THE DESIGN AND SIMULATION OF A BIO-INSPIRED MULTI-AGENT PARKING SYSTEM

A Dissertation
Presented to
The Academic Faculty

by

Amelia Tee Qiao Ying

In Partial Fulfillment
of the Requirements for the Degree
Master of Science in Mechanical Engineering

Georgia Institute of Technology

May 2018

THE DESIGN AND SIMULATION OF A BIO-INSPIRED PARKING ALGORITHM

Approved by:

Dr. Bert Bras, Advisor
School of Mechanical Engineering
Georgia Institute of Technology

Dr. Cassandra Telenko
School of Mechanical Engineering
Georgia Institute of Technology

Dr. Marc Weissburg
School of Biology
Georgia Institute of Technology

Date Approved: April 10, 2018

ACKNOWLEDGEMENTS

Firstly, I would like to thank my advisor, Dr. Bert Bras, for his guidance over the duration of this research project. His advice has helped me navigate my academic life at Georgia Tech and will continue to be very valuable as I go on to start my professional career.

I would also like to thank Dr. Marc Weissburg regarding his help with the biological aspects of this thesis. He has directed me to some very useful resources that has helped formed the foundation of the work I have done. In addition, I want to thank Dr. Cassandra Telenko for agreeing to be part of my thesis committee and taking her time to review my work.

I would also like to thank my colleagues at the Sustainable Design and Manufacturing Laboratory. I would like to especially thank Stephen Malone and Adam Cantor for laying part of the project's foundation and providing some of the code that has been essential in this project. I would like to thank the other members for their help and advice with not only my thesis, but also with other aspects of graduate life.

Last but not least, I would like to thank my family and fiancé. Their continuous love and support has sustained me through this journey at Georgia Tech. I could not have done it without them.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS AND ABBREVIATIONS

P-R	Parked-Advert, Random-Scouting Algorithm
P-G	Parked-Advert, Greedy-Scouting Algorithm
P-R (REG-NEG)	Parked-Advert, Random-Scouting Algorithm with Regular Negative Feedback
P-G (REG-NEG)	Parked-Advert, Greedy-Scouting Algorithm with Regular Negative Feedback
P-R (INS-NEG)	Parked-Advert, Random-Scouting Algorithm with Instantaneous Negative Feedback
PL-R	Parked-Leave-Advert, Random-Scouting Algorithm
PL-G	Parked-Leave-Advert, Greedy-Scouting Algorithm
PL-R (REG-NEG)	Parked-Leave-Advert, Random-Scouting Algorithm with Regular Negative Feedback
PL-G (REG-NEG)	Parked-Leave-Advert, Greedy-Scouting Algorithm with Regular Negative Feedback
PL-R (INS-NEG)	Parked-Leave-Advert, Random-Scouting Algorithm with Instantaneous Negative Feedback

SUMMARY

Finding parking is one of the common hassles of modern life. Drivers can considerable amounts of time and fuel looking for parking in congested urban centers. Therefore, it is would be very valuable to find a method that can alleviate this parking problem. However, not all solutions are cost-feasible or in some cases, even feasible. The expansion of parking infrastructure is costly and it may be more feasible to implement efficient parking practices instead.

This thesis proposes a bio-inspired smart parking solution which aims to reduce the amount of time drivers take to find parking using vehicle-to-vehicle communication and the principles of swarm intelligence. It is based on how individual honeybees communicate with each other while foraging for food. In doing so, they are able to optimize the amount of nectar collected for the colony. In the same way, vehicles can communicate with each other to identify the locations in which they are more likely to find an empty parking space.

Results shows that the proposed algorithm is usually more efficient than existing parking algorithms with the exception of a small group of particular circumstances.

CHAPTER 1. INTRODUCTION

1.1 The Necessity and Development of Smart Parking

Finding a parking space is one of the common hassles of modern life, particularly in congested metropolitan areas. It can take a long time to find an available parking lot within walkable distance of one's desired destination. This results in increased traffic on the roads caused by cars searching for parking. The average parking search time is 8.1 minutes and the search for parking can make up anywhere from 8% to a significant 74% of traffic in urban areas (Shoup, 2006). The search for parking also leads to wasted petrol and unnecessary carbon emission. It has been measured that 47,000 gallons of gasoline have been consumed and 730 tons of carbon dioxide have been produced in over one year by cars searching for parking in a small Los Angeles business district (Shoup, 2005).

Therefore, it is important to reduce the amount of time and resources wasted in the process of finding parking. But not all methods to solve this parking problem are practicable. The expansion of parking infrastructure is often costly and, in some cases, infeasible. It is more sensible to implement efficient parking management practices. One of the solutions proposed to alleviate this parking problem is the concept of 'smart parking', in which researchers apply technologies to optimize the use of parking structures. The first smart parking systems were implemented in Europe and Japan during the early 1970s. The early systems were used to display the status of a parking space or garage, such as availability of a parking spot or the number of free spaces (Chinrungrueng et al., 2007). The detection of the availability of parking space is commonly executed using sensors. Tang et al. suggested a Smart Parking Management System in which one sensor is placed in each parking spot to monitor its occupation status. The driver can then access

this information via a wireless network and use it to locate vacant parking spaces. As time went on, other smart parking methods were proposed as other areas of technology such as mobile phones, matured and became widespread. Wang and He proposed a system where drivers can locate and reserve parking spaces ahead of time. Drivers can use their personal communication devices to access the parking system via Internet and reserve a spot prior to arriving at the lot. The availability of parking spaces will again be determined by sensors placed around the lot. Khang et al. designed a parking system that will assign the car the most optimal parking spot based on the short path algorithm via SMS. Recent years have seen an increased interest in the concept of the 'Internet of Things' (IoT). Tomar et al. designed an IoT-based street parking system. Drones are used to scout and determine the parking architecture in the desired area. The vacancy of parking sports is determined using IR sensors in the parking places. The information is stored and processed in a cloud centric server, a new technology that has gained a lot of popularity over the past few years.

1.2 Disadvantages of Current Smart Parking Technology and the Rise of Connected Vehicles

The discussed parking smart parking methods in the previous section do have their limitations. Many of them require the implementation of a larger infrastructure and may be hindered by certain restrictions. Idris et al. states that implementing sensors in parking lots have cost, environmental and scale factors to consider, which may affect the feasibility of implementing such a system in rougher areas. The system may also cause some damage to the existing parking infrastructure. For example, Mimbela and Klein state that one type of sensors used in parking applications are intrusive sensors, which require invasive methods of installation. One of the sensors discussed must be installed in holes drilled into the pavement, which consequently

decreases pavement life. In addition, lane closure would be needed to install and maintain the system, which would disrupt traffic. Reinstallation would be required if the pavement needs to be repaired or renovated, creating more inconveniences.

It would be useful to invent a smart parking system whose feasibility is not limited by external factors. This is possible with the recent rise of wireless networks. This has facilitated to the creation of the concept of 'connected vehicles', which refer to vehicles who are able to communicate and interact with their internal and external environments. This encompasses vehicle-to-sensor on-board (V2S), vehicle-to-vehicle (V2V), vehicle-to-road (V2R) and vehicle-to-internet (V2I) communication. Connected vehicles will form the foundation of an 'Internet of Vehicles' (IoV) in which information is gathered, processed and shared to enable the development and implementation of intelligent transportation systems (Lu et al., 2014). Such networking reduces the need to implement physical infrastructure. The variety of applications for such networks is extensive, ranging from collecting traffic light data from vehicles as environmental sensors (Liu et al., 2013) to using parked cars to enable vehicular internet access (Crepaldi et al., 2008). As such, it is possible to use this technology in smart parking applications.

1.3 The Effectiveness of Bio-inspired Optimization Algorithms

'Big data' is on the rise as recent years have seen an explosion in data. However, it is a challenge to find the optimal solution when one is bombarded with large amounts of data. As such, scientists have been drawing inspiration from efficient biological systems and implementing their mechanisms to find the most optimal in artificial systems (Kar, 2016). For example, the concept of neural networks is based on how the human brain operates (Grossberg, 1988) and has been used to solve complex, non-linear systems (Sadegh, 1993). Genetic algorithms are based on the process

of natural selection (Darwin, 1859) and have been used to solve combinatorial and non-deterministic multi-objective problems, like the problems of scheduling and production planning in the field of operation management (Aytug et al., 2003).

Bio-inspired algorithms have also been used to optimize urban infrastructure. Zari 2015 implements natural ecosystem services (i.e. provision of energy and fuel) in context of an urban environment. Lee et al. 2008 aimed to optimize the monitoring of urban environments (i.e. traffic reporting, surveillance) via Vehicular Sensor Networks (VSN) using the foraging behaviors of the organism Escherichia coli chemotaxis. However, the application of bio-inspired algorithms in parking optimization has not yet been investigated. As such, it is plausible that bio-inspiration can help alleviate parking problems, especially in congested areas.

1.4 Thesis Organization

This thesis will propose a novel smart parking system based on Ford's smart car-to-car communication technology and existing bio-inspired algorithms. The design and mechanisms of the smart parking algorithm will be covered. The algorithm was also evaluated via computer simulation and we will discuss its effectiveness compared to existing common parking algorithms.

The structure of the thesis is as follows. Chapter 2 contains a literature review of the bio-inspired algorithms on which the new parking system is based upon. Chapter 3 will cover the design and mechanism of the novel algorithm. The algorithm's results and performance are stated and discussed in Chapter 4. The implications of the results are discussed in Chapter 5 and how this would impact parking in a real-life parking situation.

CHAPTER 2. LITERATURE REVIEW

This chapter will serve as a literature review of swarm intelligence and the bio-inspired algorithms within that field. Further details will be covered concerning the Bee's Algorithm as one will see, it is the main bio-inspiration for the novel algorithm proposed in this thesis. It will also cover the concepts of vehicular networks and vehicular cloud computing, the technology that the algorithm will depend on to function.

2.1 Swarm Intelligence

The objective of this thesis is to create a novel algorithm that will help the car find a parking spot in the shortest amount of time possible based only on information given by other cars on where they successfully or unsuccessfully found a vacant parking space. A possible source of inspiration for such an algorithm is swarm intelligence, which refers to the collective intelligent behavior of a system resulting from the simple, individual actions of independent agents (Bonabeau et al., 1999). The field is heavily inspired by the collective behavior of social creatures such as insects, birds and fish. The marvel of these particular natural system is that the members of such systems are quite simple in intelligence (i.e. ants, bees). The colonies are often characterized by decentralized control (i.e. lack of central control) and the activities of its members are self-organized. However, the collective behavior of these uncomplicated agents is complex and quite successful in ensuring the survival of the species (Blum and Li, 2008).

Bonabeau et al. define self-organization as when behavior structures appear at the global level as a result of interactions between individual parts of the system and consider it to be a key characteristic in swarm intelligence. It can be characterized by the following four attributes.

1. Positive Feedback: There exists a mechanism that enforces encourages individuals to pursue the optimal solution
2. Negative Feedback: When a solution is not or no longer favorable, there is a mechanism that discourages swarm members from following it
3. Presence of Fluctuations: The global structural behavior is present despite random alterations in the environment. In addition, fluctuations could also result in the creation of a new solution, which can impact the collective behavior of the swarm.
4. Multiple Interactions: There must be interactions between the swarm members so that an individual can benefit not only from the results of its own activity, but of others as well.

However, not all swarms can be considered to exhibit complex and intelligent behavior. Millonas lists the following set of conditions that a swarm must fulfill in order to be considered 'intelligent'.

1. Proximity Principle: The swarm must be able to perform simple space and time calculations.
2. Quality Principle: In addition to space and time, the swarm must be able to react to the quality of the object concerned (i.e. food).
3. Principle of Diverse Response: The group must not narrowly commit all its resources. Instead, it is able to distribute its sources among many options in the case that one of them may fail.
4. Principle of Stability: The group is not reactive to every fluctuation in the environment. In other words, it understands that every reaction to the surroundings is not always profitable.

5. Principle of Adaptability: If needed, the group is able to change its behavior, especially if adaptability is profitable.

Swarm intelligence encompasses a number of bio-inspired optimization algorithms which are based on these natural group behaviors. Some examples include Ant Colony Optimization (ACO), Bee Colony Optimization (BCO), Particle Swarm Optimization (PSO) and Stochastic Diffusion (SDS).

2.1.1 Ant Colony Optimization (ACO)

The Ant Colony Optimization (ACO) algorithm is based on the foraging behavior of ants. While scavenging food, Grassé discovered that an ant naturally deposit pheromone on the ground to mark an advantageous path, usually leading to a food source. Other ants can detect and follow the scent to the same food source. The most convenient (i.e. shortest) path will be used by more ants, consequently acquiring a larger amount of pheromones. Following ants will be able to identify the most optimal solution by choosing the path with stronger pheromones. In addition, ants will be able to complete the path in shorter amount of time and reach the nest first. As a result, the shortest path will receive more pheromones earlier, directing more ants down this path (Dorigo, Birattari & Stutzle, 2006).

ACO has been applied to solve a variety of problems. Dorigo and Gambardella used it to solve the traveling salesman problem, whose setting is set of cities and the distance between the locations are known. The optimal solution is the shortest path that visits each city once and only once. The algorithm creates multiple ants, which generates solution by randomly creating a path between the cities, marking it with pheromones and increasing its desirability measure. The optimal solution would the path with the highest desirability measure. The algorithm also has been used to optimize vehicle routing. Liu and Cai used ACO to solve the capacitated vehicle routing problem, in which the desired route is the most efficient path that visits a number of customers

from a central depot. Rizzoli, Montemanni, Lucibello and Gambardella applied ACO to solve a number of real-life vehicle route problems ranging from vehicle routing with time windows with time windows for a major supermarket chain to a time-dependent routing problem for a distribution company.

2.1.2 Bee Colony Optimization (BCO)

The Bee Colony Optimization algorithm is inspired by the food foraging behaviors of a honeybee colony. The process starts when bee foragers leave the hive to search for nectar sources. If a forager is successful in locating a food source, the forager takes some of the nectar with it. When it returns to the hive, it not only unloads the nectar, but also recruit other honeybees to search at its food source. It will tell other bees where it got the nectar, how far the food source is from the hive, the quality of the nectar and other important information. Using this information, other bees in the colony may go to the advertised food source to collect more nectar for the hive (Karaboga and Akay, 2009).

The Bee Colony Optimization is generally used when the optimal solution is obtained via task or effort allocation. Tedorovic and Dell'Orco applied the algorithm to optimize 'ridesharing', in which multiple people share the same vehicle to travel from a few origins to a few destinations. The BCO algorithm was used to arrange vehicle routing and scheduling of passenger pick-up/drop-offs in the most efficient manner. The algorithm has also been used to enhance task allocation, especially in the field of information technology. Nakrani and Tovey used BCO to optimize the assignment of customer requests to a finite group of internet servers, aiming to maximize profit and reduce server allocation costs.

2.1.3 Particle Swarm Optimization (PSO)

Particle Swarm Optimization is based on the social behavior of groups of animals namely flocks of birds and schools of fish. Simulations of the collective behavior of bird flocks have been

created and studied (Reynolds, 1984; Heppner and Grenander, 1990). Both models operated on the principle that the flocking behavior may be influenced by an individual bird's desire to maintain an optimal distance between itself and other members of the group. E.O. Wilson has speculated that this phenomenon originates from un understanding that an individual can greatly benefit from being in a group, acquiring the experiences and resources of other animals. One example is food foraging, where a bird does not know the food is but can find it by following other birds in group especially the one closest to the food source (Kennedy and Eberhart, 1995). In other words, group behavior is designed to optimize the benefits of an individual and perhaps even an entire group.

PSO algorithms have also been implemented in artificial systems. Kennedy, Eberhart and Shi have used the algorithm to evolve artificial neural networks. Srinivasan et al. applied their work, using PSO to train an artificial neural network used to facilitate early detection of road incidents. Zhao et al. also used the PSO algorithm to optimize the weights in the hidden and outer layers of a RBF artificial neural network which was trained to forecast the traffic flow in two adjacent intersections.

2.2 Bee Colony Optimization

In the parking problem, a driver aims to find a parking space within a certain location in the shortest amount of time possible. Once it does successfully parks, it can tell other cars where it was successful. As such, this scenario can be best related to the Bee Colony Optimization algorithm, which models how honeybees find the best nectar source within a radius of their hive and how they disseminate that information to their fellow bees. The allegory will be made clearer in Chapter 3, where the algorithm will be directly applied and specifically implemented to solve the parking problem. In this section however, the original concept of Bee Colony Algorithm will be further elaborated upon.

2.2.1 Food Foraging Behaviour of Honeybees

The processes used by honeybees to optimize the search and collection for food will be first described. Firstly, a few bees will be assigned to scout. These scouts will leave the hive and search for flower patches, which serve as food sources for the honeybee colony. Once they find one or multiple food source, they will collect its nectar and rate its quality. Once they are finished, they will return to the hive and unload the nectar they have collected. Additionally, the forager will perform the 'waggle dance' on the dance floor in an attempt to recruit other honeybees to collect more nectar from its found food sources. The dance will contain information about the location of the food source and the quality of its nectar. There may be multiple foragers performing the waggle dance concurrently. The other honeybees will randomly select one or multiple dances to watch. It is not able to watch all the dances and thus, does not acquire a global knowledge of all the flower patches. Based on the information it has collected, it will pick a flower patch to collect nectar from (Seely, 1995).

While performing the waggle dance, the forager may experience a delay while recruiting other bees. This information can be used to optimize task partitioning, that is, the number of bees scouting and the number of bees receiving and acting on the advertisements. If the proportion of foragers and receivers is suboptimal, the collection of nectar will also be subpar. For example, if there are more receivers than foragers advertising nectar sources, there will be some receivers that will be idle. The opposite is true when there are more foragers than receivers. To avoid this problem, foragers use the length of the delay to ensure that there are an optimal amount of waggle dancers and receivers. If the delay is short, there are likely more receivers than dancers. Therefore, the forager will perform a waggle dance to recruit more foragers. If the delay is long, there are probably more dancers than receivers and more honeybees will need to take on a receiving role to

optimize nectar collection. In this case, the forager will perform a ‘tremble dance’ to recruit more receivers. (Ratnieks and Anderson, 1999)

But from all the flower patches visited, how does a forager pick which food source to advertise? From all the waggle dances, how does a bystander honeybee decide which flower patch to pursue? It was concluded that one of the factors that the honeybees heavily use is the nectar source’s quality (Seeley, Camazine & Sneyd, 1991). When they presented a bee colony with one high-quality and low-quality nectar source, the honeybees mostly visited and advertised the source with a higher nectar concentration. This is because the foragers perform the waggle dance for a more profitable lot more frequently and for a longer period of time. Therefore, the bystander honeybee is more likely to see an advert for that high-quality parking lot and pursue it. Eventually, the colony as a whole was exploiting the high-quality source. The low-quality source was often abandoned. The bees that still visited that source harvested it at a slower rate and did not perform any waggle dances. Subsequently, no bees were recruited, resulting in the whole colony eventually abandoning the low-quality source.

As the hive gravitates towards high-quality nectar sources, it is important to note how bee foragers evaluate which nectar source available is the most quality and profitable. It is first postulated that the nectar sources are compared by various means, whether it is direct comparison by the honeybees or indirect comparison by the colony’s food storer. However, Seeley, Camazine and Sneyd concluded that no comparisons are made at all. Each forager only knows about its own nectar source and only calculates the source’s absolute quality. When a lower-quality nectar source was placed closer to the hive than a higher-quality nectar source, there were more ‘waggle dances’ for the nectar source for the lower-quality source as more bees have visited the source in a shorter amount of time. In addition, when the food sources were placed at equal distances form the hive,

there were more dances and harvesting at the higher-quality source. However, it is observed that the majority of the foragers have not been to both sources and as such, could not make a comparison. It seems that the higher-quality source was chosen by the probability that the honeybee saw an advert for it on the dance floor. As mentioned before, a profitable source is advertised more heavily and it is more likely that a bystander honeybee will choose to harvest it.

2.2.2 Artificial Algorithms inspired by Honeybee Foraging Behavior

There have been successful attempts to translate the food foraging behavior of bees described in Section 2.2.1 has been translated for use in artificial systems. Sato and Hagirawa designed the Bee System algorithm, which is improved version of another existing bio-inspired algorithm, the Genetic Algorithm (GA). GA is generally good at searching for solutions on a global scale but is poor at local search. This can be solved by integrating the collaborative nature of food foraging bees. Multiple bee populations can search a solution space and communicate with each other to find the local solution.

Teodorovic and Dell'Orco proposed the Bee Colony Optimization (BCO) algorithm, which is designed to solve combinatorial optimization problems. A number of bees are first initialized and are located in the hive. Then, they move locally and create partial solutions on the basis of exploration and past experience. This is called a forward pass. They may also opt to perform a backward pass, in which they return to the hive. There, they exchange information about the quality of the solutions they have created with other bees. Based on the comparisons made, a bee may choose to continue pursuing its partial solution on its own, recruit others to pursue its solution together with it or abandon its solution and create a new solution from scratch. This process is performed until the termination condition is reached.

Karabonga and Basturk wrote the Artificial Bee Colony (ABC) algorithm which focuses on optimizing multivariable functions. There are three types of bees in this algorithm: employed bees who travel to the food source, onlookers who are on the dance floor looking at waggle dances and scouts who randomly search the solution space. At the beginning, a random set of food sources are selected by employed bees. Each employed bee is assigned to investigate the nectar quality of one food source. They return to the hive and share information with the onlookers, who chooses which food source to visit. In the next stage, the employed bees not only go back to the food source visited in the last stage but also visits another food source in the neighborhood of that food source. They will return to the hive to recruit onlookers, who will again choose and possibly develop a preference for a food source based on the information provided. When a food source is exhausted and abandoned by the employed and onlooker bees, the employed bee of that food source because a scout and randomly searches the solution space for a new food source to harvest.

Pham et al. created the basic pseudo-code for the Bee's Algorithm, which basically searches a solution space for an optimal result in the same manner that a bee forager looks for food sources within a specified area. A number of scout bees are initialized to search randomly selected sites in the solution space, whose quality levels are evaluated. The sites are then ranked by their fitness and the fittest few sites are chosen. A number of forager bees are initialized to conduct a local search of the best sites while the rest of the bees are assigned to search the non-best, random locations. The bees continue searching in this manner until a global optimum is found.

2.3 Vehicle-to-Vehicle (V2V) Communication

Vehicle-to Vehicle (V2V) communication is a concept that gained popularity recently due to the emerging development of wireless technology (Yang, Liu, Vaidya & Zhao, 2004). The

technology has been explored in a variety of contexts according to Sichitiu and Kihl. One of the main areas that is studied is the use of V2V in improving traffic safety. Biswas, Tatchikou and Dion developed the Cooperative Collision Avoidance (CCA), which uses V2V technology to prevent car accidents. Another potential application is traffic management. Nadeem, Dashtinezhad, Liao and Iftode created TrafficView, which aims to disseminate and gather traffic information on the road using V2V technology. There even has been research in using V2V communication for leisure. Bucciol, Masala and Martin proposed a solution that will ameliorate inter-vehicular video communication using V2V wireless networks.

However, the use of V2V communication in searching for parking is an area that is very lightly explored. Tasseron, Martens and van der Heijden looked at using V2V communication in helping cars find on-street parking. The car will send messages in two situations: when it occupies or leaves a parking space, which will change the status of the parking space to unavailable or vacant respectively. However, the results presented in the paper shows that there is little difference in search times between V2V and regular cars except for extreme conditions in which the occupancy rate of the street network exceeds 90%.

2.4 Summary of Literature Review

In this chapter, the fundamental concepts behind the HoneyPark algorithm was covered. Firstly, the basic definition of swarm intelligence was presented, which refers to the group of algorithms that the HoneyPark algorithms belongs to. Natural examples of swarm intelligence were reviewed along with how these biological systems have inspired the design of various existing artificial systems. This was done to examine which bio-inspired algorithm was the most relevant to the contect of searching for parking.

After the review, it is found that the parking problem is more analogous to the problem honeybees face when searching for nectar as compared to other natural swarm intelligence problems. As such, the chapter goes into further detail regarding honeybee food foraging behavior. It covered the biological behavior of honey bees and how they search for nectar as both an individual and as a colony. The methods that honey bees use to determine the most profitable flower patch and consequently maximize nectar collection is also discussed. In addition, the review looks at algorithms that have already been created based on this biological system and details how they work.

After discussing the biological aspects of the algorithm, the chapter moved on to discuss the technology that will be used to implement the system: vehicle-to-vehicle (V2V) communication. Although it has been explored in a variety of contexts, it is barely investigated in the context of optimizing parking times. The one example presented here did not have any bio-inspired elements which could potentially improve parking search times. Therefore, it would be useful to see if the combination of bio-inspired concepts and V2V technology results in a heuristic that is able to significantly reduce the amount of time drivers spend in their search for parking.

CHAPTER 3. BEE PARKING ALGORITHM DESIGN

This chapter covers the design and development of the novel bee-inspired parking algorithm. First, an analogy between honeybee food foraging processes and the search for parking will be created and elaborated. This is will be followed by a discussion of the possible ways the algorithm could be implemented and a detailed description of the different variations of the bee parking algorithms investigated in this thesis. After that, two existing parking algorithms will be introduced as they will be used to evaluate whether the HoneyPark algorithm is more effective against current methods of parking.

3.1 Analogy Between Honeybee Food Foraging and Searching for Parking

3.1.1 Macro-level Analogy: Objectives and Environments

Bee-inspired algorithms were chosen as the main source of inspiration for a connected, bio-inspired algorithm because of the similarities in the objectives and environments of a honeybee searching for food and a car looking for parking in a congested, metropolitan area. The foraging behaviors of honeybees was designed to optimize the search for nectar in an extremely variable environment. According to Seeley, the nectar supply in the natural fluctuations on an hourly and daily basis as the quality of nectar varies with micro-climatic conditions, the blooming/withering cycle of flowers and exploitation. As such, the nectar intake of a colony can change by a factor of more than 100 in a single day. This is analogous to a driver looking for a parking spot in a congested city. The amount of parking spaces that are vacant/full vary greatly throughout the day depending on the time of day, rush hour, occurrence of special events and so forth. Like the

honeybee looking for nectar, the driver must find a vacant parking spot in an extremely volatile environment.

Another important characteristic that honeybee food foraging and the search for parking have in common is that both are optimization problems in which the value function is influenced by multiple factors. As mentioned in Section 2.2.1, even though the honeybee colony prefers high-quality nectar and optimizes the quality and amount of nectar collected by exploiting profitable sources, their choice of nectar source is also influenced by other variable factors such as distance and exploitation. In the same way, drivers aim to find a parking space in the shortest amount of time possible. This is too dependent on a variety of factors: the distance between the driver's location and the parking spot, the amount of traffic on the roads, the number of cars that are also looking for parking, the quantity of vacant parking spaces at a parking lot, etc.

Another commonality between honeybees looking for and drivers looking for parking is that the optimal solution depends on the optimal collection of individuals to specific regions. In the honeybee scenario, the honeybees are allocated to flower patches in such a way that the nectar collection of the hive is optimized. In the parking scenario, it is possible to send drivers to parking lot so that they all can find parking in a reasonable amount of time.

3.1.2 Micro-level Analogy

The similarities between honeybee food collection and looking for vacant parking spaces do not only exist from a 'macro perspective'. The set-up at a micro level for both these processes are also parallel. Using a similar table found in Nakrani and Tovey, we find the parking equivalent of the constituents and processes in the bee food foraging process.

Table 1 – Micro-level Analogy between Honeybee food foraging and the Search for Vacant Parking Spaces.

Honeybee Food Foraging	Search for Parking
Individual agent is a honeybee searching for nectar	Individual agent is driver searching for a vacant parking spot
There are limited number of flower patches to forage around the hive	There are limited number of parking lots to search around the driver's desired destination
Convenience of flower patch is dependent on its distance from the hive	Convenience of the parking lot is dependent on its distance from the driver's desired destination
The agents advertise where nectar was found by waggle dancing	The agents advertise where vacant parking spot was found by sending an advert to other drivers
The idle agents are more likely to collect nectar at more heavily advertised nectar sources	The idle agents are more likely to search for vacant parking spaces at more heavily advertised parking lots
Supply of Nectar is extremely variable in matter of hours and days	Availability of vacant parking spots is extremely variable in matter of hours and days
Flower patch quality is determined by the amount and quality of nectar found in it	Parking lot quality is determined by the amount and 'quality' (i.e. usability) of vacant parking spots found in it
Downtime exists when a honeybee is travelling from one nectar source to another	Downtime exists when a driver is travelling from one parking lot to another

Based on Table 1, we can describe a method to search for parking based on the processes honeybees use to find nectar. Table 2 shows the basic process of the parking algorithm based on the stages of honeybee food foraging.

Table 2 – Parking Algorithm based on Honeybee Food Foraging Processes

Step #	Honeybee Food Foraging	Search for Parking
1	A scout honeybee leaves the hive to search for nectar source by looking at any flower patches it can find	A driver arrives at desired destination and begins to search for a vacant parking spot by looking at any parking lots he/she can find
2	Honeybee collects and evaluate amount of quality of nectar at flower patch	Driver searches for vacant parking spot in parking lot
3	If the honeybee finds a suitable nectar source, it returns to the hive and advertises it other honeybees. If not, it continues scouting other flower patches for a suitable nectar source	If the driver finds a vacant parking spot, he/she parks the car and lets other drivers know that he/she found a parking spot at the parking lot by sending an If not, the driver continues scouting other parking lots for a vacancy
4	The other honeybees see the waggle dances on the dance floor and uses them to determine which flower patch they should collect nectar from	The other drivers look at the parking adverts in the server and uses them to determine which parking lot they should search for vacant parking spots

As one can see, the search for nectar can be easily adapted to the search for vacant parking spots as the processes are very similar. The process outlined in Table 2 is visualized in flowchart form in Figure 1. Note that the steps in Table 2 are pointed out in Figure 1 as well.

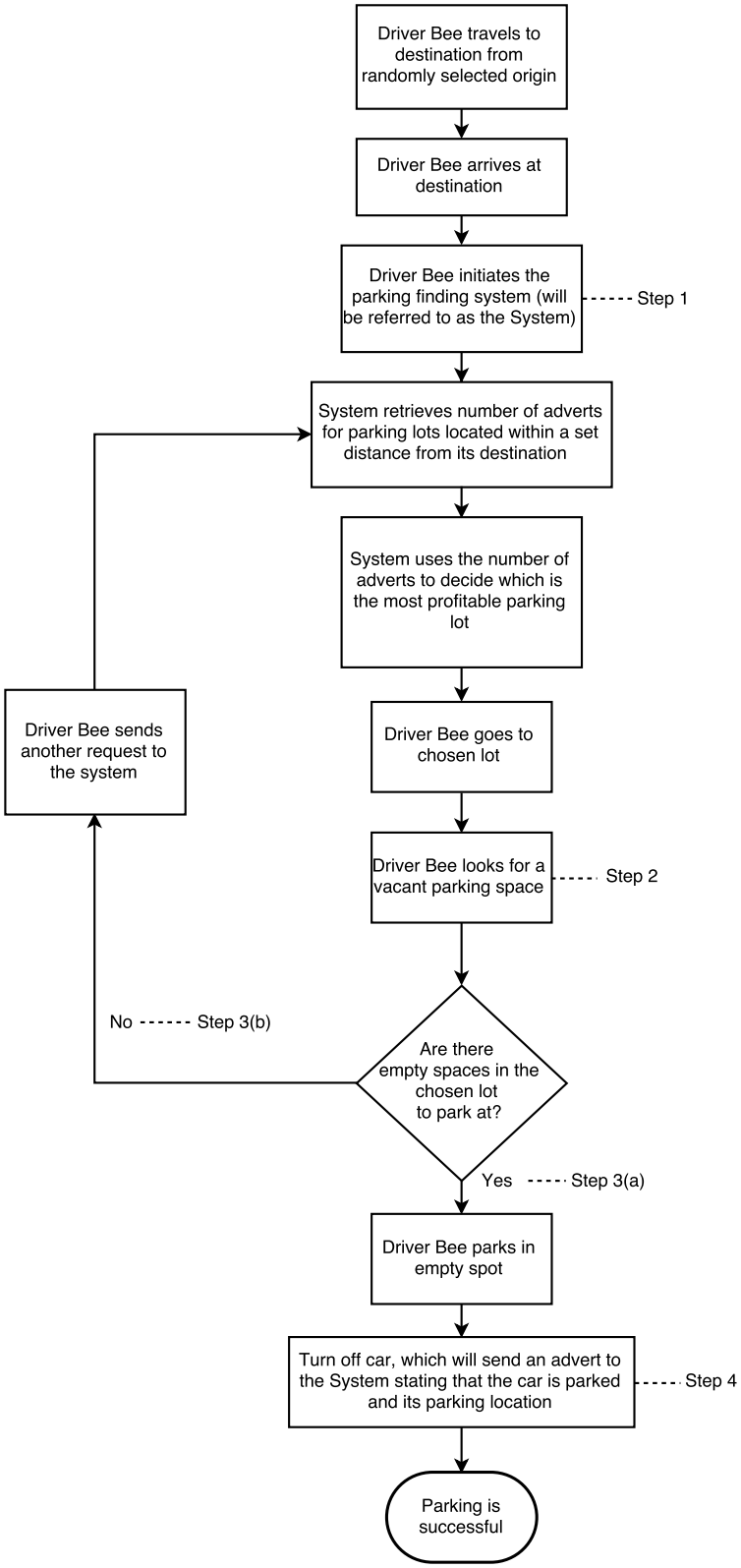


Figure 1 – Flowchart of the Basic HoneyPark Algorithm

3.2 Further Adaptation and Development of the Bee Parking Algorithm

The main structure of algorithm will take the form of the processes outlined in Figure 2. However, it is still in need of further development as the Bee's Algorithm is not always perfectly compatible will all aspects of the parking search process. The main parts of the analogy that needs to be further adapted is the scouting algorithm, how would a car or honeybee choose which parking lot or flower patch to explore respectively and the generation and processing of 'adverts'.

3.2.1 Scouting Algorithms

One detail in Table 2 that could be elaborated on is the method for scouting. In the biological environment, the hive may not know where the profitable nectar sources are located. Therefore, honeybees must scout in order to find them. This is often done by picking a random flower patch and evaluating the nectar there. In the same way, there may be times when none of the drivers know or have sufficient information to know where the profitable lots are (i.e. when the HoneyPark algorithm is first implemented and no adverts were created yet). Therefore, scouting is needed to find these profitable lots. This scouting behavior cannot be directly translated to the search of parking. In the case of the honeybee, all the honeybees are working in such a way that the colony receives the maximum amount of nectar as a whole. In the search of parking, drivers are more self-interested. They are not affected by, and therefore do not care, whether other drivers have a shorter parking time. What matters to the driver is that he/she is able to find parking in the shortest amount of time possible. Therefore, it is unreasonable to expect some drivers to scout random locations and potentially increase their parking time so that other drivers can optimize their parking search. As such, the algorithm only gets the driver to scout whenever the driver cannot find a vacant spot at the parking lot recommended by the algorithm. The driver must resort

to another method that chooses a different parking lot that is potentially more profitable. The failure of the algorithm also indicates that information in the advertisement system must be updated. Therefore, it is in the best interest of the individual driver and the drivers using the algorithm for the driver scout and find potential profitable lots.

While scouting, the locations honeybees choose to forage are also chosen randomly. One can artificially implement this behavior as well by setting the Random algorithm as the scouting algorithm. But unlike the honeybee who may not necessarily know the location of all the flower patches in an area, the parking system in this simulation knows the location of all the parking lots. Therefore, it is possible to implement other scouting algorithm such as Greedy, in which the driver will scout the parking closest to him/her.

3.2.2 Bystander Driver Decision-Making

3.2.2.1 Without Real-Time Traffic Conditions

As mentioned in Chapter 2, the honeybees do not have a global knowledge of all the flower patches explored by the scouts. This is because the waggle dance is only performed for a period of time and therefore, cannot be viewed for an infinite amount of time. In addition, bystander bees can only see part of the dance floor and therefore cannot see all the dances performed. However, this is not a significant impediment. A profitable nectar source will be advertised by a larger number of waggle dances. Therefore, it is more likely that a bystander honeybee will see an advert for a profitable source and harvest it. It is the probability that honeybee sees an advert for a particular lot that is a major factor in whether the honey visits the lot. In the parking analogy, the central cloud server will receive all advertisements. Therefore, the drivers can have a global knowledge of all the adverts sent to the system. As such, it is missing the probabilistic quality of

how honeybees choose which areas to explore. To mimic the bystander bee's judgment, a 'russian roulette' system is employed. The probability that a driver finds a vacant parking spot in a specific lot, $P_{success}$(lot) can be calculated using the number of adverts generated for that lot, N_{ads}(lot), and the total amount of adverts generated for all the possible lots that the car could park at, $N_{total\ ads}$, as seen in Equation 1.

$$P_{success}(lot) = \frac{N_{ads}(lot)}{N_{total\ ads}} \quad (1)$$

This equation fits that bee food-foraging model that a driver is more likely to successfully find a vacant parking spot at a parking lot that is more advertised. The probability is calculated for all the possible lots that the driver could visit and a cumulative probability will be made. A random number between zero and one will be generated. It will fall within the cumulative probability range for a particular parking lot, which the car will travel to and search for a vacancy.

3.2.2.2 With Real-Time Traffic Conditions

Preliminary simulations show that Equation 1 above in Section 3.2.2.1 does not work in simulations that have real-time traffic conditions. This is because it only considers the profitability of the parking lot and not the time it takes to get to the lot. In traffic conditions, one must also identify which lot is the most easily accessible (i.e. the route to the parking lot does not have a large amount of traffic). When Equation 1 was implemented in a traffic simulation, there were situations in which the HoneyPark car had a long parking time not because the lot chosen was unprofitable but because the amount of time it took to arrive at the lot was unacceptably long due to traffic. This is a notable difference from the honeybee food foraging process, where the amount

of time to get to the flower patch is considered in adverts. When a profitable flower patch is located closer to the hive, the adverts for it are more frequent than a profitable flower patch that is further away because it takes a shorter time for the bee to travel to and from the patch. Therefore, more waggle dances can be initiated for that flower patch because the honeybees can make more trips to and advertise that patch in a shorter amount of time (Seeley 1991). This is not the case in the HoneyPark algorithm. Due to the nature of wireless technology, the HoneyPark system can receive adverts almost instantly regardless of how long it takes for the driver to travel to the lot.

Therefore, one will need to explicitly program the HoneyPark algorithm to consider the amount of time it takes to travel to the parking lot. Verroios, Efstathiou and Delis formulated an equation to evaluate the profitability of parking lot by using the concept of 'lot cost', which their formulation defines as the most probable amount of time that a driver takes to travel from his/her current position to his/her final destination in person if he/she chooses a parking lot to park at. It is the sum of the time taken to travel from the driver's current position to the parking lot, the amount of time needed to find a parking space within that parking lot and the time it takes for the driver to walk from the parking lot to his/her final destination.

The above method was adapted to identify the lot that was both the most profitable and the most the easily accessible. Based on this, another formulation was created to calculate the lot cost in Equation 2. It is essentially the sum of the time it takes to travel to the parking lot from the driver's current location and the probabilistic amount of time needed to find a vacant parking space within the lot.

$$lot\ cost = T_{lot\ travel} + \left(1 - P_{success}(lot)\right) * T_{avg\ search} \quad (2)$$

where

$T_{lot\ travel}$ = amount of time it takes to travel to the parking lot

$P_{success}(lot)$ = the success rate of finding an available parking space in the lot based on adverts sent by other HoneyPark drivers (defined in Equation 1)

$T_{avg\ search}$ = the average time it took for the last few cars (five last cars in the case of the simulation) to find an available parking space in the lot.

Like $P_{success}$(lot) in the previous section, the lot cost is then arranged in a 'russian roulette system' such that parking lots with the higher cost are less likely to be chosen and parking lots with lower cost are more likely to be picked by the algorithm.

3.2.3 Advertisement Management and Processing System

When a honeybee finds a profitable source, it creates a positive feedback to harvest that source by advertising it to other honeybees in the colony. If the bystander bees find that nectar source is worth harvest, they themselves will return to the hive and advertise it, creating a chain reaction of waggle dances telling the colony to harvest this one source. This is easily implemented in the parking analogy. When a car successfully finds a parking space, it sends an advert to the central cloud network advertising the parking lot. This creates positive feedback for the profitable parking lot. Unlike honeybee food foraging where the nectar profitability is not regained by honeybee activity, a parking lot can acquire vacant parking spots and gain profitability if drivers start leaving it. As such, drivers leaving a parking lot can also send a 'leave' advert, indicating that spots are now available at a previously full lot. This can also create positive feedback and encourage other drivers to explore parking lots that are regaining their vacancy.

When a source is no longer profitable, honeybees stop advertising it. This creates a negative feedback for the nectar-poor source and bees will eventually stop expending unnecessary energy harvesting it. However, it is impractical to directly apply this analogy to the parking problem. Unlike waggle dances which 'disappear' once the dancing bee chooses to stop advertising a flower patch, the adverts in the parking system can be stored for an infinite amount of time for future drivers to look at as it is a computer system with memory. In addition, it takes time for the bees to discover that the flower patch is not profitable and slowly reduce the number of adverts for that patch. However, it is quite slow and there may be a chance that bystander honeybees will still see an advert for the unprofitable flower patch and visit it only to find that it is no longer profitable. In the same way, drivers may still an advert for a parking lot whose profitability is declining and explore it, only to find that there are no parking spaces. Therefore, one must find an artificial way to mimic this behavior in a quicker manner. In the HoneyPark algorithm, when a driver finds that a parking lot is full, it resets the number of adverts listed for that particular parking lot immediately. As such, advistement for that lot is stopped almost instantaneously and drivers are less likely to search that lot.

The HoneyPark advertisement system also differs from 'waggle dances' in the sense that it is able to give the driver a perfect knowledge of adverts available. Honeybees are limited by the fact that they can only see a portion of the dance floor so they can't see all the adverts. In addition, they can only watch the waggle dances as long as the dancer is performing it. Once the dancer stops, the dance cannot be viewed anymore. This is not the case for the HoneyPark advertisement system, which is able to display all incoming adverts to any one driver and store them for any given amount of time. As such, one can use the fact that drivers have a perfect knowledge of incoming adverts to improve the negative feedback mechanism in the HoneyPark algorithm. In

animal communication, some species of animals may send an alarm signal that warns other individuals in their communities of impending danger. In response, the receivers of the alarm call will flee and avoid the location of danger. Likewise, a driver can send out an 'alarm signal' (i.e. a Negative Advert) when he or she finds that the parking lot he or she is searching is full, telling other drivers that the lot is full and unprofitable. This is perfect for a system in which drivers can see every advert as they can all see the Negative Advert and react accordingly. In this paper, this negative feedback mechanism is implemented in two different ways:

1. Regular Negative Feedback Mechanism: When a driver sends a Negative advert for a particular parking lot, that particular lot will not be considered by other drivers that are going to look for parking in that area. It will be considered as unprofitable for as long as the Negative advert is active. The advert is deactivated when the parking receives a Leave Ads, in which a driver is leaving a parking lot and indicating that the parking lot is no longer full.
2. Instantaneous Negative Feedback Mechanism: When a driver sends a Negative advert for a parking lot, the advert will not only be considered by drivers that are going to look for parking but also by drivers that on their way to their recommended lot. If a driver is on his/her way to a lot that has received a Negative advert, the algorithm will let him/her know immediately that the chosen lot is full and will redirect him/her to another parking lot. The algorithm is designed such that the driver will only be redirected to a lot nearby to his/her current location (i.e. within a certain radius of his/her present position).

