alleviate some of the increasing costs and delays. Then, Assertion 2 contends that the lack of accepted robust, detailed scheduling practices are primarily due to implementation challenges.

These observations and assertions led to the development of two primary research questions within Chapter 2. Research Question 1 investigates how to overcome the identified challenges to implement a detailed, robust scheduling methodology. Three sub-research questions are posed to investigate different challenges. Research Question 1.1 is developed to identify the modeling technique that is appropriate to model the systems of interest for the PORRTSS methodology. In response to this research question, Hypothesis 1.1 is developed:

> Hypothesis 1.1: If discrete-event simulation is leveraged, then increasingly complex scheduling environments can be modeled effectively such that the information required for use in a selected optimization routine can be captured.

Experiment 1.1 is designed to test this hypothesis. The results from the experiment demonstrate that the discrete-event paradigm is capable of respecting the choices of the schedule model while modeling the impact of parametrically defined, detailed scheduling decisions. This allows the simulation to estimate the system-level impact of these decisions, which demonstrates that discrete-event simulation is an appropriate modeling paradigm.

With the selected modeling approach, Research Question 1.2 explores optimization techniques that are capable of exploring the multi-objective solution space. Hypothesis 1.2 is presented in response to this research question:

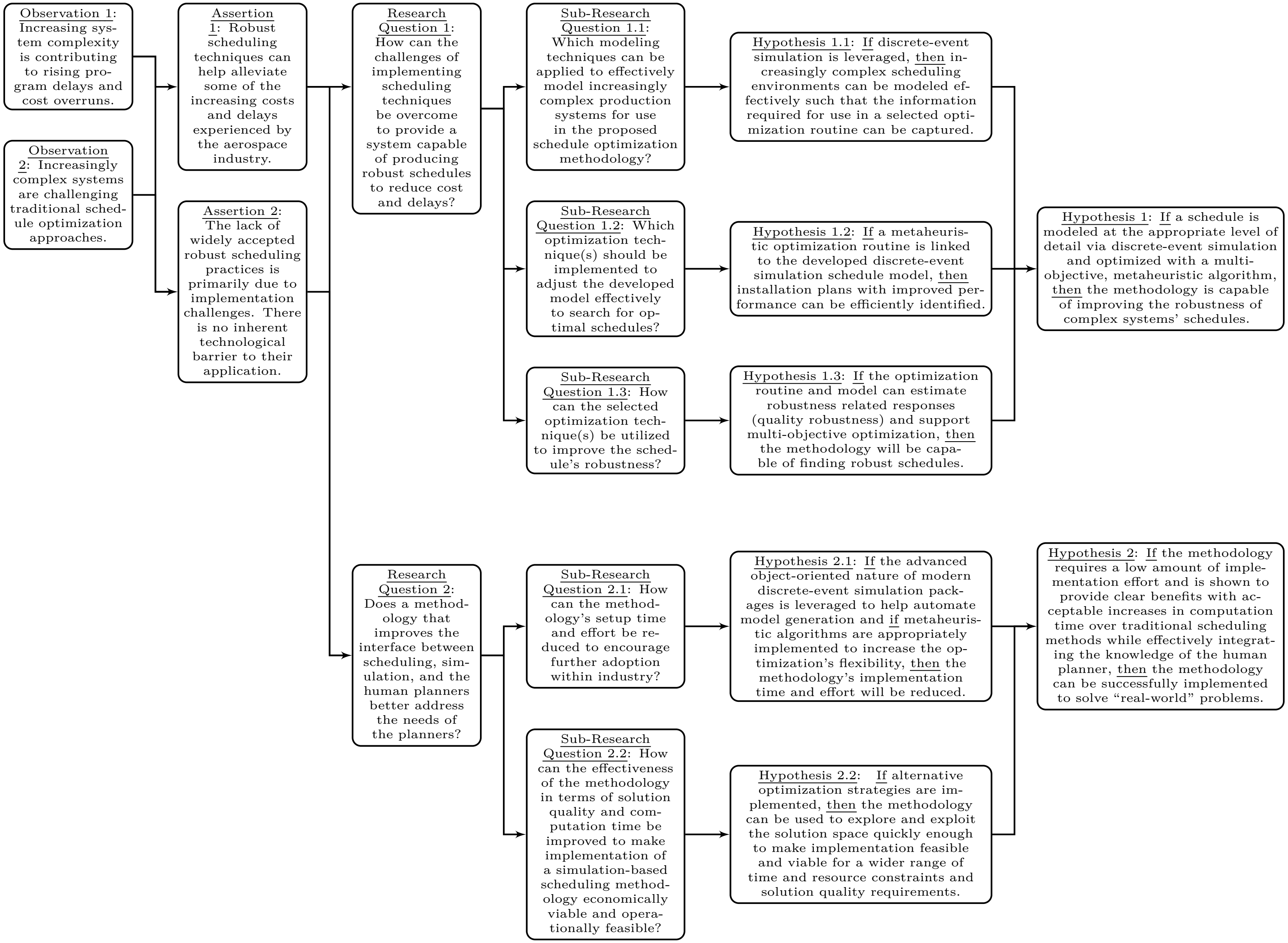

Observation 1: Increasing system complexity is contributing to rising program delays and cost overruns.
Observation 2: Increasingly complex systems are challenging traditional schedule optimization approaches.
Assertion 1: Robust scheduling techniques can help alleviate some of the increasing costs and delays experienced by the aerospace industry.
Assertion 2: The lack of widely accepted robust scheduling practices is primarily due to implementation challenges. There is no inherent technological barrier to their application.
Research Question 1: How can the challenges of implementing scheduling techniques be overcome to provide a system capable of producing robust schedules to reduce cost and delays?
Research Question 2: Does a methodology that improves the interface between scheduling, simulation, and the human planners better address the needs of the planners?
Sub-Research Question 1.1: Which modeling techniques can be applied to effectively model increasingly complex production systems for use in the proposed schedule optimization methodology?
Sub-Research Question 1.2: Which optimization technique(s) should be implemented to adjust the developed model effectively to search for optimal schedules?
Sub-Research Question 1.3: How can the selected optimization technique(s) be utilized to improve the schedule's robustness?
Sub-Research Question 2.1: How can the methodology's setup time and effort be reduced to encourage further adoption within industry?
Sub-Research Question 2.2: How can the effectiveness of the methodology in terms of solution quality and computation time be improved to make implementation of a simulation-based scheduling methodology economically viable and operationally feasible?
Hypothesis 1.1: If discrete-event simulation is leveraged, then increasingly complex scheduling environments can be modeled effectively such that the information required for use in a selected optimization routine can be captured.
Hypothesis 1.2: If a metaheuristic optimization routine is linked to the developed discrete-event simulation schedule model, then installation plans with improved performance can be efficiently identified.
Hypothesis 1.3: If the optimization routine and model can estimate robustness related responses (quality robustness) and support multi-objective optimization, then the methodology will be capable of finding robust schedules.
Hypothesis 2.1: If the advanced object-oriented nature of modern discrete-event simulation packages is leveraged to help automate model generation and if metaheuristic algorithms are appropriately implemented to increase the optimization's flexibility, then the methodology's implementation time and effort will be reduced.
Hypothesis 2.2: If alternative optimization strategies are implemented, then the methodology can be used to explore and exploit the solution space quickly enough to make implementation feasible and viable for a wider range of time and resource constraints and solution quality requirements.
Hypothesis 1: If a schedule is modeled at the appropriate level of detail via discrete-event simulation and optimized with a multi-objective, metaheuristic algorithm, then the methodology is capable of improving the robustness of complex systems' schedules.
Hypothesis 2: If the methodology requires a low amount of implementation effort and is shown to provide clear benefits with acceptable increases in computation time over traditional scheduling methods while effectively integrating the knowledge of the human planner, then the methodology can be successfully implemented to solve "real-world" problems.

Figure 9: Summary of Research Structure (Reproduced from page 74)

Hypothesis 1.2: If a metaheuristic optimization routine is linked to the developed discrete-event simulation schedule model, then installation plans with improved performance can be efficiently identified.

This Hypothesis is tested via Experiment 1.2. Results from the experiment demonstrate that the multi-objective, metaheuristic optimization paradigm is capable of finding a family of non-dominated solutions without considering quality robustness (i.e. by optimizing the deterministic model). Further, it is shown that the optimization formulation is able to efficiently generate new Pareto efficient solutions, thereby out-performing a random search.

Upon demonstrating that the optimization formulation is applicable, Research Question 1.3 is offered to determine the impact of directly including quality robustness in the optimization's formulation. The following hypothesis is put forward in response:

Hypothesis 1.3: If the optimization routine and model can estimate robustness related responses (quality robustness) and support multi-objective optimization, then the methodology will be capable of finding robust schedules.

Experiment 1.3 investigates this hypothesis by comparing results and convergence criteria when optimizing with and without uncertainty in the simulation model. Results indicate that, for the problem of interest, the optimization with robustness considerations slightly outperforms the deterministic optimization. Promising results are observed in the early generations of the optimization with robustness considerations. The first 25 generations show significant improvement, and the Pareto frontier from these initial generations indicates that a large portion of the improvement is related to the 80th quantile of process time. This particularly demonstrates that the stochastic optimization is utilizing the additional information to reduce the risk in the provided non-dominated installation plans. Finally, it is observed that the optimization without robustness does not produce Pareto optimal points as efficiently as the run with robustness considerations.

While the results of this experiment are promising, they are not sufficient to draw generalized conclusions. In truth, to generate statistically significant results, the experiment should be replicated multiple times to ensure that one run was not simply more favorable than the other. This, however, would require an exorbitant amount of time and/or computational resources while not significantly contributing to the overall research objective. Furthermore, the benefits of including robustness considerations are highly problem-specific (e.g. a well-controlled process that experiences little variation could still benefit from this methodology's planning approach without requiring evaluations of quality robustness). When devising this experiment, the stochasticity was expected to have a much larger impact on the simulation. In this scenario, it was anticipated that the robust optimization would be clearly superior and that specific decisions could be identified that significantly reduced the plan's risk. Because this turned out to not be the case, the benefits to including the quality robustness in the optimization could not be identified. Ultimately, the negative impact of increasing uncertainty on a schedule's performance is well-documented, so conclusively demonstrating that directly optimizing for robustness improves performance is not necessary to establishing the validity of the methodology.

With Hypotheses 1.1–1.3 investigated, the developed capabilities can be built up to test Hypothesis 1:

> Hypothesis 1: If a schedule is modeled at the appropriate level of detail via discrete-event simulation and optimized with a multi-objective, metaheuristic algorithm, then the methodology is capable of improving the robustness of complex systems' schedules.

Experiment 1 tests this hypothesis by completing Steps 1 and 2 of the PORRTSS methodology for the assembly and system integration flow of the use case's production process. The experiment shows that the process can be modeled at an appropriate level of detail and optimized to produce a family of solutions with improved robustness compared to a random population. Hence, Hypothesis 1 is supported, and the methodology is shown to overcome many of the schedule modeling and optimization capability gaps identified within literature.

With a technical solution leveraging simulation-based optimization implemented, Research Question 2 is developed to explore ways to bridge the second research gap. This question seeks to identify whether an improved link between scheduling, simulation, and human planners better supports the stakeholders' needs. Two sub-research questions are developed to explore the links between simulation, scheduling, and optimization. Then, a representative use case is completed to investigate the links between the analysis capabilities and the human planner. The use case is comprised of the production flow for a modern aerospace system. The production plan is provided at a medium level of detail, and the methodology is utilized to plan the installation of sensors within the defined production flow.

Research Question 2.1 investigates strategies to reduce the methodology's implementation time, which is identified as a barrier to further acceptance of simulation in industry. Hypothesis 2.1 is thus developed:

> Hypothesis 2.1: If the advanced object-oriented nature of modern discrete-event simulation packages is leveraged to help automate model generation and if metaheuristic algorithms are appropriately implemented to increase the optimization's flexibility, then the methodology's implementation time and effort will be reduced.

Experiment 2.1 is designed to examine whether the characteristics of the selected modeling and optimization paradigms work to reduce the time required to implement the methodology. The results demonstrate that the object-oriented nature of modern discrete-event simulation does facilitate automated model generation, which speeds model construction and verification. Then, the metaheuristic optimization routine can be immediately implemented for simulation models generated through this methodology. Finally, the metaheuristic paradigm is shown to be flexible to enable easy incorporation of new objective functions and optimization strategies. Hence, results from this experiment support Hypothesis 2.1.

Research Question 2.2 seeks to explore alternative optimization strategies and robustness formulations to improve the methodology's implementability for a range of circumstances. Recognizing that there can be a wide range of time and resources available for optimization, solution quality requirements, problem complexity, and knowledge of the design space, Hypothesis 2.2 is formulated:

Hypothesis 2.2: If alternative optimization strategies are implemented, then the methodology can be used to explore and exploit the solution space quickly enough to make implementation feasible and viable for a wider range of time and resource constraints and solution quality requirements.

To test this hypothesis, a 2-staged algorithm down-selection process is implemented. First, multiple heuristic and metaheuristic algorithms are investigated using the deterministic problem. Then, two promising algorithms are carried forward to investigate alternative robustness measures. The investigation of the robustness measures is inconclusive, which leads to an inconclusive result for Experiment 2.2. More computational resources would be required to properly evaluate the Hypothesis. Further, a process flow with more and better defined stochasticity could promote more distinct differences between the deterministic and stochastic optimization routines. This could help to better validate the hypothesis.

Building upon the results from the sub-research questions, Hypothesis 2 is proposed:

Hypothesis 2: If the methodology requires a low amount of implementation effort and is shown to provide clear benefits with acceptable increases in computation time over traditional scheduling methods while effectively integrating the knowledge of the human planner, then the methodology can be successfully implemented to solve "real-world" problems.

This hypothesis is tested by completing the entire case study. The purpose is to demonstrate that the developed capabilities lead to a successful, real-world implementation of the methodology. In completing the investigation, it is shown that the flexibility of the methodology allows the stakeholders to trade solution quality for a shorter turn-around time (by optimizing the deterministic model). Then, the ability to automate the generation of alternative installation plans and provide a system-level impact analysis provides improvements over the baseline, manual planning process. Finally, the integration of the human decision maker's knowledge improves transparency and increases trust. Therefore, results from the experiment demonstrate that the inclusion of the prescribed modeling, optimization, and analysis capabilities have produced a more implementable methodology.

This section has reviewed the structure of this work and summarized the results and conclusions drawn. The following section reviews the major contributions of this research.

7.3 Summary of Contributions

The contributions of this work are related to the two identified research gaps. The first major contribution is a new methodology (PORRTSS: Production Optimization to Reduce Risk Through Simulation-based Scheduling) that better integrates the strengths of simulation and scheduling practices to address the identified needs of low-volume aerospace production and assembly processes. Similar approaches can be leveraged for high-volume systems; however, low-volume projects can benefit more from improved pre-production planning practices. This is because these vehicles (e.g. satellites, prototype aircraft, launch vehicles, etc.) are typically very schedule- and cost-constrained. Additionally, there is little to no learning in these systems, so identifying a robust plan during the planning period is critical. The second key contribution is an investigation of capabilities that improve the interaction between human planners and the scheduling process. This encourages implementation of the developed methodology by increasing transparency and trust. Both contributions are discussed in the following sections.

7.3.1 Contribution 1: Improved Interaction between Scheduling and Simulation Practices

As discussed throughout the literature review, scheduling and simulation are commonly viewed as separate disciplines to be applied to different problems. As such, planning methodologies and research efforts that integrate the strengths of each are lacking. In response, this work has developed and implemented a new methodology that integrates the capacity optimization strengths of scheduling with the modeling flexibility of simulation. Ultimately, this new method is designed to solve a relevant, real-world planning problem.

For the problem at hand, traditional scheduling techniques provide a strong framework to plan and optimize, at a medium level of detail, the completion of production processes (e.g. structural assembly, system integration, etc.). Fully defining the interactions and logic required to evaluate the impact (in terms of process time) stemming from subcomponent/sensor installations in this scheduling framework is challenging. The discrete-event simulation paradigm simplifies the definition of these production rules and constraints; however, as noted in the literature review, DES models commonly require too much detail, modeling effort, and optimization time/resources to be useful during pre-production planning.

The PORRTSS methodology significantly reduces the modeling effort required to generate a suitable simulation model. The methodology extracts process and constraint information from the optimized primary process schedule. This information is condensed into a set of precedence relationships, which simplifies the modeling process and helps to reduce the time required to run the simulation model. Concepts implemented in this process mining technique, while developed for a pre-planning problem, can be extended to mine information from actual production data. This is further discussed in Section 7.5.

With the process information extracted from the input schedule, a simulation can be quickly built using an automated model generation strategy. This provides a simulation that matches the provided schedule but adds the capability to parametrically define locations within the process flow to install subcomponents. The fidelity of the simulation model can be constrained by leveraging decisions made by the schedule optimization. By treating the scheduled primary production process as truth, the simulation does not need to model many of the capacity, workforce, or physical constraints typically required in a simulation model. This helps to make simulation-based optimization more feasible. This model is then linked to an optimization routine to automatically generate and improve detailed plans for subcomponent installations. Ultimately, this enables more informed decision making by automatically generating multiple feasible alternatives and providing an evaluation of system-level performance metrics.

The second contribution focuses on how these feasible alternatives can be better explored by improving the interface between the planning system and the human planners.

7.3.2 Contribution 2: Improved Interface between Human Schedulers and the Planning Methodology

A primary barrier that limits the implementation of developed scheduling practices is poor interactions between the system and the human planners. Improved interaction is achieved by: 1) increasing the transparency of the planning process, 2) improving collaboration among the stakeholders, and 3) enabling the stakeholders to directly modify the plan. This work has identified and implemented capabilities into a decision support tool that supports this improved interface.

Increased transparency is achieved by providing both a system- and detailed-level view of the planning results. Planners with many years of experience are not likely to trust a system-level value for process time that results from a new analysis tool. Hence, the installation delay parallel plot and Gantt chart help to increase transparency by providing details about how and why the process was delayed.