3.3 Variations of Bee Parking Algorithms

Based on the principles outlined in the past two sections, eight variations of the Bee Parking Algorithm were developed. They were developed by mixing and observing how the following three different qualities affected parking search times:

1. The use of Parked Ads and Leave Ads
2. The inclusion of an explicit negative feedback mechanism
3. The algorithm used to scout for other parking lots in the case the HoneyPark algorithm is unable to direct the driver to a parking in which the driver can find a vacant parking space.

Table 3 is a summary table of all the HoneyPark variations presented in this paper.

Table 3 – Summary Table of HoneyPark Variations Grouped by Adverts and Negative Feedback

	Parked-Advert Algorithms	**Parked-Leave-Advert Algorithms**
No Negative Feedback	Parked-Advert, Random-Scouting Algorithm	Parked-Leave-Advert, Random-Scouting Algorithm
	Parked-Advert, Greedy-Scouting Algorithm	Parked-Leave-Advert, Greedy-Scouting Algorithm
Regular Negative Feedback	Parked-Advert, Random-Scouting Algorithm with Regular Negative Feedback	Parked-Leave-Advert, Random-Scouting Algorithm with Regular Negative Feedback
	Parked-Advert, Greedy-Scouting Algorithm with Regular Negative Feedback	Parked-Leave-Advert, Greedy-Scouting Algorithm with Regular Negative Feedback
Instantaneous Negative Feedback	Parked-Advert, Random-Scouting Algorithm with Instantaneous Negative Feedback	Parked-Leave-Advert, Random-Scouting Algorithm with Instantaneous Negative Feedback

3.3.1 Parked-Advert Algorithms

3.3.1.1 Parked-Advert, Random-Scouting Algorithm (P-R)

The drivers using this algorithm only consider Parked Ads when choosing a parking lot to search. If it fails to find parking in the recommended parking lot and the algorithm still insists that it should try the same lot, it will resort to the Random Algorithm to search for another lot. The whole process is visualized in a flowchart as seen in Figure 2.

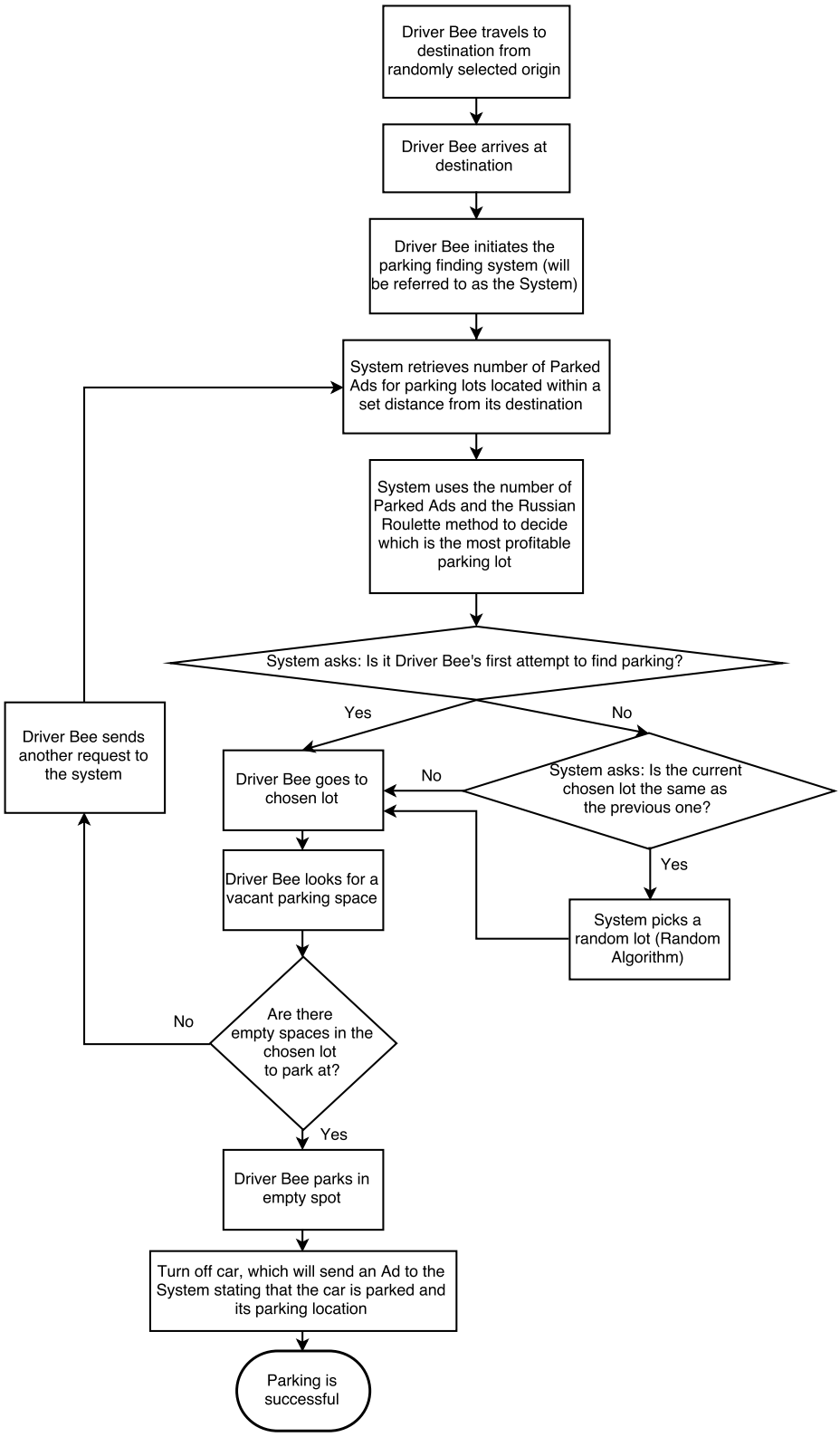


Figure 2 – Flowchart of the Parked-Advert, Random-Scouting Algorithm

3.3.1.2 Parked-Advert, Greedy-Scouting Algorithm (P-G)

This algorithm functions similarly to the Parked-Advert, Random-Scouting Algorithm. The only difference between the two algorithms is the scouting algorithm. This algorithm uses the Greedy algorithm instead of the Random Algorithm if the Bee Parking Algorithm fails to direct it to a successful parking lot. In other words, the driver will scout the parking lot closest to its location rather than choose a random lot to visit. The algorithm process is visualized in Figure 3.

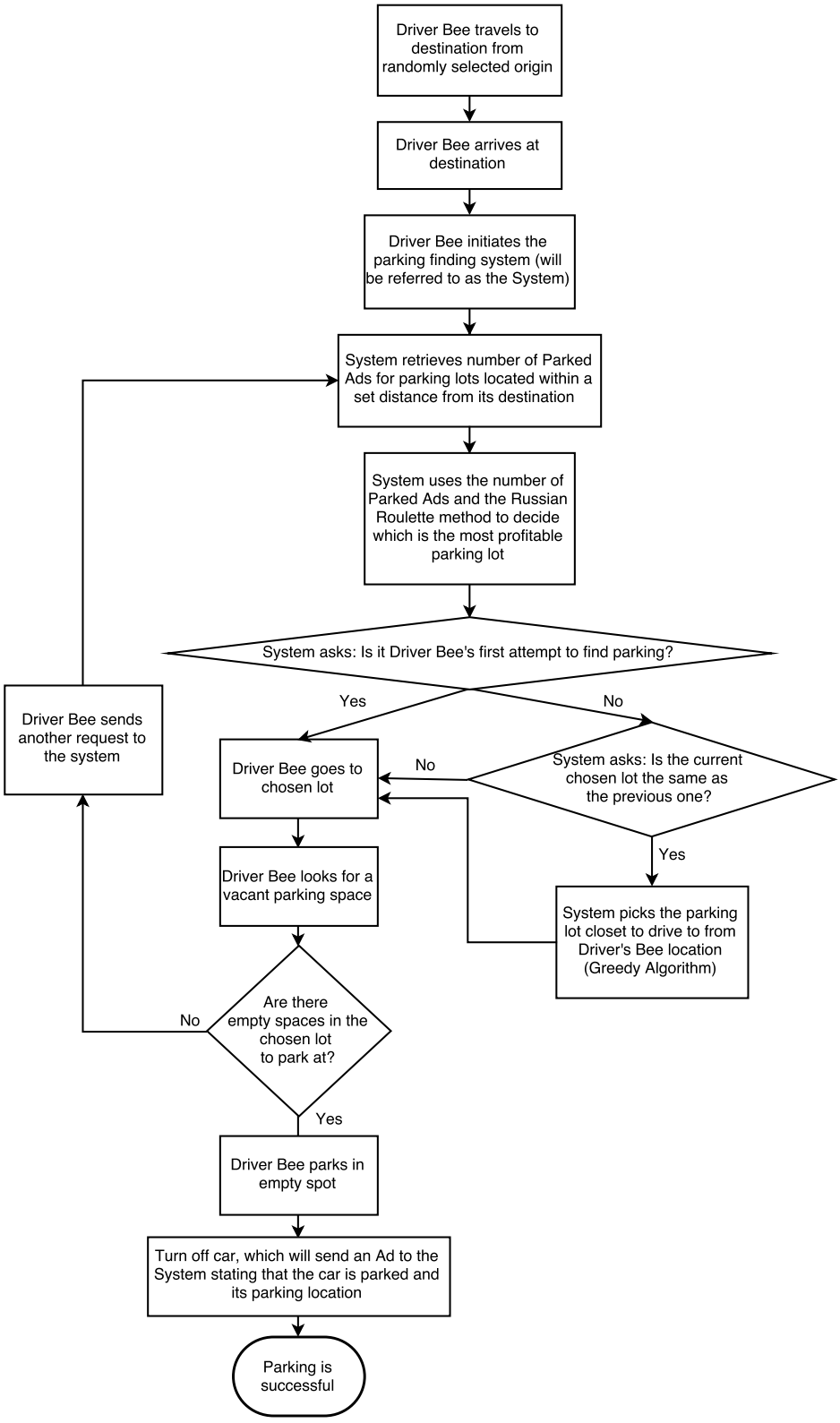


Figure 3 – Flowchart of the Parked-Advert, Greedy-Scouting Algorithm

3.3.1.3 Parked-Advert, Random-Scouting Algorithm with Regular Negative Feedback (P-R (REG-NEG))

This algorithm only looks at Parked Ads when determining the profitability of a parking lot. When the parking algorithm is unable to direct the driver to a profitable lot, the Random algorithm is used as the scouting algorithm.

An extra feature of this algorithm is that it uses regular negative feedback. With this algorithm, the driver not only sends out a Parked Ads when he or she find parking but also sends out a Negative Ad when he/she discovers a parking lot that is full, telling other drivers that the lot is unprofitable. Any lot with at one Negative Ad will be not be considered by the algorithm when subsequent drivers initiate their search for parking. The Negative Ad remains in effect until a Leave Ad is sent out by a car leaving the fully occupied lot, creating a vacancy and inferring that the lot is no longer full. The integration of Negative Ads into the Bee's algorithm can be seen in Figure 4.

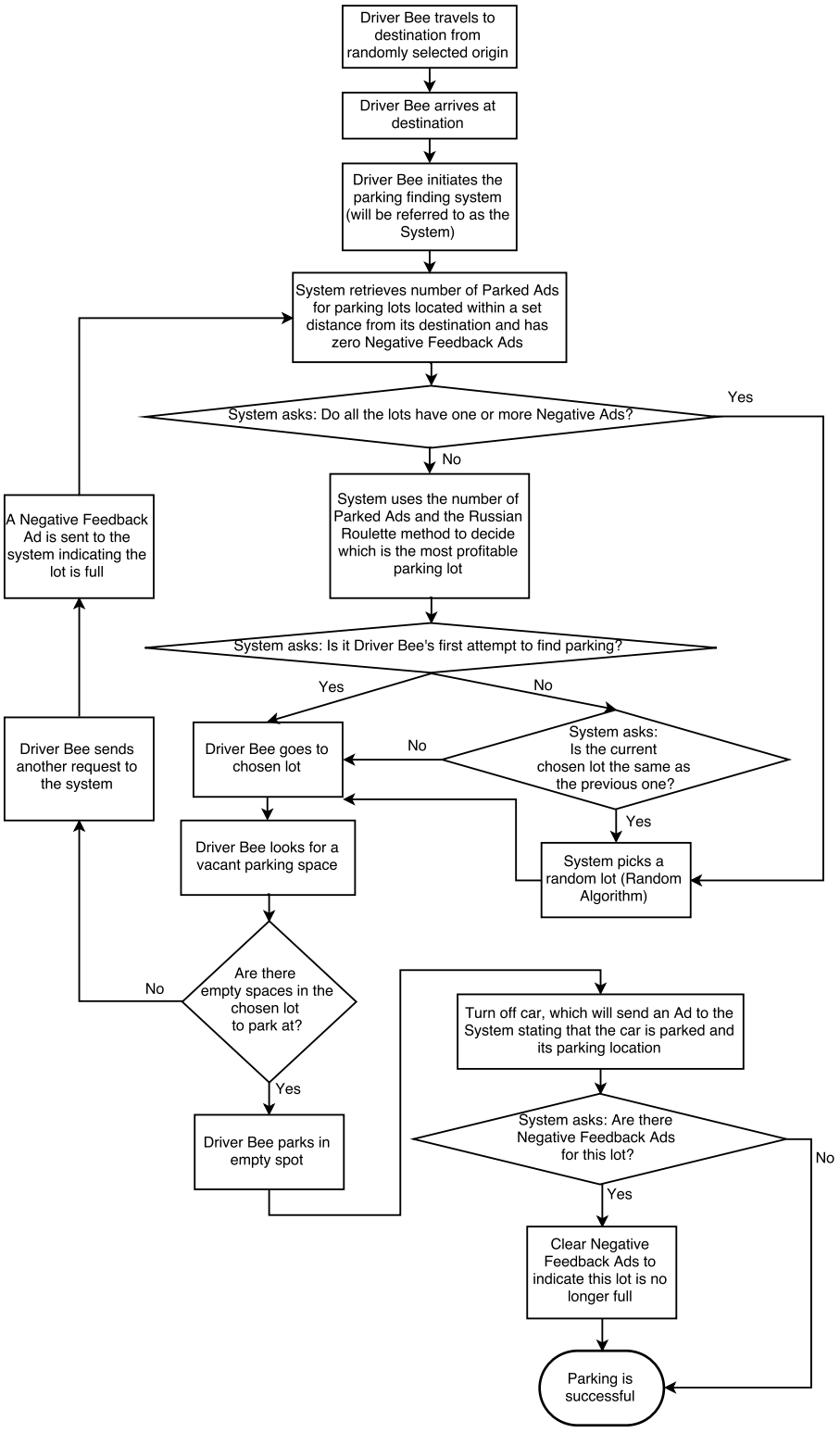


Figure 4 – Flowchart of the Parked-Advert, Random-Scouting Algorithm with Regular Negative Feedback

3.3.1.4 Parked-Advert, Greedy-Scouting Algorithm with Regular Negative Feedback (P-G (REG-NEG))

This algorithm is identical to the one described above in Section 3.3.5.1. The only difference is that the Greedy Algorithm is used as a scouting algorithm instead of the Random Algorithm. This modification is shown in Figure 5.

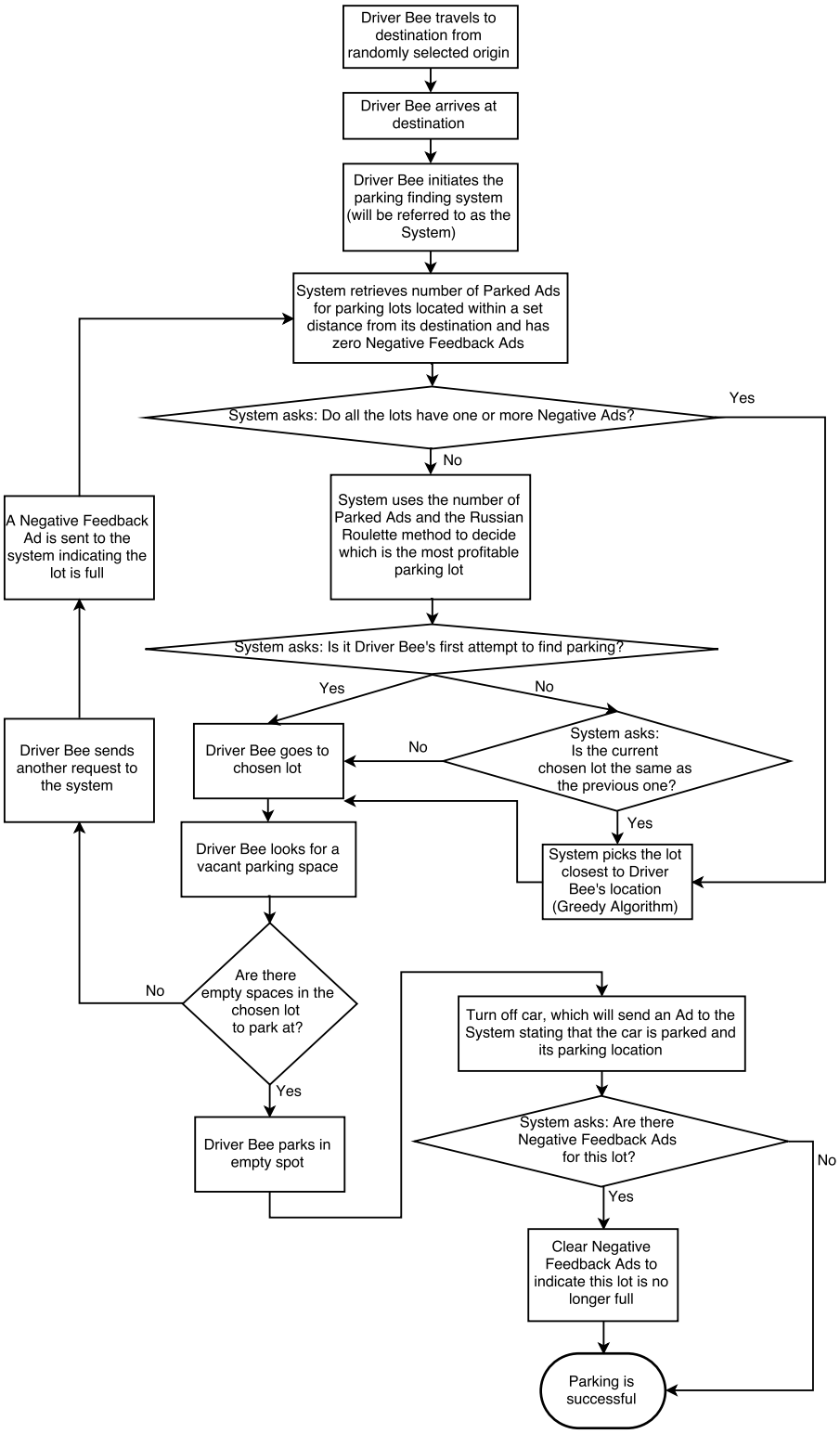


Figure 5 – Flowchart of the Parked-Advert, Greedy-Scouting Algorithm with Regular Negative Feedback

3.3.1.5 Parked-Advert, Random-Scouting Algorithm with Instantaneous Negative Feedback (P-R (INS-NEG))

This algorithm is identical to the Parked-Advertising, Random-Scouting Algorithm with Regular Negative Feedback. The only difference is that the driver is instantaneously redirected to another parking lot if his/her target parking lot is found to be full (i.e. Negative Ad was initialized for the target parking lot) while he/she is on the way to the target lot. The algorithm will then use the Bee Parking algorithm redirect the driver to another parking lot. It will be nearby to the driver's current location as explained in Section 3.2.2.2, redirecting the driver to another lot results in a longer parking time as time is wasted travelling to the parking lot. A flowchart for this algorithm is shown in Figure 6.

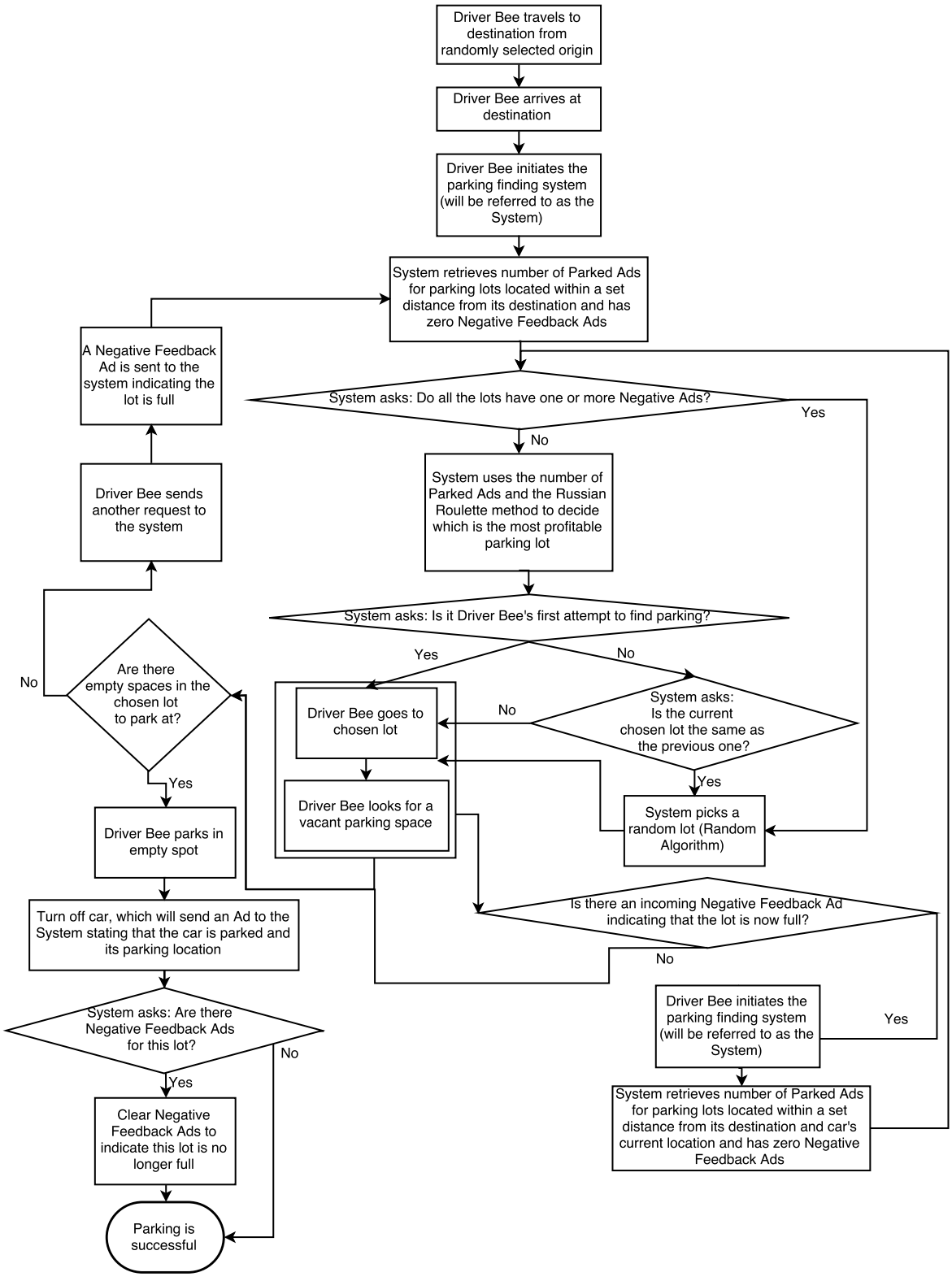


Figure 6 – Flowchart of the Parked-Advert, Random-Scouting Algorithm with Instantaneous Negative Feedback

3.3.2 Parked-Leave-Advert Algorithms

3.3.2.1 Parked-Leave-Advert, Random-Scouting Algorithm (PL-R)

The drivers using this algorithm considers both Parked Ads and Leave Ads when choosing a parking lot to search. As the Parked Ads and Leave Ads are both indicators that a lot is profitable, they are simply added together. Thus, the 'Russian roulette' system will not only consider Parked Ads, but the sum of Parked Ads and Leave Ads in a lot. Its scouting algorithm is the Random Algorithm. The whole process is visualized in a flowchart as seen in Figure 7.

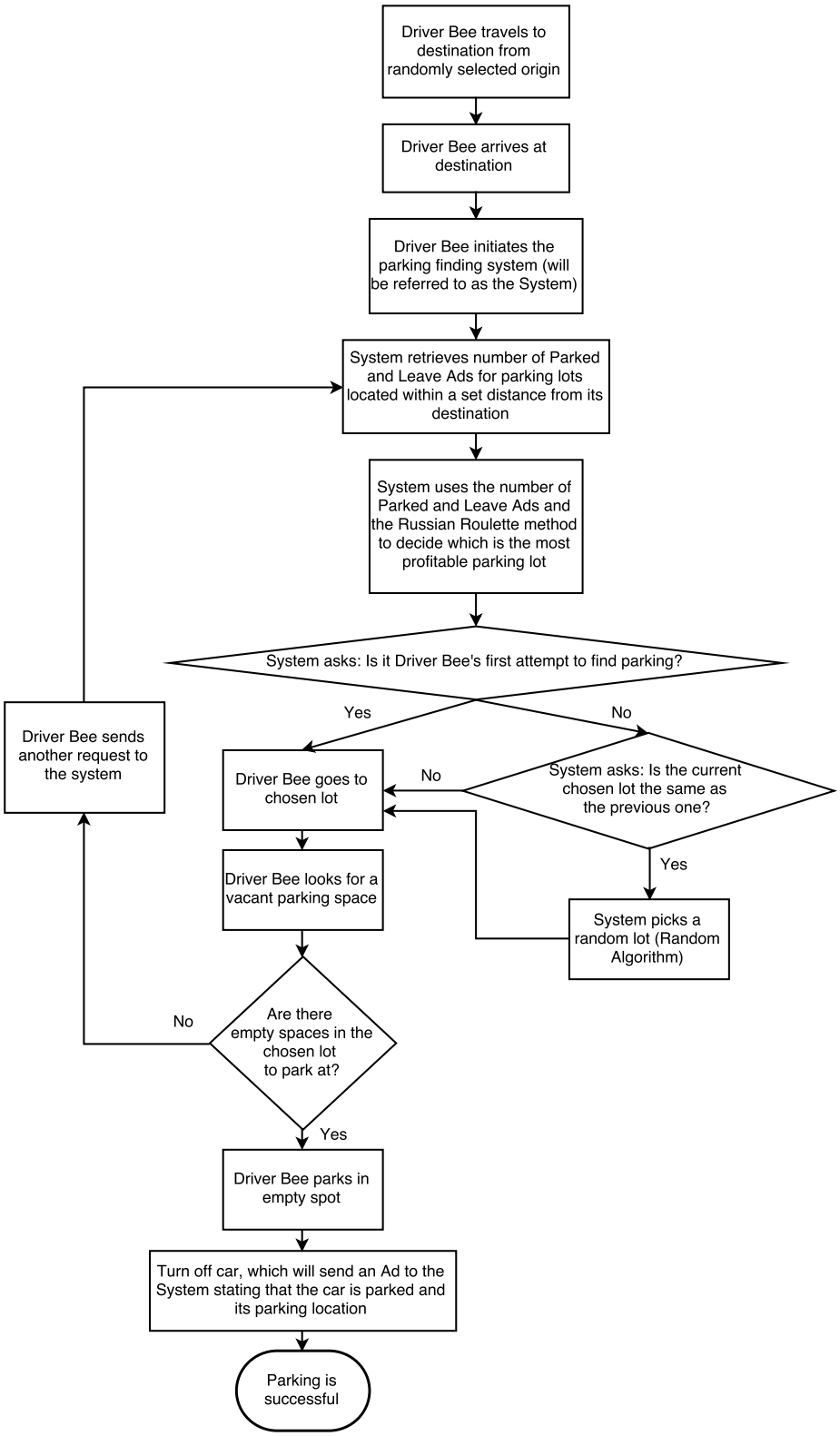


Figure 7 – Flowchart of the Parked-Leave-Advert, Random-Scouting Algorithm

3.3.2.2 Parked-Leave-Advert, Greedy-Scouting Algorithm (PL-G)

Figure 8 shows the flowchart of this algorithm, which is the same as the Parked-Leave-Advert, Random-Scouting Algorithm with the exception that the scouting algorithm is the Greedy Algorithm.

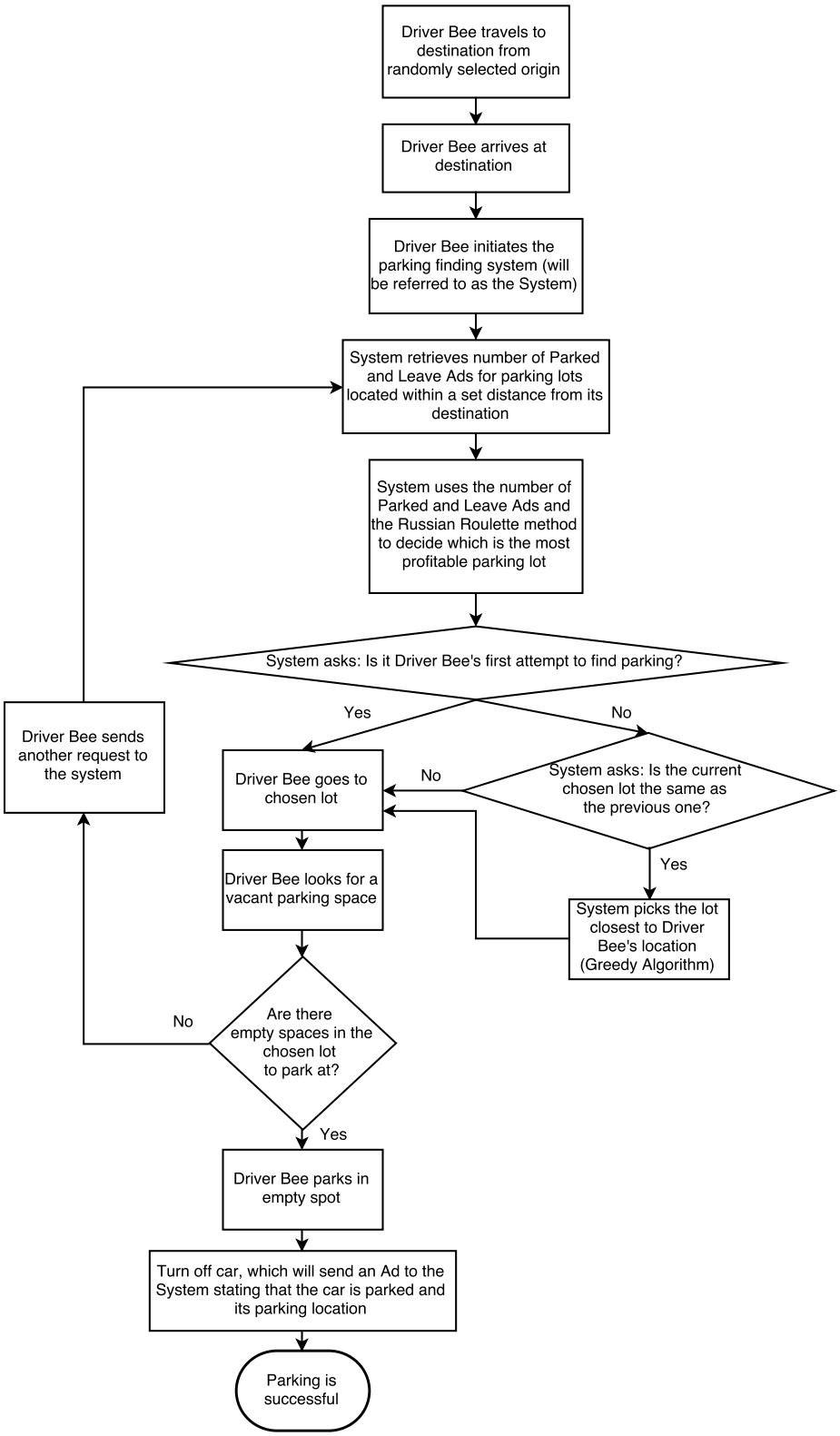


Figure 8 – Flowchart of the Parked-Leave-Advert, Greedy-Scouting Algorithm

3.3.2.3 Parked-Leave-Advertising, Random-Scouting Algorithm with Regular Negative Feedback (PL-R (REG-NEG))

This algorithm observes Parked, Leave and Negative Ads when calculating the profitability of parking lots. It utilizes the Random Algorithm as the scouting algorithm. The algorithm process is visualized in Figure 9.

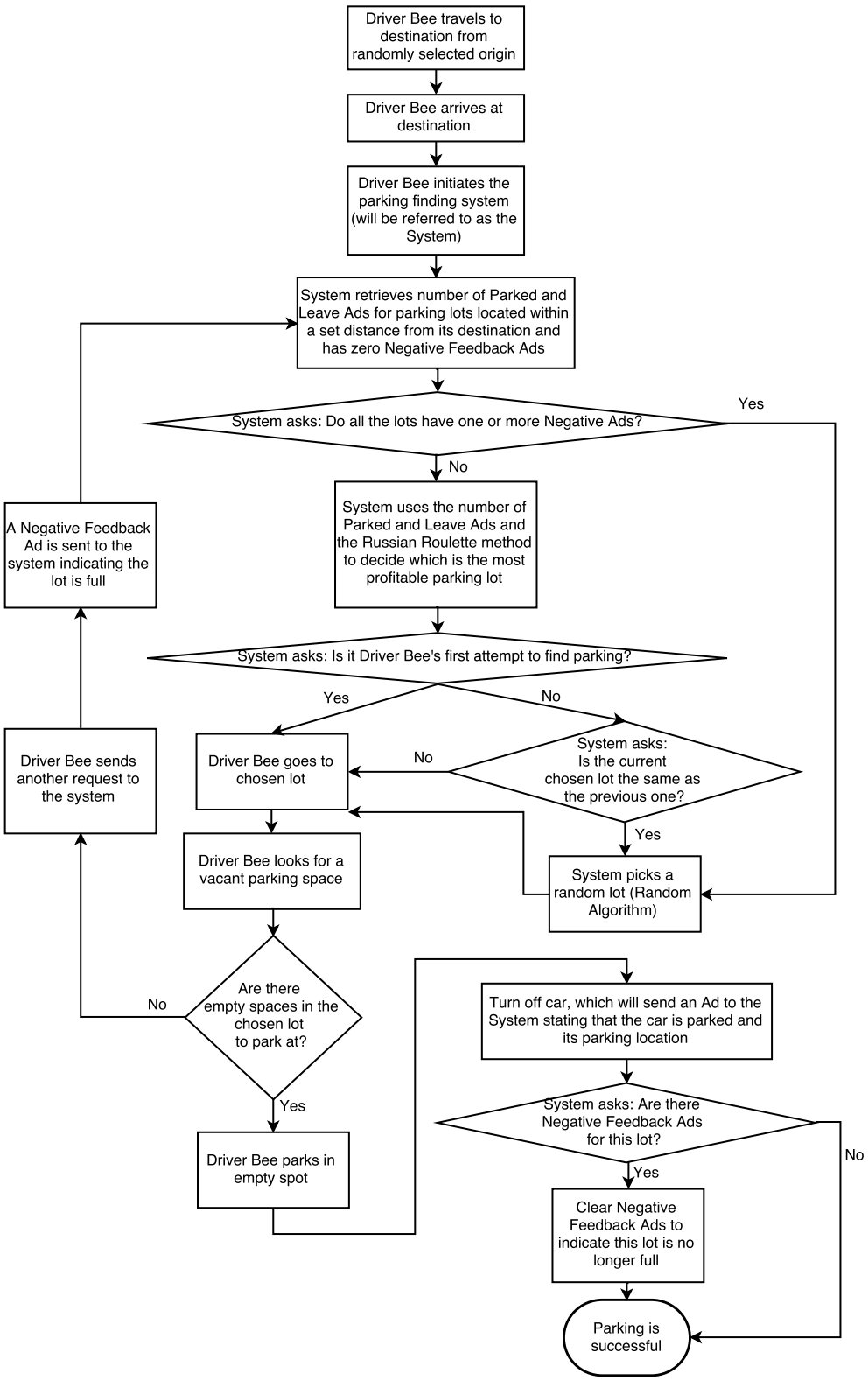


Figure 9 – Flowchart of the Parked-Leave-Advert, Random-Scouting Algorithm with Regular Negative Feedback

3.3.2.4 Parked-Leave-Advertising, Greedy-Scouting Algorithm with Regular Negative Feedback (PL-G (REG-NEG))

The algorithm is extremely identical to the Parked-Leave-Advertising, Random-Scouting Algorithm with Explicit Negative Feedback with the exception that it uses the Greedy Algorithm as its scouting method. This change is reflected in Figure 10.

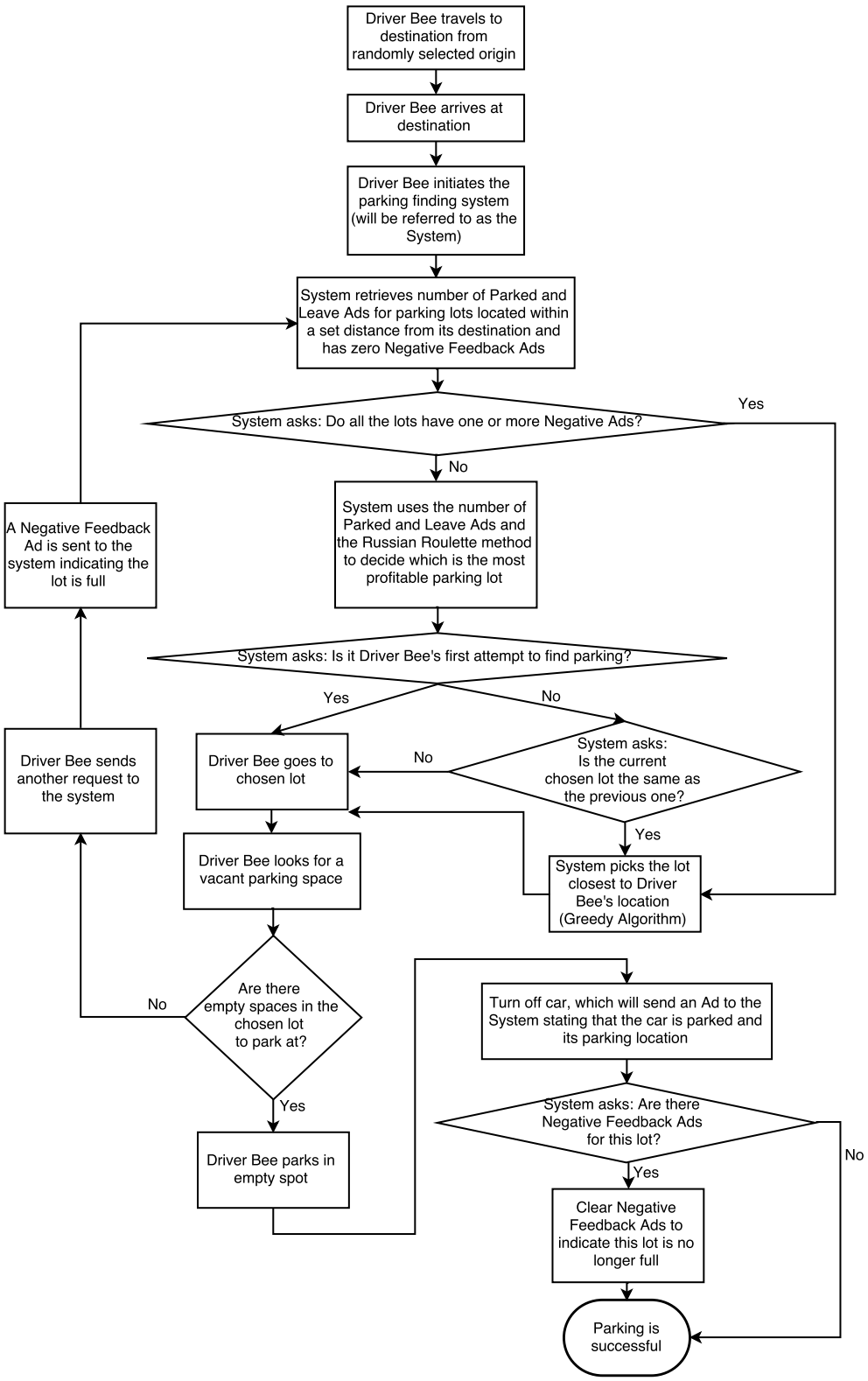


Figure 10 – Flowchart of the Parked-Leave-Advert, Random-Scouting Algorithm with Regular Negative Feedback

3.3.2.5 Parked-Leave-Advertising, Greedy-Scouting Algorithm with Instantaneous Negative Feedback (PL-R (INS-NEG))

With this algorithm, the driver is instantaneously informed and recommended a different, nearby parking lot to search the moment the system is informed that his/her target parking lot is full. Figure 11 shows a flowchart for this variation.

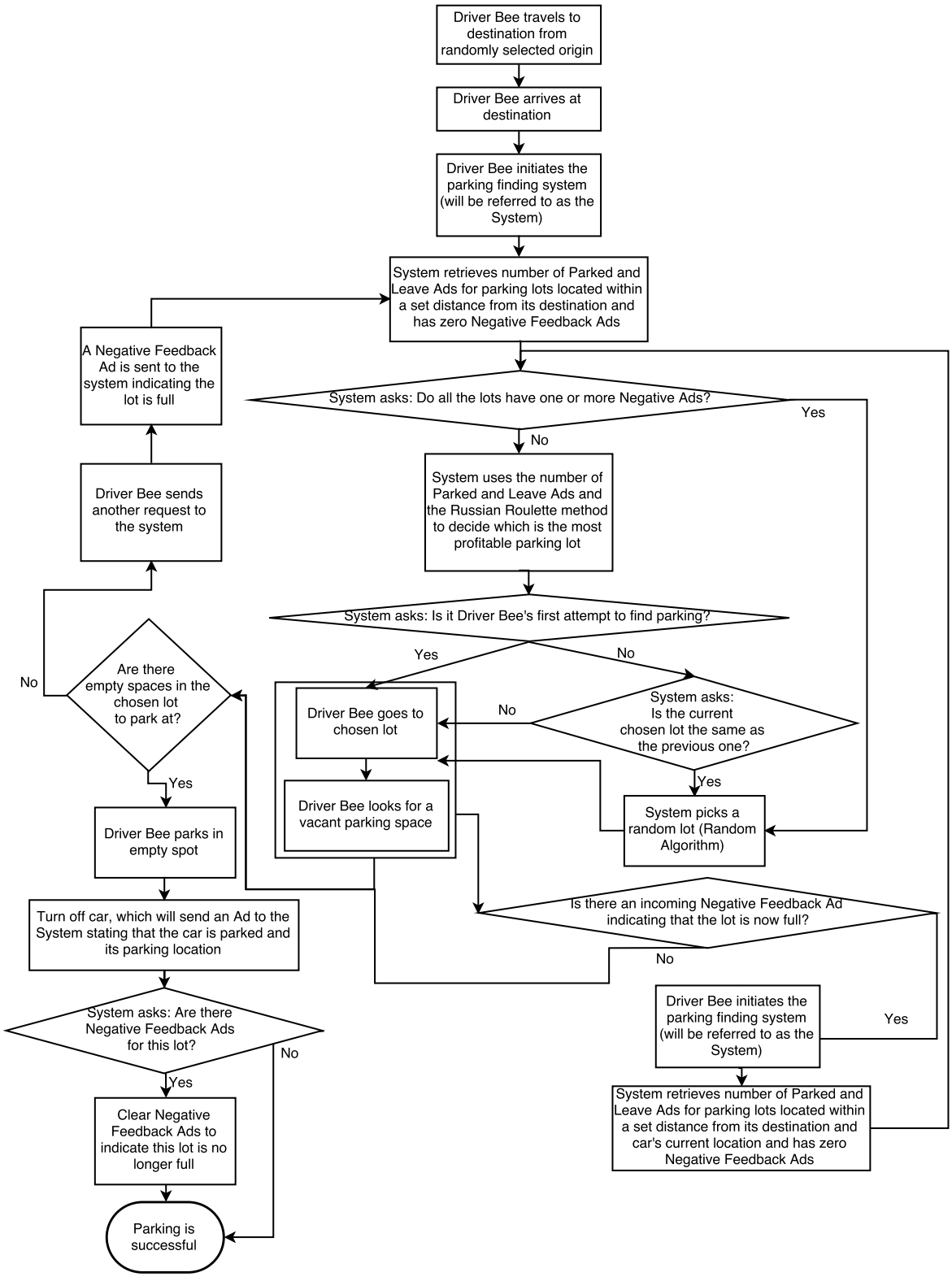


Figure 11 – Flowchart of the Parked-Leave-Advert, Random-Scouting Algorithm with Instantaneous Negative Feedback

3.4 Algorithms Used to Evaluate the HoneyPark Algorithms

It is necessary to benchmark the HoneyPark algorithm against existing parking algorithms to evaluate whether the HoneyPark algorithm more efficient than current methods used to find parking. The two algorithms that the HoneyPark algorithm is evaluated against will be the Ranodm algorithm and the Greedy algorithm.