The second, related improvement over commonly developed systems is the improved collaboration between the stakeholders. In the case of projects common to low-volume production systems, multiple stakeholders must be included in the scheduling process. The PORRTSS methodology works to facilitate collaboration through each step of the process. When developing the compatibility matrix, the implemented tool allows the IEs, MEs, and avionics experts to work together to identify process constraints. This traceable process helps the stakeholders to better define and understand the constraints that drive the analysis.

Following the model development and optimization, the decision support tool enables the stakeholders to collaborate again to identify the single plan to carry out. The combination of the scatterplot matrix and TOPSIS ranking algorithm helps the decision makers to identify regions of high performing schedules. Then, by providing a more detailed scenario comparison view, they can better understand how their decisions are impacting the process.

The re-planning capabilities enable the decision makers to incorporate any additional preferences or compromises. Many academic scheduling processes claim to provide *the* answer; however, by recognizing that there may be additional "soft" constraints, the re-planning capabilities can incorporate the valuable experience of the planners. Hence, as shown throughout this work, building the methodology with the planner in mind helps to improve the interaction between the process and the planners. This ultimately leads to a more implementable and beneficial planning methodology when compared to the many rigid methods developed in academia.

With the overall contributions described, the following section discusses the limitations of the current work.

7.4 Methodology Limitations

There are limitations present throughout all steps of the PORRTSS methodology. Due to a lack of data access, Step 1 currently requires unverifiable assumptions about the process constraints. The entire set of man-power, resource, and work area constraints are not available in the provided date. Hence, the schedule mining technique employed is only capable of identifying constraints from the process timestamps and likely does not capture the entire set of constraints. This could make the simulation model slightly optimistic and lead to plans that are infeasible. The specific problem investigated reduces the likelihood of this, however, because there are many "milestone" processes in the flow. These processes would cause the simulation to return to the baseline primary process schedule, which is necessarily feasible, and limit the extent of infeasibility.

There are also limitations related to the compatibility matrix generation. The currently implemented process requires a significant amount of input from SMEs. Along with limiting the potential for implementation, this could produce errors or oversights that could lead to an infeasible sensor installation plan. Improvements to this process could work to increase the validity of the results and speed the methodology's implementation time.

Progressing to Step 2 of the PORRTSS methodology, the optimization algorithm could be improved to increase efficiency. The results are currently ranked without any considerations for user preferences. In reality, many non-dominated solutions may never be chosen for execution. For instance, it is unlikely that the user would choose to sacrifice a significant amount of process time for a small increase in slack time. The optimization routine also does not intelligently select the number of replications to evaluate for each population member, which leads to reduced search efficiency.

The scope of experiments 1.3 and 2.2 that could be completed also limit the conclusions that can be drawn. The limited set of test problems and computational time and resources available restrict the extent of the conclusions. While the optimization algorithms investigated generally appear appropriate for the specific problem, further investigations are necessary to help generalize the experiments' results.

Finally, Step 3 of the methodology is limited in the scope and flexibility of the scheduling decisions available. Ideally, the decision support system would enable modifications to both the primary production plan and the sensor installation plan. By only allowing changes to the installation plan, opportunities for improved plans may be missed. Further, the decision support system does not support the modification of process constraints. This capability could be useful if infeasible processes are identified when analyzing the data.

The following section identifies opportunities for future work to overcome some of these identified limitations.

7.5 Recommendations for Future Work

This section discusses potential areas of future work that can expand and improve upon the limitations of the PORRTSS methodology. The first area involves improving and automating the compatibility matrix generation. Ultimately, this improvement can be leveraged to help plan the initial primary production process.

7.5.1 Compatibility Matrix & Primary Production Flow Generation

One major hurdle to the implementation of the PORRTSS methodology is the time and effort required to construct the compatibility matrix. When in an active vehicle development program, the planners may not have the time to develop constraints for an unproven planning methodology. Therefore, identifying a process to automatically generate the compatibility matrix would help to ease the burden of the planners and better encourage implementation.

The majority of the constraints identified by the stakeholders are related to installation site access. For example, if a large avionics rack or structural component is installed that blocks access to a subcomponent installation site, the subcomponent must be installed first. These constraints could potentially be identified by simulating the physical installation process and identifying the components that interfere with an installation.

Video game development platforms are well-suited for these simulations. Detecting and reacting to collisions between objects is an important and well-developed capability of game design software. Simplified motion paths to simulate the installation process (e.g. movement of the component and a representative operator from the center of the vehicle to the installation site) is readily possible. Then, by identifying the objects that collide with the operator and/or component, compatibility constraints can be determined. This technique could be utilized to initially generate a compatibility matrix or check the feasibility of a defined matrix.

With further development, this technique could be expanded to determine complete, feasible assembly sequences for all steps and components. By identifying constraining components, a sequence could be identified that minimizes interference between all steps. This knowledge can then be fed into the developed methodology to help automatically identify plans with low process time and high robustness. Therefore, by automating these sequence and compatibility generation steps, the effort required to implement the methodology could be significantly reduced.

The following section describes potential expansions to the schedule process mining process to better support simulation of complex production systems.

7.5.2 Schedule Mining to Support Simulation

The barriers to the use of simulation tools in complex production systems have been well-documented throughout this work. One major barrier is the time required to collect, analyze, and model process data and constraints. The process mining technique described throughout this work represents a first step towards a more general methodology that can reduce this implementation effort.

Trades and analyses of interest to a more general production system are different from those discussed throughout this work. Instead of process optimization, higher volume systems are commonly more interested in overarching strategies that can improve the flow or mitigate disruptions. Supporting these goals with simulation requires that multiple, representative processes can be quickly modeled to support virtual experimentation.

By leveraging the actual schedule completion information, which inherently respects process constraints, precedence relationships can be extracted from the data. Similar to this work's implementation, both "firm" and "preference" precedence constraints can be identified. The *firm* predecessors are identified as the processes that always (or almost always) are completed before the current process. This helps to give an overall structure to the sequence.

Preference relations, which represent processes that commonly end closely prior the current process's start, can then be stochastically added to provide variation between testing schedules. This variation is important to test the robustness of any implemented selection rule on a wide selection of scenarios. A preference precedence relationship can be added from Process_i to Process_j based on a weighted probability defined by:

$$\frac{\text{Percent of historical schedules where Process}_\text{i}\text{ finishes before Process}_\text{j}}{\text{Median distance (in terms of sequence position) between Process}_\text{i}\text{ and Process}_\text{j}}$$

By following this process, schedules that are representative of the complex and highly variable processes encountered in aerospace production environments can potentially be quickly identified for simulation.

Similar analyses can be leveraged to identify constraints within the process. A common approach to modeling constraints is to divide a vehicle into work zones. Each work zone can then only support a single task at any given time. By linking each process to a physical location, historical process information can be utilized to optimize the location of work zone divisions. The work zone optimization could be implemented as follows:

1. Assign an initial set of work zone divisions

2. Classify each process into a work zone (or multiple zones) based on their defined physical location(s)

3. Evaluate an objective function by counting the number of processes occurring at the same time in the same work zone. For example, two processes occurring simultaneously in the same work zone indicate that the work zone may not be correctly defined.

4. Modify the work zone divisions based on an optimization procedure and repeat until convergence

The identified work zones can then be modeled as seizable resources in the simulation.

Incorporating these two strategies can help to reduce the time to model a baseline process. Then additional fidelity can be added to the simulation model to support trades of interest. Strategies such as this can help to further incorporate simulation into production planning environments.

The following section discusses potential areas of investigation relating to simulation-based optimization.

7.5.3 Expanded Investigation of the Robust Simulation-based Optimization

The importance of including quality robustness as an objective function (instead of simply a deterministic evaluation of process time) should be more rigorously examined. The use case model included in this work saw a minimal impact from the stochasticity; hence, generating a model of a process with increased uncertainty could better exhibit the importance of including quality robustness. Furthermore, the low number of computing resources available limited the optimization cases that could be reasonably evaluated. As such, the impact of including the quality robustness measures is not certain and requires further investigation.

Many other optimization techniques and strategies can also be investigated. Dynamically selecting the number of replications to run can reduce the number of cases to evaluate. Modifying the solution ranking criteria to incorporate user preferences could make the multi-objective search more efficient. For instance, because reducing the process time is the most important objective function, assigning a constraint to the increase in process time to trade for gains in slack time can help to guide the optimization. Investigating alternative algorithms and objective functions can also better tailor the methodology to new problems.

The next section describes further capabilities that can be investigated to improve the data analysis.

7.5.4 Decision Support Improvements

Improvements to the decision support tool can also be investigated. Primarily, additional capabilities to improve the interaction between the planner and the planning process can be incorporated. First, providing the capability to update the compatibility matrix in the decision support tool could help to identify individual cases where the initial matrix is overly constraining. Furthermore, providing the ability to show the assembly's progression in the CAD model can better illustrate the process and provide a final feasibility check. Ultimately, the best decision support visuals and prioritization algorithms are problem dependent; however, providing improved linkages between system-level metrics, process details, and the actual CAD information can improve the planning process.