3.4.1 Random Algorithm

The Random algorithm causes the car to pick a random lot and search for an available parking lot there. The process is visualized in Figure 12.

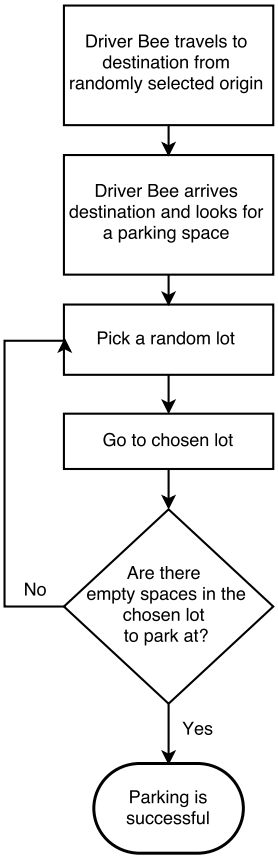


Figure 12 – Flowchart of the Random Algorithm

3.4.2 Greedy Algorithm

Drivers using the Greedy algorithm will attempt to search the parking lot closest to his or her destination causes the car to pick a random lot and search for an available parking lot there. If the lot he or she is searching is full, he or she will then proceed to search the next closest lot. The flowchart for this algorithm in Figure 13.

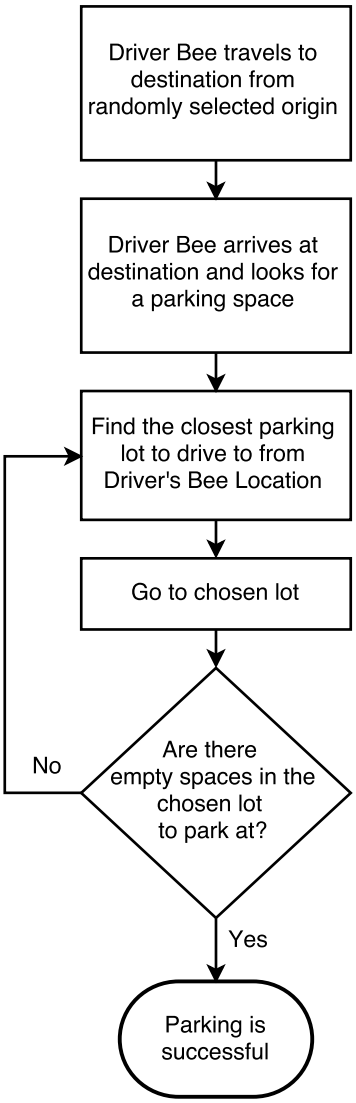


Figure 13 – Flowchart of the Greedy Algorithm

3.5 Summary of Bee Parking Algorithm Design

This chapter covers the design and structure of the HoneyPark algorithm. Firstly, an comparision between honeybee food foraging and the search for parking was created on a macro and micro level. This analogy is important as it will serve as the foundation of the HoneyPark algorithm. However, certain key aspects of honeybee behavior cannot be directly translated to a parking application such as the advertising system and the scouting algorithms. Therefore, they were modified to ensure that the Bee's algorithm can be implemented in the context of the search for parking.

All this information was then consolidated to create eight variations of the HoneyPark algorithms, which will be tested and examined over the course of this thesis. First, algorithms with no explicit negative feedback were created. Four variations could be created by changing the scouting algorithm and whether the algorithm just looked at Parked Adverts or both Parked and Leave Adverts. This is demonstrated in Figure 14.

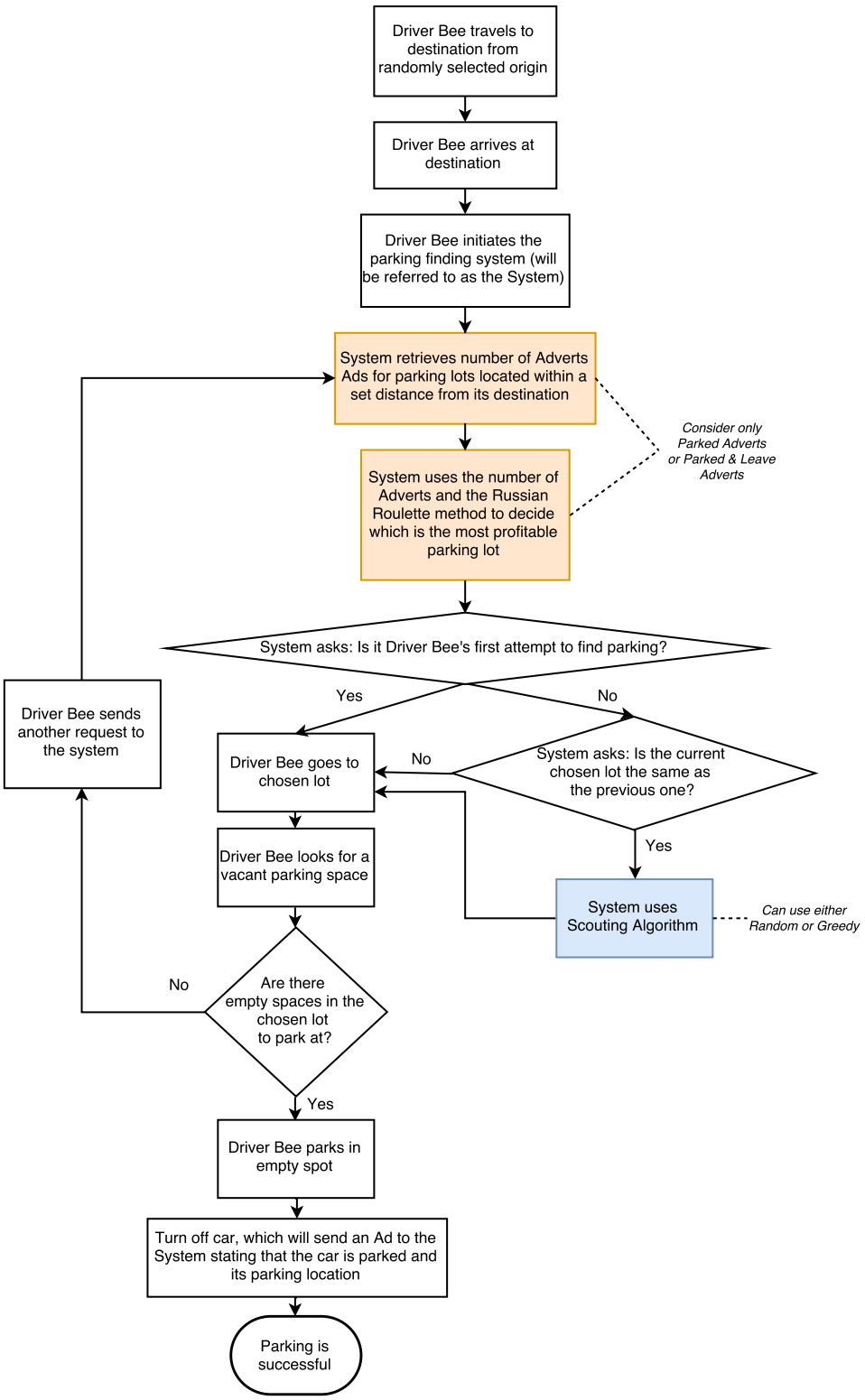


Figure 14 – Flowchart of HoneyPark Algorithms with No Explicit Negative Feedback Mechanism

Another four HoneyPark variations were created by implementing the regular negative feedback mechanism, along with changing the scouting algorithm and which adverts the algorithm considered. This is demonstrated in Figure 15.

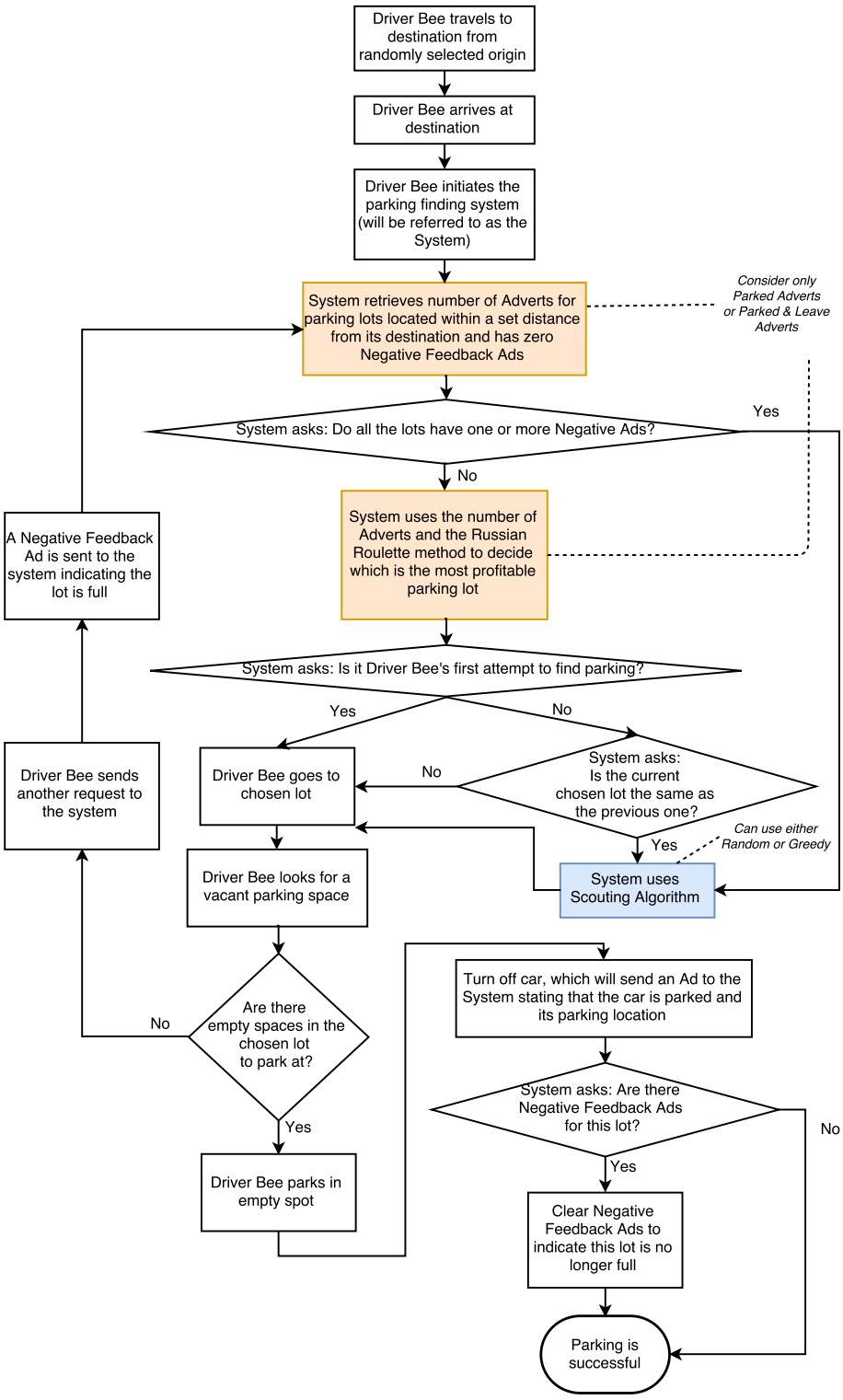


Figure 15 – Flowchart of HoneyPark Algorithms with Regular Negative Feedback Mechanism

The final two algorithms were created by implementing the instantaneous negative feedback mechanism and changing whether the algorithm looked at Parked adverts or Parked and Leave adverts, as shown in Figure 16.

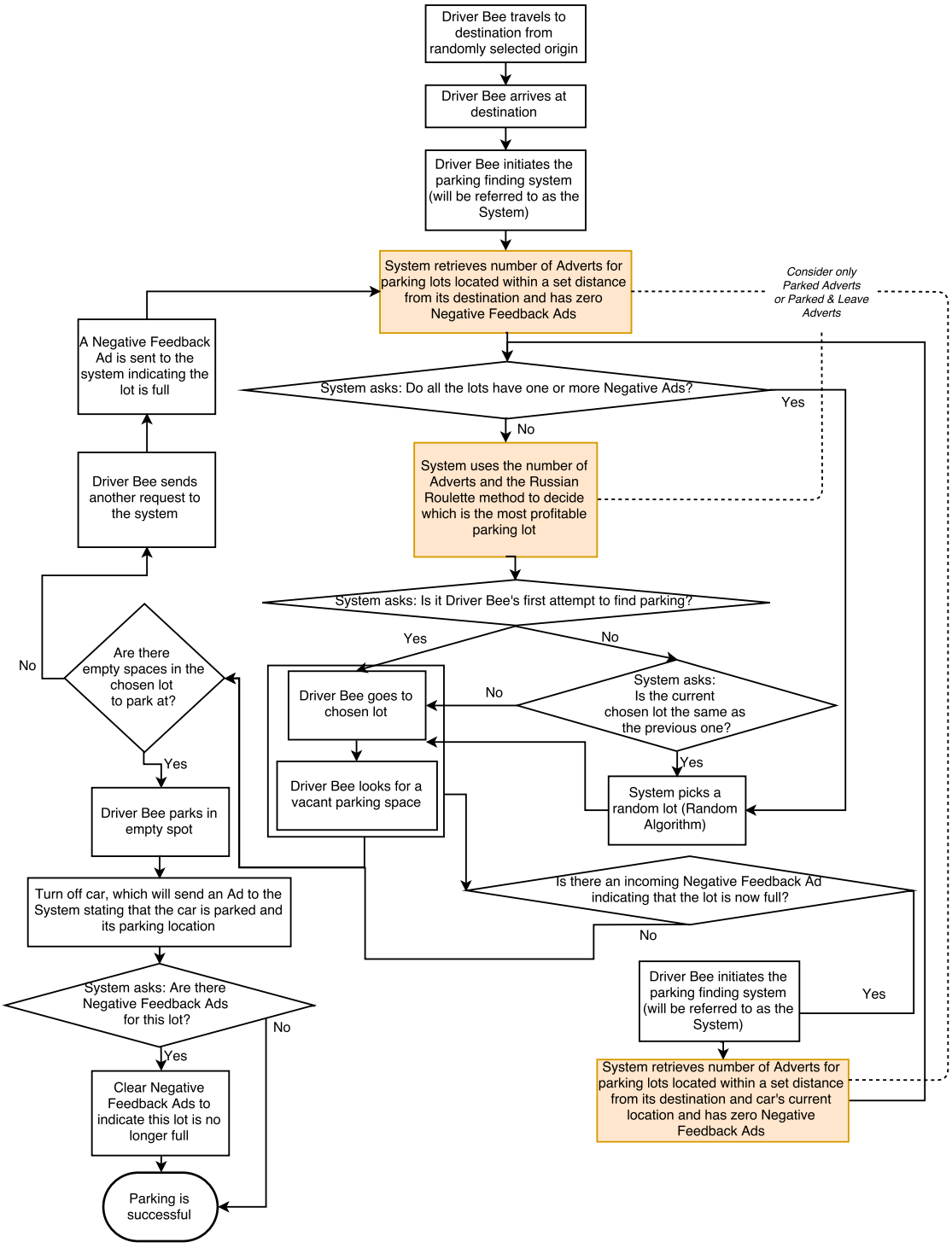


Figure 16 – Flowchart of HoneyPark Algorithms with Instantaneous Negative Feedback Mechanism

The chapter also introduced two existing algorithms, Random and Greedy, that will be used to evaluate the HoneyPark algorithm against other parking algorithms.

CHAPTER 4. EXPERIMENTAL STRUCTURE

The HoneyPark algorithms were evaluated using computational modeling and simulation techniques. This chapter will cover the setup of the parking simulation, how the parking system operates within the simulation and the routine all drivers will follow once they are initialized. It will also cover the techniques used to ensure that the results collected from the simulation are accurate and correctly interpreted.

4.1 Simulation Environment and Setting

It is important to ensure that the algorithms are tested and evaluated in a realistic environment to ensure that the simulation will give an accurate measurement of algorithm performance under real time road infrastructure. As such, the simulation was carried out using the map and parking layout of an existing city, namely San Francisco, California. This city was chosen due to its variety of road structures and has a mix of grid and non-grid road layouts as seen in Figure 17. In addition, there is relatively more parking data available for the city which can be downloaded online. The road map of San Francisco was provided by OpenStreetMaps while the location of the parking lots was acquired using both OpenStreetMaps and Overpass API. It should be noted that there may be more parking lots than those implemented in the simulation due to the limited quantity of parking data. A total of 221 parking lots was placed on the simulation map, which are marked with red circles in Figure 17.

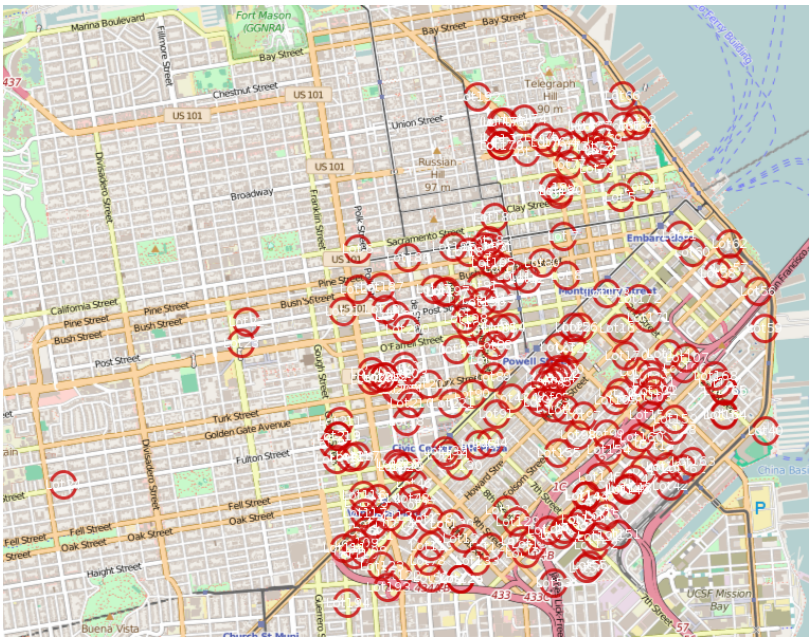


Figure 17 – Layout of Simulation Environment Marked with Parking Lots used in Simulation

The drivers in the simulation could travel to, and consequently finding parking, at 60 destinations within the city which are randomly picked by the author. Each destination is represented by the red dots in Figure 18.

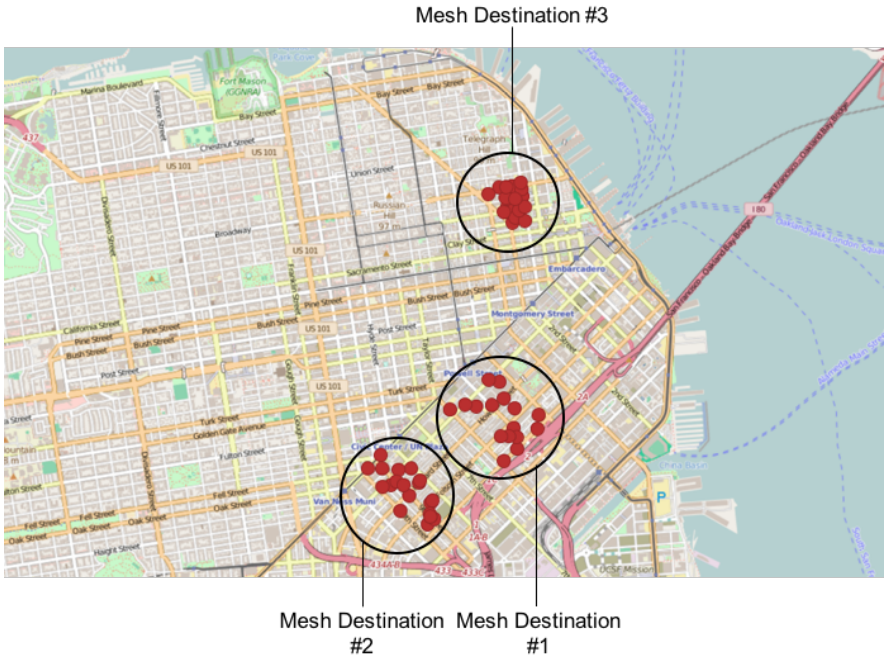


Figure 18 – Destinations Used in Simulation

Note that the author has also picked the destinations in such a way they are clustered in three geographical groups called 'Mesh Destinations', which are marked by black circular boundaries. In each group, the destinations are close enough in proximity in such a way the drivers may have different destinations but will still search the same parking lots. These Mesh Destinations were created to see if drivers targeting slightly different destinations but searching the same set of parking lots will have any impact on algorithmic performance.

4.2 Parking System

The simulation is executed by incrementing timesteps of one second. At each timestep, the simulation will update the status of the following components:

- Drivers: Every timestep, the simulation will loop through all the drivers and update their status. Drivers in transit to their destination or parking lots are moved forward towards

their target location by one second. For drivers that are determining which lot to search, it is assumed that the parking algorithms take one second to determine the most suitable lot to explore.

- Lots: As drivers park at and leave parking lots, the simulation will keep track of how many lots have been occupied or freed at each timestep as it loops through and updates the statuses of all the drivers. After it has done so at the end of each timestep, it will update the actual lot data using the data collected so that it seems that the parking spaces were concurrently occupied/freed.
- Advert System: The advert system will only be updated at each timestep to simulate the fact the drivers are searching for parking simultaneously. As such, the drivers will first send their adverts to a temporary array, which will be used to update the advert system at the next timestep.

The parking system is also designed to handle special situations that may occur. One example is that a situation may occur in which there may be more drivers that arrive and search at a parking lot in a timestep than there are available parking lots at the time. In this case, each available parking lot will be assigned randomly to one of the bees searching. The remaining bees will finish searching the lot before it uses the parking algorithm to find another potential lot to search if they are still unsuccessful.

4.3 Parking Methodology

In the simulation, all the drivers follow the same routine. First, a driver is initialized at random places within or just outside San Francisco city and will start travelling to its desired destination at a time provided by a random number generator. Once it is within 0.05km of their

destination, it will start looking for a parking lot to explore. It will only consider parking at lots that is within a specified distance of its destination as it doesn't want to park its car at a location that is unreasonably far away from its destination. In this simulation, this distance is set to 0.4 km. It will use one of the parking algorithms to determine which lot to explore. When it arrives at a parking lot, it will take some time to look for a parking space. In this simulation, it is assumed that the driver will take one second to observe a parking space and determine if it is occupied or empty and will look through all the occupied parking spots before finding an empty space. As such, the amount of time it takes for a driver to find a parking space in a lot or determine that a parking lot is full can be calculated using Equation 3.

$$t_{search} = N_{total} - N_{empty} \quad (3)$$

where t_{search} = amount of time driver spends searching one parking lot

N_{total} = total number of parking spaces in the parking lot

N_{empty} = number of empty parking spaces in the parking lot

For example, a driver explores a parking lot has 60 spaces in total, 20 of which are empty. It will take him/her 40 seconds (60 seconds – 20 seconds) to find a parking space. Likewise, if the parking lot is completely occupied, it will take the driver 60 seconds (60 seconds – 0 seconds) to determine that the entire lot is full and it will consult the parking algorithms again to suggest another parking lot to search. This system mimics the fact that it will take a driver a longer time to take a vacant parking space in a more occupied parking lot than an emptier one.

Once the driver has found a parking spot, the simulation calculates the time it took for the driver to find parking, which is calculated using Equation 4.

$$t_{park} = T_{dest} - T_{parked} \quad (4)$$

where t_{search} = amount of time driver spent to find an empty parking space

T_{dest} = the time at which the driver arrived within 0.05km of destination

T_{parked} = the time at which the driver found an empty parking space

The driver's car will stay there and the parking space will be considered to be occupied for a period of time, which will be referred to as the 'errand time' in this paper. Unless specified, each driver will have a randomized errand time ranging from five minutes to three hours in most simulations. After the errand time has elapsed, the driver will leave the parking lot and the parking space will be marked as unoccupied.

4.4 Evaluating and Ensuring Accuracy of Results

A couple of mechanisms were implemented to ensure that the results collected was accurate and correctly interpreted. The first measure was to set up an automated system that is able to run the simulation several times for a particular scenario. This is because the parking environment is volatile and each simulation run produces different results. As such, the number of simulations performed was manually increased until a consistent result was obtained.

Another measure taken to ensure that the results produced were accurate was to give each driver the same starting position, starting time and errand time for a single scenario. This is to

ensure that the algorithms were tested fairly. That way, one would not mistakenly think that one algorithm was more efficient than another algorithm when in reality, the more 'efficient' algorithm was tested using more favorable simulation settings. Thus, it is best to test the algorithms using the same parameters in a single scenario. However, we must also make sure that the algorithms are able to perform in a variety of simulation settings. Therefore, the driver's starting position, starting time and errand times are changed when one of the simulation variables are changed (i.e. change in the number of drivers, change in parking congestion). As such, one is able to ensure that the algorithms are tested fairly and at the same time, measure their performance in a variety of situations.

It is also important to ensure that our data is not only consistent, but also correctly interpreted. As the parking environment is constantly changing, parking times can vary greatly and it can be difficult to tell if the results produced by one algorithm is significantly different from another. A solution to this problem is to use the Wilcoxon Rank-Sum Test, which compares the median between two populations, or in this case the results produced by two different parking algorithms. The null hypothesis is that the two populations are equal. This holds true if the p-value is larger than the 0.05 significance level. If this hypothesis is rejected, that is if the p-value is less than 0.05, the two populations are significantly different from each other (Yau). In addition to the p-value, a test statistic W is also produced by the Wilcoxon Rank-Sum Test. It is derived by first calculating and ranking the difference between individual values in the two populations. Then the rankings of the positive and negative differences are summed separately and substituted into Equation 5 and Equation 6.

$$W_1 = n_1 n_2 + \frac{n_1(n_1 + 1)}{2} - R_1 \quad (5)$$

$$W_2 = n_1 n_2 + \frac{n_2(n_2 + 1)}{2} - R_2 \quad (6)$$

where n_1 = number of samples in the first population

n_2 = number of samples in the second population

R_1 = the sum of the ranks of the first population (that is, the ranks of the differences in which individual value of the first population exceeds that of the second population

R_2 = the sum of the ranks of the second population (that is, the ranks of the differences in which individual value of the second population exceeds that of the first population

The final test statistic W is the lowest value between W_1 and W_2. It is meant to reflect the extent that the populations are similar or different from each other (LaMorte, 2017).

The Wilcoxon Rank-Sum Test was chosen to evaluate the simulation results for two main reasons. The first reason is that it is a non-parametric test (LaMorte, 2017). One cannot assume that the parking times produced by the HoneyPark algorithms fits any distribution. Therefore, it is

best to use a distribution-free test to evaluate the simulation results. The second reason is that the test is designed to compare two distinct, unrelated populations (Yau). As the performance of one algorithm does not necessarily affect that of another algorithm, one can say that the results produced by the two algorithms are independent of each other.

4.5 Summary of Experimental Structure

The importance of this chapter is to describe the environment in which the algorithms were tested and evaluated. This included the physical arrangement and setting of the simulation. It also comprised of the 'software' of the simulation, namely how the drivers were managed and proceeded through the simulation from the moment he or she enters to the time he or she exits the simulation. This chapter also covers how the efficiency of the parking algorithms will be measured, which will be done through the use of 'parking times', which is the time it takes for the driver to find a parking space.

CHAPTER 5. EXPERIMENTAL RESULTS

In this chapter, the performance of the HoneyPark algorithms was measured and evaluated using computer simulation. The results will be discussed and used to evaluate the effectiveness of the algorithms in a range of situations. The application of the algorithm in varying levels of parking demand, diverse lengths of errand times, parking congestion, variable amounts of road traffic and environments with a mix of Honey and Non-Honey cars as well as a combination of V2V and Non-V2V vehicles will be examined and used to determine the factors and environs in which the algorithms thrive or disappoint.

5.1 Algorithmic Performance in Varying Levels of Parking Demand

This section will examine how algorithmic performance varies with parking demand relative to parking supply. In practical terms, we want to see how the HoneyPark algorithms fare in crowded areas where parking demand is much higher than the parking spaces available or in spare areas where parking supply often exceeds parking demand. The relative amount of parking demand in controlled by varying the 'lot-to-bee ratio', which is defined as the ratio of the number of bees looking for parking in a simulation run to the number of parking spaces available that is located within a certain distance of the bee's location. The lot-to-bee ratios examined in this thesis is 0.50, 0.75, 1.00, 2.00 and 4.00.

5.1.1 One Destination

This section will address the scenario in which all the drivers wish to travel to one destination. The destination, shown as a solid red dot, used in this section is displayed in Figure 19 along with its surrounding parking lots which are marked are red circles.

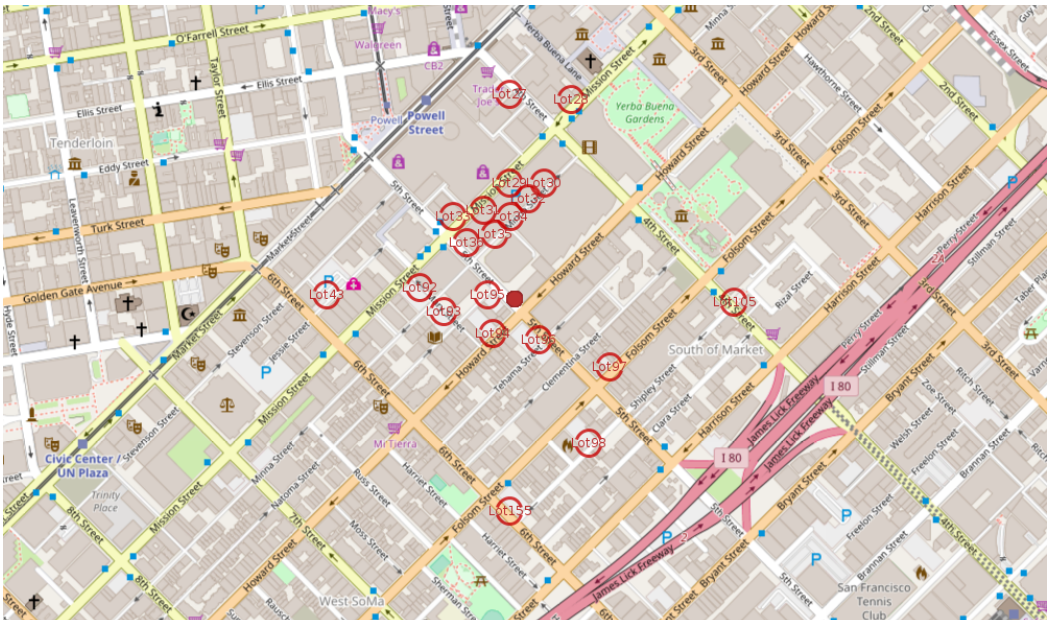


Figure 19 – Destination and Parking Lot Used in One Destination Scenario

All the results collected in this scenario is displayed in Figure 20. The average parking times are displayed by algorithm and lot-to-bee ratio. This section will break the data down further and expound upon it.

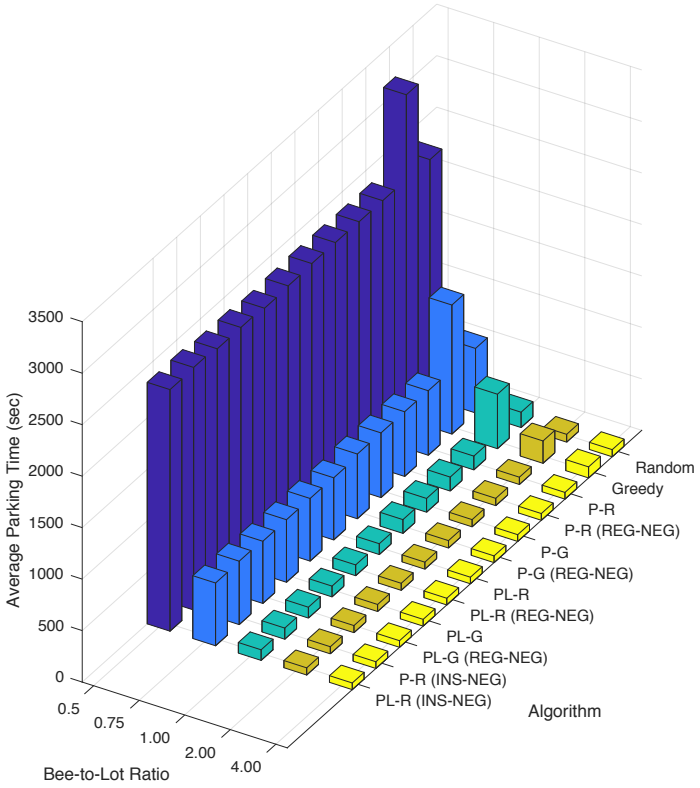


Figure 20 – Average Parking Times Based on the Algorithm Used and Lot-to-Bee Ratio

5.1.1.1 Parking Supply is Equivalent to Parking Demand

Figure 21 shows the average time it took a driver to find parking when using each of the algorithms. Table 4 expresses the average parking times in numerical form along with how many simulations were executed for each algorithm in this scenario. We will first discuss the case in the which the parking supply equals to parking demand. In other words, the lot-to-bee ratio is 1.00 and there are roughly as many parking space available as there are drivers who wish to park.

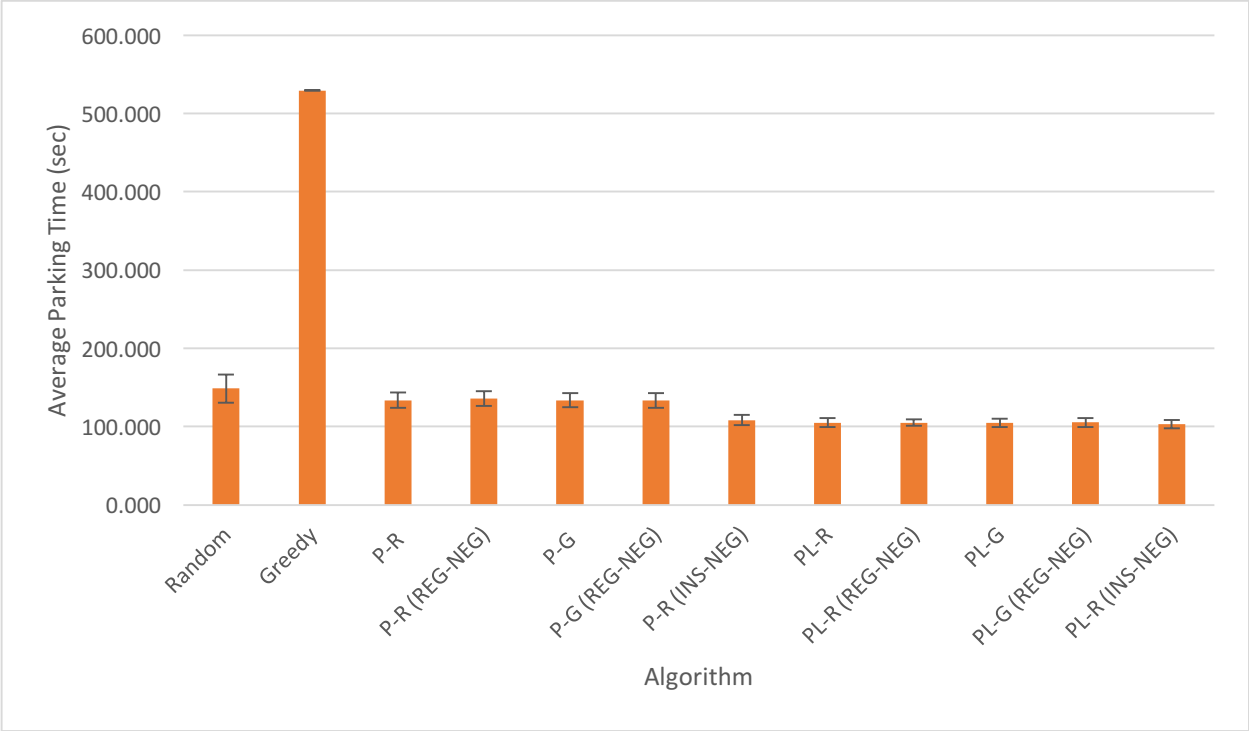


Figure 21 – Average Parking Times Produced by Each Algorithm in One Destination Scenario when Lot-to-Bee Ratio = 1

Table 4 – Average Parking Times and Variances produced in One Destination Scenario (Lot-to-Bee Ratio = 1)

Algorithm	No. of Simulations Executed	Average Parking Time (sec)	Standard Deviation of Average Parking Time (sec)	Ratio of Parking Time to Random Parking Time
Random	70	148.872	17.997	1.000
Greedy	70	529.165	0.000	3.554
P-R	70	133.755	10.123	0.898
P-R (REG-NEG)	70	135.679	9.490	0.911
P-G	70	133.721	8.766	0.898
P-G (REG-NEG)	70	133.577	9.255	0.897
P-R (INS-NEG)	70	108.413	6.734	0.728
PL-R	70	105.181	5.764	0.707
PL-R (REG-NEG)	70	104.938	4.173	0.705
PL-G	70	104.663	5.369	0.703
PL-G (REG-NEG)	70	105.290	5.527	0.707
PL-R (INS-NEG)	70	102.901	5.198	0.691

Table 3 shows that the standard deviation is relatively small compared to the average parking times. This means that the deviation in average parking times is small. This is due to the fact that a sufficient number of simulations were performed. The number of simulations executed was increased until a consistent average parking time was produced for all algorithms. As Table 3 shows, 70 simulations were needed to ensure that the average parking time was consistent and its variance was reasonably low. It is also important to note that the standard deviation for Greedy algorithm is consistently zero. This is because drivers using the Greedy algorithm follow the same route in which they start their search at the nearest parking lot. If they are unsuccessful, they will progress to the next nearest parking lot and so forth. As the position of the lots does not change, all the drivers follow the same route. Given that the drivers have the same starting times, positions and errand times for a single case as mentioned in Section 4.4, it is easy to see why Greedy will produce the same parking times in every simulation. The Random and HoneyPark algorithms, on the other hand, do not send drivers to the exact same lots in each simulation run. The Random algorithm sends drivers to random lots while the HoneyPark algorithm sends drivers to parking lots that considered profitable at the time, which can be extremely variable. Therefore, the results are different for each simulation run and the standard deviation is greater than zero.

As one can see, all variations of the HoneyPark algorithms performed better than either the Random or Greedy algorithm. The ratios in Table 4 indicate that the HoneyPark algorithms consistently produces lower parking times than either Random or Greedy. The Wilcoxon Rank-Sum Test confirms that the difference between the Random/Greedy and the HoneyPark algorithms are statistically significant. As one can see in Table 5, when all the HoneyPark variations were compared with the Random algorithm, the Wilcoxon Rank-Sum Test produced p-values that are less than 0.05, indicating that the average parking times produced by the HoneyPark algorithms

statistically significantly lower than the Random algorithm. Likewise, the Wilcoxon Rank-Sum Test proves that the HoneyPark algorithm performs significantly better than the Greedy algorithm the W-value is calculated to be 0 and the p-value $< 2.2 \times 10^{-16}$ for all HoneyPark algorithms when they are compared with the Greedy algorithm.

Table 5 – Results of Wilcoxon Rank-Sum Test to Compare the Performance of HoneyPark Algorithms against the Random Algorithm (Lot-to-Bee Ratio = 1)

Algorithm	W	p-value
P-R - Random	1115	2.67E-08
P-R (REG-NEG) - Random	1289	1.32E-06
P-G - Random	1110	2.37E-08
P-G (REG-NEG) - Random	1148.5	5.89E-08
P-R (INS-NEG) - Random	18	< 2.2E-16
PL-R - Random	4	< 2.2E-16
PL-R (REG-NEG) - Random	0	< 2.2E-16
PL-G - Random	1	< 2.2E-16
PL-G (REG-NEG) - Random	2	< 2.2E-16
PL-R (INS-NEG) - Random	1	< 2.2E-16

The reason for this is trivial. Drivers using the HoneyPark algorithms have an indication of which parking lots are profitable before picking a lot to search, increasing their chances of finding a vacant parking spot on their first attempt. Drivers using the Random and Greedy algorithms do not factor any indication which lots are profitable, making it less likely that they will find a parking spot at the chosen search lot and consequently prolonging their parking search time. This is disadvantageous especially in the case of the Greedy algorithm. Figure 21 shows that the Greedy algorithm performs markedly worse than either the Random or HoneyPark algorithms. This is because it always sends the drivers to the nearest parking lot regardless of its profitability, overloading the lot. However, even though the nearest lot is overfilled and extremely unprofitable,

the Greedy algorithm still insists on sending drivers there. This results in drivers exploring all the nearest lots first (which are also packed with other drivers using the Greedy algorithm) before considering searching lots further away, leading to abnormally long parking times.

Figure 21 also shows that HoneyPark algorithms that used both Parked and Leave adverts to determine which parking lots they should search performed much better than algorithms that only utilized Parked adverts. This is because the use of Leave adverts opens up the definition of a profitable lot – it not only encompasses lots where drivers have success in finding parking but also lots in which drivers are leaving and repopulating vacant parking spots. Consequently, the drivers are given more potential parking lots to consider which increases the chances of finding a parking spot and shortens parking times.

Table 6 – Results of Wilcoxon Rank-Sum Test to Compare the Performance between Random-Scouting and Greedy-Scouting Algorithms (Lot-to-Bee Ratio = 1)

Algorithm	W	p-value
P-R - P-G	2340	0.6481
P-R (REG-NEG) - P-G (REG-NEG)	2688.5	0.3213
PL-R - PL-G	2436.5	0.9568
PL-R (REG-NEG) - PL-G (REG-NEG)	2336.5	0.6377

The results of the Wilcoxon Signed-Ranked test displayed in Table 6 shows that there is generally little difference in the performance of HoneyPark algorithms that utilized the Random algorithm as their scouting algorithm against their counterparts that used the Greedy algorithm. The simulation shows that only a couple of drivers utilized the scouting algorithm in the duration of one simulation run and nearly, if not all, the bees successfully found a parking spot without using the scouting algorithms. As the drivers did not need to resort to using a scouting algorithm,

there should be no difference between HoneyPark algorithms that utilized the Random scouting algorithm and their Greedy scouting counterparts as they follow the same heuristic with the exception of the type of scouting method, which they did not need to use at this lot-to-bee ratio.

Table 7 – Results of Wilcoxon Rank-Sum Test to Compare the Performance between HoneyPark Algorithms that Use no Negative Feedback, Regular Negative Feedback and Instantaneous Negative Feedback (Lot-to-Bee Ratio = 1)

Algorithm	W	p-value
P-R - P-R (REG-NEG)	2173	0.2492
P-G - P-G (REG-NEG)	2421	0.9055
PL-R - PL-R (REG-NEG)	2576.5	0.5995
PL-G - PL-G (REG-NEG)	2288	0.5009
P-R - P-R (INS-NEG)	4841	< 2.2E-16
P-R (REG-NEG) - P-R (INS-NEG)	4871	< 2.2E-16
PL-R - PL-R (INS-NEG)	3034	0.01502
PL-R (REG-NEG) - PL-R (INS-NEG)	3037	0.01451

Results were also compared between algorithms that utilized negative feedback and their counterparts than did not. Table 7 indicates that the difference in performance between algorithms that did not use negative feedback and those that used regular negative feedback is quite negligible. This is because simulation data shows that regular negative feedback is quite redundant. As mentioned in Chapter 3, the number of Parked/Leave adverts for a lot resets to a low number when it is full. For non-negative feedback adverts, the chances of sending a driver to that lot is already low due to its low number of advertisements. All the regular negative feedback mechanism does is put an additional barrier to the lot by sending out a Negative advert prevent any drivers from exploring it. The simulation results show that this extra block is redundant as the drivers are already avoiding the parking lot due to the fact it is lowly advertised. As such, it makes sense why the

average parking times produced by algorithms that use a regular negative feedback algorithm and those who use no negative feedback algorithm are similar.

However, Table 7 shows that algorithms that used instantaneous negative feedback performed significantly better than algorithms that either did not used negative feedback or used regular negative feedback. It is intuitive to see why instantaneous negative feedback would result in better performance than no negative feedback at all. Instantaneous negative feedback immediately lets a driver know when his or her target search lot becomes full and redirects him or her to another potentially profitable lot. This saves the driver time as he or she does not need to waste time travelling and searching a parking lot that is already full, consequently reducing his or her parking time.

In addition, it is important to note that the instantaneous negative feedback redirects the driver to another lot that is close to the driver's current location. Past simulations have shown that algorithms that use instantaneous negative feedback but do not redirect the drivers to nearby parking may even result in worse performance than algorithms that utilized regular negative feedback. This is because when the driver is informed that his or her target search lot is no longer profitable, he or she has already spent time travelling to it. Simulation data shows that the driver will actually take longer to find parking if the alternative lot is too far away as it takes a longer time for the car to reroute than if it went to its target lot and waited for a parking space to become available. Therefore, a reduction in parking time is only seen with the instantaneous negative feedback mechanism on the condition that the alterative lot is not too far away from the car's current position.

5.1.1.2 Parking Supply is less than Parking Demand

Figure 22 and Figure 23 show the parking average times produced by each algorithm when the lot-to-bee ratio is 0.75 and 0.5 respectively. Table 8 and Table 9 gives this information in numeric form along with the number of simulations executed for each algorithm.

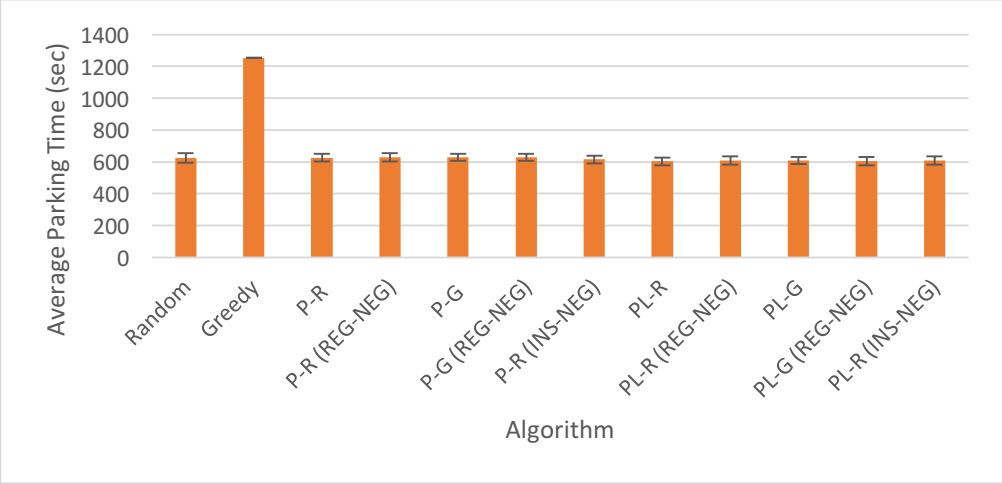


Figure 22 – Average Parking Times Produced by Each Algorithm in One Destination Scenario when Lot-to-Bee Ratio = 0.75

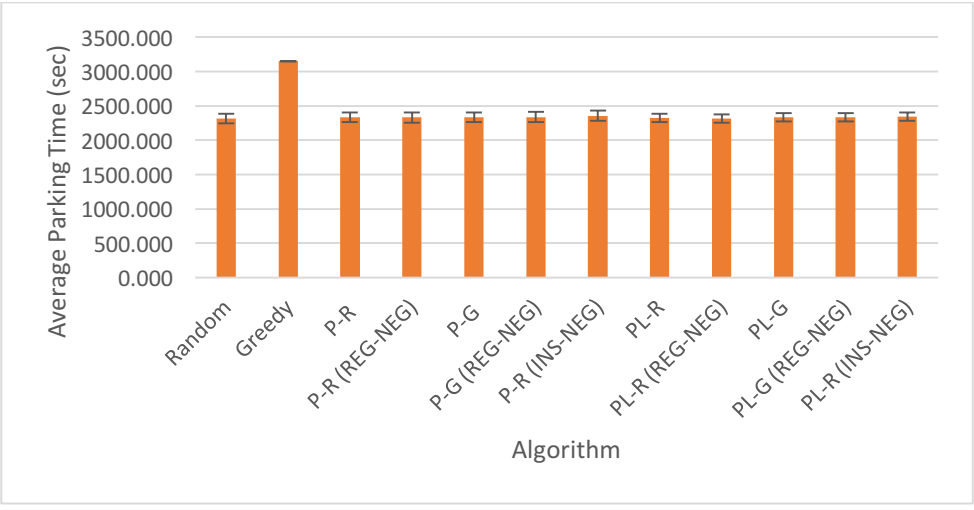


Figure 23 – Average Parking Times Produced by Each Algorithm in One Destination Scenario when Lot-to-Bee Ratio = 0.5

Table 8 – Average Parking Times and Variances produced in One Destination Scenario (Lot-to-Bee Ratio = 0.75)

Algorithm	No. of Simulations Executed	Average Parking Time (sec)	Standard Deviation of Average Parking Time (sec)	Ratio of Parking Time to Random Parking Time
Random	70	626.256	30.430	1.000
Greedy	70	1253.382	0.000	2.001
P-R	70	626.604	25.355	1.001
P-R (REG-NEG)	70	628.178	25.565	1.003
P-G	70	628.380	23.320	1.003
P-G (REG-NEG)	70	628.685	21.886	1.004
P-R (INS-NEG)	70	616.020	24.209	0.984
PL-R	70	603.593	24.978	0.964
PL-R (REG-NEG)	70	608.310	25.800	0.971
PL-G	70	607.530	22.572	0.970
PL-G (REG-NEG)	70	603.961	25.718	0.964
PL-R (INS-NEG)	70	609.673	25.678	0.974

Table 9 – Average Parking Times and Variances produced in One Destination Scenario (Lot-to-Bee Ratio = 0.5)

Algorithm	No. of Simulations Executed	Average Parking Time (sec)	Standard Deviation of Average Parking Time (sec)	Ratio of Parking Time to Random Parking Time
Random	70	2317.987	69.611	1.000
Greedy	70	3153.414	0.000	1.360
P-R	70	2330.839	68.854	1.006
P-R (REG-NEG)	70	2331.481	72.624	1.006
P-G	70	2334.978	67.703	1.007
P-G (REG-NEG)	70	2336.742	72.673	1.008
P-R (INS-NEG)	70	2355.719	73.961	1.016
PL-R	70	2325.484	62.123	1.003
PL-R (REG-NEG)	70	2314.032	62.032	0.998
PL-G	70	2332.812	58.732	1.006
PL-G (REG-NEG)	70	2330.883	59.136	1.006
PL-R (INS-NEG)	70	2345.941	62.216	1.012

The HoneyPark algorithms still perform significantly better than the Greedy algorithm. All the HoneyPark-Greedy pairs for both lot-to-bee ratios of 0.75 and 0.5 produce a W value of zero and a p-value of < 2.2E-16, indicating that the overlap in the average parking times produced by the Greedy and HoneyPark algorithms is small. However, the efficiency of the Random algorithm is comparable to the Parked-Advert HoneyPark algorithms at a lot-to-bee ratio of 0.75. This is shown in Table 8 where the ratios between the parking times of Parked-Advert algorithms to Random algorithm is close to the value of one. This fact is further supported by the data in Table 10, where the p-values produced by Random-Parked-Advert HoneyPark pairs exceed 0.05 and thus, indicating that there is no significant difference in performance between the two algorithms. As the lot-to-bee ratio decreases even further to 0.5, the efficiency of the Random algorithm and all of the HoneyPark algorithms are no longer significantly different. As one can see in Table 9, the ratios are close to one, indicating that the HoneyPark and Random algorithm produce similar parking times. This is further supported by Table 11, which shows that there is no significant dissimilarity in any of the HoneyPark-Random pairs.

Table 10 – Results of Wilcoxon Rank-Sum Test to Compare the Performance of HoneyPark Algorithms against the Random Algorithm (Lot-to-Bee Ratio = 0.75)

Algorithm	W	p-value
P-R - Random	2501.5	0.8317
P-R (REG-NEG) - Random	2586	0.5723
P-G - Random	2583	0.5808
P-G (REG-NEG) - Random	2640	0.4297
P-R (INS-NEG) - Random	1934	0.03168
PL-R - Random	1337	3.55E-06
PL-R (REG-NEG) - Random	1570	2.47E-04
PL-G - Random	1532	0.0001314
PL-G (REG-NEG) - Random	1392	1.05E-05
PL-R (INS-NEG) - Random	1648	0.0008396

Table 11 – Results of Wilcoxon Rank-Sum Test to Compare the Performance of HoneyPark Algorithms against the Random Algorithm (Lot-to-Bee Ratio = 0.5)

Algorithm	W	p-value
P-R - Random	2731	0.2424
P-R (REG-NEG) - Random	2684	0.3305
P-G - Random	2850	0.09592
P-G (REG-NEG) - Random	2815	0.1287
P-R (INS-NEG) - Random	3176	0.002498
PL-R - Random	2601	0.5305
PL-R (REG-NEG) - Random	2384	0.7849
PL-G - Random	2776	0.1749
PL-G (REG-NEG) - Random	2725	0.2526
PL-R (INS-NEG) - Random	3033	0.0152

In short, the data shows that generally, the efficiency of the Random and HoneyPark algorithm become more comparable as the lot-to-bee ratio decreases. As such, one can conclude that the performance of the parking algorithms is to an extent limited by the ability of existing parking infrastructure to handle parking demand. The simulation shows that average parking times become similar among the Random and HoneyPark algorithms as there are not enough parking spaces to accommodate the number of drivers looking for parking. Therefore, a driver will experience long parking times regardless of how efficiently a parking algorithm may direct him or her to a profitable lot.

As the lot-to-bee decreases to 0.5, Figure 23 and Table 11 show that the instantaneous negative feedback algorithm's performance is significantly worse than that of the Random algorithm. Due to the large demand for parking spaces, the number of parking spaces are quickly filled and the lots are occupied for a longer period of time. Simulation data shows that drivers using instantaneous negative feedback tend to experience either very short parking times as they

are able to find a parking space after considering a small number of lots or extremely long times being stuck in limbo as the algorithm is constantly trying to redirect the driver as all the parking lots are marked as full for longer periods of time. As a result, the driver spends more time deciding which parking lot to go to than going to a parking lot and finding a parking space. Therefore, using instantaneous negative feedback in an extremely crowded parking environment where the lots are marked as unprofitable for an exceptionally long amount of time results in ineffectiveness.

Table 12 – Results of Wilcoxon Rank-Sum Test to Compare the Performance between HoneyPark Algorithms that Use No Negative Feedback, Regular Negative Feedback and Instantaneous Negative Feedback (Lot-to-Bee Ratio = 0.75)

Algorithm	W	p-value
P-R - P-R (REG-NEG)	2338	0.6422
P-G - P-G (REG-NEG)	2370	0.7404
PL-R - PL-R (REG-NEG)	2110.5	0.1577
PL-G - PL-G (REG-NEG)	2677.5	0.3441
P-R - P-R (INS-NEG)	3039	0.01418
P-R (REG-NEG) - P-R (INS-NEG)	3158	0.003193
PL-R - PL-R (INS-NEG)	2115	0.1633
PL-R (REG-NEG) - PL-R (INS-NEG)	2384	0.7849

Table 13 – Results of Wilcoxon Rank-Sum Test to Compare the Performance between HoneyPark Algorithms that Use No Negative Feedback, Regular Negative Feedback and Instantaneous Negative Feedback (Lot-to-Bee Ratio = 0.5)

Algorithm	W	p-value
P-R - P-R (REG-NEG)	2445	0.985
P-G - P-G (REG-NEG)	2440.5	0.9701
PL-R - PL-R (REG-NEG)	2669	0.3625
PL-G - PL-G (REG-NEG)	2483.5	0.8906
P-R - P-R (INS-NEG)	1978	0.04941
P-R (REG-NEG) - P-R (INS-NEG)	1992	0.05656
PL-R - PL-R (INS-NEG)	1996	0.05876
PL-R (REG-NEG) - PL-R (INS-NEG)	1768	0.004509

Now, let's investigate the performance of the regular negative feedback mechanism. Table 12 and Table 13 infers that there is no difference in performance between HoneyPark algorithms that use regular negative feedback and those who use no negative feedback at all. This is probably because the low lot-to-bee ratio results in full parking lots for the majority of the time. In this case, drivers that do not use negative feedback will search a potentially full lot. Driver that use negative feedback will likely see that a Negative advert has been initiated for all the parking lots and default to a scouting algorithm, which we have established in the previous paragraph is used more as a stalling technique to wait for parking spaces to open up rather than to find a profitable parking lot. Therefore, the speed at which a driver is able to find parking is dependent on the rate at which the parking lots open up rather than the ability of drivers to avoid unprofitable lots by means of negative feedback as all the lots are full.

Table 14 – Results of Wilcoxon Rank-Sum Test to Compare the Performance between Random-Scouting and Greedy-Scouting Algorithms (Lot-to-Bee Ratio = 0.75)

Algorithm	W	p-value
P-R - P-G	2366	0.7278
P-R (REG-NEG) - P-G (REG-NEG)	2408	0.8627
PL-R - PL-G	2191	0.2813
PL-R (REG-NEG) - PL-G (REG-NEG)	2727	0.2492

Table 15 – Results of Wilcoxon Rank-Sum Test to Compare the Performance between Random-Scouting and Greedy-Scouting Algorithms (Lot-to-Bee Ratio = 0.5)

Algorithm	W	p-value
P-R - P-G	2366	0.7278
P-R (REG-NEG) - P-G (REG-NEG)	2345	0.6632
PL-R - PL-G	2277	0.4722
PL-R (REG-NEG) - PL-G (REG-NEG)	2060.5	0.105

It is important to note that there is still no difference between the performance of algorithms that use a Random scouting algorithm and a Greedy scouting algorithm even though simulation data shows that drivers scouted more often due to the lack of vacant parking spots. The explanation could be found in the fact that scouting algorithm were not necessarily used to locate a profitable parking lot. This is supported by the data displayed in Table 14 and Table 15, whose p-values indicate there is no significant difference in performance between algorithm that use different scouting algorithms. On average, only 1.62% of drivers in a single simulation found a parking space while scouting when the lot-to-bee ratio is 0.75 and 0.9% when the lot-to-bee ratio is 0.5. This is obvious when one tracks an individual driver in the simulation. It is found that drivers resorted to scouting as a way of waiting for an advert to indicate that a parking space has opened up rather as an efficient means to look for parking. This makes sense as one of the features of the HoneyPark algorithm is that it defaults to the scouting algorithm when there were no adverts registered in the system or the Negative adverts indicate that all the lots are full. The algorithm stops using the scouting algorithm once a Parked or Leave advert comes in and indicates there is a parking space has opened up in the area of interest.

5.1.1.3 Parking Supply is more than Parking Demand

Figure 24 and Figure 25 show the parking average times produced by each algorithm when the lot-to-bee ratio is 2.0 and 4.0 respectively. Table 16 and Table 17 shows the same data in numeric form along with the number of simulation runs executed for each algorithm.

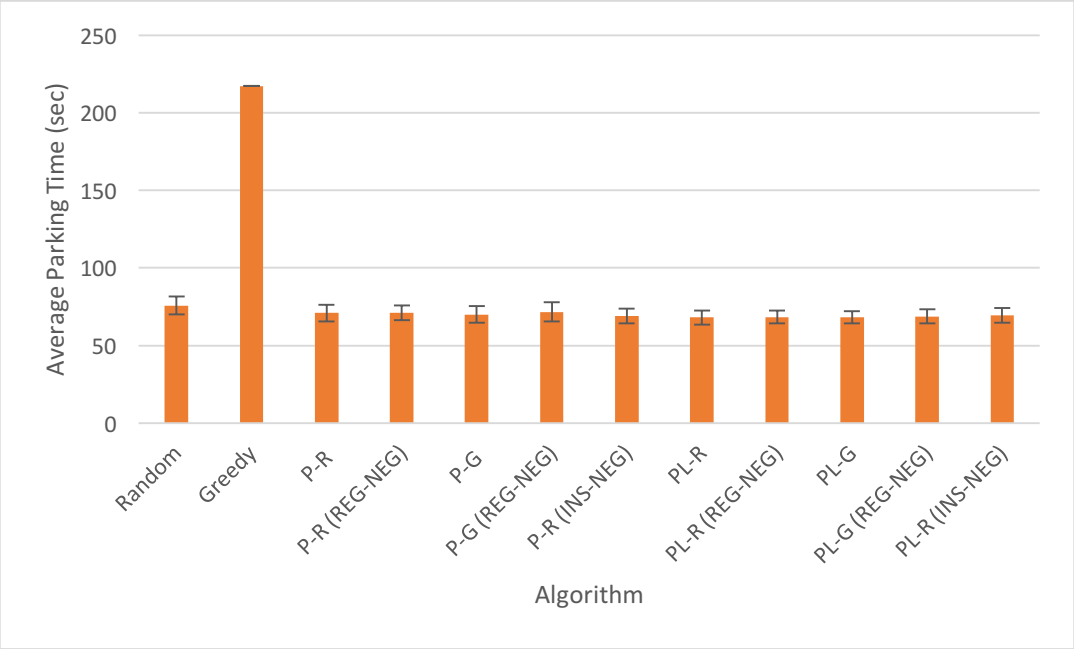


Figure 24 – Average Parking Times Produced by Each Algorithm in One Destination Scenario when Lot-to-Bee Ratio = 2

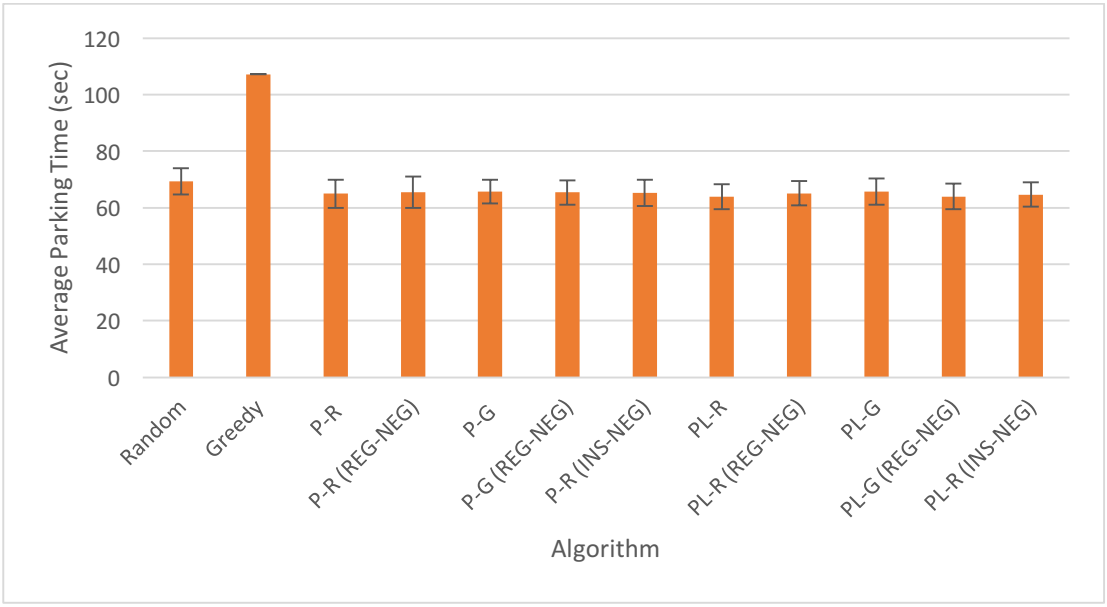


Figure 25 – Average Parking Times Produced by Each Algorithm in One Destination Scenario when Lot-to-Bee Ratio = 4

Table 16 – Average Parking Times and Variances produced in One Destination Scenario (Lot-to-Bee Ratio = 2)

Algorithm	No. of Simulations Executed	Average Parking Time (sec)	Standard Deviation of Average Parking Time (sec)	Ratio of Parking Time to Random Parking Time
Random	70	75.873	5.684	1.000
Greedy	70	217.269	0.000	2.864
P-R	70	70.976	5.381	0.935
P-R (REG-NEG)	70	71.044	4.824	0.936
P-G	70	70.098	5.346	0.924
P-G (REG-NEG)	70	71.724	6.231	0.945
P-R (INS-NEG)	70	69.054	4.894	0.910
PL-R	70	68.047	4.585	0.897
PL-R (REG-NEG)	70	68.424	4.034	0.902
PL-G	70	68.350	3.890	0.901
PL-G (REG-NEG)	70	68.840	4.357	0.907
PL-R (INS-NEG)	70	69.539	4.644	0.917

Table 17 – Average Parking Times and Variances produced in One Destination Scenario (Lot-to-Bee Ratio = 4)

Algorithm	No. of Simulations Executed	Average Parking Time (sec)	Standard Deviation of Average Parking Time (sec)	Ratio of Parking Time to Random Parking Time
Random	70	69.277	4.648	1.000
Greedy	70	107.242	0.000	1.548
P-R	70	64.930	5.035	0.937
P-R (REG-NEG)	70	65.446	5.613	0.945
P-G	70	65.709	4.263	0.948
P-G (REG-NEG)	70	65.415	4.310	0.944
P-R (INS-NEG)	70	65.203	4.638	0.941
PL-R	70	63.829	4.404	0.921
PL-R (REG-NEG)	70	65.084	4.284	0.939
PL-G	70	65.709	4.699	0.948
PL-G (REG-NEG)	70	63.992	4.557	0.924
PL-R (INS-NEG)	70	64.661	4.237	0.933

The important thing to note is that the Greedy algorithm significantly improves as the lot-to-bee ratio increases. Looking at Table 16 and Table 17, the ratio of the average parking time

produced by the Greedy algorithm to that of the Random algorithm drops drastically from 2.864 to 1.548. This is due to an excess of lots and low number of competing drivers. One is mostly guaranteed to find a parking space regardless of the parking lot visited. The only factor dictating the length of the parking time is the distance and amount of time it takes for the bee to travel to the parking lot. Consequently, it is faster for the driver to park at the closest lot. Additional simulations show that the performance of the Greedy algorithm begins to overtake that of the HoneyPark algorithm at a lot-to-bee ratio between four and six. However, it seems that the parking supply has to exceptionally exceed parking demand for the Greedy algorithm to be effective.

Table 18 shows that the HoneyPark algorithms still perform significantly better than the Random algorithm at lot-to-bee ratios of two and four. However, the difference in performance between the Random and HoneyPark algorithms is smaller as the demand for parking is extremely low to the extent that one can find a parking space regardless of which lot one chooses to search. Therefore, the average parking times produced by the Random algorithm is further reduced.

Table 18 – Results of Wilcoxon Rank-Sum Test to Compare the Performance of HoneyPark Algorithms against the Random Algorithm (Lot-to-Bee Ratio = 2)

Algorithm	W	p-value
P-R - Random	1319.5	2.49E-06
P-R (REG-NEG) - Random	1276.5	1.02E-06
P-G - Random	1099.5	1.84E-08
P-G (REG-NEG) - Random	1433.5	2.29E-05
P-R (INS-NEG) - Random	846.5	2.38E-11
PL-R - Random	707	3.81E-13
PL-R (REG-NEG) - Random	721.5	5.95E-13
PL-G - Random	949.5	4.07E-10
PL-G (REG-NEG) - Random	821.5	1.16E-11
PL-R (INS-NEG) - Random	949.5	4.07E-10

Table 19 – Results of Wilcoxon Rank-Sum Test to Compare the Performance of HoneyPark Algorithms against the Random Algorithm (Lot-to-Bee Ratio = 4)

Algorithm	W	p-value
P-R - Random	1271.5	9.13E-07
P-R (REG-NEG) - Random	1445.5	2.86E-05
P-G - Random	1420	1.78E-05
P-G (REG-NEG) - Random	1383	8.80E-06
P-R (INS-NEG) - Random	1361	5.72E-06
PL-R - Random	989	1.15E-09
PL-R (REG-NEG) - Random	1277	1.03E-06
PL-G - Random	1133	4.10E-08
PL-G (REG-NEG) - Random	992	1.25E-09
PL-R (INS-NEG) - Random	1165	8.63E-08

It is also important to note that as the lot-to-bee ratio increases, the difference in performance between HoneyPark algorithms that use Parked adverts and their counterparts that use both Parked and Leave adverts becomes less distinct. In fact, Table 19 shows that at a lot-to-bee ratio of four, there is no significant difference in average parking times between algorithms that use only Parked adverts and those who use both Parked and Leave adverts. This is because the demand is relatively so low compared to supply and a driver can find parking at any parking lot.

5.1.2 One Mesh Destination

Mesh Destination 1 from Figure 18 was used to evaluate algorithmic performance in one mesh destination. The mesh destination is pictured again below in Figure 26, along with its accompanying parking lots. The destinations are marked with solid red dots while the parking lots are marked with red circles.

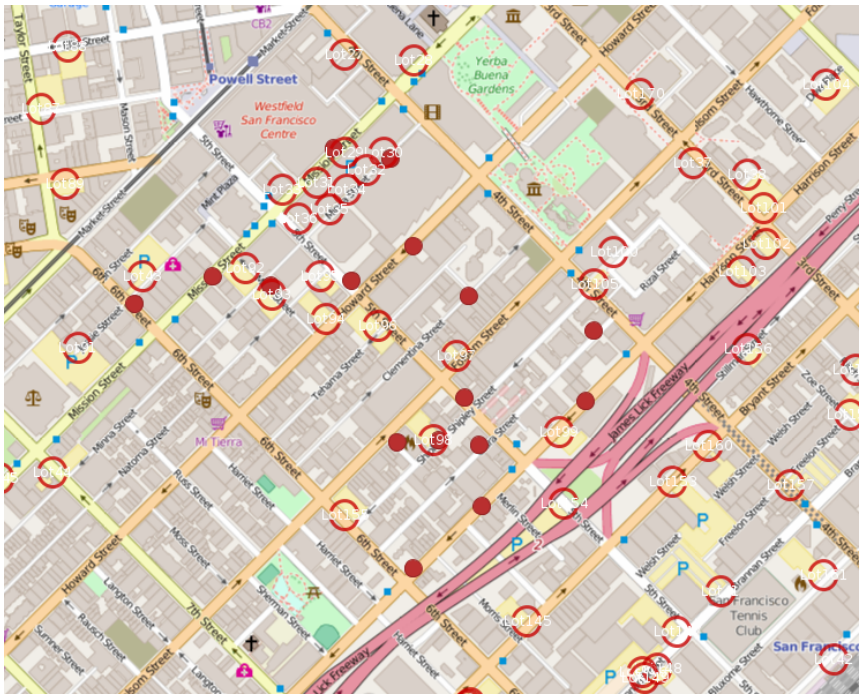


Figure 26 – Destination and Parking Lots Used in One Mesh Destination Scenario

In a mesh destination, the bees are assigned to a random destination within a particular area. As such, it is difficult to conduct an analysis based on the relative amount of supply and demand as more drivers may be directed to one destination than another. As such, the demand for certain parking lots will be higher than others and it is hard to say that the lot-to-bee ratio is equal throughout the simulation environment. Therefore, instead of adjusting the lot-to-bee ratio as in the One Destination scenario, the absolute number of drivers will be varied instead.

Figure 27, Figure 28, Figure 29 and Figure 30 show how the performance of the HoneyPark algorithms vary as the number of drivers in the simulation is increased. Table 20, Table 21, Table 22 and Table 23 shows the same data in numeric data along with the number of simulation executed for each algorithm. Note that the number of simulations performed for each algorithm is greater than that of the One Destination scenario. This is because there are more destinations that drivers

can go to, which creates more variation in the average parking times. Therefore, it takes a larger number of simulations to produce a consistent average parking time.

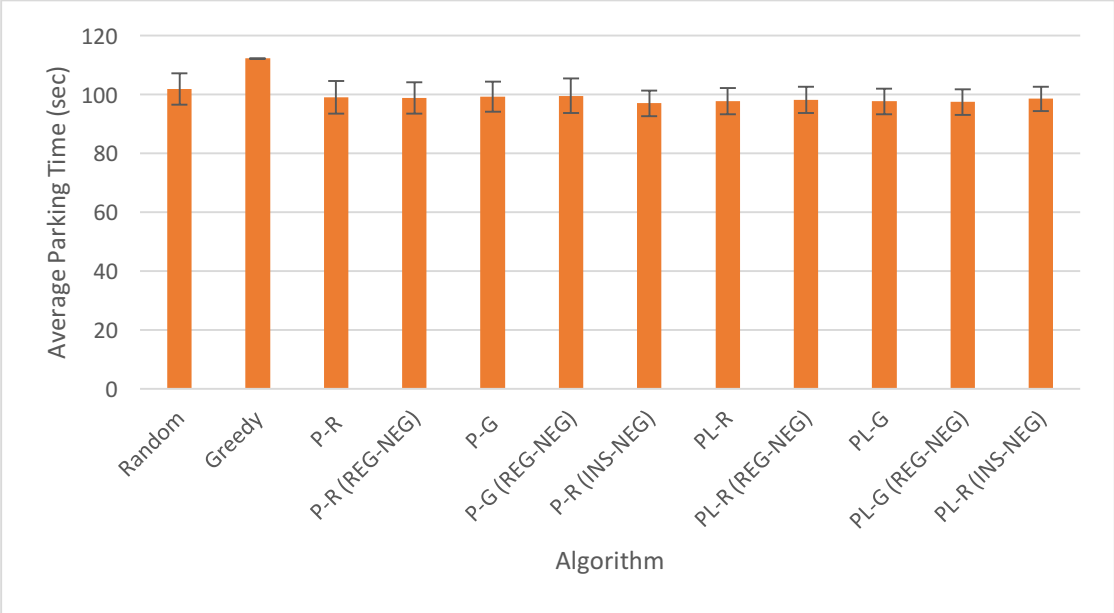


Figure 27 – Average Parking Times when a Small Number of Drivers were Initialized in the Simulation

Table 20 – Average Parking Times and Variances produced in One Destination Scenario when a Small Number of Drivers were Initialized in the Simulation

Algorithm	No. of Simulations Executed	Average Parking Time (sec)	Standard Deviation of Average Parking Time (sec)	Ratio of Parking Time to Random Parking Time
Random	200	101.7868	5.244784	1.000
Greedy	200	112.11	0	1.101
P-R	200	98.9991	5.561735	0.973
P-R (REG-NEG)	200	98.69895	5.32152	0.970
P-G	200	99.0759	5.092296	0.973
P-G (REG-NEG)	200	99.4668	5.828625	0.977
P-R (INS-NEG)	200	96.90745	4.365831	0.952
PL-R	200	97.6692	4.476285	0.960
PL-R (REG-NEG)	200	98.01815	4.447413	0.963
PL-G	200	97.54995	4.283353	0.958
PL-G (REG-NEG)	200	97.33605	4.41295	0.956
PL-R (INS-NEG)	200	98.44245	4.178693	0.967

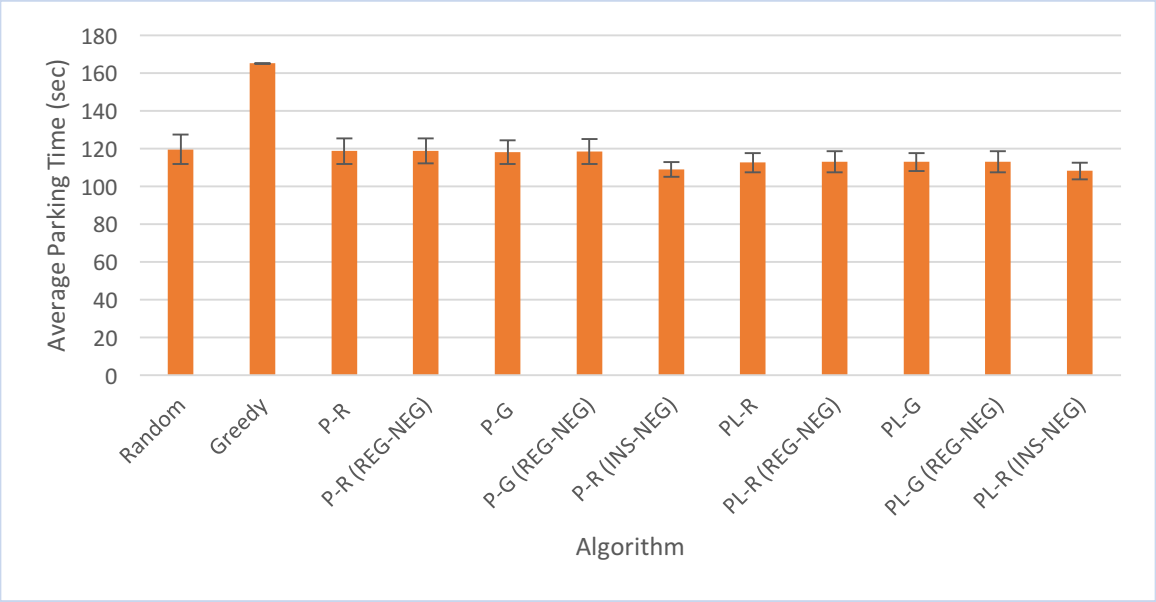


Figure 28 – Average Parking Times when a Moderate Number of Drivers were Initialized in the Simulation

Table 21 – Average Parking Times and Variances produced in One Destination Scenario when a Moderate Number of Drivers were Initialized in the Simulation

Algorithm	No. of Simulations Executed	Average Parking Time (sec)	Standard Deviation of Average Parking Time (sec)	Ratio of Parking Time to Random Parking Time
Random	200	119.585	7.698	1.000
Greedy	200	165.073	0.000	1.380
P-R	200	118.689	6.798	0.993
P-R (REG-NEG)	200	118.706	6.591	0.993
P-G	200	118.036	6.268	0.987
P-G (REG-NEG)	200	118.417	6.748	0.990
P-R (INS-NEG)	200	108.834	3.892	0.910
PL-R	200	112.511	4.956	0.941
PL-R (REG-NEG)	200	112.995	5.472	0.945
PL-G	200	112.895	4.771	0.944
PL-G (REG-NEG)	200	113.068	5.657	0.946
PL-R (INS-NEG)	200	108.098	4.300	0.904

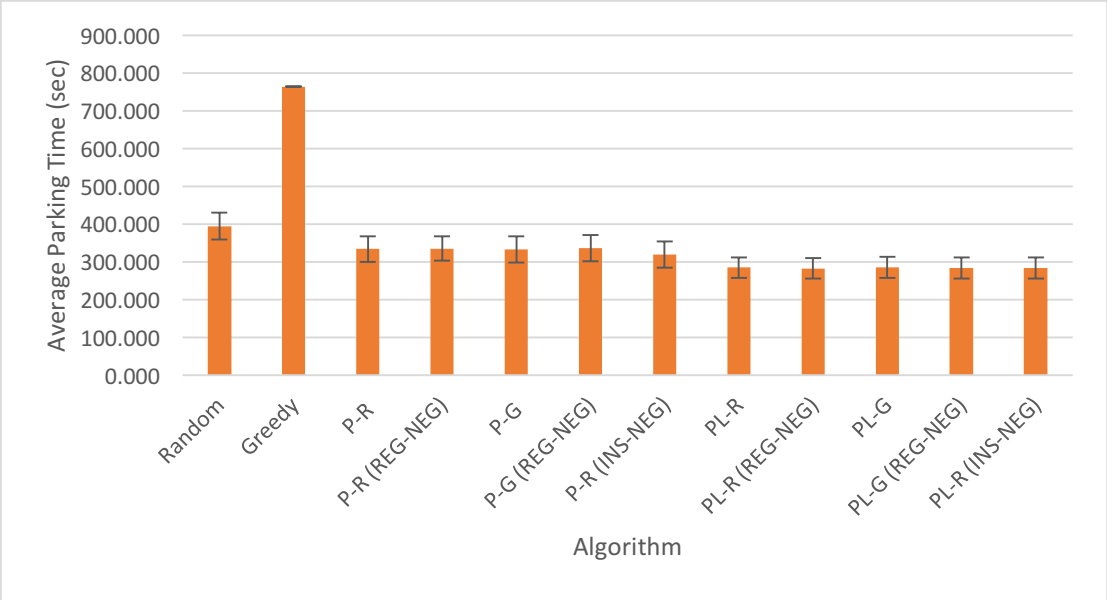


Figure 29 – Average Parking Times when a Large Number of Drivers were Initialized in the Simulation

Table 22 – Average Parking Times and Variances produced in One Destination Scenario when a Large Number of Drivers were Initialized in the Simulation

Algorithm	No. of Simulations Executed	Average Parking Time (sec)	Standard Deviation of Average Parking Time (sec)	Ratio of Parking Time to Random Parking Time
Random	200	394.525	35.355	1.000
Greedy	200	763.103	0.000	1.934
P-R	200	333.902	33.859	0.846
P-R (REG-NEG)	200	334.848	32.361	0.849
P-G	200	333.068	34.423	0.844
P-G (REG-NEG)	200	335.965	34.338	0.852
P-R (INS-NEG)	200	319.463	35.172	0.810
PL-R	200	284.978	27.280	0.722
PL-R (REG-NEG)	200	282.700	26.978	0.717
PL-G	200	284.855	27.726	0.722
PL-G (REG-NEG)	200	284.220	27.954	0.720
PL-R (INS-NEG)	200	284.300	28.203	0.721

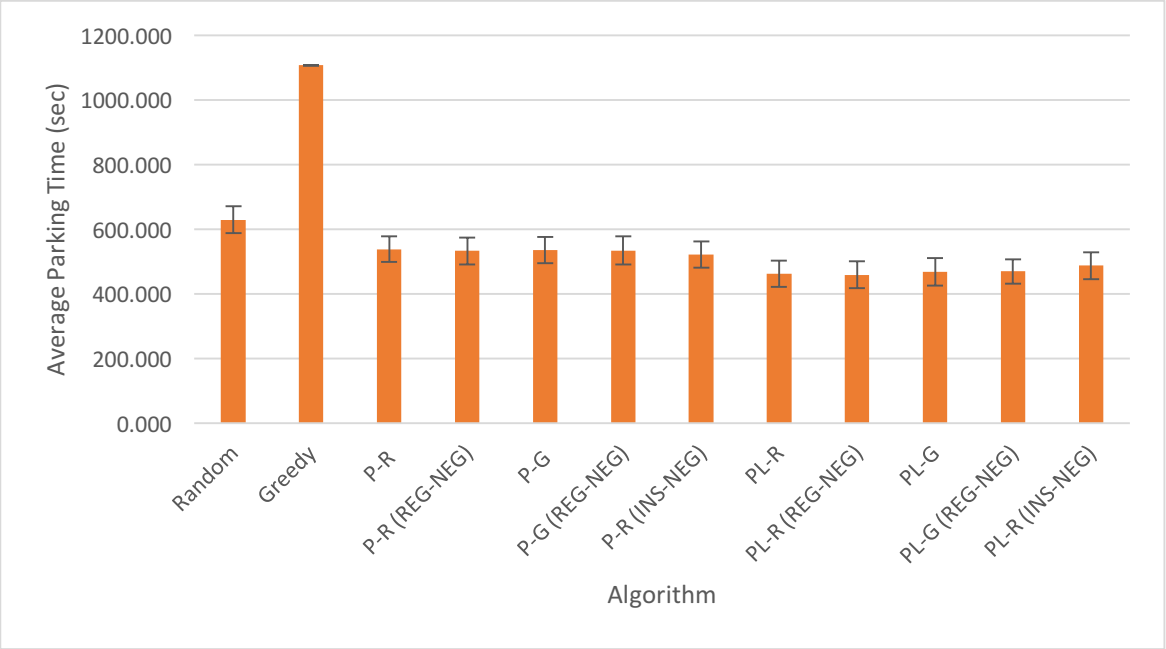


Figure 30 – Average Parking Times when a very Large Number of Drivers were Initialized in the Simulation

Table 23 – Average Parking Times and Variances produced in One Destination Scenario when a very Large Number of Drivers were Initialized in the Simulation

Algorithm	No. of Simulations Executed	Average Parking Time (sec)	Standard Deviation of Average Parking Time (sec)	Ratio of Parking Time to Random Parking Time
Random	200	629.317	40.834	1.000
Greedy	200	1106.983	0.000	1.759
P-R	200	537.855	40.045	0.855
P-R (REG-NEG)	200	532.702	42.207	0.846
P-G	200	535.546	40.443	0.851
P-G (REG-NEG)	200	534.277	44.263	0.849
P-R (INS-NEG)	200	521.978	40.722	0.829
PL-R	200	461.686	40.921	0.734
PL-R (REG-NEG)	200	459.022	41.223	0.729
PL-G	200	467.526	42.173	0.743
PL-G (REG-NEG)	200	469.478	37.051	0.746
PL-R (INS-NEG)	200	487.432	41.487	0.775

The algorithms, for the most part, follow the same trends derived in the scenario in which all the drivers drove to one destination when parking demand is varied. When parking demand is low, the average parking time of the Greedy algorithm is relatively lower and closer to that produced by the other algorithms. The average parking times produced by the HoneyPark variations are comparable to each other. When the number of drivers increases in Figure 28 and Figure 29, differences in performance are observed within the HoneyPark variations. Like the One Destination Scenario, Parked-Leave Advert HoneyPark algorithms perform significantly better than Parked Adverts HoneyPark algorithms for the same reason explained in Section 5.1.1. Likewise, algorithms that use instantaneous negative feedback stood out as the most efficient algorithms. When parking demand increased further in Figure 30, the Parked-Advert HoneyPark with instantaneous negative feedback performed better than other Parked-Advert HoneyPark algorithm while Parked-Leave Algorithm that used instantaneous negative feedback produced similar times as other Parked-Leave Algorithms. This same trend is observed in the One Destination Scenario case in which the lot-to-bee ratio is 0.75.