APPENDIX A

OPTIMIZATION ALGORITHM PSEUDO-CODE

Algorithm 1: Shifting Bottleneck-Inspired Algorithm

```
Input: $k$: Maximum number of schedules to try per iteration
Initialize random schedule: $x$
Evaluate initial schedule: $f(x) \leftarrow evaluateSchedule(x)$
while stopping condition not met do
    Determine performance of each individual sensor in $f(x)$
    if algorithm==greedy then
        Select worst performing sensor for re-planning: $s_{mod}$
        if All available options for $s_{mod}$ have been attempted then
            Find new $s_{mod}$ whose options have not all been attempted
        end
    else
        Select sensor for replanning based on weighted probability distribution:
        $s_{mod}$
        if All available options for $s_{mod}$ have been attempted then
            Find new $s_{mod}$ whose options have not all been attempted
        end
    end
    if $s_{mod}$ does not equal $s_{mod,old}$ then
        Generate up to $k$ schedules by moving $s_{mod}$
    else
        Generate up to $k$ schedules by moving $s_{mod}$ that are not the same as
        those already evaluate
    end
    Evaluate $f(x)$ for each possible position of the selected sensor
    Set $x$ as the point that produces the best value of $f(x)$
    $s_{mod,old} \leftarrow s_{mod}$
    Check stopping criteria
end
```

Algorithm 2: Stochastic Neighborhood Search Algorithm Procedure

```
Initialize random schedule: $x$
Evaluate initial schedule: $f(x) \leftarrow evaluateSchedule(x)$
while stopping condition not met do
    Determine performance of each individual sensor in $f(x)$
    Determine total number of sensors to re-plan: $n \leftarrow rand\,[1, 0.05n_{tot}]$
    for $i \leftarrow 1$ to $n$ do
        Select sensor for re-planning based on weighted probability distribution
        Add selected sensor to list for re-planning
    end
    Randomly select new location to install each sensor in the re-planning list
    Generate $x_{new}$ using the new locations to install the selected sensors
    $f\left(x_{new}\right) \leftarrow evaluateSchedule(x_{new})$
    if $f\left(x_{new}\right)$ is better than $f(x)$ then
        $x \leftarrow x_{new}$
    end
    Check stopping criteria
end
```

Algorithm 3: Simulated Annealing Algorithm

```
Input: Initial Temperature: $T_o$
Initialize random schedule: $x$
Evaluate initial schedule: $f(x) \leftarrow evaluateSchedule(x)$
while stopping condition not met do
    Find $x_{new}$ and $f(x_{new})$ using the stochastic neighborhood search
    (Algorithm 2)
    if $f\left(x_{new}\right)$ is better than $f(x)$ then
        Accept $x_{new}$
    else
        Accept $x_{new}$ with probability $p\left(\Delta f\right) = \exp\left(\frac{-\Delta f}{T_k}\right)$
    end
    Reduce temperature based on annealing schedule
    Check stopping criteria
end
```

Algorithm 4: Fast Simulated Annealing Algorithm

```
Input: Initial Temperature: $T_o$
Initialize random schedule: $x$
Evaluate initial schedule: $f(x) \leftarrow evaluateSchedule(x)$
while stopping condition not met do
    Find $x_{new}$ and $f(x_{new})$ using the stochastic neighborhood search
    (Algorithm 2)
    if $f(x_{new})$ is better than $f(x)$ then
        Accept $x_{new}$
    else
        Accept $x_{new}$ with probability $p(\Delta f) = \exp\left(\frac{-\Delta f}{T_k}\right)$
    end
    $n \leftarrow n + 1$
    Set temperature based on annealing schedule described in Equation 8
    Check stopping criteria
end
```

Algorithm 5: Pareto Dominance-based Simulated Annealing Algorithm

```
Input: Initial Temperature: $T_o$
Initialize random schedule: $x$
Evaluate initial schedule: $f(x) \leftarrow evaluateSchedule(x)$
while stopping condition not met do
    Find $x_{new}$ and $f(x_{new})$ using the stochastic neighborhood search
    (Algorithm 2)
    if Pareto Rank $\vec{f_{new}} \leq \vec{f_{old}}$ then
        Accept $x_{new}$
    else
        Calculate $\Delta f = \vec{\alpha}\left(\vec{f_{new}} - \vec{f_{old}}\right)$, where $\vec{\alpha}$ is a vector of random values
        between 0 and 1
        Accept $x_{new}$ with probability $p(\Delta f) = \exp\left(\frac{-\Delta f}{T_k}\right)$
    end
    Reduce temperature based on annealing schedule
    Check stopping criteria
end
```

REFERENCES

[1] "Orion exploration flight test-1." Online.

[2] "DoD integrated product and process development handbook." Online, July 1998.

[3] "Airthm - airbus real time health monitoring." Online, 2015.

[4] "The job-shop problem, the disjunctive model and benchmark data." Online, 2015.

[5] "Exponential growth of system complexity." Online, 2016.

[6] Abdulmalek, F. A. and Rajgopal, J., "Analyzing the benefits of lean manufacturing and value stream mapping via simulation: A process sector case study," *International Journal of Production Economics*, vol. 107, no. 1, pp. 223 – 236, 2007. Special Section on Building Core-Competence through Operational Excellence.

[7] Adler, P. S., "Interdepartmental interdependence and coordination: The case of the design/manufacturing interface," *Organization science*, vol. 6, no. 2, pp. 147–167, 1995.

[8] Al-Fawzan, M. and Haouari, M., "A bi-objective model for robust resource-constrained project scheduling," *International Journal of Production Economics*, vol. 96, no. 2, pp. 175 – 187, 2005.

[9] Andrienko, G. and Andrienko, N., "Constructing parallel coordinates plot for problem solving," in *1st International Symposium on Smart Graphics*, pp. 9–14, 2001.

[10] Antill, J. M. and Woodhead, R. W., *Critical path methods in construction practice.* John Wiley & Sons, 1990.

[11] Arena, M. V., Younoss, O., Brancato, K., Blickstein, I., and Grammich, C. A., "Why has the cost of fixed-wing aircraft risen? a macroscopic examination of the trends in u.s. military aircraft costs over the past several decades.," Santa Monica, CA: RAND Corporation, 2008.

[12] Arora, J., *Introduction to optimum design.* Academic Press, 2004.

[13] Axten, E. and Schier, J., "Inertial sensor performance for diverse integration strategies in automotive safety," *Advanced Microsystems for Automotive Applications*, pp. 251–263, 2007.

[14] Aytug, H., Lawley, M. A., McKay, K., Mohan, S., and Uzsoy, R., "Executing production schedules in the face of uncertainties: A review and some future directions," *European Journal of Operational Research*, vol. 161, no. 1, pp. 86 – 110, 2005. IEPM: Focus on Scheduling.

[15] Bagchi, T. P., *Multiobjective scheduling by genetic algorithms.* Springer Science & Business Media, 1999.

[16] Balas, E., "Disjunctive programming and a hierarchy of relaxations for discrete optimization problems," *SIAM Journal on Algebraic Discrete Methods*, vol. 6, no. 3, pp. 466–486, 1985.

[17] Banks, J., ed., *Handbook of Simulation.* Engineering and Management Press, 1998.

[18] Banks, J., II, J. S. C., and Nelson, B. L., *Descrete-Event System Simulation.* Prentice Hall, 1996.

[19] Barua, A., Raghavan, N., Upasani, A., and Uzsoy, R., "Implementing global factory schedules in the face of stochastic disruptions," *International Journal of Production Research*, vol. 43, no. 4, pp. 793–818, 2005.

[20] Beasley, D., Paredis, J., Palmer, C. C., Kershenbaum, A., Iba, H., Nordin, P., Mitchell, M., Crutchfield, J. P., Das, R., Schaffer, J. D., and others, "Handbook of evolutionary computation," 1997.

[21] Becz, S., Pinto, A., Zeidner, L., Khire, R., Reeve, H., and Banaszuk, A., "Design system for managing complexity in aerospace systems," in *Aviation Technology, Integration, and Operations (ATIO) Conferences*, pp. –, American Institute of Aeronautics and Astronautics, Sept. 2010.

[22] Ben-Ameur, W., "Computing the initial temperature of simulated annealing," *Computational Optimization and Applications*, vol. 29, no. 3, pp. 369–385, 2004.

[23] Benjaafar, S., "Intelligent simulation for flexible manufacturing systems: An integrated approach," *Computers & Industrial Engineering*, vol. 22, no. 3, pp. 297 – 311, 1992.

[24] Bergman, B., de Mare, J., Svensson, T., and Loren, S., *Robust Design Methodology for Reliability: Exploring the Effects of Variation and Uncertainty.* John Wiley & Sons, 1 ed., August 2009.

[25] Berry, W. L. and Rao, V., "Critical ratio scheduling: An experimental analysis," *Management Science*, vol. 22, no. 2, pp. 192–201, 1975.

[26] Bhote, K. R., *World class quality: Using design of experiments to make it happen.* AMACOM Div American Mgmt Assn, 1999.

[27] Bianchi, L., Dorigo, M., Gambardella, L. M., and Gutjahr, W. J., "A survey on metaheuristics for stochastic combinatorial optimization," *Natural Computing: an international journal*, vol. 8, no. 2, pp. 239–287, 2009.

[28] Błażewicz, J., Ecker, K. H., Pesch, E., Schmidt, G., and Weglarz, J., *Scheduling computer and manufacturing processes.* Springer Science & Business Media, 1996.

[29] Blum, C. and Roli, A., "Metaheuristics in combinatorial optimization: Overview and conceptual comparison," *ACM Comput. Surv.*, vol. 35, pp. 268–308, Sept. 2003.