The algorithms, for the most part, follow the same trends derived from the One Destination Scenario in Section 5.1.1. Therefore, it seems that sending drivers to a mesh destination does not significantly change algorithmic behavior and performance and the same principles derived in Section 5.1.1 still apply. The only difference noticed between the two scenario is that one needs to vary the number of drivers more significantly in a One Mesh Destination Scenario than in a One Destination Scenario to observe the changes that occur when the amount of parking demand is varied. This is understandable as a mesh destination is larger and typically encompasses more parking supply than a single destination. Therefore, one must change the absolute number of drivers by a larger amount to see any impact on algorithmic performance caused by the relative

change between parking demand and supply. As such, it seems the relative amount of parking demand and supply is one of the main factors affecting algorithmic performance rather than whether the drivers are looking for parking in a single or mesh destination.

5.2 Algorithmic Performance with Varying Errand Times

In this section, the performance of each algorithm was evaluated when the drivers in the simulation were given errand times within a certain range. As Section 5.1 shows that the results are dependent on the amount of relative demand for parking, the experiment was conducted using three lot-to-bee ratios: when the lot-to-bee ratio was equal to two as seen in Figure 31, lot-to-bee was one in Figure 32 and when the lot-to-bee ratio was 0.75 in Figure 33. Table 24, Table 25 and Table 26 expresses the same data in numeric form along with the number of simulations performed for each algorithm. Note that this section was conducted using the same set-up as the One Destination scenario. Therefore, only 70 simulation runs are needed to obtain a consistent average parking time value.

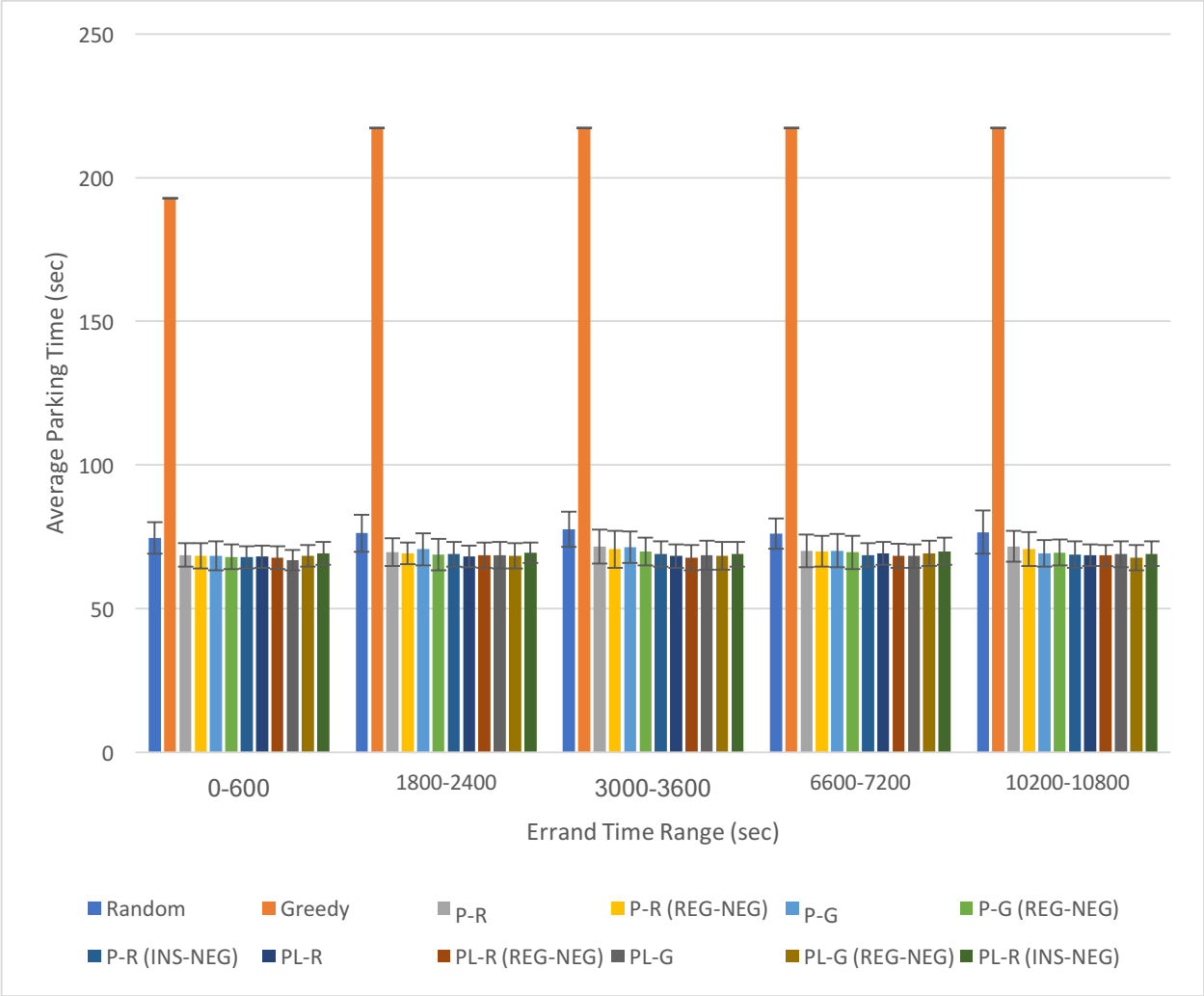


Figure 31 – Average Parking Times when the Errand Time Length was Varied (Lot-to-Bee Ratio = 2)

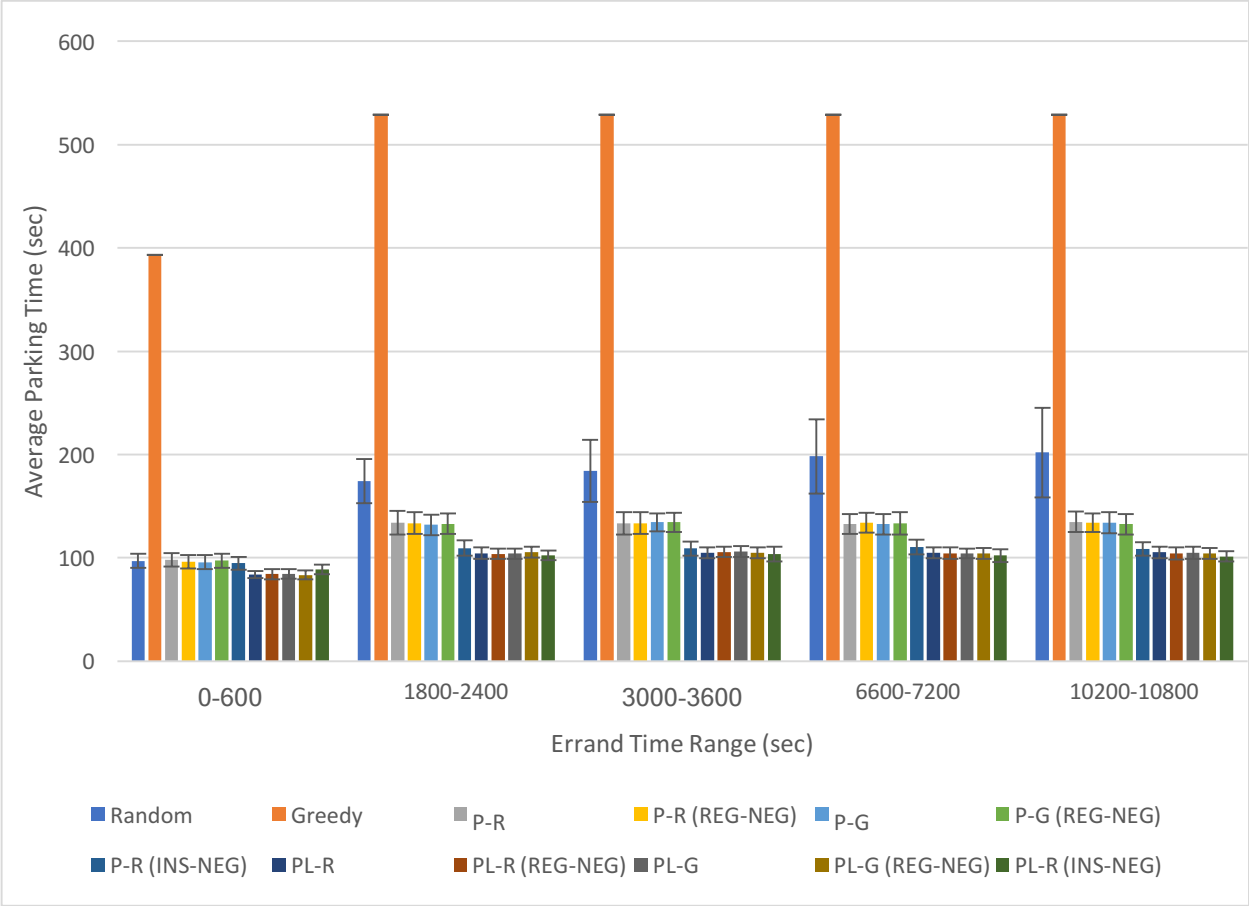


Figure 32 – Average Parking Times when the Errand Time Length was Varied (Lot-to-Bee Ratio = 1)

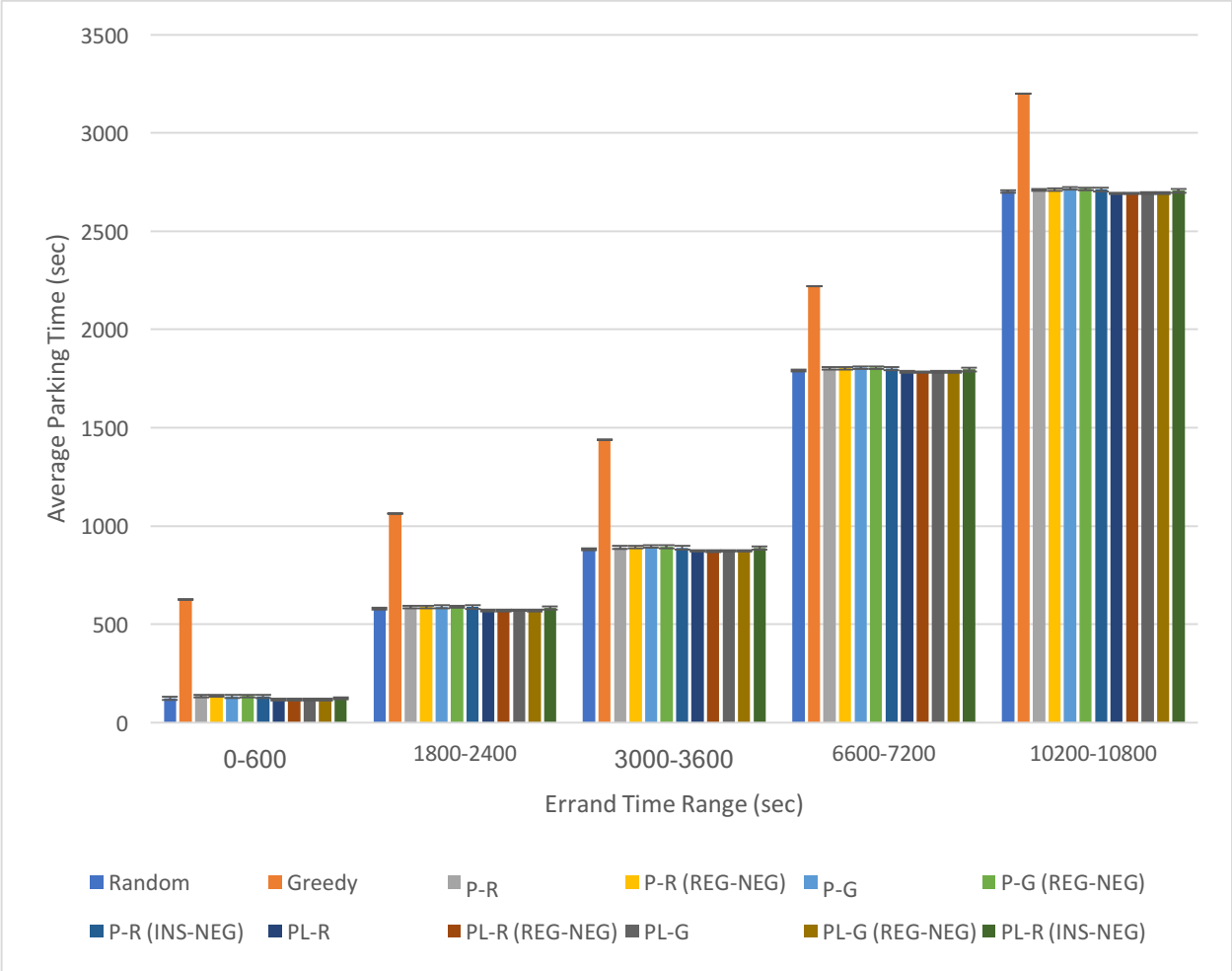


Figure 33 – Average Parking Times when the Errand Time Length was Varied (Lot-to-Bee Ratio = 0.75)

Table 24 – Average Parking Times and Variances produced at a Variety of Errand Times when Lot-to-Bee Ratio = 2

	Errand Time Range of 0-600 sec			Errand Time Range of 1800-2400 sec			Errand Time Range of 3000-3600 sec			Errand Time Range of 6600-7200 sec			Errand Time Range of 10200-10800 sec		
Algorithm	No. of Simulations Executed	Average Parking Time (sec)	Standard Deviation of Average Parking Time (sec)	No. of Simulations Executed	Average Parking Time (sec)	Standard Deviation of Average Parking Time (sec)	No. of Simulations Executed	Average Parking Time (sec)	Standard Deviation of Average Parking Time (sec)	No. of Simulations Executed	Average Parking Time (sec)	Standard Deviation of Average Parking Time (sec)	No. of Simulations Executed	Average Parking Time (sec)	Standard Deviation of Average Parking Time (sec)
Random	70	74.643	5.424	70	76.262	6.432	70	77.663	6.091	70	76.101	5.201	70	76.554	7.513
Greedy	70	192.791	0.000	70	217.269	0.000	70	217.269	0.000	70	217.269	0.000	70	217.269	0.000
P-R	70	68.660	4.021	70	69.683	4.830	70	71.622	5.899	70	70.183	5.678	70	71.691	5.353
P-R (REG-NEG)	70	68.304	4.366	70	69.194	3.787	70	70.647	6.417	70	69.961	5.372	70	70.702	5.957
P-G	70	68.373	5.095	70	70.633	5.496	70	71.351	5.467	70	70.171	5.864	70	69.148	4.626
P-G (REG-NEG)	70	68.002	4.354	70	68.861	5.515	70	69.844	4.853	70	69.567	5.856	70	69.492	4.551
P-R (INS-NEG)	70	67.843	3.796	70	68.936	4.280	70	69.085	4.413	70	68.652	4.087	70	68.840	4.637
PL-R	70	68.080	3.789	70	68.101	3.785	70	68.279	4.015	70	69.271	4.013	70	68.533	3.764
PL-R (REG-NEG)	70	67.719	4.046	70	68.636	4.379	70	67.738	4.336	70	68.325	4.188	70	68.503	3.698
PL-G	70	66.875	3.556	70	68.551	4.685	70	68.614	5.026	70	68.332	4.044	70	68.933	4.503
PL-G (REG-NEG)	70	68.268	3.754	70	68.337	4.394	70	68.387	4.825	70	69.235	4.351	70	67.772	4.355
PL-R (INS-NEG)	70	69.284	3.988	70	69.461	3.617	70	68.946	4.240	70	69.925	4.720	70	69.108	4.360

Table 25 – Average Parking Times and Variances produced at a Variety of Errand Times when Lot-to-Bee Ratio = 1

	Errand Time Range of 0-600 sec			Errand Time Range of 1800-2400 sec			Errand Time Range of 3000-3600 sec			Errand Time Range of 6600-7200 sec			Errand Time Range of 10200-10800 sec		
Algorithm	No. of Simulations Executed	Average Parking Time (sec)	Standard Deviation of Average Parking Time (sec)	No. of Simulations Executed	Average Parking Time (sec)	Standard Deviation of Average Parking Time (sec)	No. of Simulations Executed	Average Parking Time (sec)	Standard Deviation of Average Parking Time (sec)	No. of Simulations Executed	Average Parking Time (sec)	Standard Deviation of Average Parking Time (sec)	No. of Simulations Executed	Average Parking Time (sec)	Standard Deviation of Average Parking Time (sec)
Random	70	96.968	6.692	70	174.204	21.646	70	184.209	30.022	70	198.105	35.928	70	201.825	43.141
Greedy	70	393.414	0.000	70	529.165	0.000	70	529.165	0.000	70	529.165	0.000	70	529.165	0.000
P-R	70	97.939	6.415	70	133.997	11.266	70	133.349	10.536	70	132.723	9.869	70	134.782	10.092
P-R (REG-NEG)	70	96.183	6.720	70	133.457	10.544	70	133.424	10.496	70	134.054	9.404	70	133.767	9.023
P-G	70	95.794	6.782	70	132.060	9.890	70	134.362	8.718	70	132.426	9.927	70	133.940	10.437
P-G (REG-NEG)	70	97.250	6.798	70	132.927	9.757	70	134.353	9.411	70	133.379	10.827	70	132.456	10.049
P-R (INS-NEG)	70	94.660	6.334	70	109.287	7.500	70	109.098	6.752	70	110.253	7.031	70	108.498	6.515
PL-R	70	83.770	3.528	70	104.339	5.542	70	104.948	5.462	70	104.798	5.019	70	105.256	5.727
PL-R (REG-NEG)	70	84.097	4.665	70	103.797	4.930	70	105.652	4.970	70	104.408	5.397	70	103.966	5.920
PL-G	70	84.360	4.577	70	103.935	4.980	70	106.003	5.422	70	104.177	4.604	70	104.679	5.997
PL-G (REG-NEG)	70	83.367	4.289	70	105.392	5.479	70	104.837	5.364	70	104.187	5.323	70	104.258	5.148
PL-R (INS-NEG)	70	88.596	4.773	70	102.552	4.754	70	103.493	6.987	70	102.193	6.062	70	101.434	5.089

Table 26 – Average Parking Times and Variances produced at a Variety of Errand Times when Lot-to-Bee Ratio = 0.75

	Errand Time Range of 0-600 sec			Errand Time Range of 1800-2400 sec			Errand Time Range of 3000-3600 sec			Errand Time Range of 6600-7200 sec			Errand Time Range of 10200-10800 sec		
Algorithm	No. of Simulations Executed	Average Parking Time (sec)	Standard Deviation of Average Parking Time (sec)	No. of Simulations Executed	Average Parking Time (sec)	Standard Deviation of Average Parking Time (sec)	No. of Simulations Executed	Average Parking Time (sec)	Standard Deviation of Average Parking Time (sec)	No. of Simulations Executed	Average Parking Time (sec)	Standard Deviation of Average Parking Time (sec)	No. of Simulations Executed	Average Parking Time (sec)	Standard Deviation of Average Parking Time (sec)
Random	70	123.428	7.532	70	579.820	6.285	70	881.703	6.226	70	1790.950	6.157	70	2702.054	7.186
Greedy	70	625.916	0.000	70	1063.438	0.000	70	1440.635	0.000	70	2219.270	0.000	70	3198.753	0.000
P-R	70	135.896	6.663	70	587.441	5.438	70	891.515	7.148	70	1801.933	6.728	70	2710.633	5.635
P-R (REG-NEG)	70	136.232	6.228	70	587.934	6.585	70	892.559	5.963	70	1802.518	7.606	70	2711.351	6.640
P-G	70	133.445	7.753	70	589.891	7.043	70	895.549	7.142	70	1805.898	6.542	70	2717.015	6.542
P-G (REG-NEG)	70	135.291	6.876	70	589.125	6.203	70	893.793	7.327	70	1805.992	6.737	70	2716.055	6.625
PL-R	70	117.159	4.565	70	570.188	3.434	70	872.837	3.905	70	1784.514	4.569	70	2692.826	4.033
PL-R (REG-NEG)	70	116.692	4.600	70	569.413	3.877	70	873.409	4.606	70	1783.467	3.600	70	2693.051	3.972
PL-G	70	116.312	4.980	70	570.993	3.978	70	874.627	3.682	70	1784.598	4.110	70	2693.896	3.575
PL-G (REG-NEG)	70	116.228	4.585	70	570.308	4.104	70	873.399	4.069	70	1784.580	3.386	70	2694.406	4.135
P-R (INS-NEG)	70	133.098	8.152	70	588.290	9.734	70	889.679	8.538	70	1800.307	7.567	70	2711.934	8.618
PL-R (INS-NEG)	70	122.982	5.169	70	582.606	7.399	70	887.327	7.926	70	1795.911	8.614	70	2706.373	9.844

For lot-to-bee ratios that are equal or less than one, parking times are quicker at shorter errand times. Simulation results show that when errand times are shorter, there are more vacant parking spots available throughout the simulation as drivers are only occupying their parking spot for a smaller period of time which makes it easier for drivers to find parking. Likewise, at larger errand times, parked drivers are occupying parking spaces for a longer period of time and consequently, there are less vacant parking spaces available which leads to an increase in parking times. When parking demand is relatively low, that is when the lot-to-bee ratio is more than one, there is barely any difference in average parking time as the errand time length is varied. This is because there are enough parking spaces for all the drivers, even if each driver were to stay in their parking space for the entire simulation. Therefore, drivers can still find parking independent of errand time length.

When one compares the individual performance of each HoneyPark variation to other algorithms, Figure 31, Figure 32 and Figure 33 show that the trends produced at all lot-to-bee ratios at nearly all errand time lengths follow the basic patterns covered in Section 5.1. When the lot-to-bee ratio is greater than one, there is no significant difference in performance between the different HoneyPark variations. When the lot-to-bee ratio is equal to one, Parked-Leave Advert algorithms perform better than their Parked Advert counterparts. Algorithms that used instantaneous negative feedback consistently performed better than those that did not. When the lot-to-bee ratio is less than one, Parked Adverts performed worse than the Random algorithms and the use of instantaneous feedback did not necessarily result in an improvement in algorithm performance. As such, it seems that a change in errand time length did not have much impact on the performance rankings of the HoneyPark variations.

Table 27 – Ratio between the Average Parking Times of the HoneyPark Algorithm to the Random Algorithm

Lot-to-Bee Ratio	Algorithm	Errand Time Range				
		0-600	1800-2400	3000-3600	6600-7200	10200-10800
	P-R	1.101	1.013	1.011	1.006	1.003
	P-R (REG-NEG)	1.104	1.014	1.012	1.006	1.003
	P-G	1.081	1.017	1.016	1.008	1.006
	P-G (REG-NEG)	1.096	1.016	1.014	1.008	1.005
	PL-R	0.949	0.983	0.990	0.996	0.997
	PL-R (REG-NEG)	0.945	0.982	0.991	0.996	0.997
	PL-G	0.942	0.985	0.992	0.996	0.997
	PL-G (REG-NEG)	0.942	0.984	0.991	0.996	0.997
	P-R (INS-NEG)	1.078	1.015	1.009	1.005	1.004
0.75	PL-R (INS-NEG)	0.996	1.005	1.006	1.003	1.002
	P-R	1.010	0.769	0.724	0.670	0.668
	P-R (REG-NEG)	0.992	0.766	0.724	0.677	0.663
	P-G	0.988	0.758	0.729	0.668	0.664
	P-G (REG-NEG)	1.003	0.763	0.729	0.673	0.656
	PL-R	0.864	0.599	0.570	0.529	0.522
	PL-R (REG-NEG)	0.867	0.596	0.574	0.527	0.515
	PL-G	0.870	0.597	0.575	0.526	0.519
	PL-G (REG-NEG)	0.860	0.605	0.569	0.526	0.517
	P-R (INS-NEG)	0.976	0.627	0.592	0.557	0.538
1	PL-R (INS-NEG)	0.914	0.589	0.562	0.516	0.503
	P-R	0.920	0.914	0.922	0.922	0.936
	P-R (REG-NEG)	0.915	0.907	0.910	0.919	0.924
	P-G	0.916	0.926	0.919	0.922	0.903
	P-G (REG-NEG)	0.911	0.903	0.899	0.914	0.908
	PL-R	0.912	0.893	0.879	0.910	0.895
	PL-R (REG-NEG)	0.907	0.900	0.872	0.898	0.895
	PL-G	0.896	0.899	0.883	0.898	0.900
	PL-G (REG-NEG)	0.915	0.896	0.881	0.910	0.885
	P-R (INS-NEG)	0.909	0.904	0.890	0.902	0.899
2	PL-R (INS-NEG)	0.928	0.911	0.888	0.919	0.903

However, one can see a difference in how well the HoneyPark algorithms perform compared to the Random algorithm as the errand time is varied. At a lot-to-bee ratio of one, Table 27 shows that the ratio between the average parking time of HoneyPark and Random algorithms as errand time is varied, which decreases as errand time increases. This indicates that the HoneyPark algorithm performance improves relative to that of the Random algorithm as the errand time is lengthened. This is because as errand time increases, drivers are occupying parking lots for a longer period of time and there are less parking spaces available. In this case, the Random algorithm will perform worse as it is sending drivers to random lots, making it very likely that it will send a driver to a less profitable lot. On the other hand, the HoneyPark algorithm considers the profitability of the lot, making it more likely that the driver will find a parking space.

But at a lot-to-bee ratio of 0.75, Table 27 shows that the ratio between the average parking times produced by the Parked-Leave-Advert algorithms and Random algorithm doesn't vary too much as errand time length is varied because demand exceeds supply. It is difficult to find parking regardless of whether the driver uses the Random or the Parked-Leave algorithm. However, one can see that the ratios converges to one as errand time is increased. This is because the parking lots are more occupied at longer errand times, making it even more difficult for any driver to find parking regardless of the parking algorithm he or she uses. Therefore, the parking times produced by the Random and HoneyPark algorithms become more similar.

At a lot-to-bee ratio of two, Table 27 shows that the ratio doesn't vary much either as the number of parking spaces exceeds the number of drivers looking for parking. Therefore, a driver is likely to find a parking space at any lot it explores regardless of whether he or she uses the Random or HoneyPark algorithm.

The only scenario that deviates from the trends described in Section 5.1 is the case in which errand times are exceptionally short (0-600 seconds) at a lot-to-bee ratio of one. Figure 32 shows that the average parking times produced by Parked Advert algorithms are comparable to that produced by the Random algorithm. This trend can be analyzed by comparing Figure 34 with Figure 35 and Figure 36, which show the number of lots each driver searched before successfully finding a parking space for the Random and Parked-Advert algorithms. This deviates from the trend that Parked Advert algorithms usually perform better than the Random algorithm as observed in Section 5.1 and at other errand times. Note that graphs are arranged that it shows the data for each driver in the order of the time at which they initiated the search for parking.

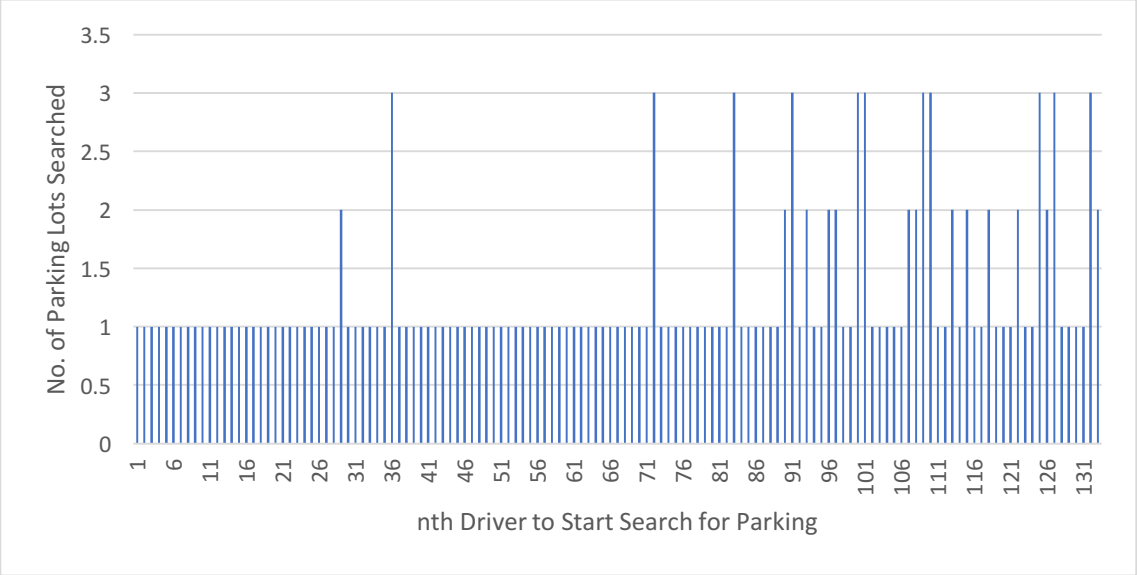


Figure 34 – Number of Parking Lots Searched by Each Driver in the Simulation when the Random Algorithm was used

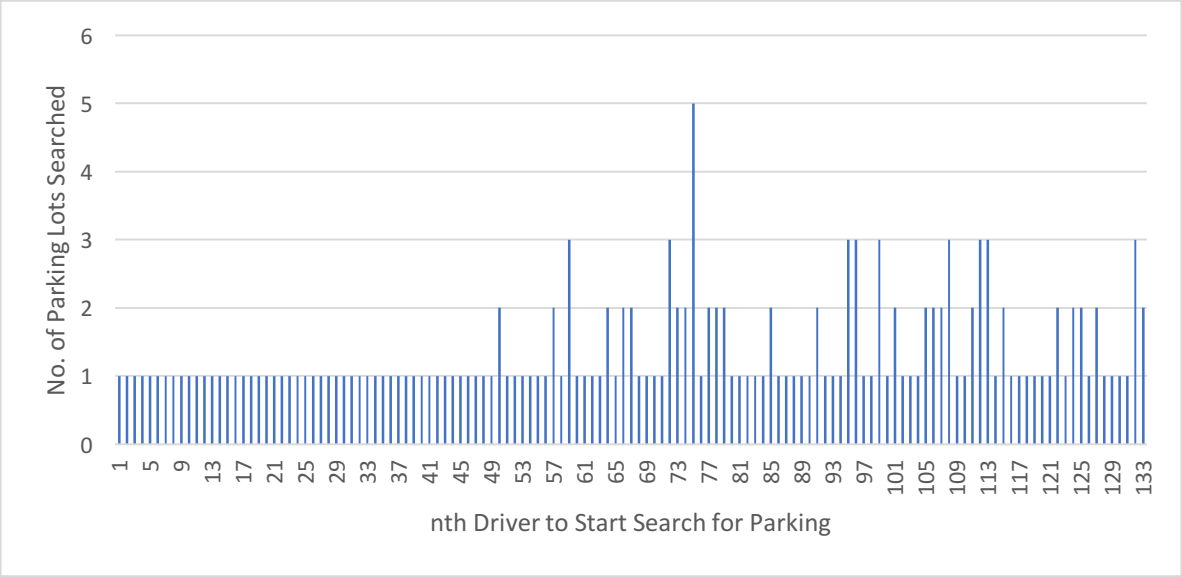


Figure 35 – Number of Parking Lots Searched by Each Driver in the Simulation when the Parked-Advert Random-Scouting Algorithm was used

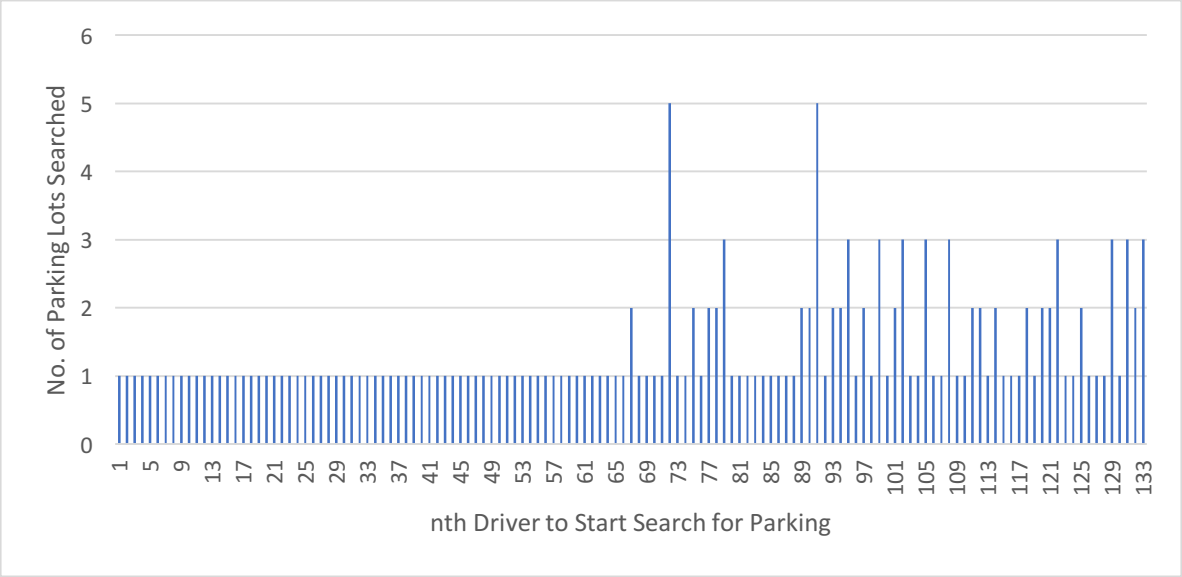


Figure 36 – Number of Parking Lots Searched by Each Driver in the Simulation when the Parked-Advert Greedy-Scouting Algorithm was used

As one can see, the number of lots for the Random algorithm fluctuates throughout the entire simulation and does not follow a certain trend, which is understandable as the Random algorithm sends the drivers to random parking lots. For the Parked-Advert Algorithm, the number of lots searched increases greatly for drivers who initiate their parking search later in the simulation. As the lots are generally more profitable due to the short errand times, it is very likely that the drivers can find a parking space regardless of which parking lot was visited on the condition that they are not especially overcrowded. Therefore, it is possible for drivers using the Random algorithm to find parking in a shorter amount of time in this situation. Figure 34 shows that the majority, around 70% to be exact, of drivers find a parking space on their first try when they use the Random algorithm. On the other hand, algorithms that used only Parked adverts tend to send drivers to what is a small group of profitable lots. This method is initially effective as Figure 35 and Figure 36 show that earlier drivers are able to find parking on their first try. As time elapses, Parked Advert algorithms continue to bombard the small group of lots with more drivers, saturating them and making it harder for subsequent drivers to find parking on their first try. This is reflected in the graphs, where drivers that come later in the simulations have to search more lots before they are able to successfully park. Unlike the Random algorithm, the Parked Advert algorithms aren't flexible enough to take advantage of the fact that most of the parking lots are profitable when errand times are short and continue to send most, if not all, of the drivers to a limited selection of 'profitable' lots.

When errand times are short, the performance of instantaneous negative feedback adverts also deviates from the usual trend described in Section 5.1. Figure 32 shows that for lot-to-bee ratio of 1, using instantaneous negative feedback did not have any impact on performance when drivers only looked at Parked adverts, but was less efficient when driver considered both Parked

and Leave adverts. As mentioned previously, the performance of Parked advert algorithms tends to be poor when errand times are very short because they consider a small select group of lots as profitable. Therefore, the inclusion of an instantaneous negative feedback mechanism will result in a substantial improvement in algorithmic performance. However, this is not the case for Parked-Leave adverts, who consider a bigger group of profitable and are thus more accurate in determining which lots are profitable. A real-time observation of the simulation shows that instantaneous negative feedback results in longer times in this case because it immediately redirects drivers away from parking lots that become full. As errand times are short, parked drivers will not occupy their parking space for a long time and the probability is high that a parking space will become vacant while the driver is on his or her way to the parking lot. Even though a lot may have filled up, it will on average take a shorter time to find parking if the driver continued to pursue the lot than it is to redirect the driver to another parking lot.

5.3 Algorithmic Performance with Varying Levels of Parking Congestion

Figure 37, Figure 38 and Figure 39 show how algorithm performance is affected by the amount of parking congestion at the lot-to-bee ratios of 0.75, one and two respectively. Likewise, Table 28, Table 29 and Table 30 show the same data in numeric form along with the number of simulations performed for each algorithm. The amount of parking congestion was controlled by changing the time it takes to initiate all the drivers in the simulation. For example, if all the drivers enter the simulation in a shorter amount of time, there would be more congestion as there will be more drivers searching for parking at the same time. Likewise, if all the drivers entered the simulation within a longer period of time, there will be less drivers present in the simulation simultaneously, reducing the amount of congestion. Similar to the Errand Time simulations in

Section 5.2, this experiment was also conducted at lot-to-bee ratios of 0.75, 1 and 2 to observe how the algorithm fares in varying levels of parking demand.

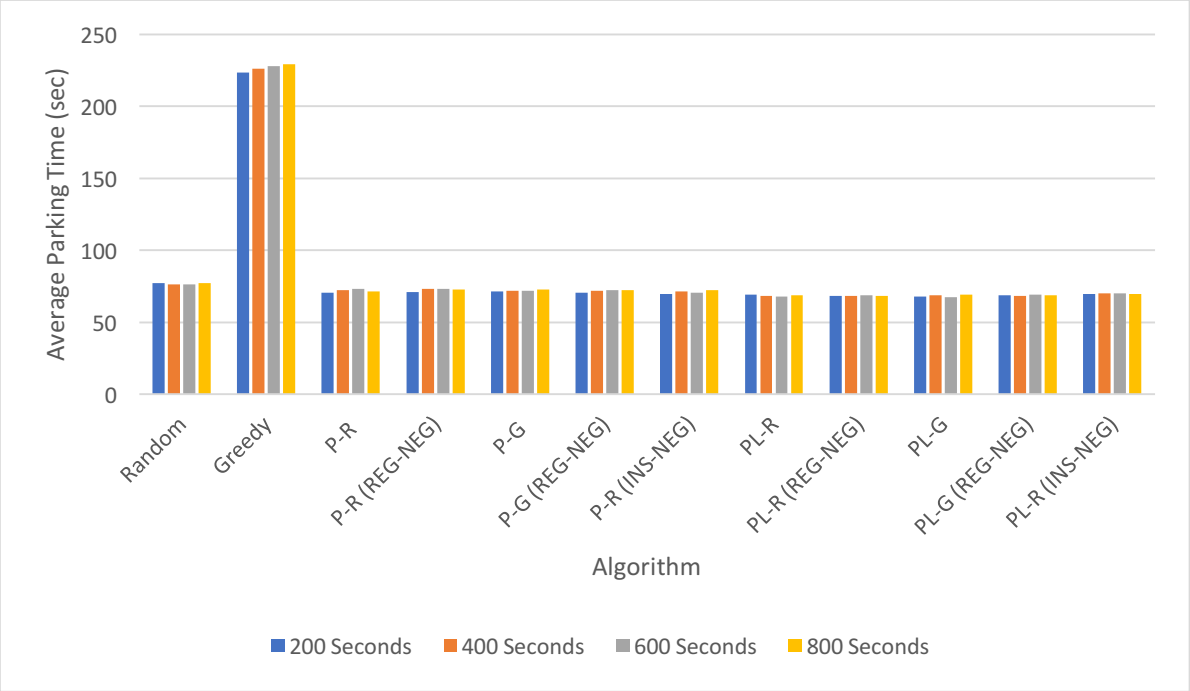


Figure 37 – Average Parking Times when All Drivers Enter the Simulation in 200 Seconds, 400 Seconds, 600 seconds and 800 Seconds when the Lot-to-Bee Ratio = 2

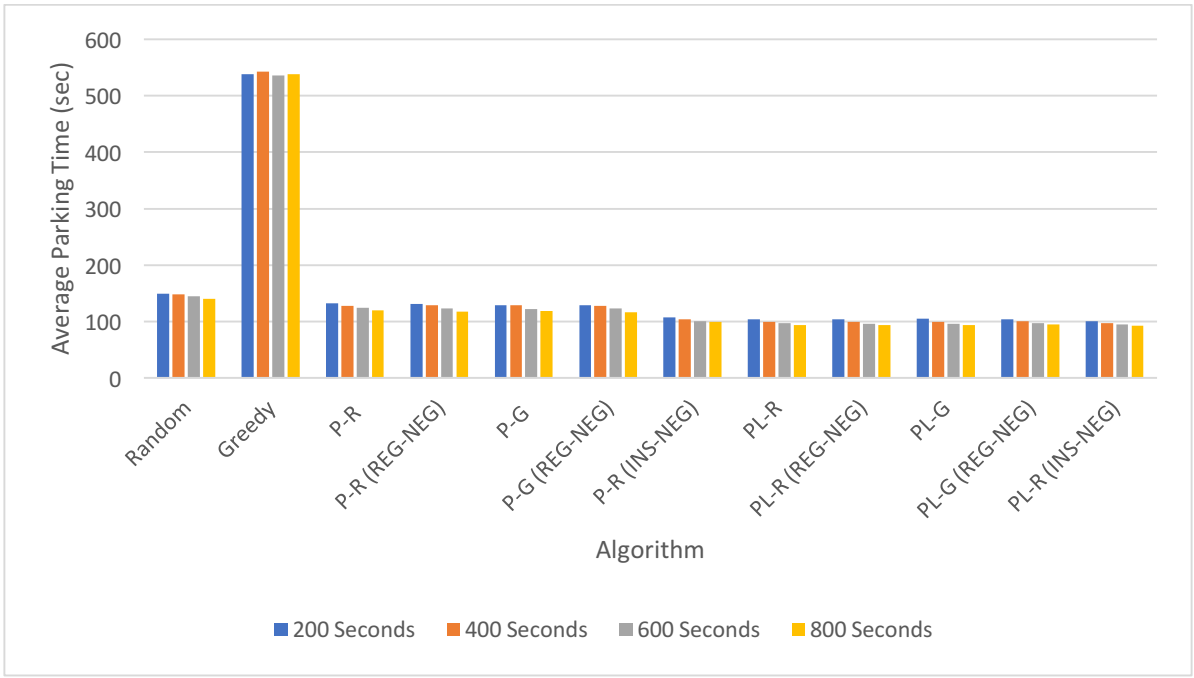


Figure 38 – Average Parking Times when All Drivers Enter the Simulation in 200 Seconds, 400 Seconds, 600 seconds and 800 Seconds when the Lot-to-Bee Ratio = 1

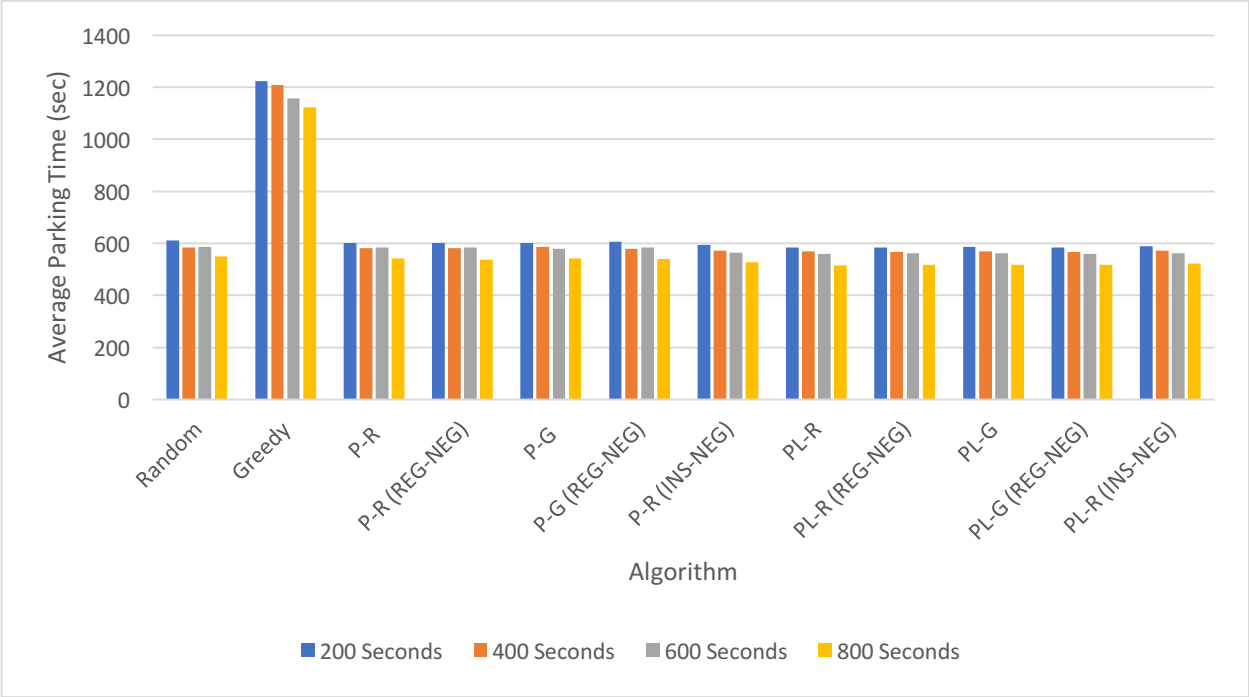


Figure 39 – Average Parking Times when All Drivers Enter the Simulation in 200 Seconds, 400 Seconds, 600 seconds and 800 Seconds when the Lot-to-Bee Ratio = 0.75

Table 28 – Average Parking Times and Variances at Varying Congestion Levels when Bee-to-lot Ratio = 2

	Algorithm	No. of Simulations Executed	Average Parking Time (sec)	Standard Deviation of Average Parking Time (sec)	Ratio of Parking Time to Random Parking Time
200 Seconds for All Drivers to Enter Simulation	Random	70	77.168	5.735	1.000
	Greedy	70	223.448	0.000	2.896
	P-R	70	70.591	6.384	0.915
	P-R (REG-NEG)	70	70.846	5.245	0.918
	P-G	70	71.553	5.420	0.927
	P-G (REG-NEG)	70	70.451	5.257	0.913
	P-R (INS-NEG)	70	69.641	4.674	0.902
	PL-R	70	69.092	4.003	0.895
	PL-R (REG-NEG)	70	68.468	4.727	0.887
	PL-G	70	67.730	3.637	0.878
	PL-G (REG-NEG)	70	68.644	4.076	0.890
	PL-R (INS-NEG)	70	69.887	3.814	0.906
400 Seconds for All Drivers to Enter Simulation	Random	70	76.505	6.638	1.000
	Greedy	70	226.254	0.000	2.957
	P-R	70	72.382	5.146	0.946
	P-R (REG-NEG)	70	73.071	4.901	0.955
	P-G	70	71.896	5.837	0.940
	P-G (REG-NEG)	70	71.776	6.096	0.938
	P-R (INS-NEG)	70	71.456	4.856	0.934
	PL-R	70	68.542	3.958	0.896
	PL-R (REG-NEG)	70	68.496	3.760	0.895
	PL-G	70	69.021	3.941	0.902
	PL-G (REG-NEG)	70	68.403	4.030	0.894
	PL-R (INS-NEG)	70	70.138	4.153	0.917
600 Seconds for All Drivers to Enter Simulation	Random	70	76.231	5.907	1.000
	Greedy	70	228.045	0.000	2.991
	P-R	70	73.154	5.671	0.960
	P-R (REG-NEG)	70	73.257	5.857	0.961
	P-G	70	71.797	5.819	0.942
	P-G (REG-NEG)	70	72.594	5.588	0.952
	P-R (INS-NEG)	70	70.823	4.455	0.929
	PL-R	70	68.115	3.818	0.894
	PL-R (REG-NEG)	70	68.766	3.768	0.902
	PL-G	70	67.285	3.748	0.883
	PL-G (REG-NEG)	70	69.048	3.867	0.906
	PL-R (INS-NEG)	70	70.184	4.030	0.921
800 Seconds for All Drivers to Enter Simulation	Random	70	77.288	6.242	1.000
	Greedy	70	229.463	0.000	2.969
	P-R	70	71.465	5.008	0.925
	P-R (REG-NEG)	70	72.713	4.941	0.941
	P-G	70	72.798	5.531	0.942
	P-G (REG-NEG)	70	72.406	5.290	0.937
	P-R (INS-NEG)	70	72.462	4.742	0.938
	PL-R	70	68.809	3.250	0.890
	PL-R (REG-NEG)	70	68.542	3.853	0.887
	PL-G	70	69.158	4.218	0.895
	PL-G (REG-NEG)	70	68.630	3.959	0.888
	PL-R (INS-NEG)	70	69.615	4.310	0.901

Table 29 – Average Parking Times and Variances at Varying Congestion Levels when Bee-to-lot Ratio = 1

	Algorithm	No. of Simulations Executed	Average Parking Time (sec)	Standard Deviation of Average Parking Time (sec)	Ratio of Parking Time to Random Parking Time
200 Seconds for All Drivers to Enter Simulation	Random	70	149.001	17.029	1.000
	Greedy	70	538.015	0.000	3.611
	P-R	70	131.901	9.287	0.885
	P-R (REG-NEG)	70	131.625	11.136	0.883
	P-G	70	129.514	9.365	0.869
	P-G (REG-NEG)	70	128.814	8.174	0.865
	P-R (INS-NEG)	70	108.039	6.230	0.725
	PL-R	70	104.038	5.269	0.698
	PL-R (REG-NEG)	70	103.943	5.598	0.698
	PL-G	70	104.869	5.332	0.704
	PL-G (REG-NEG)	70	103.612	4.837	0.695
	PL-R (INS-NEG)	70	100.518	5.738	0.675
400 Seconds for All Drivers to Enter Simulation	Random	70	148.343	18.565	1.000
	Greedy	70	542.865	0.000	3.660
	P-R	70	127.925	9.474	0.862
	P-R (REG-NEG)	70	129.016	10.986	0.870
	P-G	70	128.628	8.787	0.867
	P-G (REG-NEG)	70	127.388	10.887	0.859
	P-R (INS-NEG)	70	103.887	6.474	0.700
	PL-R	70	99.768	5.495	0.673
	PL-R (REG-NEG)	70	99.314	5.590	0.669
	PL-G	70	100.083	5.225	0.675
	PL-G (REG-NEG)	70	100.547	4.615	0.678
	PL-R (INS-NEG)	70	96.950	4.643	0.654
600 Seconds for All Drivers to Enter Simulation	Random	70	145.382	16.095	1.000
	Greedy	70	535.308	0.000	3.682
	P-R	70	123.959	9.731	0.853
	P-R (REG-NEG)	70	123.102	9.890	0.847
	P-G	70	121.926	9.123	0.839
	P-G (REG-NEG)	70	123.238	8.677	0.848
	P-R (INS-NEG)	70	100.609	5.267	0.692
	PL-R	70	96.929	5.448	0.667
	PL-R (REG-NEG)	70	96.699	4.188	0.665
	PL-G	70	96.153	5.494	0.661
	PL-G (REG-NEG)	70	97.797	4.801	0.673
	PL-R (INS-NEG)	70	94.669	3.551	0.651
800 Seconds for All Drivers to Enter Simulation	Random	70	140.196	14.651	1.000
	Greedy	70	537.571	0.000	3.834
	P-R	70	119.537	8.561	0.853
	P-R (REG-NEG)	70	117.285	10.050	0.837
	P-G	70	119.155	9.122	0.850
	P-G (REG-NEG)	70	116.981	9.087	0.834
	P-R (INS-NEG)	70	99.085	5.682	0.707
	PL-R	70	94.123	3.665	0.671
	PL-R (REG-NEG)	70	94.279	4.650	0.672
	PL-G	70	94.431	4.574	0.674
	PL-G (REG-NEG)	70	95.308	4.331	0.680
	PL-R (INS-NEG)	70	92.926	4.220	0.663

Table 30 – Average Parking Times and Variances at Varying Congestion Levels when Bee-to-lot Ratio = 0.75

	Algorithm	No. of Simulations Executed	Average Parking Time (sec)	Standard Deviation of Average Parking Time (sec)	Ratio of Parking Time to Random Parking Time
200 Seconds for All Drivers to Enter Simulation	Random	70	610.428	24.689	1.000
	Greedy	70	1222.719	0.000	2.003
	P-R	70	600.801	21.067	0.984
	P-R (REG-NEG)	70	600.458	22.468	0.984
	P-G	70	602.496	23.627	0.987
	P-G (REG-NEG)	70	606.586	23.311	0.994
	P-R (INS-NEG)	70	593.286	27.322	0.972
	PL-R	70	583.112	21.540	0.955
	PL-R (REG-NEG)	70	584.466	23.177	0.957
	PL-G	70	585.333	20.797	0.959
	PL-G (REG-NEG)	70	583.597	19.539	0.956
	PL-R (INS-NEG)	70	588.188	23.024	0.964
400 Seconds for All Drivers to Enter Simulation	Random	70	584.582	24.080	1.000
	Greedy	70	1207.730	0.000	2.066
	P-R	70	582.087	21.527	0.996
	P-R (REG-NEG)	70	581.721	19.356	0.995
	P-G	70	586.407	24.954	1.003
	P-G (REG-NEG)	70	580.115	23.420	0.992
	P-R (INS-NEG)	70	572.820	19.530	0.980
	PL-R	70	569.326	19.621	0.974
	PL-R (REG-NEG)	70	567.705	20.435	0.971
	PL-G	70	569.696	18.174	0.975
	PL-G (REG-NEG)	70	567.989	20.145	0.972
	PL-R (INS-NEG)	70	572.587	18.475	0.979
600 Seconds for All Drivers to Enter Simulation	Random	70	585.433	21.996	1.000
	Greedy	70	1157.011	0.000	1.976
	P-R	70	584.567	25.904	0.999
	P-R (REG-NEG)	70	584.321	21.723	0.998
	P-G	70	579.317	29.232	0.990
	P-G (REG-NEG)	70	584.117	23.502	0.998
	P-R (INS-NEG)	70	565.028	19.825	0.965
	PL-R	70	558.585	18.486	0.954
	PL-R (REG-NEG)	70	562.044	22.795	0.960
	PL-G	70	560.761	22.943	0.958
	PL-G (REG-NEG)	70	560.010	21.116	0.957
	PL-R (INS-NEG)	70	562.056	19.666	0.960
800 Seconds for All Drivers to Enter Simulation	Random	70	548.798	17.463	1.000
	Greedy	70	1122.691	0.000	2.046
	P-R	70	542.939	21.357	0.989
	P-R (REG-NEG)	70	538.064	21.988	0.980
	P-G	70	542.995	20.994	0.989
	P-G (REG-NEG)	70	540.853	19.404	0.986
	P-R (INS-NEG)	70	526.983	18.779	0.960
	PL-R	70	515.050	16.192	0.939
	PL-R (REG-NEG)	70	517.292	15.827	0.943
	PL-G	70	517.686	16.193	0.943
	PL-G (REG-NEG)	70	516.914	18.425	0.942
	PL-R (INS-NEG)	70	522.061	16.491	0.951

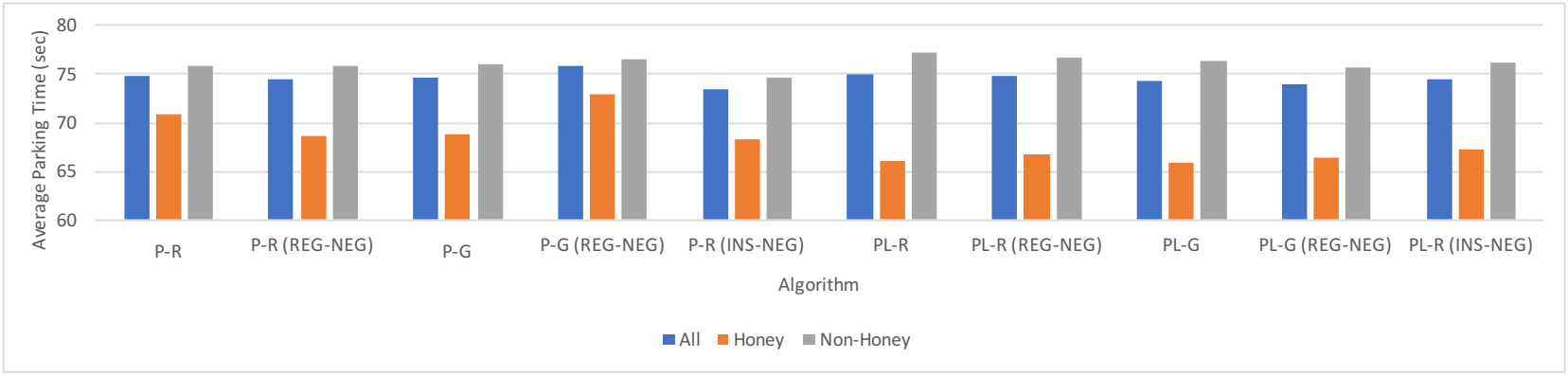
As expected, Figure 37, Figure 38 and Figure 39 show that the average parking times shorten as parking congestion decreases for lot-to-bee ratios equal or less than one. The reason is intuitive. When there is less parking congestion, there is less competition for vacant spots at any one time and it is easier for a driver to find parking. However, the results produced by simulations set a lot-to-bee ratio of two do not follow this rule. This is because parking demand and congestion is already low at that lot-to-bee ratio and as such, increasing the amount of time is takes for drivers to enter the simulation would not have a large effect. When it comes to comparing the relative performance of each algorithm, the results in Figure 37, Figure 38 and Figure 39 follow the trends outlined in Section 5.1.

5.4 Algorithmic Performance with a Mix of Honey and Non-Honey Cars

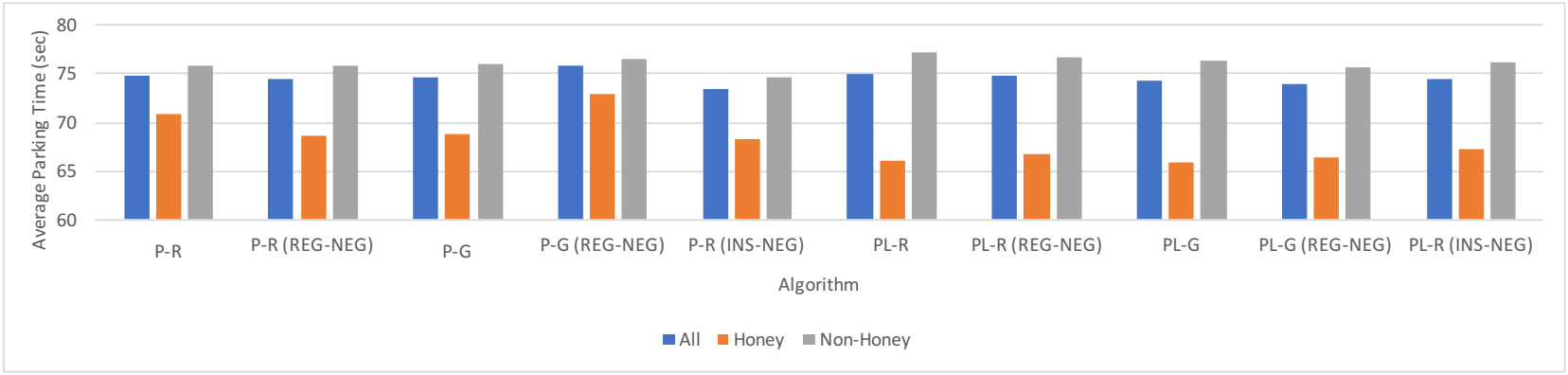
This section discusses the performance of the HoneyPark algorithm when it used alongside the other two parking algorithms, namely the Random and Greedy algorithms.

5.4.1 Mix of Honey and Random Cars

Figure 40, Figure 41 and Figure 42 show the performance of the HoneyPark algorithms when HoneyPark drivers and Random drivers are looking for parking simultaneously.

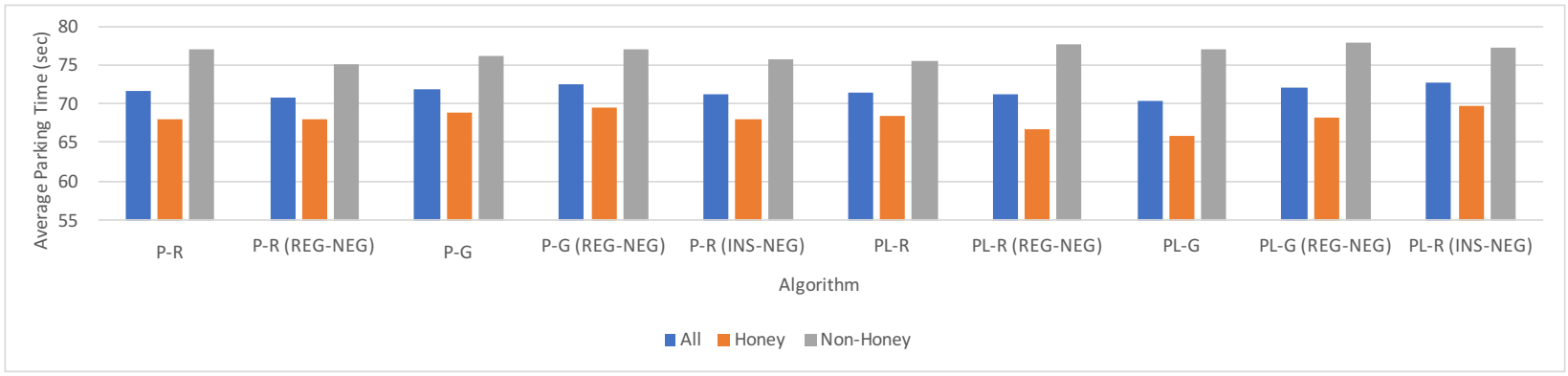


(a)

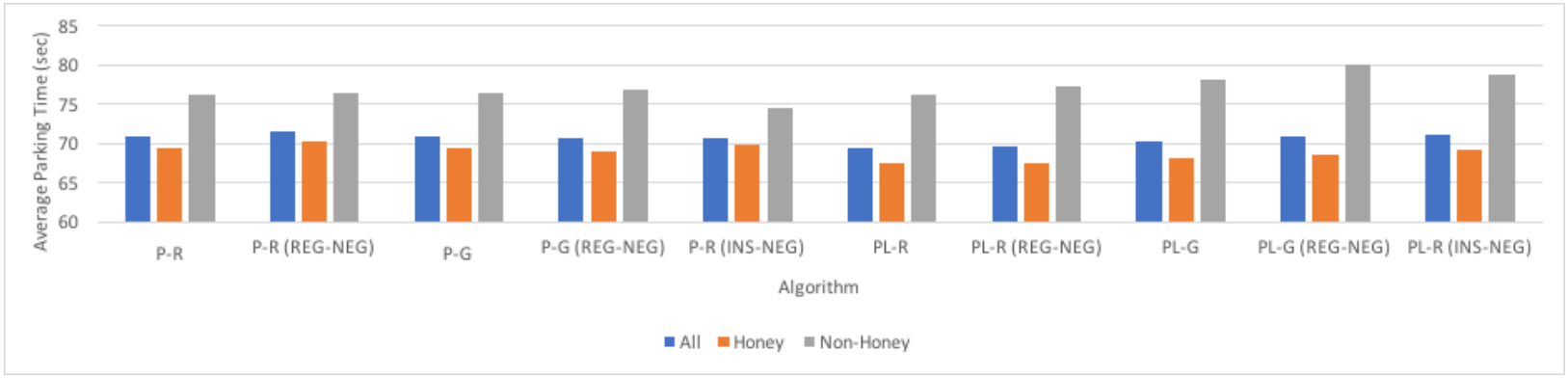


(b)

Figure 40 – Average Parking Times when the HoneyPark Car Proportion is (a) 20% (b) 40% (c) 60% (d) 80% when the Lot-to-Bee Ratio = 2

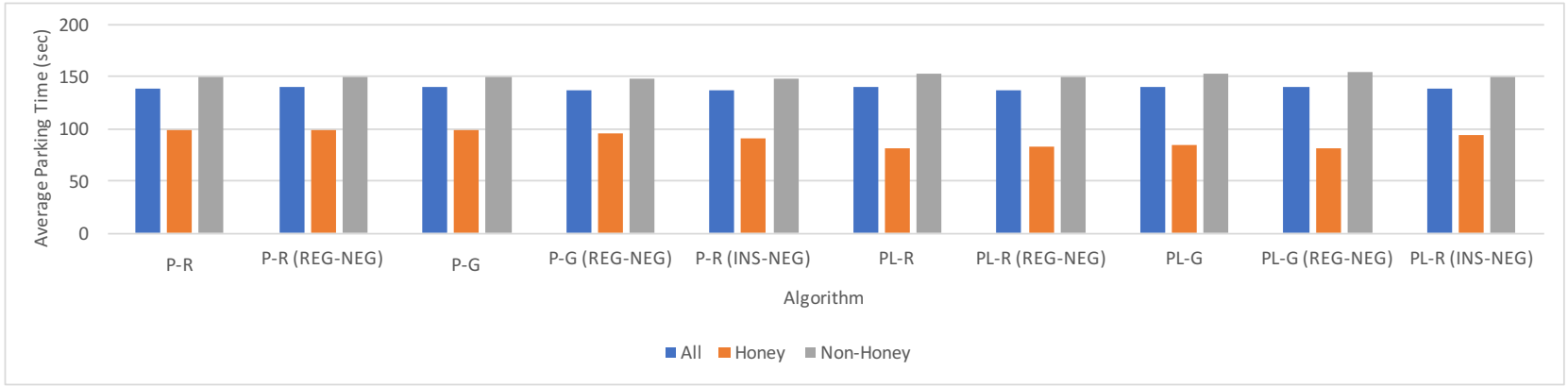


(c)

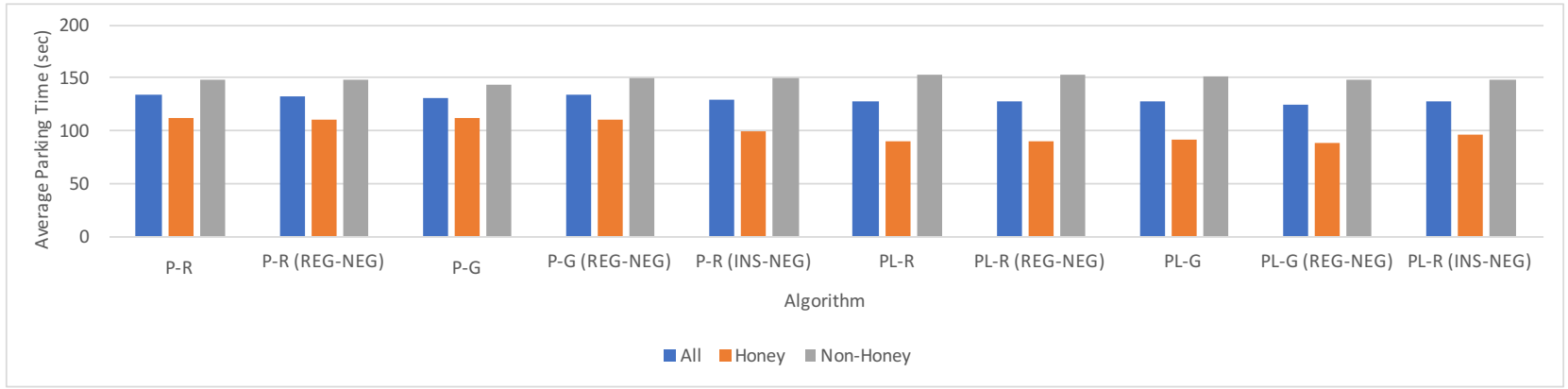


(d)

Figure 40 (continued)– Average Parking Times when the HoneyPark Car Proportion is (a) 20% (b) 40% (c) 60% (d) 80% when the Lot-to-Bee Ratio = 2

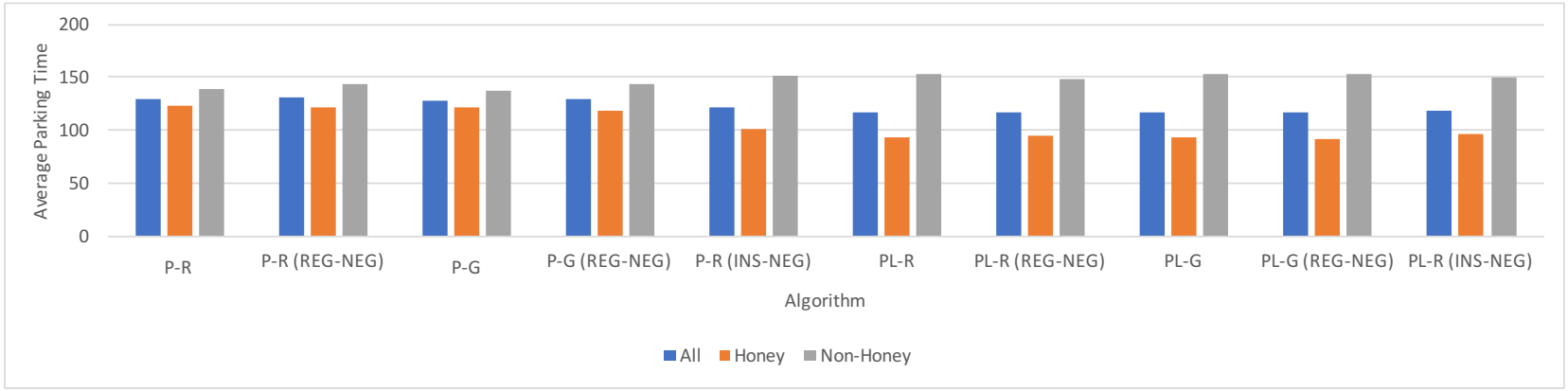


(a)

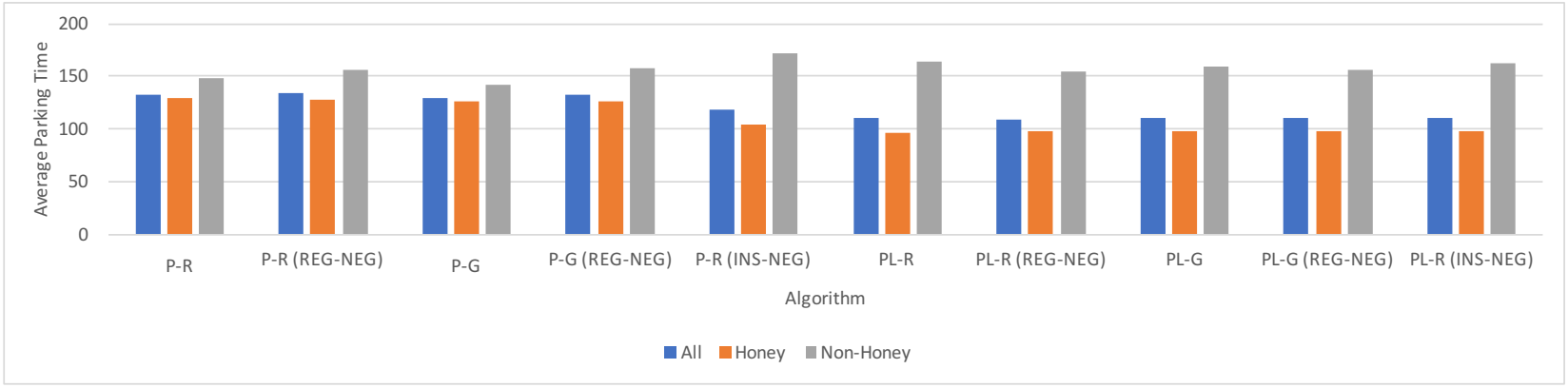


(b)

Figure 41 – Average Parking Times when the HoneyPark Car Proportion is (a) 20% (b) 40% (c) 60% (d) 80% when the Lot-to-Bee Ratio = 1

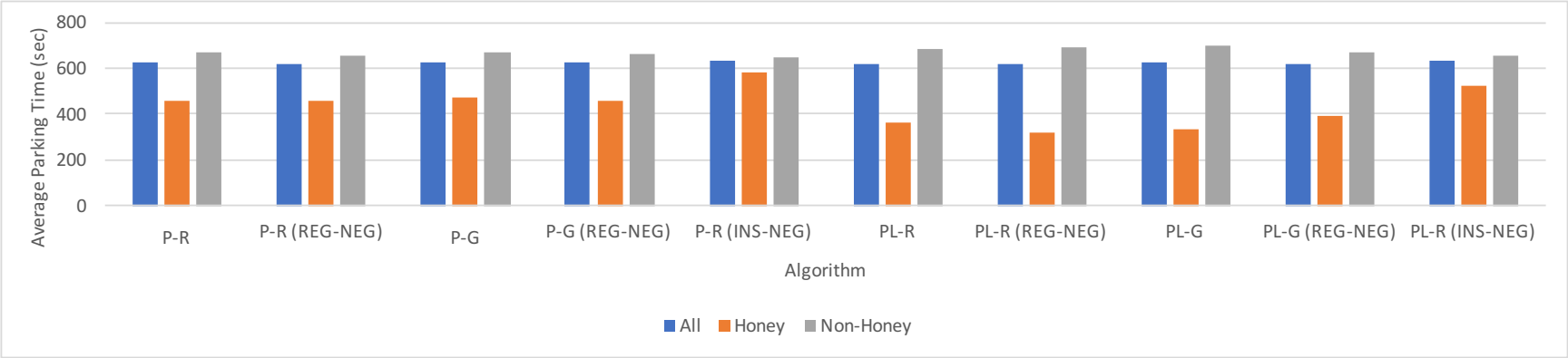


(c)

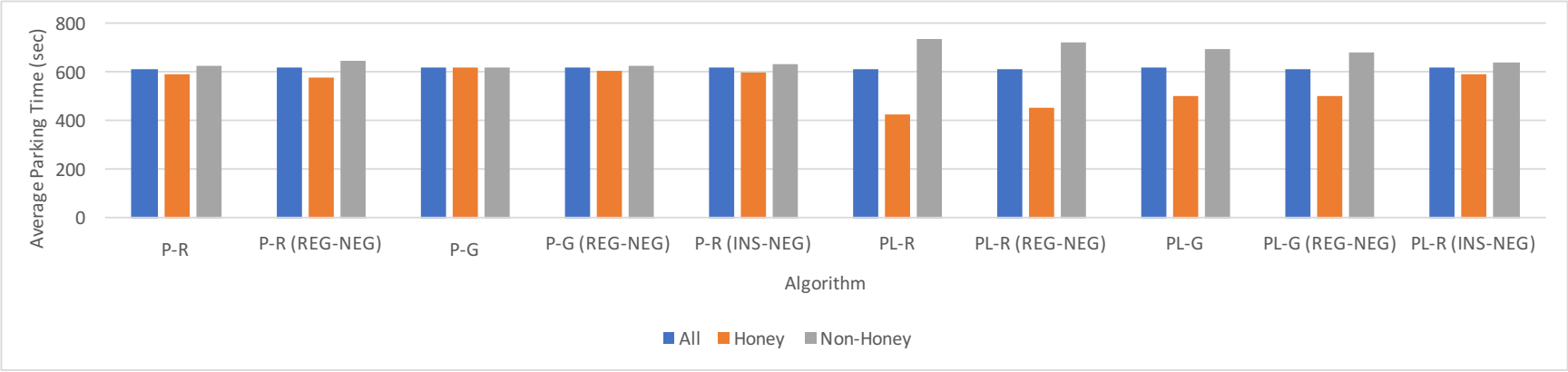


(d)

Figure 41 (continued) – Average Parking Times when the HoneyPark Car Proportion is (a) 20% (b) 40% (c) 60% (d) 80% when the Lot-to-Bee Ratio = 1

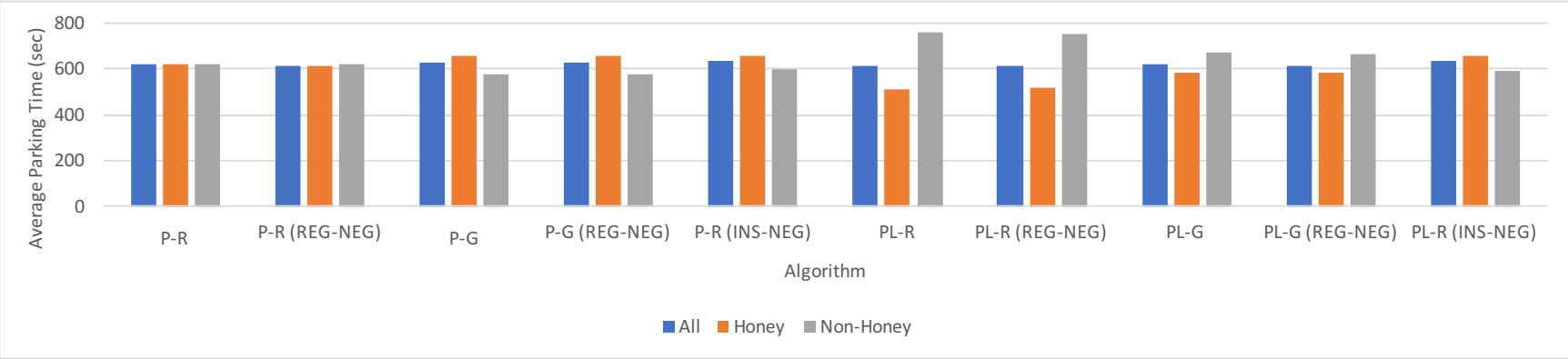


(a)

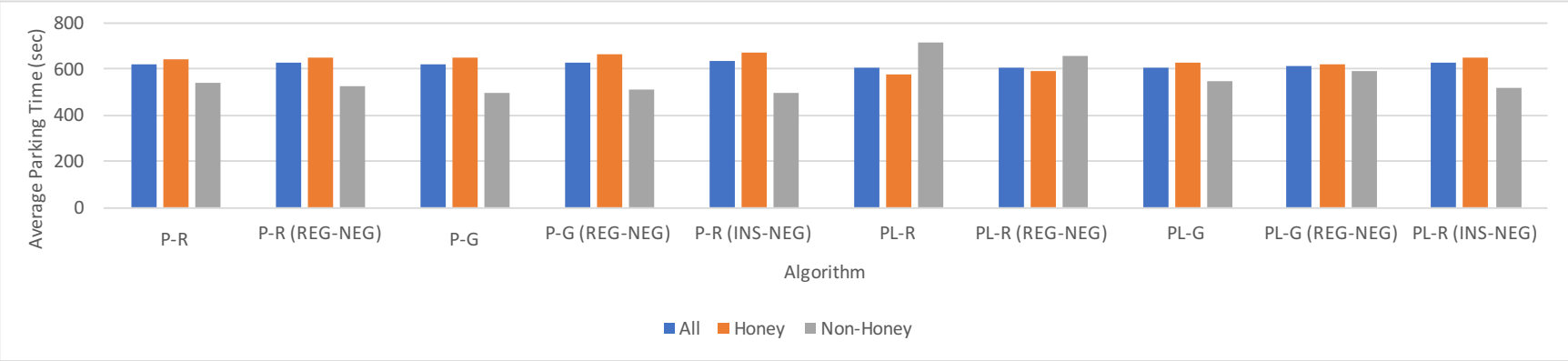


(b)

Figure 42 – Average Parking Times when the HoneyPark Car Proportion is (a) 20% (b) 40% (c) 60% (d) 80% when the Lot-to-Bee Ratio = 0.75



(c)



(d)

Figure 42 – Average Parking Times when the HoneyPark Car Proportion is (a) 20% (b) 40% (c) 60% (d) 80% when the Lot-to-Bee Ratio = 0.75

Figure 40 and Figure 41 show that for the lot-to-bee ratios of one and 2.0, the average parking times of all HoneyPark variations is lower than that of the Random algorithm at all HoneyPark car percentages. This is because the HoneyPark system is decentralized and based on real-time data. As long as there are HoneyPark cars sending adverts, the system can detect the changes in the success rate of each lot and thus correctly direct the driver to the most profitable lot. As such, the general trend is that the overall average parking time decreases as the proportion of HoneyPark drivers to Random drivers increases. As the HoneyPark algorithm is generally more efficient than the Random algorithm, it makes sense that when more drivers use the HoneyPark algorithm, it results in a lower overall average parking time.

Now, let's rearrange the data such that the only average parking times of HoneyCars are considered and are arranged by algorithm and HoneyPark Car Percentage. The results are displayed in Figure 43, Figure 44 and Figure 45.

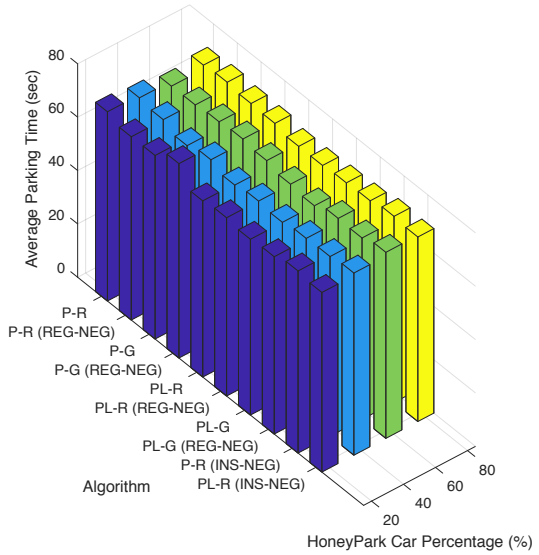


Figure 43 – Average Parking Times of HoneyPark Drivers arranged by HoneyPark Car Percentage when Lot-to-Bee Ratio is 2

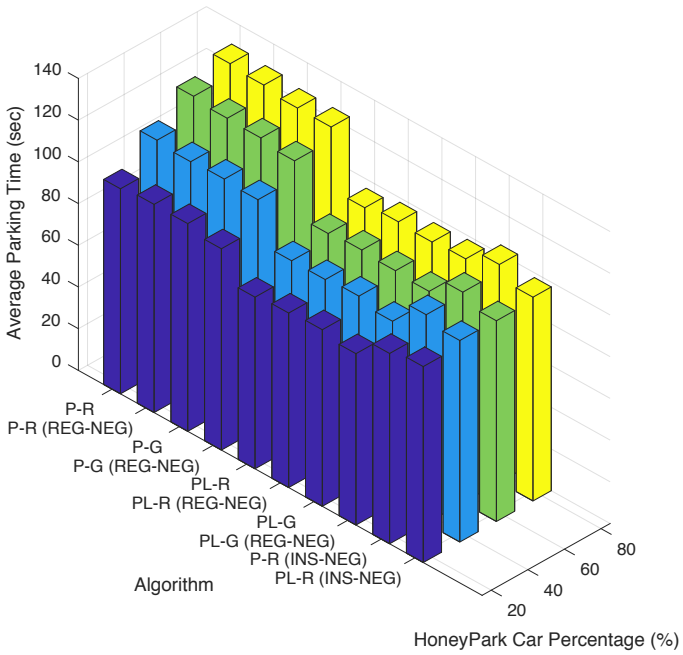


Figure 44 – Average Parking Times of HoneyPark Drivers arranged by HoneyPark Car Percentage when Lot-to-Bee Ratio is 1

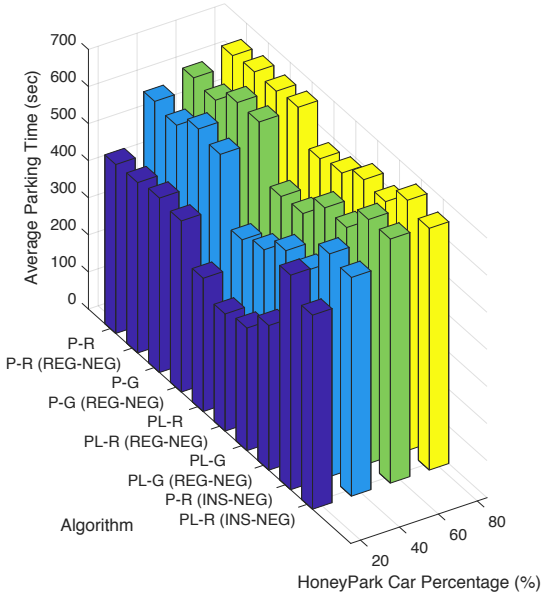


Figure 45 – Average Parking Times of HoneyPark Drivers arranged by HoneyPark Car Percentage when Lot-to-Bee Ratio is 0.75

It is interesting to note that for a lot-to-bee ratio of one and 0.75, the average parking time of HoneyPark drivers is lengthened as the percentage of HoneyPark drivers increases. This is because an increase in the number of drivers using the HoneyPark algorithms results in a greater number of drivers are directed to the most profitable lots. This results in more competition within these lots and some cars may have to take additional time once the profitable lot is fully occupied. Therefore, it seems that the HoneyPark algorithm performs best when there are enough drivers to collect information regarding the profitability of the parking lots, but not too many that the profitable lots become competitive. At a lot-to-bee ratio of two, the average parking time doesn't seem to change significantly as the HoneyPark car percentage is varied. This is because there is more parking supply than demand and it is likely that a driver can find parking easily regardless of the HoneyPark car percentage.

When the lot-to-bee ratio is equal to one, the Parked-Leave algorithm with instantaneous negative feedback is less efficient than other Parked-Leave algorithms at lower HoneyPark car percentages. In fact, it is also less efficient than its Parked-Advert equivalent at HoneyPark car percentages below 60%. A real-time analysis shows that the inclusion of Leave adverts in instantaneous negative feedback can be detrimental to algorithm performance in an environment that is not mostly controlled by the HoneyPark system. When a car leaves a full parking lot, it sends out a Leave advert and drivers can consider searching the parking lot. However, there is only one free parking space in that lot which can be easily be occupied by another driver using the Random algorithm especially when the HoneyPark car percentage is low. The simulations show that a driver using instantaneous negative feedback will consider such a lot and drive there only to find that the vacant spot has already been occupied. Contrary, the Parked-Advert algorithm has a shorter average parking time because there are no Leave adverts and as a result, does not tend to

consider parking lots that have only a few vacant spots. Therefore, the driver can choose a lot that is more profitable and will still be vacant when he or she arrives there.

At a lot-to-bee ratio of 0.75, the relative performance of the HoneyPark and the Random algorithm is more variable. With the exception of the Parked-Leave Advert algorithm (Random Scouting algorithm), the other variations of the HoneyPark algorithm are proven to be less effective than the Random algorithm at higher HoneyPark car percentages. As mentioned in previous sections, the HoneyPark algorithm tends to send more drivers to small selective group of parking lots, which is disadvantageous when the demand for parking is higher than supply. This is further supported by the fact that the Parked-Leave, Random-Scouting Algorithm is the only variation that consistently performs better than the Random algorithm at all tested HoneyPark car percentages. Its use of both Parked and Leave adverts as well as the Random scouting algorithm allows drivers to explore more parking lots, especially relative to the other variants of the HoneyPark algorithm.

There is also a deviation from the usual trends observed in Section 5.1 when looks at the relative performance of the HoneyPark algorithms. When the lot-to-bee ratio is less than one in a mixed environment of HoneyPark and Random drivers, the scouting algorithm does have an impact on algorithmic performance when only a portion of the drivers use the HoneyPark algorithm. Simulation data shows that drivers are more likely to find a parking space using their scouting algorithms when only a portion of active drivers are using the HoneyPark algorithm as compared to when all drivers in the simulation are HoneyPark algorithm. This can be seen in Figure 46, which shows what percentage of scouting attempts made by drivers in the simulation resulted in the driver successfully finding a vacant parking space at varying HoneyPark car percentages. As one can see, a driver is more likely to find parking using a scouting algorithm

when only a portion of the simulation population use the HoneyPark algorithm. This quantity decreases as the HoneyPark Car Proportion increases. This is likely caused by the fact that at lower HoneyPark car percentages, the HoneyPark system predicts the profitability of the parking lots with less accuracy as it is not receiving adverts from a larger number of the drivers in the simulation. Therefore, it is necessary to include a scouting algorithm that is able to redirect the driver to a more profitable lot and correct the HoneyPark system in the event that the information collected by adverts is insufficient or incorrect.