[30] Bölte, A. and Thonemann, U. W., "Optimizing simulated annealing schedules with genetic programming," *European Journal of Operational Research*, vol. 92, no. 2, pp. 402–416, 1996.

[31] Borshchev, A. and Filippov, A., "From System Dynamics and Discrete Event to Practical Agent Based Modeling: Reasons, Techniques, Tools,"

[32] Branke, J. and Deb, K., "Integrating user preferences into evolutionary multi-objective optimization," in *Knowledge incorporation in evolutionary computation*, pp. 461–477, Springer, 2005.

[33] Brown, D. E., Marin, J. A., and Scherer, W. T., *A survey of intelligent scheduling systems.* Springer, 1995.

[34] Butterfield, J., Crosby, S., Curran, R., Price, M., Armstrong, C. G., Raghunathan, S., McAleenan, D., and Gibson, C., "Optimization of aircraft fuselage assembly process using digital manufacturing," *Journal of Computing and Information Science in Engineering*, vol. 7, pp. 269–275, June 2007.

[35] Captain, T., “Can we afford our own future? why a&d programs are late and over-budget and what can be done to fix the problem.” Online, 2009.

[36] Caridi, M. and Cavalieri, S., “Multi-agent systems in production planning and control: an overview,” *Production Planning & Control*, vol. 15, no. 2, pp. 106–118, 2004.

[37] Chance, F., Robinson, J., and Fowler, J. W., “Supporting manufacturing with simulation: Model design, development, and deployment,” in *Proceedings of the 28th Conference on Winter Simulation*, WSC ’96, (Washington, DC, USA), pp. 114–121, IEEE Computer Society, 1996.

[38] Chen, C.-H., *Stochastic simulation optimization: an optimal computing budget allocation*, vol. 1. World scientific, 2010.

[39] Chen, C.-H., Lin, J., Yücesan, E., and Chick, S., “Simulation budget allocation for further enhancing the efficiency of ordinal optimization,” *Discrete Event Dynamic Systems*, vol. 10, no. 3, pp. 251–270, 2000.

[40] Chen, T.-C. and Chang, Y.-W., “Modern floorplanning based on b*-tree and fast simulated annealing,” *IEE Transactions on Somputer-Aided Design of Integrated Circuits and Systems*, vol. 25, pp. 637–650, April 2006.

[41] Chong, C., Sivakumar, A. I., and Gay, R., “Simulation-based scheduling for dynamic discrete manufacturing,” in *Simulation Conference, 2003. Proceedings of the 2003 Winter*, vol. 2, pp. 1465–1473 vol.2, Dec 2003.

[42] Chretienne, P., Coffman, E. G., Lenstra, J. K., and Liu, Z., eds., *Scheduling Theory and its Applications.* John Wiley and Sons, 1995.

[43] Clark, N., “The airbus saga: Crossed wires and a multibillion-euro delay.” Online, December 2006.

[44] Cottrell, W. D., "Simplified program evaluation and review technique (pert)," *Journal of construction Engineering and Management*, vol. 125, no. 1, pp. 16–22, 1999.

[45] Crawford, S. and Wiers, V. C., "From anecdotes to theory: a review of existing knowledge on human factors of planning and scheduling," *Human performance in planning and scheduling*, pp. 15–43.

[46] Davenport, A., Gefflot, C., and Beck, C., "Slack-based techniques for robust schedules," in *Sixth European Conference on Planning*, 2014.

[47] De Boer, R., *Resource-constrained multi-project management.* PhD thesis, PhD thesis, University of Twente, The Netherlands, 1998.

[48] de Vonder, S. V., Demeulemeester, E., Herroelen, W., and Leus, R., "The use of buffers in project management: The trade-off between stability and makespan," *International Journal of Production Economics*, vol. 97, no. 2, pp. 227 – 240, 2005.

[49] Deb, K. and Jain, S., "Running performance metrics for evolutionary multi-objective optimization," 2002.

[50] Deb, K., Pratap, A., Agarwal, S., and Meyarivan, T., "A fast and elitist multiobjective genetic algorithm: Nsga-ii," *Evolutionary Computation, IEEE Transactions on*, vol. 6, no. 2, pp. 182–197, 2002.

[51] Deb, K., Thiele, L., Laumanns, M., and Zitzler, E., "Scalable multi-objective optimization test problems," in *Evolutionary Computation, 2002. CEC'02. Proceedings of the 2002 Congress on*, vol. 1, pp. 825–830, IEEE, 2002.

[52] Demeulemeester, E. L. and Herroelen, W. S., *Project scheduling: a research handbook*, vol. 49. Springer Science & Business Media, 2006.

[53] Dempster, M. A., *Deterministic and Stochastic Scheduling: Proceedings of the NATO Advanced Study and Research Institute on Theoretical Approaches to Scheduling Problems held in Durham, England, July 6–17, 1981*, vol. 84. Springer Science & Business Media, 2012.

[54] Dieter, G. E. and Schmidt, L. C., *Engineering design*, vol. 3. McGraw-Hill New York, 2013.

[55] Finnegan, M., "Boeing 787s to create half a terabyte of data per flight, says virgin atlantic." Online, March 2013.

[56] Fisher, K., "Agent-based design of holonic manufacturing systems," *Robotics and Autonomous Systems*, vol. 27, pp. 3 – 13, 1999. Multi-Agent Systems Applications.

[57] Fishman, G. S., *Discrete-Event Simulation: Modeling, Programming, and Analysis*. Springer Science & Business Media, 2001.

[58] Fowler, J. W. and Rose, O., "Grand challenges in modeling and simulation of complex manufacturing systems," *SIMULATION*, vol. 80, no. 9, pp. 469–476, 2004.

[59] Fox, M. S. and Smith, S. F., "Isis - a knowledge-based system for factory scheduling," *Expert systems*, vol. 1, no. 1, pp. 25–49, 1984.

[60] Framinan, J. M., Leisten, R., and García, R. R., *Manufacturing Scheduling Systems: An Integrated View on Models, Methods and Tools*. Springer, 2014.

[61] French, S., *Sequencing and scheduling: an introduction to the mathematics of the job-shop*. New York [etc.]: Horwood, 1982.

[62] Fu, M. C., "Optimization for simulation: Theory vs. practice," *INFORMS Journal on Computing*, vol. 14, no. 3, pp. 192–215, 2002.

[63] Fu, M. C., *Handbook of simulation optimization.* Springer, 2015.

[64] Gass, S. I., *Linear programming.* Wiley Online Library, 1958.

[65] Gaudin, S., “Nasa’s first orion launch will test tech that could take humans to mars.” Online, December 2014.

[66] Gen, M. and Cheng, R., *Genetic Algorithms & Engineering Optimization*, vol. 7 of *Wiley Series in Engineering Desifn and Automation.* John Wiley & Sons, 2000.

[67] Gilbert, J. K., “Models and modelling: Routes to more authentic science education,” *International Journal of Science and Mathematics Education*, vol. 2, no. 2, pp. 115–130, 2004.

[68] Gilbert, S. W., “Model building and a definition of science,” *Journal of Research in Science Teaching*, vol. 28, no. 1, pp. 73–79, 1991.

[69] Glover, F., “Tabu search: A tutorial,” *Interfaces*, vol. 20, no. 4, pp. 74–94, 1990.

[70] Glover, F. and Kochenberger, G. A., *Handbook of metaheuristics.* Springer Science & Business Media, 2003.

[71] Goel, T. and Stander, N., “A study on the convergence of multiobjective evolutionary algorithms,” in *Preprint submitted to the 13th AIAA/ISSMO conference on Multidisciplinary Analysis Optimization*, pp. 1–18, 2010.

[72] GOLDRATT, E. M., “Computerized shop floor scheduling,” *International Journal of Production Research*, vol. 26, no. 3, pp. 443–455, 1988.

[73] Golenko-Ginzburg, D. and Gonik, A., “Stochastic network project scheduling with non-consumable limited resources,” *International Journal of Production Economics*, vol. 48, no. 1, pp. 29 – 37, 1997.

[74] Gosling, K., "E-enabled capabilities of the 787 dreamliner." Online, 2009.

[75] Grant, T., "Lessons for or from ai: A scheduling case study," *Journal of the Operational Research Society*, pp. 41–57, 1986.

[76] Graves, S. C., "Uncertainty and production planning." Online, December 2008.

[77] Gu, J., Gu, X., and Gu, M., "A novel parallel quantum genetic algorithm for stochastic job shop scheduling," *Journal of Mathematical Analysis and Applications*, vol. 355, no. 1, pp. 63 – 81, 2009.

[78] Heer, J., Van Ham, F., Carpendale, S., Weaver, C., and Isenberg, P., "Creation and collaboration: Engaging new audiences for information visualization," in *Information Visualization*, pp. 92–133, Springer, 2008.

[79] Herrmann, J. W., *Handbook of production scheduling*, vol. 89. Springer Science & Business Media, 2006.

[80] Herroelen, W. and Leus, R., "Robust and reactive project scheduling: a review and classification of procedures," *International Journal of Production Research*, vol. 42, no. 8, pp. 1599–1620, 2004.

[81] Herroelen, W. and Leus, R., "Project scheduling under uncertainty: Survey and research potentials," *European Journal of Operational Research*, vol. 165, no. 2, pp. 289 – 306, 2005. Project Management and Scheduling.

[82] Highsmith, H.-H., Brock, J., and Stephens, D., "Space launch system (sls) data acquisition and sensor system for human space flight," in *Aerospace Conference, 2015 IEEE*, pp. 1–9, March 2015.

[83] Hobday, M., "Product complexity, innovation and industrial organisation," *Research Policy*, vol. 26, no. 6, pp. 689 – 710, 1998.

[84] Hopp, W. J. and Spearman, M. L., *Factory Physics.* Waveland Press, 2011.

[85] Houston, D., *Structural Sensing, Health Monitoring and Performance Evaluation.* Taylor & Francis, 2010.

[86] Huang, G. Q., *Design for X: Concurrent Engineering Imperatives.* Springer Science & Business Media, 1996.

[87] Humphries, K. K., "Risk analysis and contingency determination using range estimating," *Project Control Professional*, vol. 47, no. 4, p. 16, 2009.

[88] Iassinovski, S., Artiba, A., Bachelet, V., and Riane, F., "Integration of simulation and optimization for solving complex decision making problems," *International Journal of Production Economics*, vol. 85, no. 1, pp. 3 – 10, 2003. Planning and Control of Productive Systems.

[89] Isaai, M. and Cassaigne, N., "Predictive and reactive approaches to the train-scheduling problem: a knowledge management perspective," *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, vol. 31, pp. 476–484, Nov 2001.

[90] Jaeggi, D., Parks, G. T., Kipouros, T., and Clarkson, P. J., "The development of a multi-objective tabu search algorithm for continuous optimisation problems," *European Journal of Operational Research*, vol. 185, no. 3, pp. 1192–1212, 2008.

[91] Jahangirian, M., Eldabi, T., Naseer, A., Stergioulas, L. K., and Young, T., "Simulation in manufacturing and business: A review," *European Journal of Operational Research*, vol. 203, no. 1, pp. 1 – 13, 2010.

[92] Jones, D., Mirrazavi, S., and Tamiz, M., “Multi-objective meta-heuristics: An overview of the current state-of-the-art,” *European Journal of Operational Research*, vol. 137, no. 1, pp. 1 – 9, 2002.

[93] Józefowska, J. and Weglarz, J., *Perspectives in modern project scheduling*, vol. 92. Springer Science & Business Media, 2006.

[94] Jung, Y. and Woo, S., “Flexible work breakdown structure for integrated cost and schedule control,” *Journal of Construction Engineering and Management*, vol. 130, no. 5, pp. 616–625, 2004.

[95] Kalpakjian, S., *Manufacturing engineering and technology.* Pearson Education India, 2001.

[96] Karayanakis, N. M., *Advanced system modelling and simulation with block diagram languages.* CRC Press, 1995.

[97] Keim, D. A., Andrienko, G., Fekete, J.-D., Görg, C., Kohlhammer, J., and Melançon, G., *Information Visualization*, vol. 4950 of *Lectures Notes in Computer Science*, ch. Visual Analytics: Definition, Process, and Challenges, pp. 154–175. Springer Berlin / Heidelberg, 2008.

[98] Keim, D. A., Mansmann, F., and Thomas, J., “Visual analytics: How much visualization and how much analytics,” *SigKDD Explorations Journal*, December 2009.

[99] Kelton, W. D., Sadowski, R. P., and Sturrock, D. T., *Simulation with Arena.* McGraw Hill Boston, 4 ed., 2007.

[100] Kelton, W. D., Smith, J. S., and Sturrock, D. T., *Simio and Simulation: Modeling, Analysis, and Applications.* Simio LLC, 3 ed., 2013.

[101] Khan, M. B. and Zhou, X., “Stochastic optimization model and solution algorithm for robust double-track train-timetabling problem,” *Intelligent Transportation Systems, IEEE Transactions on*, vol. 11, no. 1, pp. 81–89, 2010.

[102] Khuri, A. I. and Mukhopadhyay, S., “Response surface methodology,” *Wiley Interdisciplinary Reviews: Computational Statistics*, vol. 2, no. 2, pp. 128–149, 2010.

[103] Kirkpatrick, S., Gelatt, C. D., Vecchi, M. P., and others, “Optimization by simmulated annealing,” *Science*, vol. 220, no. 4598, pp. 671–680, 1983.

[104] Klemmt, A., Horn, S., Weigert, G., and Wolter, K.-J., “Simulation-based optimization vs. mathematical programming: A hybrid approach for optimizing scheduling problems,” *Robotics and Computer-Integrated Manufacturing*, vol. 25, no. 6, pp. 917 – 925, 2009. 18th International Conference on Flexible Automation and Intelligent Manufacturing.

[105] Kobyłanński, P. and Kuchta, D., “A note on the paper by m. a. al-fawzan and m. haouari about a bi-objective problem for robust resource-constrained project scheduling,” *International Journal of Production Economics*, vol. 107, no. 2, pp. 496 – 501, 2007. Operations Management in China.

[106] Koren, Y., Heisel, U., Jovane, F., Moriwaki, T., Pritschow, G., Ulsoy, G., and Brussel, H. V., “Reconfigurable manufacturing systems,” *CIRP Annals - Manufacturing Technology*, vol. 48, no. 2, pp. 527 – 540, 1999.

[107] Koren, Y. and Shpitalni, M., “Design of reconfigurable manufacturing systems,” *Journal of Manufacturing Systems*, vol. 29, no. 4, pp. 130 – 141, 2010.

[108] Kutanoglu, E. and Sabuncuoglu, I., "Experimental investigation of iterative simulation-based scheduling in a dynamic and stochastic job shop," *Journal of Manufacturing Systems*, vol. 20, no. 4, pp. 264 – 279, 2001.

[109] Lacksonen, T., "Empirical comparison of search algorithms for discrete event simulation," *Computers & Industrial Engineering*, vol. 40, pp. 133 – 148, 2001.

[110] Law, A. M., *Simulation modeling and analysis.* McGraw Hill Boston, 4 ed., 2007.

[111] Leitão, P., "Agent-based distributed manufacturing control: A state-of-the-art survey," *Engineering Applications of Artificial Intelligence*, vol. 22, no. 7, pp. 979 – 991, 2009. Distributed Control of Production Systems.

[112] Lepadatu, L., "Risk determination in projects. the advantages and disadvantages of stochastic methods," *Acta Universitatis Danubius. Œconomica*, vol. 5, no. 1, 2010.

[113] Li, Z. and Ierapetritou, M., "Process scheduling under uncertainty: Review and challenges," *Computers & Chemical Engineering*, vol. 32, no. 4–5, pp. 715 – 727, 2008. Festschrift devoted to Rex Reklaitis on his 65th Birthday.

[114] Ligetti, C., Simpson, T. W., Frecker, M., Barton, R. R., and Stump, G., "Assessing the impact of graphical design interfaces on design efficiency and effectiveness," *Journal of Computing and Information Science in Engineering*, vol. 3, pp. 144–154, June 2003.

[115] Lin, X., Janak, S. L., and Floudas, C. A., "A new robust optimization approach for scheduling under uncertainty: I. bounded uncertainty," *Computers & Chemical Engineering*, vol. 28, pp. 1069 – 1085, 2004. FOCAPO 2003 Special issue.

[116] Lockett, H., Fletcher, S., and Luquet, N., “Applying design for assembly principles in computer aided design to make small changes that improve the efficiency of manual aircraft systems installations,” *SAE International Journal of Aerospace*, vol. 7, no. 2014-01-2266, pp. 284–291, 2014.

[117] Luenberger, D. G. and Ye, Y., *Linear and nonlinear programming*, vol. 116. Springer Science & Business Media, 2008.

[118] Macal, C. M. and North, M. J., “Tutorial on agent-based modeling and simulation,” in *Proceedings of the 37th conference on Winter simulation*, pp. 2–15, Winter Simulation Conference, 2005.

[119] Majid, M. A., Aickelin, U., and Siebers, P.-O., “Comparing simulation output accuracy of discrete event and agent based models: a quantitative approach,” in *Proceedings of the 2009 Summer Computer Simulation Conference*, pp. 177–184, Society for Modeling & Simulation International, 2009.

[120] Markgraf, B., “The disadvantages of pert in project management.” Online.

[121] Mathaisel, D. F., Manary, J. M., and Criscimagna, N. H., *Engineering for sustainability.* CRC Press, 2012.

[122] MATLAB, *version 8.6.0 (R2015b).* Natick, Massachusetts: The MathWorks Inc., 2015.

[123] Mattfeld, D. C., *Evolutionary search and the job shop.* Production and Logistics, Physica-Verlag, 1996.

[124] Mavris, D. N., Pinon, O. J., and Fullmer Jr., D., “Systems design and modeling: A visual analytics approach,” in *27th Congress of International Council of the Aeronautical Sciences (ICAS)*, 2010.

[125] McKay, K. N., Buzacott, J. A., Charness, N., and Safayeni, F. R., *Artificial Intelligence in Operational Research*, ch. The Scheduler's Predictive Expertise: An Interdisciplinary Perspective, pp. 139–150. London: Macmillan Education UK, 1992.