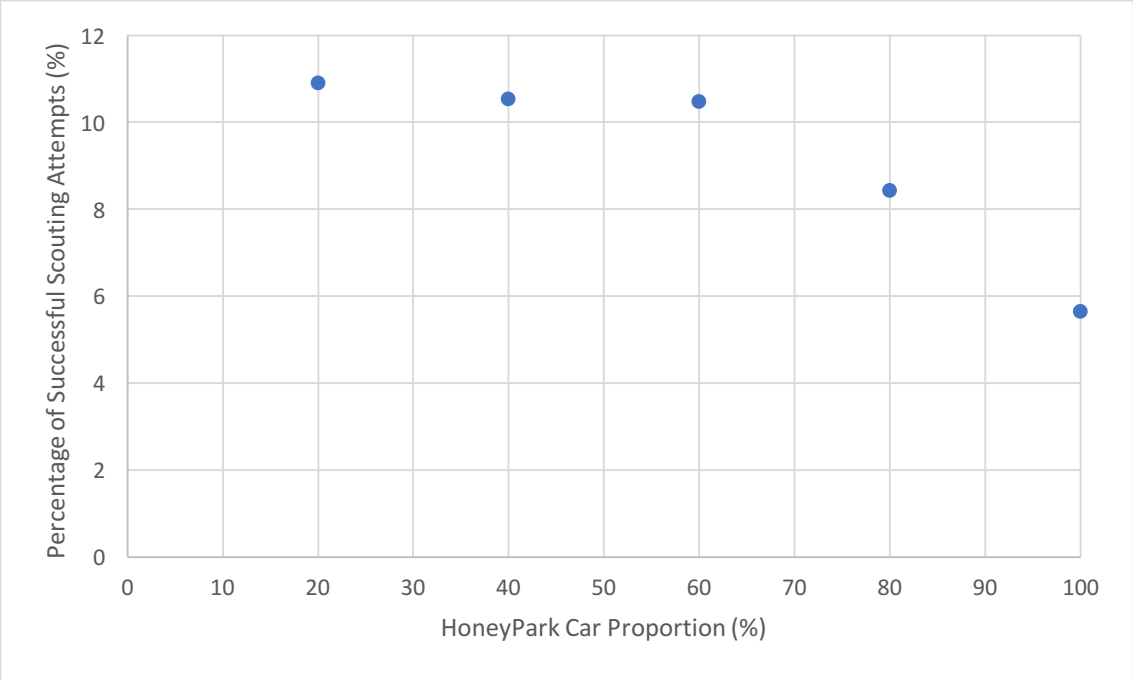
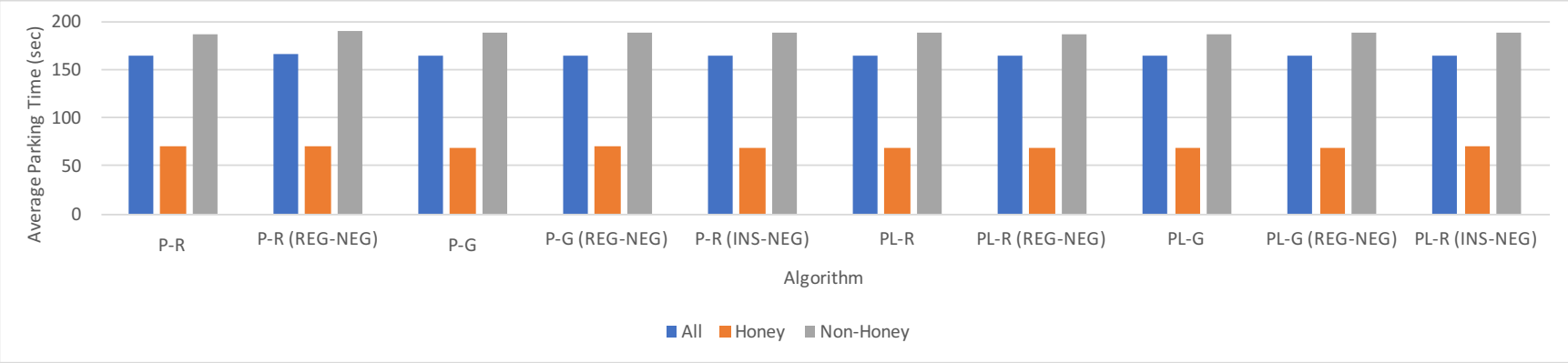


Figure 46 – Percentage of Scouting Drivers that are Able to Find a Vacant Parking Spot at different HoneyPark Car Proportions

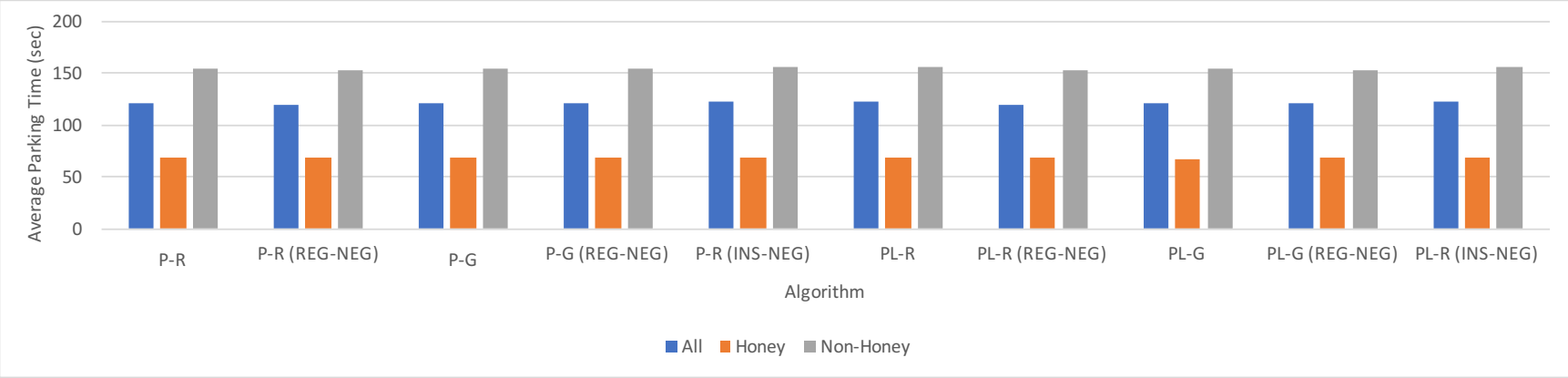
As the HoneyPark algorithm increases from 60% to 80%, one can see that the HoneyPark algorithms that use the Greedy scouting algorithm become less efficient sooner than their counterparts that use the Random scouting algorithm. Simulation data shows that drivers that use

the Greedy scouting algorithm will often search a lot assigned to them by the HoneyPark algorithm, fail to find parking there and proceed to search the nearest parking lot. As the HoneyPark algorithm tends to direct the drivers to a smaller subset of parking lots, simulation data shows that the drivers using the Greedy scouting algorithm tend to scout the same parking lots when they fail to find parking at their assigned lots. This increases competition for those locations, consequently lengthening parking time. In contrast, drivers that use the random scouting algorithm explore a wider variety of parking lots increasing their chances of success finding a profitable parking lot.

5.4.2 Mix of Honey and Greedy Cars

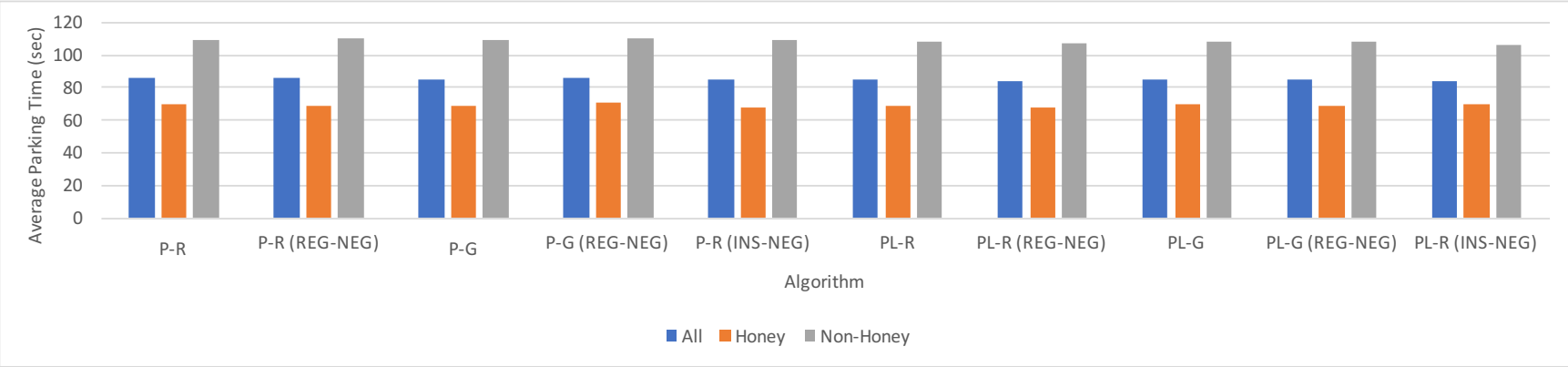


(a)

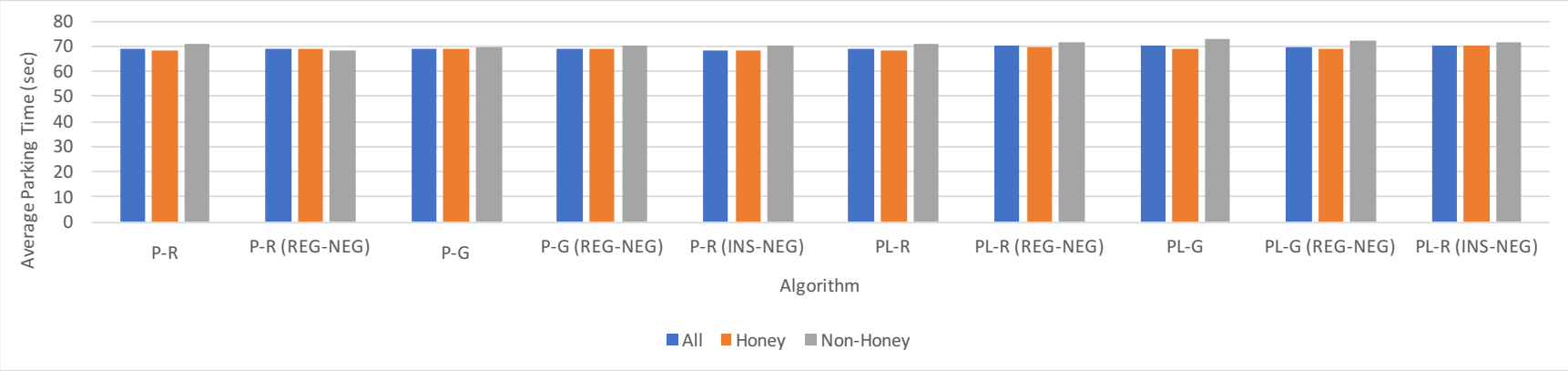


(b)

Figure 47 – Average Parking Times when the HoneyPark Car Proportion is (a) 20% (b) 40% (c) 60% (d) 80% when the Lot-to-Bee Ratio = 2

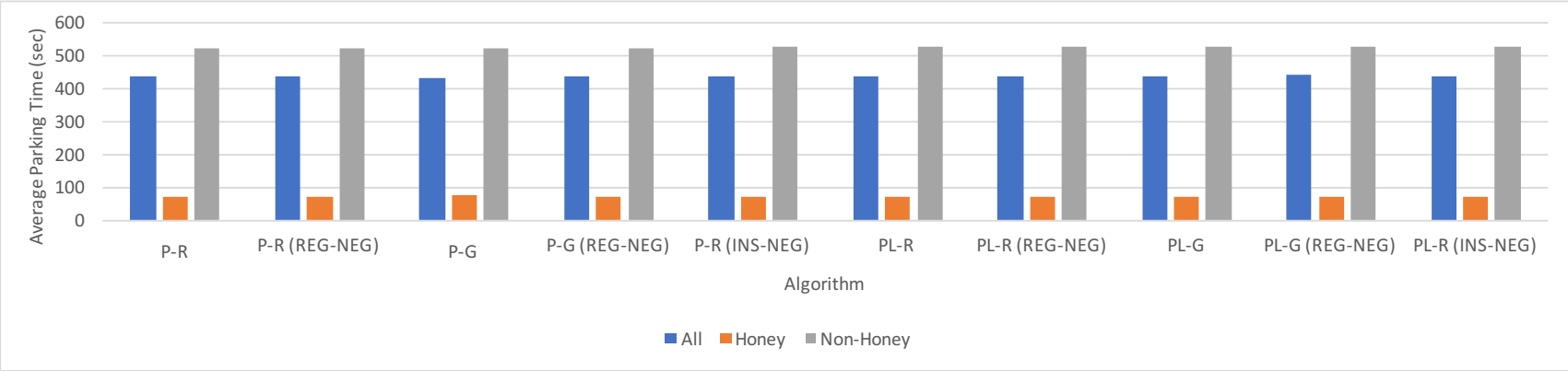


(c)

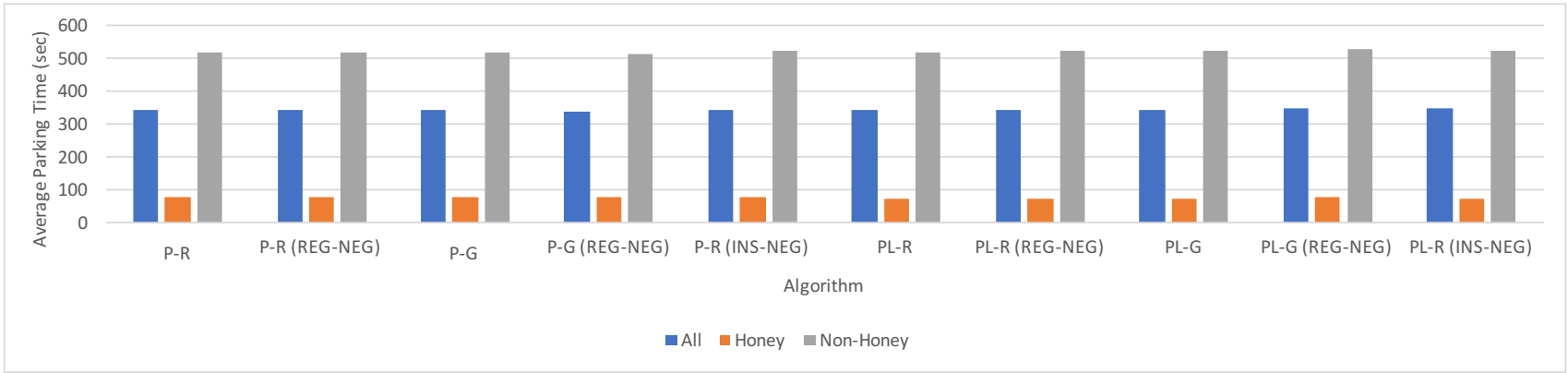


(d)

Figure 47 (continued) – Average Parking Times when the HoneyPark Car Proportion is (a) 20% (b) 40% (c) 60% (d) 80% when the Lot-to-Bee Ratio = 2

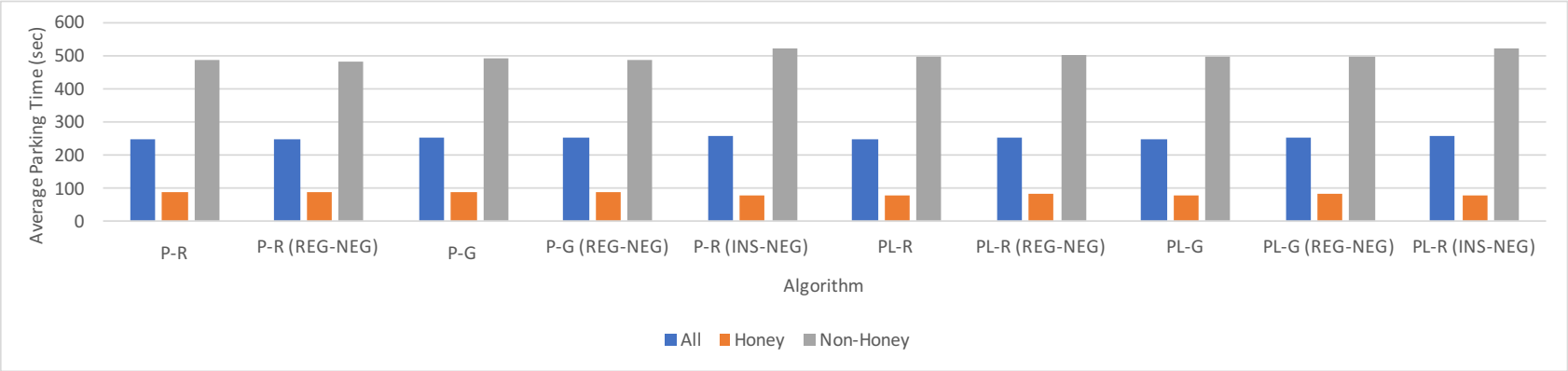


(a)

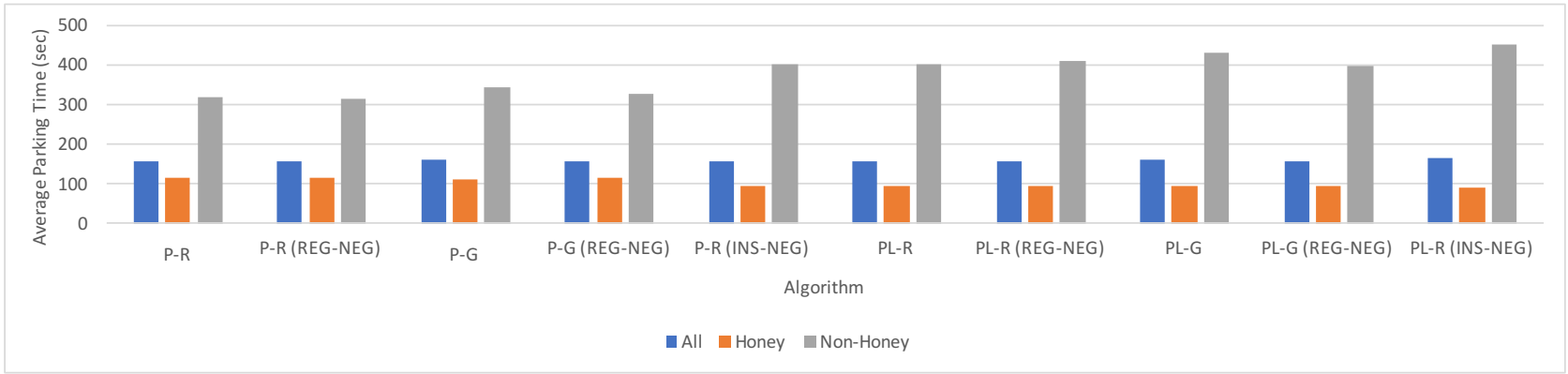


(b)

Figure 48 – Average Parking Times when the HoneyPark Car Proportion is (a) 20% (b) 40% (c) 60% (d) 80% when the Lot-to-Bee Ratio = 1

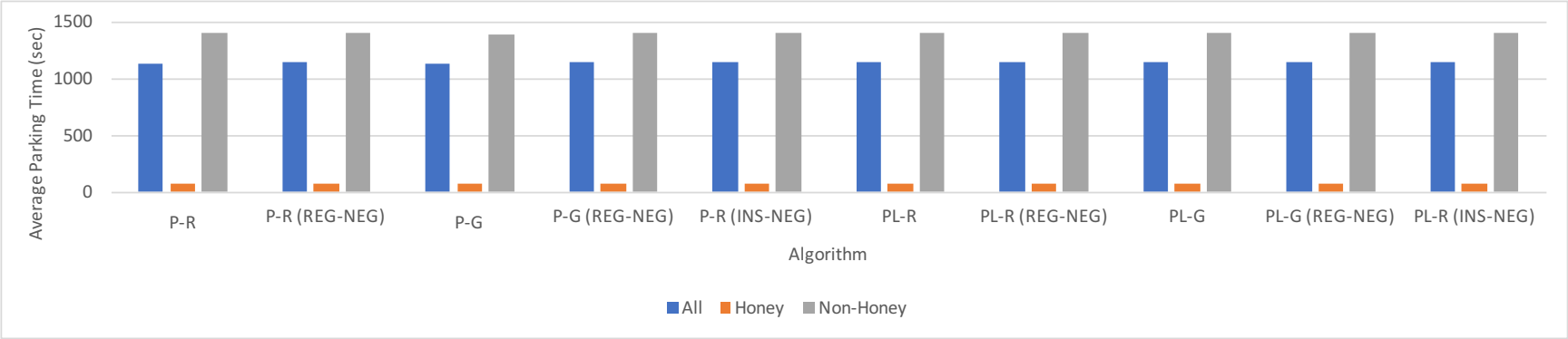


(c)

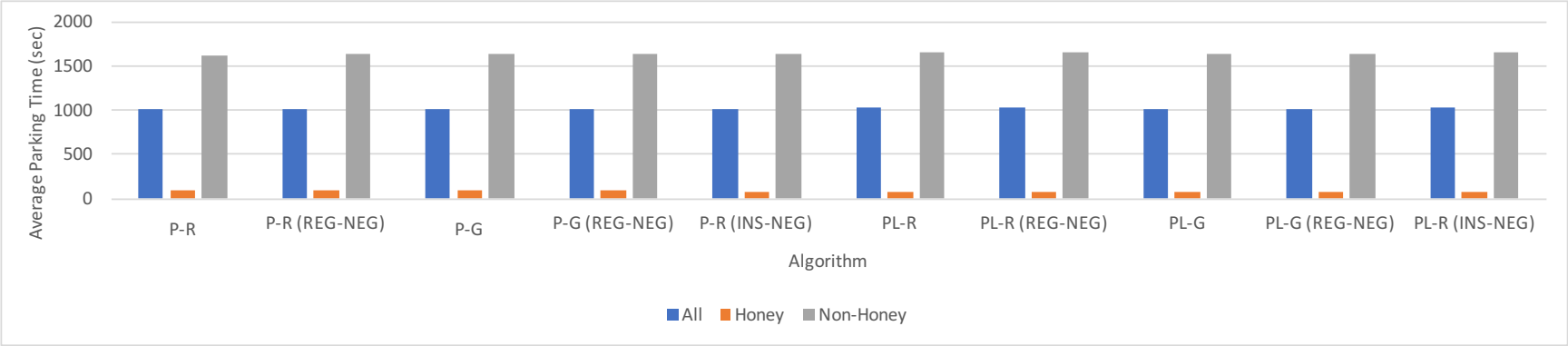


(d)

Figure 48 (continued) – Average Parking Times when the HoneyPark Car Proportion is (a) 20% (b) 40% (c) 60% (d) 80% when the Lot-to-Bee Ratio = 1

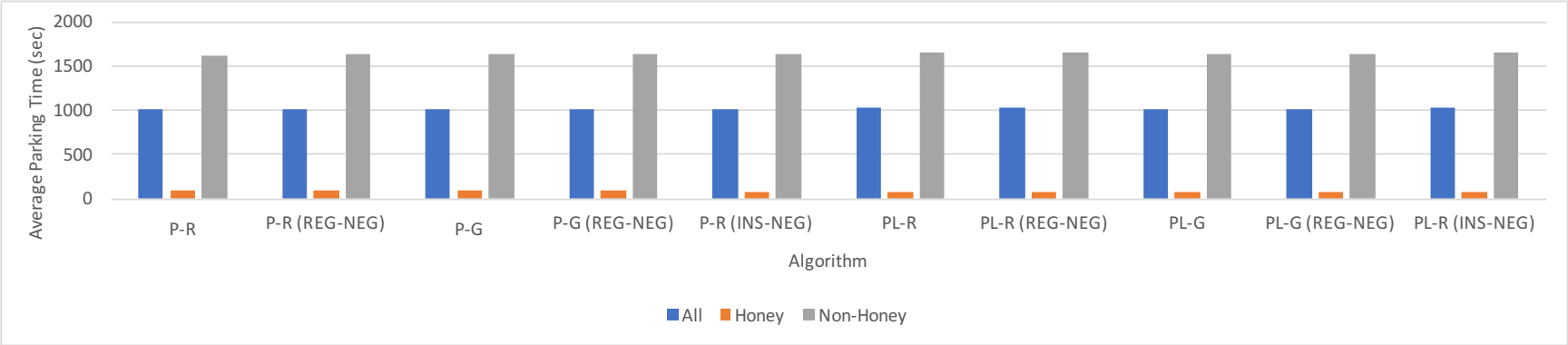


(a)

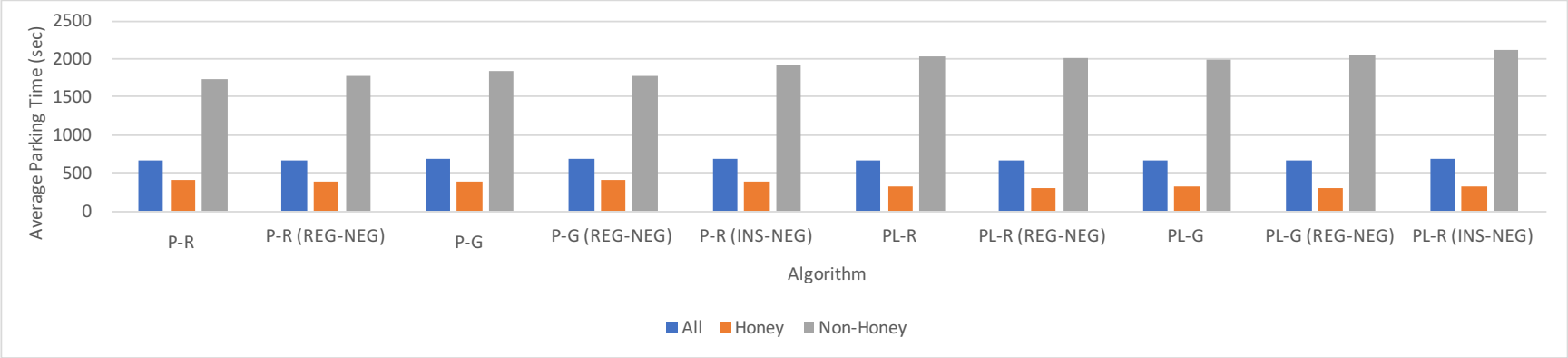


(b)

Figure 49 – Average Parking Times when the HoneyPark Car Proportion is (a) 20% (b) 40% (c) 60% (d) 80% when the Lot-to-Bee Ratio = 0.75



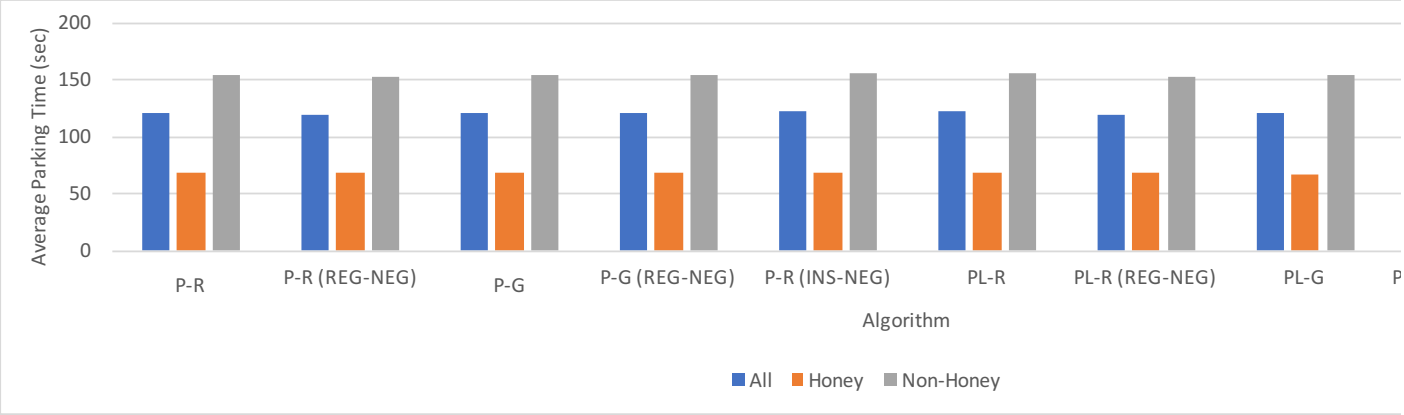
(c)



(d)

Figure 49 (continued) – Average Parking Times when the HoneyPark Car Proportion is (a) 20% (b) 40% (c) 60% (d) 80% when the Lot-to-Bee Ratio = 0.75

The average HoneyPark parking times in (a)



(b)

Figure 47, Figure 48 and Figure 49 also increase as the proportion of drivers using the HoneyPark algorithm increases. The only interesting difference is that unlike the Random algorithm, the average parking time of the Greedy algorithm sharply decreases as the number of drivers using the HoneyPark algorithm increases, especially above 90%. This is because there are less drivers using the Greedy algorithm and therefore, there is less competition for the parking spots closest to the desired destination. In addition, the proximity of the parking spot to the destination shortens the amount of time it takes the driver to travel to the lot, which consequently reduces the parking time further. Therefore, the Greedy algorithm will be more efficient than the HoneyPark algorithm if the absolute number of drivers using the Greedy algorithm is small enough. This is reflected in Figure 50 and Figure 51, which arranges the Greedy average parking time by HoneyPark car percentages.

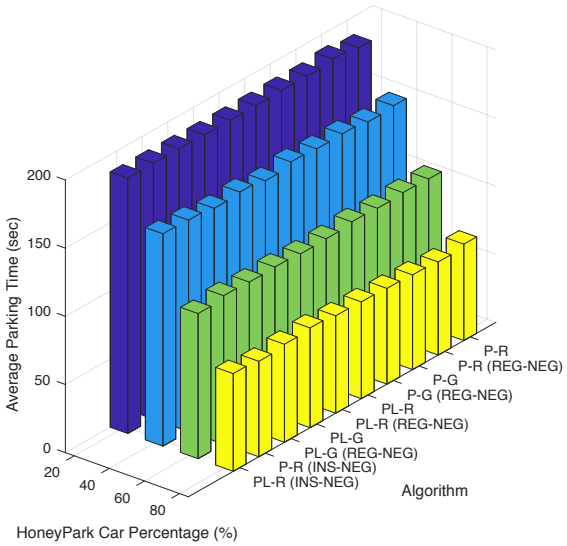


Figure 50 – Average Parking Times of Greedy Drivers arranged by HoneyPark Car Percentage when Lot-to-Bee Ratio is 2

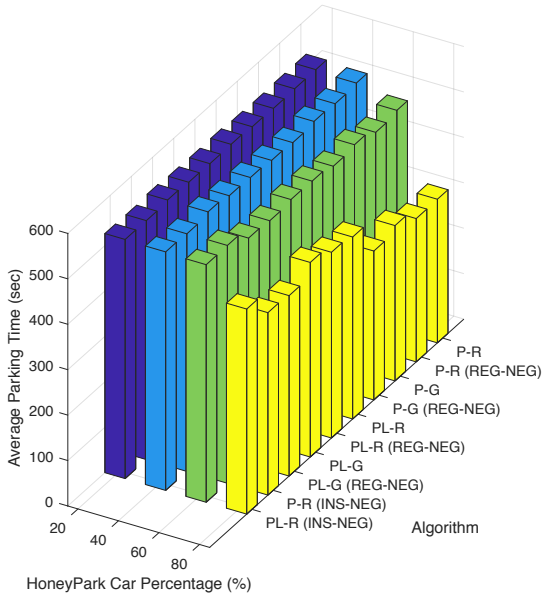


Figure 51 – Average Parking Times of Greedy Drivers arranged by HoneyPark Car Percentage when Lot-to-Bee Ratio is 1

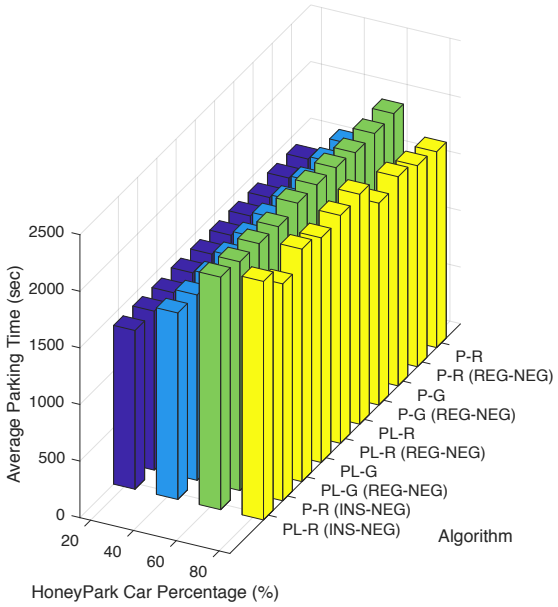


Figure 52 – Average Parking Times of Greedy Drivers arranged by HoneyPark Car Percentage when Lot-to-Bee Ratio is 0.75

However, it is important to note that the Greedy average parking times in Figure 52 does not decrease as HoneyPark car percentage increases. This is because the lot-to-bee ratio is too low and there are too many cars in simulation. Therefore, there is competition for the closest due to the simple fact that there is more competition at this lot-to-bee ratio in general.

5.5 Algorithmic Performance with a Mix of V2V and Non-V2V Cars

There is a possibility that certain cars may not have vehicle-to-vehicle communication capability and cannot send adverts to the HoneyPark system. However, it may be possible for them to use the HoneyPark algorithm to shorten their parking time. Figure 53, Figure 54 and Figure 55 shows the performance of each algorithm when all drivers are using the HoneyPark algorithm but some of them are not capable of sending adverts. Table 31, Table 32 and Table 33 provides the

same data as the figures in numeric form with the addition of how many simulation runs were performed for each algorithm

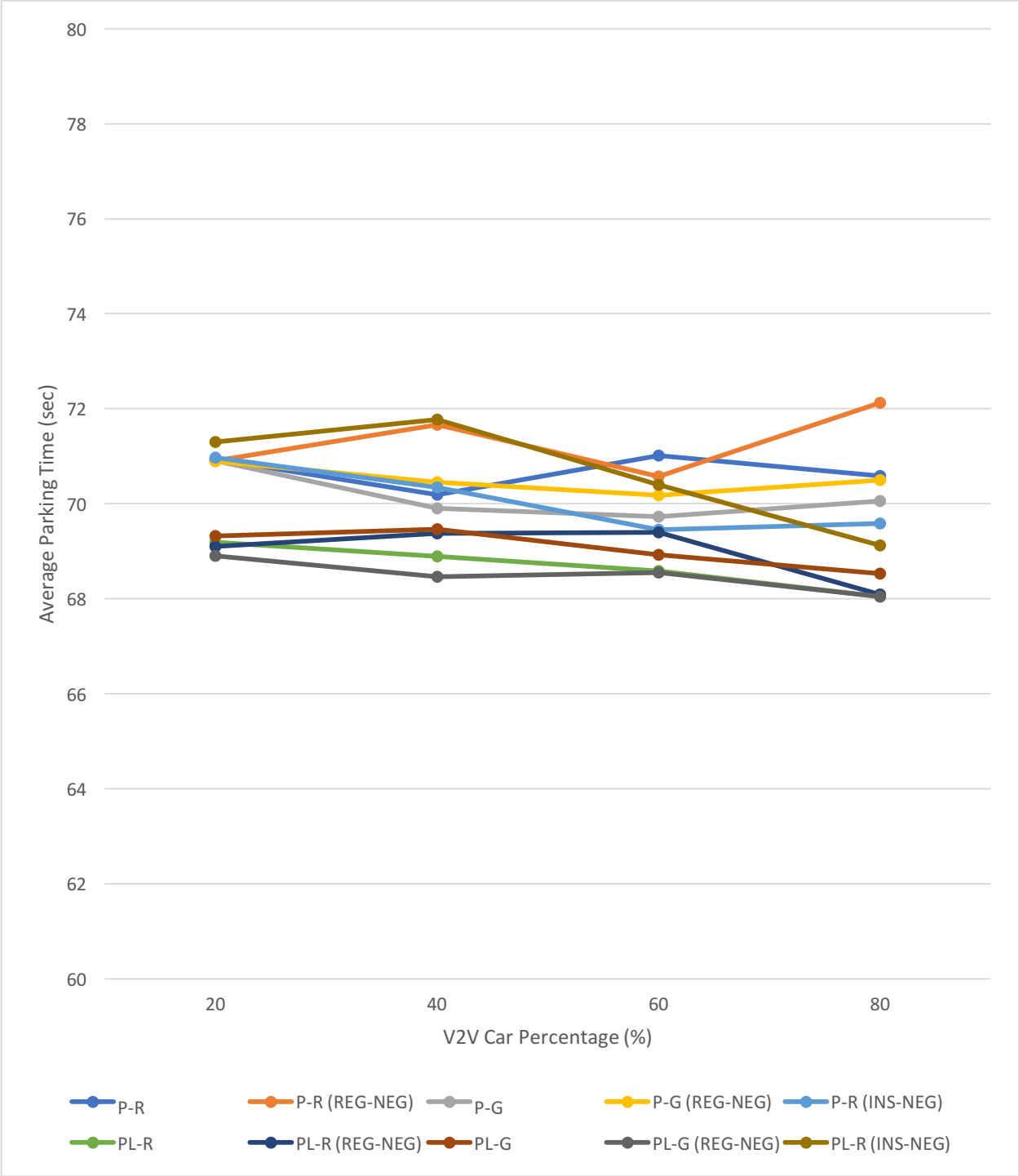


Figure 53 – Average Parking Times when V2V Car Percentage is varied (Lot-to-Bee Ratio = 2)

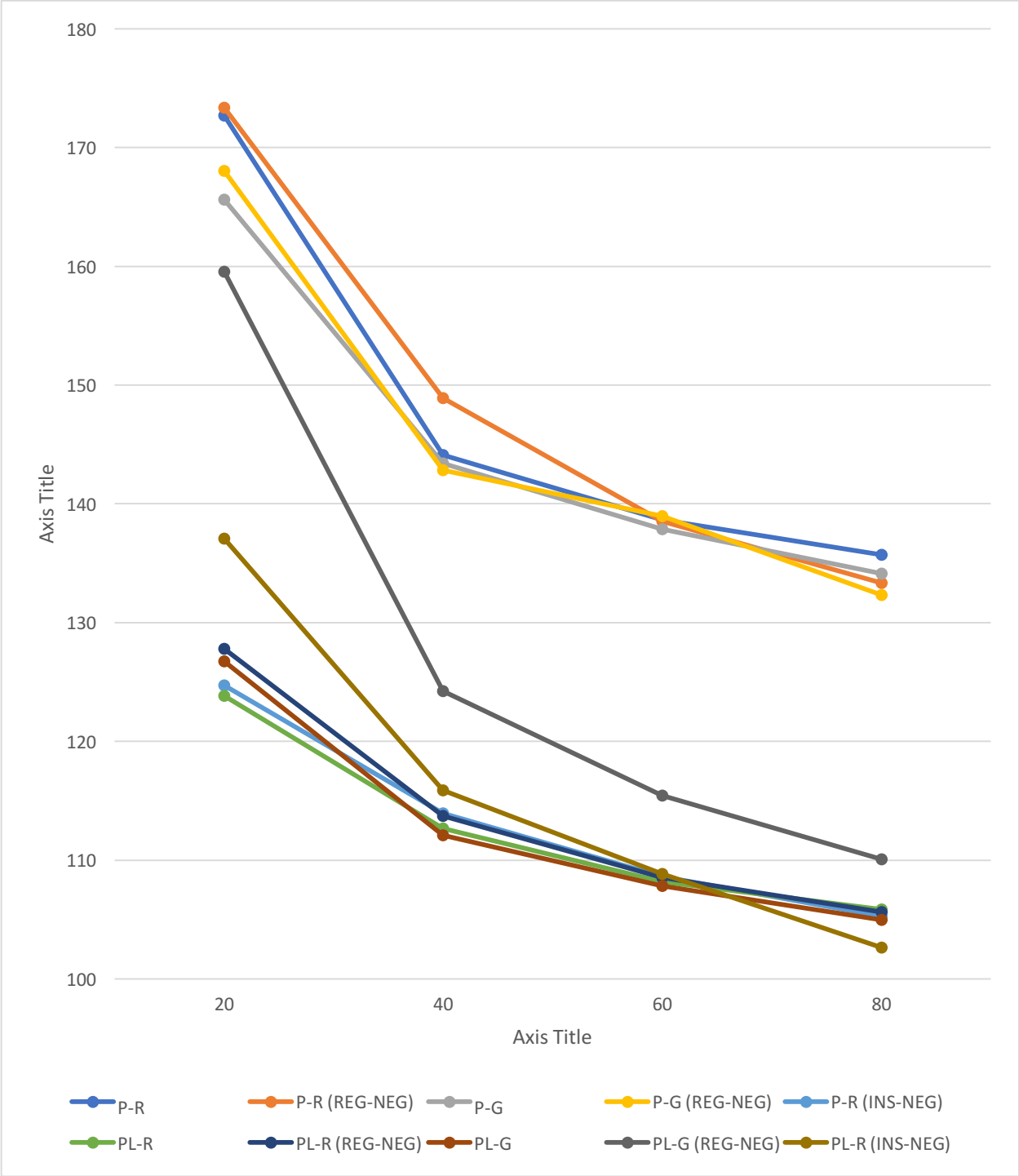


Figure 54 – Average Parking Times when V2V Car Percentage is varied (Lot-to-Bee Ratio = 1)

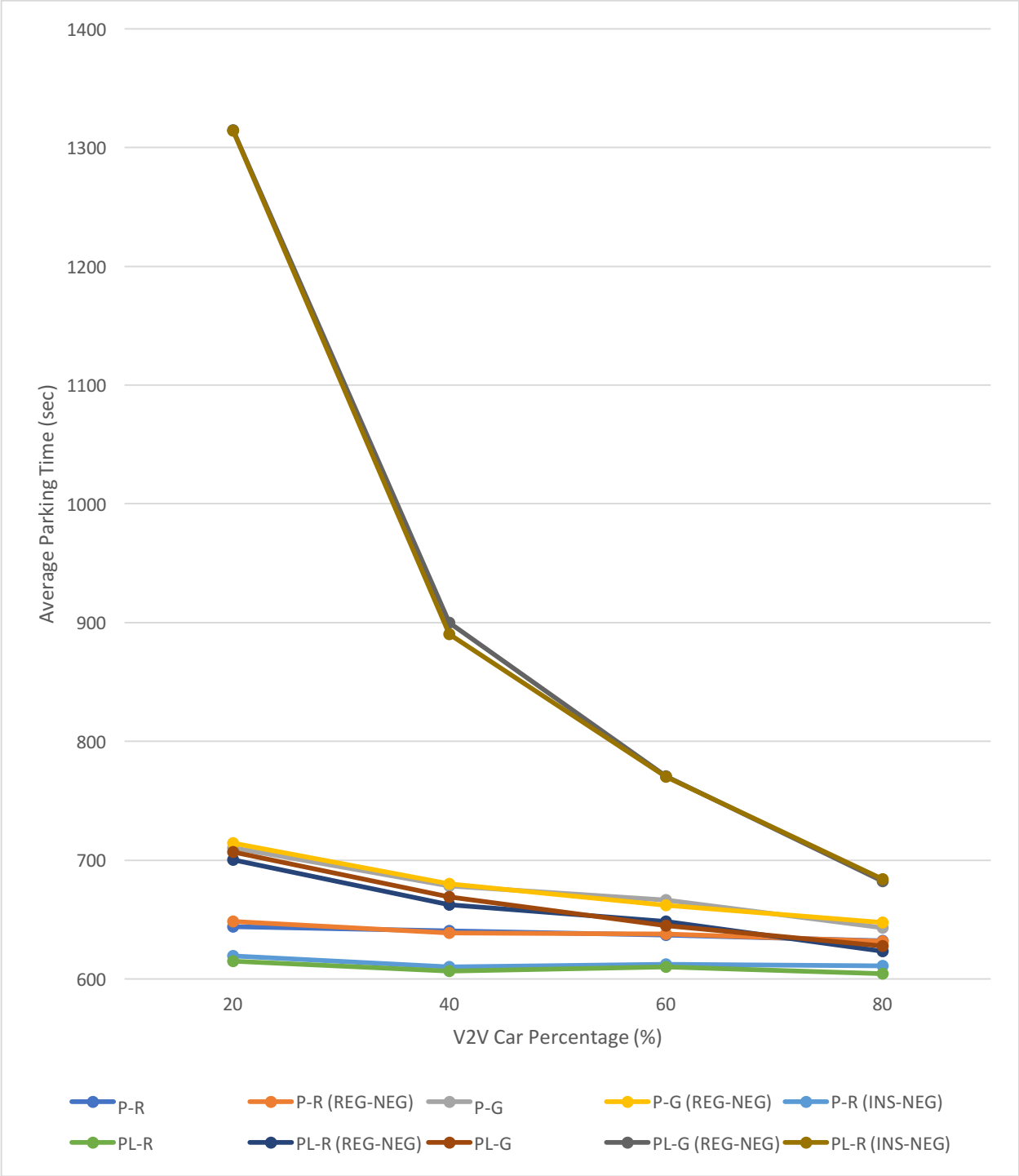


Figure 55 – Average Parking Times when V2V Car Percentage is varied (Lot-to-Bee Ratio = 0.75)