[126] McKay, K., Pinedo, M., and Webster, S., "Practice-focused research issues for scheduling systems," *Production and Operations Management*, vol. 11, no. 2, p. 249, 2002.

[127] McKay, K. N., Buzacott, J. A., and Safayeni, F. R., "The schedulerŠs knowledge of uncertainty: the missing link," *Knowledge based production management systems*, pp. 171–189, 1989.

[128] McKay, K. N. and Wiers, V. C., "Unifying the theory and practice of production scheduling," *Journal of Manufacturing Systems*, vol. 18, no. 4, pp. 241 – 255, 1999. Special issue on scheduling: From Research Into Practice.

[129] McMullen, P. R. and Frazier, G., "Using simulated annealing to solve a multiobjective assembly line balancing problem with parallel workstations," *International Journal of Production Research*, vol. 36, no. 10, pp. 2717–2741, 1998.

[130] Medeiros, A. K. A. D. and Weijters, A. J. M. M., "Genetic process mining," in *Applications and Theory of Petri Nets 2005, volume 3536 of Lecture Notes in Computer Science*, pp. 48–69, Springer-Verlag, 2005.

[131] Meixner, H., *Sensors, Micro-and Nanosensor Technology: Trends in Sensor Markets*, vol. 8. John Wiley & Sons, 2008.

[132] Melchiors, P., *Dynamic and Stochastic Multi-project Planning*, vol. 673. Springer, 2015.

[133] Michaels, D., "Hit by delays, airbus tries new way of building planes," July 2012.

[134] Momoh, A., Roy, R., and Shehab, E., "Challenges in enterprise resource planning implementation: state of the art," *Business Process Management Journal*, vol. 16, no. 4, pp. 537–565, 2010.

[135] Monostori, L., Va'ncza, J., and Kumara, S., "Agent-based systems for manufacturing," *CIRP Annals - Manufacturing Technology*, vol. 55, no. 2, pp. 697 – 720, 2006.

[136] Morton, T. and Pentico, D. W., *Heuristic scheduling systems: with applications to production systems and project management*, vol. 3. John Wiley & Sons, 1993.

[137] Mourtzis, D., Doukas, M., and Bernidaki, D., "Simulation in manufacturing: Review and challenges," *Procedia CIRP*, vol. 25, pp. 213 – 229, 2014. 8th International Conference on Digital Enterprise Technology - DET 2014 Disruptive Innovation in Manufacturing Engineering towards the 4th Industrial Revolution.

[138] Na, W., Wuliang, P., and Hua, G., "A robustness simulation method of project schedule based on the monte carlo method," *The Open Cybernetics & Systemics Journal*, vol. 8, pp. 254–258, 2014.

[139] Nam, D. and Park, C. H., "Multiobjective simulated annealing: A comparative study to evolutionary algorithms," *International Journal of Fuzzy Systems*, vol. 2, no. 2, pp. 87–97, 2000.

[140] Nasution, S., "Fuzzy critical path method," *Systems, Man and Cybernetics, IEEE Transactions on*, vol. 24, pp. 48–57, Jan 1994.

[141] Nebylov, A., *Aerospace Sensors*. EBSCO ebook academic collection, Momentum Press, 2012.

[142] O'Leary, D. E., *Enterprise resource planning systems: systems, life cycle, electronic commerce, and risk*. Cambridge university press, 2000.

[143] Oppenhein, B. W., *Lean for Systems Engineering with Lean Enablers for System Engineering*. John Wiley & Sons Inc., 2011.

[144] Opricovic, S. and Tzeng, G.-H., "Compromise solution by MCDM methods: A comparative analysis of VIKOR and TOPSIS," *European Journal of Operational Research*, vol. 156, no. 2, pp. 445 – 455, 2004.

[145] Ouelhadj, D. and Petrovic, S., "A survey of dynamic scheduling in manufacturing systems," *Journal of Scheduling*, vol. 12, no. 4, pp. 417–431, 2009.

[146] Parejo, J., Ruiz-Cort?es, A., Lozano, S., and Fernandez, P., "Metaheuristic optimization frameworks: a survey and benchmarking," *Soft Computing*, vol. 16, no. 3, pp. 527–561, 2012.

[147] Pezzella, F., Morganti, G., and Ciaschetti, G., "A genetic algorithm for the flexible job-shop scheduling problem," *Computers & Operations Research*, vol. 35, no. 10, pp. 3202 – 3212, 2008. Part Special Issue: Search-based Software Engineering.

[148] Pinedo, M., *Planning and scheduling in manufacturing and services*, vol. 24. Springer, 2005.

[149] Pinedo, M. L., *Scheduling: theory, algorithms, and systems*. Springer Science & Business Media, 2012.

[150] Powell, A., Mander, K., and Brown, D., "Strategies for lifecycle concurrency and iteration: A system dynamics approach," *Journal of Systems and Software*, vol. 46, pp. 151 – 161, 1999.

[151] Pugh, G. A., "Agent-based simulation of discrete-event systems," in *Proceedings of the 2006 Illinois-Indiana and North Central Joint Section Conferences,(Mar 31-Apr 1)*, Citeseer, 2006.

[152] Pujawan, I. N., "Schedule nervousness in a manufacturing system: a case study," *Production planning & control*, vol. 15, no. 5, pp. 515–524, 2004.

[153] Quarteroni, A., "Mathematical models in science and engineering," *Notices of the AMS*, vol. 56, no. 1, pp. 10–19, 2009.

[154] Raymer, D. P., *Aircraft Design: A Conceptual Approach.* Reston: American Institue of Aeronautics and Astronautics, fourth ed., 2006.

[155] Ruiz, R. and Maroto, C., "A comprehensive review and evaluation of permutation flowshop heuristics," *European Journal of Operational Research*, vol. 165, no. 2, pp. 479 – 494, 2005. Project Management and Scheduling.

[156] Sabuncuoglu, I. and Bayiz, M., "Analysis of reactive scheduling problems in a job shop environment," *European Journal of Operational Research*, vol. 126, no. 3, pp. 567 – 586, 2000.

[157] Salvendy, G., *Handbook of industrial engineering: technology and operations management.* John Wiley & Sons, 2001.

[158] Sampigethaya, K., Poovendran, R., Bushnell, L., Li, M., Robinson, R., and Lintelman, S., "Secure wireless collection and distribution of commercial airplane health data," *Aerospace and Electronic Systems Magazine, IEEE*, vol. 24, pp. 14–20, July 2009.

[159] Sanderson, P. M., "The human planning and scheduling role in advanced manufacturing systems: an emerging human factors domain," *Human Factors: The Journal of the Human Factors and Ergonomics Society*, vol. 31, no. 6, pp. 635–666, 1989.

[160] Schaffer, J. R. and Kim, M.-J., "Number of replications required in control chart monte carlo simulation studies," *Communications in Statistics - Simulation and Computation*, vol. 36, no. 5, pp. 1075–1087, 2007.

[161] Schwartz, J. D., Wang, W., and Rivera, D. E., "Simulation-based optimization of process control policies for inventory management in supply chains," *Automatica*, vol. 42, no. 8, pp. 1311 – 1320, 2006. Optimal Control Applications to Management Sciences.

[162] Shah, D., "Millions of data points flying in tight formation." Online, December 2014.

[163] Shapiro, A., "Monte carlo sampling methods," in *Stochastic Programming* (Ruszczynski, A. and Shapiro, A., eds.), vol. 10 of *Handbooks in Operations Research and Management Science*, pp. 353 – 425, Elsevier, 2003.

[164] Siedlak, D. J., Schlais, P. R., Pinon, O. J., and Mavris, D. N., "Supporting affordability-based design decisions in the presence of demand variability," in *Proceedings of ASME 2015 International Manufacturing Science and Engineering Conference*, no. MSEC2015-9422, American Society of Mechanical Engineers, June 2015.

[165] Siedlak, D. J., Schmidt, T. M., Pinon, O. J., and Mavris, D. N., "A methodology for the parametric exploration of the impact of production

planning on the early stages of design," in *Proceedings of ASME 2014 International Manufacturing Science and Engineering Conference*, no. MSEC2014-3974, American Society of Mechanical Engineers, June 2014.

[166] Simio, *version 8.141.* Sewickley, Pennsylvania: Simio LLC, 2016.

[167] Simio LLC, "Simulation versus optimization based scheduling." Online, 2016.

[168] Stark, J., *Product Lifecycle Management: 21st Century Paradigm for Product Realisation.* Springer, 2005.

[169] Staszewski, W., Boller, C., and Tomlinson, G., *Health Monitoring of Aerospace Structures: Smart Sensor Technologies and Signal Processing.* Wiley, 2004.

[170] Stork, F., *Stochastic Resource-constrained Project Scheduling.* PhD thesis, Technische Universit?t Berlin, 2001.

[171] Studor, G., ""fly-by-wireless": A revolution in aerospace vehicle architecture for instrumentation and control." Online, January 2007.

[172] Stump, G., Simpson, T. W., Yukish, M., and Bennett, L., "Multidimensional visualization and its application to a design by shopping paradigm," in *Proceedings of the 9th AIAA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, no. AIAA-2002-5622, (Atlanta, GA, USA), 4-6 September 2002.

[173] Sullivan, M. J., "Defense acquisitions: Assessments of selected weapon programs," Tech. Rep. GAO-15-342SP, U.S. Government Accountability Office, April 2015.

[174] Suman, B. and Kumar, P., “A survey of simulated annealing as a tool for single and multiobjective optimization,” *Journal of the Operational Research Society*, vol. 57, pp. 1143–1160, 2006.

[175] Sundaresan, C., Tang, Z., Ceisel, J., and Mavris, D., “A methodology for parametric production planning in preliminary aircraft design,” in *The 28th Congress of the International Council of the Aeronautical Sciences (ICAS 2012)*, (Brisbane, Australia), 2012.

[176] Swenson, L. A., Little, J., Manning, J., and van der Krogt, R., “Using sensitivity analysis to illustrate and improve schedule robustness,” January 2010.

[177] Taguchi, G., *Introduction to quality engineering: designing quality into products and processes.* 1986.

[178] Taillard, E., “Robust taboo search for the quadratic assignment problem,” *Parallel Computing*, vol. 17, pp. 443 – 455, 1991.

[179] Thomas, J. J., *Illuminating the path: the research and development agenda for visual analytics.* IEEE Computer Society, 2005.

[180] T’kindt, V. and Billaut, J.-C., *Multicriteria scheduling: theory, models and algorithms.* Springer Science & Business Media, 2 ed., 2006.

[181] Tobiasz, M., Isenberg, P., and Carpendale, S., “Lark: Coordinating co-located collaboration with information visualization,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 15, pp. 1065–1072, Nov 2009.

[182] Touran, A. and Wiser, E., “Monte carlo technique with correlated random variables,” *Journal of Construction Engineering and Management*, vol. 118, no. 2, pp. 258–272, 1992.

[183] Tsai, Y.-W. and Gemmill, D. D., "Using tabu search to schedule activities of stochastic resource-constrained projects," *European Journal of Operational Research*, vol. 111, no. 1, pp. 129 – 141, 1998.

[184] Usher, J. M., Roy, U., and Parsaei, H. R., *Integrated Product and Process Develoment: Methods, Tools, and Techologies.* John Wiley & Sons Inc., 1998.

[185] Uzsoy, R., "Manufacturing scheduling: Are the times a-changin'?," *Journal of Manufacturing Systems*, vol. 18, no. 4, pp. i – ii, 1999. Special issue on scheduling: From Research Into Practice.

[186] Van de Vonder, S., Demeulemeester, E., and Herroelen, W., "An investigation of efficient and effective predictive-reactive project scheduling procedures," *DTEW Research Report 0466*, pp. 1–29, 2004.

[187] Van de Vonder, S., Demeulemeester, E., and Herroelen, W., "A classification of predictive-reactive project scheduling procedures," *Journal of Scheduling*, vol. 10, no. 3, pp. 195–207, 2007.

[188] van der Aalst, W. M. P., de Beer, H. T., and van Dongen, B. F., *On the Move to Meaningful Internet Systems 2005: CoopIS, DOA, and ODBASE: OTM Confederated International Conferences, CoopIS, DOA, and ODBASE 2005, Agia Napa, Cyprus, October 31 - November 4, 2005, Proceedings, Part I*, ch. Process Mining and Verification of Properties: An Approach Based on Temporal Logic, pp. 130–147. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005.

[189] van der Aalst, W., Reijers, H., Weijters, A., van Dongen, B., de Medeiros, A. A., Song, M., and Verbeek, H., "Business process mining: An industrial application," *Information Systems*, vol. 32, no. 5, pp. 713 – 732, 2007.

[190] Van Dyke Parunak, H., Savit, R., and Riolo, R., "Agent-based modeling vs. equation-based modeling: A case study and users guide," in *Multi-Agent Systems and Agent-Based Simulation* (Sichman, J., Conte, R., and Gilbert, N., eds.), vol. 1534 of *Lecture Notes in Computer Science*, pp. 10–25, Springer Berlin Heidelberg, 1998.

[191] van Laarhoven, P. J. M., Aarts, E. H. L., and Lenstra, J. K., "Job shop scheduling by simulated annealing," *Operations Research*, vol. 40, no. 1, pp. 113–125, 1992.

[192] Van Veldhuizen, D. A., *Multiobjective evolutionary algorithms: classifications, analyses, and new innovations.* PhD thesis, Air Force Institute of Technology, June 1999.

[193] Van Veldhuizen, D. A. and Lamont, G. B., "Evolutionary computation and convergence to a pareto front," in *Late breaking papers at the genetic programming 1998 conference*, pp. 221–228, Citeseer, 1998.

[194] Vanderplaats, G. N., *Multidiscipline Design Optimization.* Monterey, CA: Vanderplaats Research & Development, Inc., 1st ed., 2007.

[195] Vanhoucke, M., "Monte-carlo simulations: Linking critical path schedules to project control," April 2012.

[196] Velasquez, M. and Hester, P. T., "An analysis of multi-criteria decision making methods," *International Journal of Operations Research*, vol. 10, no. 2, pp. 56–66, 2013.

[197] Veldhuizen, D. A. V. and Lamont, G. B., "On measuring multiobjective evolutionary algorithm performance," in *Evolutionary Computation, 2000. Proceedings of the 2000 Congress on*, vol. 1, pp. 204–211 vol.1, 2000.

[198] Vernon, C., "Lingering amongst the lingerie: An observation-based study into support for scheduling at a garment manufacturer," *Human performance in planning and scheduling*, pp. 135–163, 2001.

[199] Vieira, G., Herrmann, J., and Lin, E., "Rescheduling manufacturing systems: A framework of strategies, policies, and methods," *Journal of Scheduling*, vol. 6, no. 1, pp. 39–62, 2003.

[200] Wang, J., "A fuzzy project scheduling approach to minimize schedule risk for product development," *Fuzzy Sets and Systems*, vol. 127, no. 2, pp. 99 – 116, 2002.

[201] Webster, S., "A case study of scheduling practice at a machine tool manufacturer," *Human performance in planning and scheduling*, pp. 67–81, 2001.

[202] White, K., "Advances in the theory and practice of production scheduling," *Advances in Control and Dynamic Systems*, pp. 115–157, 2012.

[203] Widmer, M. and Hertz, A., "A new heuristic method for the flow shop sequencing problem," *European Journal of Operational Research*, vol. 41, no. 2, pp. 186–193, 1989.

[204] Wiedenman, N., "Adaptive vehicle make." Online, January 2013.

[205] Wiers, V. C., "A quantitative field study of the decision behaviour of four shopfloor schedulers," *Production Planning & Control*, vol. 7, no. 4, pp. 383–392, 1996.

[206] Wiers, V. C., "Human-computer interaction in production scheduling: Analysis and design of decision support systems for production scheduling tasks," 1997.

[207] Wiers, V. C. and Van Der Schaaf, T. W., "A framework for decision support in production scheduling tasks," *Production Planning & Control*, vol. 8, no. 6, pp. 533–544, 1997.

[208] Wilhelm, M. R. and Ward, T. L., "Solving quadratic assignment problems by 'simulated annealing'," *IIE transactions*, vol. 19, no. 1, pp. 107–119, 1987.

[209] Williamson, D. F., Parker, R. A., and Kendrick, J. S., "The box plot: a simple visual method to interpret data," *Annals of internal medicine*, vol. 110, no. 11, pp. 916–921, 1989.

[210] Wilson, J. M., "Gantt charts: A centenary appreciation," *European Journal of Operational Research*, vol. 149, no. 2, pp. 430 – 437, 2003. Sequencing and Scheduling.

[211] Wolpert, D. H. and Macready, W. G., "No free lunch theorems for optimization," *IEEE transactions on evolutionary computation*, vol. 1, no. 1, pp. 67–82, 1997.

[212] Wood, J., Wright, H., and Brodie, K., "Collaborative visualization," in *Visualization '97., Proceedings*, pp. 253–259, Oct 1997.

[213] Wu, S. D., Storer, R. H., and Pei-Chann, C., "One-machine rescheduling heuristics with efficiency and stability as criteria," *Computers & Operations Research*, vol. 20, no. 1, pp. 1–14, 1993.

[214] Xhafa, F. and Abraham, A., *Metaheuristics for Scheduling in Industrial and Manufacturing Applications.* Studies in Computational Intelligence, Springer, 2008.

[215] Zhang, Y. and Chen, H., “A knowledge-based dynamic job-scheduling in low-volume/high-variety manufacturing,” *Artificial Intelligence in Engineering*, vol. 13, no. 3, pp. 241 – 249, 1999.

[216] Zitzler, E., Deb, K., and Thiele, L., “Comparison of multiobjective evolutionary algorithms: Empirical results,” *Evolutionary computation*, vol. 8, no. 2, pp. 173–195, 2000.

[217] Zitzler, E., Thiele, L., Laumanns, M., Fonseca, C. M., and Da Fonseca, V. G., “Performance assessment of multiobjective optimizers: an analysis and review,” *Evolutionary Computation, IEEE Transactions on*, vol. 7, no. 2, pp. 117–132, 2003.

[218] Zuo, X., Mo, H., and Wu, J., “A robust scheduling method based on a multi-objective immune algorithm,” *Information Sciences*, vol. 179, no. 19, pp. 3359 – 3369, 2009.