Table 31 – Average Parking Times and Variances at Varying V2V Car Percentages when Bee-to-lot Ratio = 2

	V2V Car Percentage (%)											
	20			40			60			80		
Algorithm	No. of Simulations Executed	Average Parking Time (sec)	Standard Deviation of Average Parking Time (sec)	No. of Simulations Executed	Average Parking Time (sec)	Standard Deviation of Average Parking Time (sec)	No. of Simulations Executed	Average Parking Time (sec)	Standard Deviation of Average Parking Time (sec)	No. of Simulations Executed	Average Parking Time (sec)	Standard Deviation of Average Parking Time (sec)
P-R	70	70.937	5.826	70	70.191	5.137	70	71.018	5.523	70	70.589	5.115
P-R (REG-NEG)	70	70.906	5.294	70	71.659	5.972	70	70.573	5.520	70	72.128	6.123
P-G	70	70.905	5.363	70	69.905	4.982	70	69.732	5.355	70	70.054	5.484
P-G (REG-NEG)	70	70.890	5.365	70	70.452	5.086	70	70.184	5.086	70	70.501	5.711
PL-R	70	69.191	4.185	70	68.889	4.325	70	68.582	4.053	70	68.051	4.478
PL-R (REG-NEG)	70	69.104	4.038	70	69.372	4.386	70	69.403	4.453	70	68.094	4.093
PL-G	70	69.318	4.333	70	69.470	4.020	70	68.929	4.125	70	68.526	4.435
PL-G (REG-NEG)	70	68.901	4.499	70	68.470	4.423	70	68.551	4.266	70	68.052	4.514
P-R (INS-NEG)	70	70.975	5.653	70	70.342	5.330	70	69.450	4.734	70	69.587	3.966
PL-R (INS-NEG)	70	71.302	5.587	70	71.779	5.628	70	70.400	4.897	70	69.125	4.190

Table 32 – Average Parking Times and Variances at Varying V2V Car Percentages when Bee-to-lot Ratio = 1

	V2V Car Percentage (%)											
	20			40			60			80		
Algorithm	No. of Simulations Executed	Average Parking Time (sec)	Standard Deviation of Average Parking Time (sec)	No. of Simulations Executed	Average Parking Time (sec)	Standard Deviation of Average Parking Time (sec)	No. of Simulations Executed	Average Parking Time (sec)	Standard Deviation of Average Parking Time (sec)	No. of Simulations Executed	Average Parking Time (sec)	Standard Deviation of Average Parking Time (sec)
P-R	70	172.674	20.121	70	144.104	11.276	70	138.654	10.229	70	135.867	10.613
P-R (REG-NEG)	70	173.340	19.008	70	148.879	11.446	70	138.539	10.552	70	134.606	10.044
P-G	70	165.610	18.660	70	143.403	12.062	70	137.879	10.037	70	135.031	10.381
P-G (REG-NEG)	70	168.013	17.822	70	142.818	10.020	70	138.951	9.633	70	134.545	10.372
PL-R	70	124.695	11.085	70	113.952	8.577	70	108.599	5.542	70	106.306	5.724
PL-R (REG-NEG)	70	123.837	10.052	70	112.645	6.535	70	108.177	6.116	70	106.301	6.003
PL-G	70	127.797	11.396	70	113.716	7.452	70	108.523	6.530	70	106.549	5.573
PL-G (REG-NEG)	70	126.739	11.963	70	112.096	6.932	70	107.846	5.925	70	105.621	5.543
P-R (INS-NEG)	70	159.553	23.670	70	124.212	10.936	70	115.417	6.691	70	113.199	7.298
PL-R (INS-NEG)	70	137.079	12.993	70	115.874	8.164	70	108.839	6.939	70	105.122	6.073

Table 33 – Average Parking Times and Variances at Varying V2V Car Percentages when Bee-to-lot Ratio = 0.75

	V2V Car Percentage (%)											
	20			40			60			80		
Algorithm	No. of Simulations Executed	Average Parking Time (sec)	Standard Deviation of Average Parking Time (sec)	No. of Simulations Executed	Average Parking Time (sec)	Standard Deviation of Average Parking Time (sec)	No. of Simulations Executed	Average Parking Time (sec)	Standard Deviation of Average Parking Time (sec)	No. of Simulations Executed	Average Parking Time (sec)	Standard Deviation of Average Parking Time (sec)
P-R	70	644.190	27.428	70	640.257	26.248	70	636.738	25.699	70	632.127	21.367
P-R (REG-NEG)	70	648.411	30.675	70	638.884	28.309	70	637.942	24.376	70	631.462	29.528
P-G	70	710.182	27.778	70	678.303	36.383	70	666.306	31.594	70	642.903	31.431
P-G (REG-NEG)	70	714.530	32.527	70	679.896	30.582	70	662.054	32.372	70	647.557	32.586
PL-R	70	619.420	24.363	70	610.270	26.475	70	612.211	23.025	70	610.946	23.763
PL-R (REG-NEG)	70	614.933	25.009	70	606.659	24.410	70	610.261	23.564	70	604.506	24.435
PL-G	70	700.394	25.585	70	662.515	25.060	70	648.566	27.169	70	623.360	23.113
PL-G (REG-NEG)	70	706.848	29.305	70	668.928	30.931	70	644.812	29.528	70	627.694	27.321
P-R (INS-NEG)	70	1314.585	86.191	70	900.070	55.226	70	770.522	39.476	70	682.233	30.428
PL-R (INS-NEG)	70	1313.988	89.504	70	890.017	56.536	70	770.071	38.854	70	683.974	32.703

Simulation results show that using the HoneyPark algorithm without sufficient V2V cars generally results in longer average parking times for both V2V and non-V2V cars for ratios that are equal to one. Figure 54 shows that the average parking time for all drivers decreases as the proportion of V2V driver increases for all variations of the HoneyPark algorithm. This is because as non-V2V drivers are occupying and leaving parking spaces without updating the system, the system is consequently measuring the profitability of the parking lot based on incorrect, outdated data. Therefore, there must be enough V2V drivers in the simulation to keep the system updated so that the algorithm can accurately calculate the profitability of each lot.

The same general trend can also be observed when the ratio is 0.75 as shown in Figure 55. The only major deviation is the fact that algorithms that use the Greedy scouting algorithm perform significantly worse than their counterparts that use the Random algorithm. Similar to Section 5.5, drivers in a mixed environment of V2V and non-V2V cars rely more on scouting algorithms as the HoneyPark system does not receive adverts from all drivers and therefore does not always have an accurate measure of the profitability of parking lots in the simulation. The underperformance of the Greedy scouting algorithm is likely due to the fact that it only scouts the nearest lots to a small group of profitable lots. As a result, it greatly increases competition for a small subset of lots and prevents drivers from exploring more profitable options.

Another significant thing to note in Figure 55 is that the instantaneous negative feedback algorithms performs significantly worse at lower V2V car percentages when the ratio is 0.75. Their performance also improves drastically as the V2V car percentage increases. This is because the instantaneous negative feedback feature keeps drivers from going to parking lots that are marked as full at a point in time. As the HoneyPark system is not consistently accurate as it is not receiving adverts from all drivers in the simulation, it is possible that ‘false negatives’ may occur.

In this sense, a ‘false negative’ arises when parking lots are still marked as full even though they may no longer be fully occupied. Instantaneous negative feedback algorithms may keep drivers from visiting such lots, preventing drivers from finding spaces within them. This will obviously result in a longer parking time. Increasing the V2V car percentage will drastically improve the performance of such algorithms because there are more drivers sending adverts. As such, the HoneyPark system is able to mark which lots are filled with greater accuracy and consequently reduce the occurrence of a ‘false negative’.

For ratios exceeding one, Figure 53 shows that a change in proportion of V2V drivers does not significantly affect the overall average parking time. This is mainly due to the fact that there is more parking supply than demand and it is likely that drivers can find a parking space regardless of which lot they choose to search. In such a case, the average parking time is less affected by the accuracy of the HoneyPark system simply because it is likely that drivers can find a vacant parking spot as all the parking lots are profitable. Therefore, as shown in Figure 53, changing the proportion of V2V drivers in the simulation will not have a substantial effect on the average parking time.

5.6 Algorithmic Performance in Real-Time Traffic Environment

This section discusses the performance of the Random, Greedy and HoneyPark algorithms in a real-time traffic environment. Traffic is added to the simulation environment using the travel times provided by the Google Distance Matrix API. The travel times provided by the Google API is much longer than that provided by OpenStreetMaps API used in previous simulations, simulating the fact that the driver experience traffic on the roads and will take a longer time to reach their destination. The Google API provides three traffic models that we will use to adjust the level of traffic in the simulation:

1. Optimistic Model: Provides data when traffic has been good (i.e. less traffic) based on historical traffic data.
2. Best-Guess Model: Estimates what traffic would be like based on historical traffic data.
3. Pessimistic Model: Provides data when traffic has been bad (i.e. more traffic) based on historical traffic data.

As the API has a daily limit, the author only had time to test one variation of the HoneyPark algorithm. The Parked-Leave, Random-Scouting Algorithm was chosen as it was the only algorithm that was consistently the most efficient in all scenarios.

Figure 56 shows the performance of the algorithms under varying real-life traffic conditions when the ratio is 0.75, 1 and 2 respectively. Table 34 displays the data in its numeric form along with the number of simulations executed for each algorithm. With real-life traffic conditions, more simulations were needed to obtain a stable answer. Therefore, 100 simulations were performed for the Random and HoneyPark algorithms. As previous sections have discussed, the Greedy algorithm is extremely slow and there was not enough time and computational resources to execute multiple simulations for it. As the algorithm is proven to consistently underperform, there is no need to rigorously test it. Thus, only one simulation was done for the Greedy algorithm so that one has an idea of its performance in a real-life traffic situation.

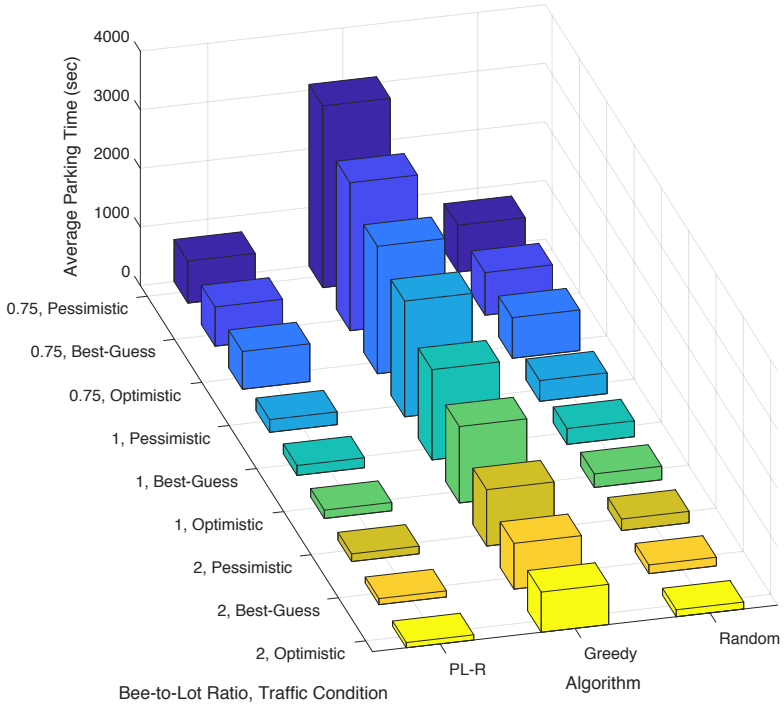


Figure 56 – Average Parking Times in Varying Traffic Conditions at Various Ratios and Traffic Conditions

Table 34 – Average Parking Times and Variances in Varying Traffic Conditions at Ratios of 0.75, 1 and 2

Lot-to-Bee Ratio	Algorithm	Google Traffic Condition											
		Pessimistic				Best Guess				Optimistic			
		No. of Simulations Executed	Average Parking Time (sec)	Standard Deviation of Average Parking Time (sec)	Ratio of Parking Time to Random Parking	No. of Simulations Executed	Average Parking Time (sec)	Standard Deviation of Average Parking Time (sec)	Ratio of Parking Time to Random Parking	No. of Simulations Executed	Average Parking Time (sec)	Standard Deviation of Average Parking Time (sec)	Ratio of Parking Time to Random Parking
0.75	Random	100	798.961	62.052	1.000	100	726.384	33.802	1.000	100	692.930	32.422	1.000
	Greedy	1	3100.579	-	3.881	1	2524.438	NA	3.475	1	2175.376	-	3.139
	PL-R	100	717.677	80.691	0.898	100	669.296	30.078	0.921	100	644.184	25.531	0.930
1	Random	100	354.214	68.643	1.000	100	270.963	39.474	1.000	100	231.505	33.387	1.000
	Greedy	1	1975.263	-	5.576	1	1540.158	NA	5.684	1	1304.737	-	5.636
	PL-R	100	217.175	35.563	0.613	100	169.425	16.585	0.625	100	142.425	9.733	0.615
2	Random	100	196.714	41.960	1.000	100	148.467	23.174	1.000	100	118.530	12.872	1.000
	Greedy	1	962.567	-	4.893	1	781.328	NA	5.263	1	685.284	-	5.782
	PL-R	100	131.340	22.684	0.668	100	103.761	12.304	0.699	100	88.320	7.050	0.745

As one can see, pessimistic traffic conditions consistently results in longer parking times, followed by best-guess and optimistic traffic conditions. The reason for this occurrence is simply the fact that it takes a longer time to travel to a parking lot when traffic is bad. This consequently lengthens average parking time.

Figure 56 also shows that the HoneyPark algorithm consistently performs better than either the Random or Greedy parking algorithms. This is because the HoneyPark algorithms first uses data collected from adverts and Google Traffic to determine if a parking is profitable and if it is too much trouble to travel to the lot respectively. As such, it is more likely to send drivers to parking lots where they can find parking and/or where there is less traffic. Random does not consider any of these factors and sends drivers to random parking lots, which can result in long parking times in bad conditions. Greedy insists on sending drivers to the nearest parking lots, which can quickly increase competition for those lots and result in long parking times. Coupled with bad traffic conditions, this can result in abnormal parking times. As the data shows, the average parking time produced by the Greedy algorithm can exceed 10 times that of the HoneyPark algorithm.

5.7 Summary of Experimental Results

This chapter essentially discusses if and why the HoneyPark algorithms are effective or ineffective in a variety of situations. Table 35 shows the relative performance of each HoneyPark variation in each scenario as compared to the Random algorithm. The numbers are calculated by dividing the HoneyPark average parking time by the Random average parking time. Therefore, values less than one indicate that the HoneyPark algorithm performs better than Random algorithm and vice versa.

Table 35 – Relative Performance of all HoneyPark Variations in All Tested Scenarios

Varying Lot-to-Bee Ratio (One Dest)	Lot-to-Bee Ratio	P-R	P-R (REG-NEG)	P-G	P-G (REG-NEG)	P-R (INS-NEG)	PL-R	PL-R (REG-NEG)	PL-G	PL-G (REG-NEG)	PL-R (INS-NEG)
	0.5	1.006	1.006	1.007	1.008	1.016	1.003	0.998	1.006	1.006	1.012
	0.75	1.001	1.003	1.003	1.004	0.984	0.964	0.971	0.970	0.964	0.974
	1	0.898	0.911	0.898	0.897	0.728	0.707	0.705	0.703	0.707	0.691
	2	0.935	0.936	0.924	0.945	0.910	0.897	0.902	0.901	0.907	0.917
	4	0.937	0.945	0.948	0.944	0.941	0.921	0.939	0.948	0.924	0.933
Varying Lot-to-Bee Ratio (One Mesh Dest)	Number of Drivers	P-R	P-R (REG-NEG)	P-G	P-G (REG-NEG)	P-R (INS-NEG)	PL-R	PL-R (REG-NEG)	PL-G	PL-G (REG-NEG)	PL-R (INS-NEG)
	Small	0.973	0.970	0.973	0.977	0.952	0.960	0.963	0.958	0.956	0.967
	Moderate	0.993	0.993	0.987	0.990	0.910	0.941	0.945	0.944	0.946	0.904
	Large	0.846	0.849	0.844	0.852	0.810	0.722	0.717	0.722	0.720	0.721
	Very Large	0.855	0.846	0.851	0.849	0.829	0.734	0.729	0.743	0.746	0.775

Table 35 (continued) – Relative Performance of all HoneyPark Variations in All Tested Scenarios

	Errand Time Length (Lot-to-Bee Ratio = 0.75)	P-R	P-R (REG-NEG)	P-G	P-G (REG-NEG)	P-R (INS-NEG)	PL-R	PL-R (REG-NEG)	PL-G	PL-G (REG-NEG)	PL-R (INS-NEG)
Varying Errand Time Length	0-600	1.101	1.104	1.081	1.096	0.949	0.945	0.942	0.942	1.078	0.996
	1800-2400	1.013	1.014	1.017	1.016	0.983	0.982	0.985	0.984	1.015	1.005
	3000-3600	1.011	1.012	1.016	1.014	0.990	0.991	0.992	0.991	1.009	1.006
	6600-7200	1.006	1.006	1.008	1.008	0.996	0.996	0.996	0.996	1.005	1.003
	10200-10800	1.003	1.003	1.006	1.005	0.997	0.997	0.997	0.997	1.004	1.002
	Errand Time Length (Lot-to-Bee Ratio = 1)	P-R	P-R (REG-NEG)	P-G	P-G (REG-NEG)	P-R (INS-NEG)	PL-R	PL-R (REG-NEG)	PL-G	PL-G (REG-NEG)	PL-R (INS-NEG)
	0-600	1.010	0.992	0.988	1.003	0.864	0.867	0.870	0.860	0.976	0.914
	1800-2400	0.769	0.766	0.758	0.763	0.599	0.596	0.597	0.605	0.627	0.589
	3000-3600	0.724	0.724	0.729	0.729	0.570	0.574	0.575	0.569	0.592	0.562
	6600-7200	0.670	0.677	0.668	0.673	0.529	0.527	0.526	0.526	0.557	0.516
	10200-10800	0.668	0.663	0.664	0.656	0.522	0.515	0.519	0.517	0.538	0.503
	Errand Time Length (Lot-to-Bee Ratio = 2)	P-R	P-R (REG-NEG)	P-G	P-G (REG-NEG)	P-R (INS-NEG)	PL-R	PL-R (REG-NEG)	PL-G	PL-G (REG-NEG)	PL-R (INS-NEG)
	0-600	0.920	0.915	0.916	0.911	0.912	0.907	0.896	0.915	0.909	0.928
	1800-2400	0.914	0.907	0.926	0.903	0.893	0.900	0.899	0.896	0.904	0.911
	3000-3600	0.922	0.910	0.919	0.899	0.879	0.872	0.883	0.881	0.890	0.888
	6600-7200	0.922	0.919	0.922	0.914	0.910	0.898	0.898	0.910	0.902	0.919
	10200-10800	0.936	0.924	0.903	0.908	0.895	0.895	0.900	0.885	0.899	0.903

Table 35 (continued) – Relative Performance of all HoneyPark Variations in All Tested Scenarios

Varying Congestion Levels	Time for all Drivers to Enter Simulation (Lot-to-Bee Ratio = 0.75)	P-R	P-R (REG-NEG)	P-G	P-G (REG-NEG)	P-R (INS-NEG)	PL-R	PL-R (REG-NEG)	PL-G	PL-G (REG-NEG)	PL-R (INS-NEG)
	200	0.984	0.984	0.987	0.994	0.972	0.955	0.957	0.959	0.956	0.964
	400	0.996	0.995	1.003	0.992	0.980	0.974	0.971	0.975	0.972	0.979
	600	0.999	0.998	0.990	0.998	0.965	0.954	0.960	0.958	0.957	0.960
	800	0.989	0.980	0.989	0.986	0.960	0.939	0.943	0.943	0.942	0.951
	Time for all Drivers to Enter Simulation (Lot-to-Bee Ratio = 1)	P-R	P-R (REG-NEG)	P-G	P-G (REG-NEG)	P-R (INS-NEG)	PL-R	PL-R (REG-NEG)	PL-G	PL-G (REG-NEG)	PL-R (INS-NEG)
	200	0.885	0.883	0.869	0.865	0.725	0.698	0.698	0.704	0.695	0.675
	400	0.862	0.870	0.867	0.859	0.700	0.673	0.669	0.675	0.678	0.654
	600	0.853	0.847	0.839	0.848	0.692	0.667	0.665	0.661	0.673	0.651
	800	0.853	0.837	0.850	0.834	0.707	0.671	0.672	0.674	0.680	0.663
	Time for all Drivers to Enter Simulation (Lot-to-Bee Ratio = 2)	P-R	P-R (REG-NEG)	P-G	P-G (REG-NEG)	P-R (INS-NEG)	PL-R	PL-R (REG-NEG)	PL-G	PL-G (REG-NEG)	PL-R (INS-NEG)
	200	0.915	0.918	0.927	0.913	0.902	0.895	0.887	0.878	0.890	0.906
	400	0.946	0.955	0.940	0.938	0.934	0.896	0.895	0.902	0.894	0.917
	600	0.960	0.961	0.942	0.952	0.929	0.894	0.902	0.883	0.906	0.921
	800	0.925	0.941	0.942	0.937	0.938	0.890	0.887	0.895	0.888	0.901

Table 35 (continued) – Relative Performance of all HoneyPark Variations in All Tested Scenarios

Mix of HoneyPark and Random Cars	HoneyPark Car Perecentage (Lot-to-Bee Ratio = 0.75)	P-R	P-R (REG-NEG)	P-G	P-G (REG-NEG)	P-R (INS-NEG)	PL-R	PL-R (REG-NEG)	PL-G	PL-G (REG-NEG)	PL-R (INS-NEG)
	20	0.683	0.698	0.707	0.694	0.899	0.525	0.455	0.474	0.581	0.799
	40	0.939	0.896	0.998	0.962	0.950	0.579	0.629	0.727	0.736	0.918
	60	0.997	0.984	1.141	1.150	1.105	0.671	0.688	0.873	0.882	1.117
	80	1.188	1.244	1.316	1.304	1.365	0.810	0.896	1.144	1.050	1.251
	HoneyPark Car Perecentage (Lot-to-Bee Ratio = 1)	P-R	P-R (REG-NEG)	P-G	P-G (REG-NEG)	P-R (INS-NEG)	PL-R	PL-R (REG-NEG)	PL-G	PL-G (REG-NEG)	PL-R (INS-NEG)
	20	0.659	0.663	0.662	0.654	0.613	0.535	0.559	0.552	0.534	0.627
	40	0.759	0.748	0.771	0.737	0.669	0.587	0.590	0.599	0.592	0.651
	60	0.883	0.848	0.884	0.826	0.665	0.609	0.635	0.610	0.604	0.642
	80	0.873	0.816	0.886	0.796	0.607	0.582	0.637	0.613	0.626	0.603
	HoneyPark Car Perecentage (Lot-to-Bee Ratio = 2)	P-R	P-R (REG-NEG)	P-G	P-G (REG-NEG)	P-R (INS-NEG)	PL-R	PL-R (REG-NEG)	PL-G	PL-G (REG-NEG)	PL-R (INS-NEG)
	20	0.936	0.906	0.905	0.952	0.915	0.855	0.871	0.862	0.878	0.884
	40	0.906	0.892	0.871	0.885	0.884	0.855	0.881	0.863	0.861	0.876
	60	0.881	0.905	0.902	0.901	0.897	0.906	0.859	0.856	0.876	0.902
	80	0.912	0.917	0.908	0.899	0.937	0.886	0.872	0.873	0.857	0.879

Table 35 (continued) – Relative Performance of all HoneyPark Variations in All Tested Scenarios

	V2V Car Perecentage (Lot-to-Bee Ratio = 0.75)	P-R	P-R (REG-NEG)	P-G	P-G (REG-NEG)	P-R (INS-NEG)	PL-R	PL-R (REG-NEG)	PL-G	PL-G (REG-NEG)	PL-R (INS-NEG)
Mix of V2V and non-V2V Cars	20	1.029	1.035	1.134	1.141	0.989	0.982	1.118	1.129	2.099	2.098
	40	1.022	1.020	1.083	1.086	0.974	0.969	1.058	1.068	1.437	1.421
	60	1.017	1.019	1.064	1.057	0.978	0.974	1.036	1.030	1.230	1.230
	80	1.009	1.008	1.027	1.034	0.976	0.965	0.995	1.002	1.089	1.092
	V2V Car Perecentage (Lot-to-Bee Ratio = 1)	P-R	P-R (REG-NEG)	P-G	P-G (REG-NEG)	P-R (INS-NEG)	PL-R	PL-R (REG-NEG)	PL-G	PL-G (REG-NEG)	PL-R (INS-NEG)
	20	1.160	1.164	1.112	1.129	1.072	0.838	0.832	0.858	0.851	0.921
	40	0.968	1.000	0.963	0.959	0.834	0.765	0.757	0.764	0.753	0.778
	60	0.931	0.931	0.926	0.933	0.775	0.729	0.727	0.729	0.724	0.731
	80	0.912	0.896	0.901	0.889	0.739	0.707	0.711	0.709	0.705	0.689
	V2V Car Perecentage (Lot-to-Bee Ratio = 2)	P-R	P-R (REG-NEG)	P-G	P-G (REG-NEG)	P-R (INS-NEG)	PL-R	PL-R (REG-NEG)	PL-G	PL-G (REG-NEG)	PL-R (INS-NEG)
	20	0.935	0.935	0.935	0.934	0.935	0.912	0.911	0.914	0.908	0.940
	40	0.925	0.944	0.921	0.929	0.927	0.908	0.914	0.916	0.902	0.946
	60	0.936	0.930	0.919	0.925	0.915	0.904	0.915	0.908	0.903	0.928
	80	0.930	0.951	0.923	0.929	0.917	0.897	0.897	0.903	0.897	0.911

Table 35 (continued) – Relative Performance of all HoneyPark Variations in All Tested Scenarios

Real-Time Traffic Environment	Google Traffic Model (Lot-to-Bee Ratio = 0.75)	P-R	P-R (REG-NEG)	P-G	P-G (REG-NEG)	P-R (INS-NEG)	PL-R	PL-R (REG-NEG)	PL-G	PL-G (REG-NEG)	PL-R (INS-NEG)
	Pessimistic	-	-	-	-	-	0.898	-	-	-	-
	Best Guess	-	-	-	-	-	0.921	-	-	-	-
	Optimistic	-	-	-	-	-	0.930	-	-	-	-
	Google Traffic Model (Lot-to-Bee Ratio = 1)	P-R	P-R (REG-NEG)	P-G	P-G (REG-NEG)	P-R (INS-NEG)	PL-R	PL-R (REG-NEG)	PL-G	PL-G (REG-NEG)	PL-R (INS-NEG)
	Pessimistic	-	-	-	-	-	0.613	-	-	-	-
	Best Guess	-	-	-	-	-	0.625	-	-	-	-
	Optimistic	-	-	-	-	-	0.615	-	-	-	-
	Google Traffic Model (Lot-to-Bee Ratio = 2)	P-R	P-R (REG-NEG)	P-G	P-G (REG-NEG)	P-R (INS-NEG)	PL-R	PL-R (REG-NEG)	PL-G	PL-G (REG-NEG)	PL-R (INS-NEG)
	Pessimistic	-	-	-	-	-	0.668	-	-	-	-
	Best Guess	-	-	-	-	-	0.699	-	-	-	-
	Optimistic	-	-	-	-	-	0.745	-	-	-	-

Table 36 is an overall ranking of all the HoneyPark variations based on how they performed in the six scenarios presented in this chapter. Remember that the regular negative feedback is a redundant feature and therefore produces similar parking times to algorithms that have no negative feedback. This is why algorithms with no and regular negative feedback are grouped together in the rankings.

Table 36 – Overall Ranking of HoneyPark Algorithms

Rank	Algorithm	Explanation for Rank
1	Parked-Leave-Advert, Random-Scouting Algorithm with No or Regular Negative Feedback	The only set of algorithms that consistently performs better than the other algorithms given that the scenarios are not too extreme (i.e. too high/low lot-to-bee ratio). This is because the Parked-Leave-Advert feature allows the drivers to explore a larger group of profitable lots. The Random scouting algorithm works best as it explores a more diverse group of parking lots in the case where the simulation is not completely populated by HoneyPark cars
2	Parked-Leave-Advert, Greedy-Scouting Algorithm with No or Regular Negative Feedback	Perform as well as the algorithms ranked number one with the exception that the Greedy scouting algorithm doesn't perform as well in an environment that is not purely occupied with HoneyPark cars.
3	Parked-Leave-Advert, Random-Scouting Algorithm with Instantaneous Negative Feedback	For Parked-Leave-Advert algorithms, the instantaneous Negative feedback results in a slight improvement in some cases (i.e. lot-to-bee ratio = 1) and worse performance in others (i.e. lot-to-bee ratio = 0.75). This is because the Parked-Leave-Advert feature is already efficient in allocating lots, so adding instantaneous negative feedback doesn't lead to much improvement.
4	Parked-Advert, Random-Scouting Algorithm with Instantaneous Negative Feedback	The Parked-Advert feature only considers a smaller pool of profitable lots, which leads to much worse performance. Thus, it always helps to implement instantaneous negative feedback in Parked-Advert algorithms as significant time can be saved by redirecting drivers heading toward full lots to other lots that have not been considered by the algorithm.
5	Parked-Advert, Random-Scouting Algorithm with No or Regular Negative Feedback	The Parked-Advert feature leads to significantly worse parking times as it only considers a small group of profitable lots. When these profitable lots become full, it takes time for the algorithm to react to the change, leading to longer parking times.
6	Parked-Advert, Greedy-Scouting Algorithm with No or Regular Negative Feedback	Perform as well as the algorithms ranked number five with the exception that the Greedy scouting algorithm doesn't perform as well in an environment that is not purely occupied with HoneyPark cars.

In addition, the algorithm was tested in a setting where not all the drivers have V2V capabilities but can access the HoneyPark system and therefore, obtain lot recommendations using the HoneyPark algorithm. However, it seems that all the HoneyPark algorithms are ineffective in this scenario as the non-V2V drivers heavily disrupts the accuracy of the HoneyPark system by targeting lots deemed profitably by the HoneyPark algorithm without sending the system any adverts.

CHAPTER 6. CONCLUSION AND DISCUSSION

This chapter will give an overview of the findings presented in this thesis. It will start with a discussion of the two main factors proven to affect the average time taken to find parking: algorithm design and the parking environment. This will be followed by a section suggesting further work that can be done to further develop and improve the HoneyPark algorithm.

6.1 Overview

6.1.1 Algorithm Design

Obviously, algorithm design is an important factor in algorithmic performance. The findings in Chapter 5 have highlighted two important aspects that drastically affect the performance of the HoneyPark algorithm: range of lots explored and construction of negative feedback mechanism.

6.1.1.1 Range of Lots Explored

Chapter 5 shows that one of the keys to creating an effective parking algorithm is finding the balance between only considering sending the driver to a smaller group of profitable lots, but not making that group so small to the extent that the algorithm is trying to meet parking demand with a limited number of parking spaces. The narrowing of possible parking to a smaller group of more profitable parking lots is indeed beneficial. Simulation results shows that the HoneyPark algorithm consistently performs better than existing algorithms, namely Random and Greedy as it considers and are more likely to direct drivers to profitable lots where he or she are more likely to find a profitable lot. However, if the algorithm only identifies and send drivers to a small group of

profitable lots, such as in the case of Parked-Advert algorithms, it can also increase the amount of time it takes for the driver to find parking. Simulation results show that the volatility of the parking environment and the impact of the algorithm on parking profitability is responsible for this occurrence.

The parking environment is volatile as the profitability of a parking lot is constantly fluctuating due to movement of drivers and how they choose to park. As discussed in Chapter 5, a parking algorithm may identify a small, group of very profitable lots. The first few drivers they send to these lots spend minimal time looking for parking. However, the profitability of parking lot can change drastically. It is possible that the lots can be quickly occupied by other non-HoneyPark and non-V2V drivers. But the decline in profitability is often caused by the fact that the algorithm is parking more cars at that lot and as such, the lot is occupied at a faster rate. As the algorithm did not consider a larger number of options, simulation data shows that the moment that identified 'profitable' parking lots are full, such algorithms results in long parking times as it has sent a large number of drivers to look for parking in an unprofitable parking lot. It will take some time for the algorithm to readjust and find another group of profitable lot. In other words, the decay of positive feedback is unable to keep up with the decaying profitability of the parking lot. On the other hand, algorithms that look at a larger group of profitable parking lots will send fewer drivers to one particular lot. Therefore, the profitability of the parking lot decays at a lower rate that positive feedback is able to keep up with. In addition, as the algorithm is considering a larger number of parking lots, there are plenty of other backup options in the case that the most profitable lots are no longer available.

Therefore, it is important to strike a balance between identifying a select group of profitable parking lot and selecting a group of profitable lots that is large enough to keep up with the changing

profitability of parking lots. This is the reason why, according to simulation results, Park-Leave Adverts algorithms have been the most successful algorithms. They do have a process that uses advertisements to identify the most profitable lots. However, as they consider both Parked and Leave Adverts, the group of identified profitable lots is bigger and consequently, they are able to meet parking demand more efficiently.

6.1.1.2 Design of Negative Feedback Mechanism

The results in Chapter 5 shows that the design of negative feedback mechanism can have an impact on algorithmic performance. Throughout the entire chapter, the performance of regular negative feedback algorithms does not differ much from their non-negative feedback counterparts. As mentioned in Section 5.1, the design of regular negative feedback creates redundancy as it prevents cars from exploring lots that they are less likely to explore anyways. The impact of the instantaneous negative feedback mechanism is greater. In some cases, it shortens the average parking time as it prevents drivers from wasting time searching lots that are unprofitable. However in other cases, such as that drivers need to present at a full lot to successfully find an empty parking space (i.e. when parking demand exceeds parking supply), the mechanism can greatly hurt the driver's chances of success.

6.1.2 Parking Environment

The results in Chapter 5 shows that the relative performance of the HoneyPark algorithm is not only dependent not only on the algorithm design but also on the parking environment. Fundamentally, the results show that the main factors affecting algorithm performance is the relative amount of parking demand as compared to parking demand and the type of algorithms that other drivers are using.

6.1.2.1 Parking Demand vs. Parking Supply

Section 5.1 has explicitly shown that the relative difference between parking supply and demand has a large impact on algorithm performance. When parking demand is much lower than parking supply, there is not much variation in performance among the HoneyPark algorithms as it is very likely that a driver will find parking regardless of which lot he or she goes to. If the parking demand is exceptionally lower than parking supply, the Greedy algorithm, which underperforms in most situations, becomes the most effective algorithm as it simply directs the driver to the closest lot, which will very likely to have empty spaces.

When parking demand rivals parking supply, there is a difference in algorithmic performance among the HoneyPark variations as it is no longer certain that a driver will be able to get a parking spot at any parking lot. Therefore, the parking algorithm will have to be effective to identifying which lots are profitable and which lots are not. When parking demand exceeds parking supply, it is crucial that the algorithm explores all the lots possible due to the limited amount of parking spaces. Therefore, algorithms that only consider sending drivers to a small group of lots, such as Parked-Advert algorithms, do not perform as effectively in such situations. There is also an element of luck involved. There are so many drivers searching for parking at oversaturated lots that one has be present at a lot when a parking space opens up. If not, the parking space will quickly be taken by competing drivers. Therefore, the algorithms will need to take this 'luck' into consideration. Those who don't, such as in the case of instantaneous negative feedback algorithms, will see their performance drop.

6.1.2.2 Other Drivers

Section 5.5 and 5.6 also shows that algorithmic performance is also influenced by the parking algorithms that other drivers are using as well. One of the main factors that affects algorithmic performance in this regard is the competition for parking lots. For example, Section 5.5 shows that more drivers using the HoneyPark algorithms results in more drivers being sent to the identified profitable lots, increasing competition for those specific lots and resulting in longer parking times. Another factor to consider is whether the parking methods used by other drivers affect the accuracy of the data gathered by the HoneyPark system. Section 5.6 shows that the presence of non-V2V drivers using the HoneyPark system without updating it greatly lengthens parking for all the drivers. This is because it renders the data collected by the HoneyPark system inaccurate, especially on the lots that it considers profitable.

6.2 Potential Future Work

Based on the findings of this thesis, one avenue for further algorithm development is the creation of a HoneyPark variation that will perform in a mixture of V2V and non-V2V vehicles. This can be done by introducing an 'error' element in the advertisement system which acknowledges that the raw advertisement data may need correction. With this component, the HoneyPark algorithm may be able to still identify correctly which lots are more profitable and perform efficiently as result.

This thesis was only responsible for identifying the general trends that affected algorithm performance. As such, further investigations can be done to further detail the performance of the HoneyPark algorithm. One example is finding the true limits of the HoneyPark variations at which it starts or stops becoming effective. The thesis identified that negative feedback algorithms

performed worse when parking demand was higher than parking supply. But it did not identify exactly at what point (i.e. the exact lot-to-bee ratio) certain algorithms become more or less efficient. Therefore, more simulations can be conducted to find the numerical limit at which certain algorithms become more or less effective.

One could also look into further optimizing the HoneyPark algorithm by getting rid of some of the constraints imposed in the simulations analyzed in this thesis. For example, the simulations were programmed in such a way that only parking lots within a certain distance of the driver's destination is considered. The algorithm could be modified so that this distance is increased when all the parking lots within the current range is full, allowing the driver to explore other feasible drivers when all current options are exhausted.

Further investigation can be conducted regarding the feasibility of the HoneyPark algorithm in a real-world setting. Most of the environments created in this thesis was done on a theoretical basis and may not necessarily reflect realistic parking situations. For example, this thesis did investigate when parking demand was varied in relation to parking demand but did not look at what the lot-to-bee ratio is for the majority of parking situations that occur in daily life. Therefore, one could try to find real-world parking data and investigate whether the HoneyPark algorithm will fare under such circumstances.

All algorithms were tested in the same simulation setting. Only the road and parking infrastructure of San Francisco was used to evaluate algorithm performance and was modified in any way. Therefore, one could look at how the HoneyPark algorithm performs in a variety of parking infrastructures. This could include testing the algorithm in variety of cities or adding and taking away parking lots at strategic geographical points of the environment with this intention of

seeing the geographical effect on algorithmic performance. In addition, all algorithms were only tested using parking lots. The simulation environment did not include other types of parking such as street parking. Therefore, future tests can include a variety of parking spaces and examine if the HoneyPark algorithm is also effective in measuring the profitability of different kinds of parking spaces.

REFERENCES

Aytug, H., Khouja, M., & Vergara, F. E. (2003). Use of Genetic Algorithms to Solve Production and Operations Management Problems: A Review. *International Journal of Production Research, 41*(17), 3955-4009.

Biswas, S., Tatchikou, R., & Dion, F. (2006). Vehicle-to-vehicle wireless communication protocols for enhancing highway traffic safety. *IEEE Communications Magazine, 44,* 74-82.

Blum, C., & Li, X. (2008). Swarm Intelligence in Optimization *Swarm Intelligence* (pp. 43-85): Springer Berlin Heidelberg.

Bonabeau, E., Dorigo, M., & Theraulaz, G. (1999). *Swarm Intelligence: From Natural to Artificial Systems*. New York, NY: Oxford University Press.

Bucciol, P., Masala, E., & Martin, J. C. D. (2005). *Dynamic Packet Size Selection for 802.11 Inter-Vehicular Video Communications*. Paper presented at the 1st International Workshop on Vehicle-to-Vehicle Communication.

Chinrungrueng, J., Sunantachaikul, U., & Triamlumlerd, S. (2007). *Smart Parking: an Application of optical Wireless Sensor Network.* Paper presented at the Proceedings of the 2007 International Symposium on Applications and the Internet Workshops.

Crepaldi, R., Beavers, R., Ehrat, B., Sze, J., Jaeger, M., Biersteker, S., & Kravets, R. (2012). *LoadingZones: Leveraging Street Parking to Enable Vehicular Internet Access*. Paper presented at the CHANTS'12 - 7th ACM International Workshop on Challenged Networks, Istanbul, Turkey.

Darwin, C. (1859). *On the Origin of Species*: John Murray.

Dorigo, M., Birattari, M., & Stutzle, T. (2006). Ant colony optimization. *IEEE Computational Intelligence Magazine, 1,* 28-39.

Dorigo, M., & Gambardella, L. M. (1997). Ant colony system: a cooperative learning approach to the traveling salesman problem. *IEEE Transactions on Evolutionary Computation, 1*(1), 53-66.

Grassé, P. P. (1946). *Les insects dans leur univers*. Paris: Ed. du Palais de la d´ecouverte.

Grossberg, S. (1988). Nonlinear Neural Networks: Principles, Mechanisms, and Architectures. *Neural Networks, 1*, 17-61.

Heppner, F. H., & Grenander, U. (1990). A Stochastic Nonlinear Model for Coordinate Bird Flocks. In S. Krasner (Ed.), *The Ubiquity of Chaos* (pp. 233-238): American Association for the Advancement of Science.

Idris, M. Y. I., Leng, Y. Y., Tamil, E. M., Noor, N. M., & Razak, Z. (2009). Car Park System: A Review of Smart Parking System and its Technology. *Information Technology Journal, 8*(2), 101-113.

Kar, A. K. (2016). Bio inspired computing – A review of algorithms and scope of applications. *Expert Systems With Applications, 59*, 20-32.

Karaboga, D., & Akay, B. (2009). A comparative study of Artificial Bee Colony algorithm. *Applied Mathematics and Computation, 214*(1), 108-132.

Karaboga, D., & Basturk, B. (2007). A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm. *Journal of Global Optimization, 39*(3), 459-471.

Kennedy, J., & Eberhart, R. (1995). *Particle Swarm Optimization*. Paper presented at the IEEE International Conference on Neural Networks, Perth, WA, Australia.

Kennedy, J., Eberhart, R. C., & Shi, Y. (2001). *Swarm Intelligence*. San Mateo, CA: Morgan Kaufmann.

Khang, S. C., Hong, T. J., Chin, T. S., & Wang, S. (2010). *Wireless Mobile-based Shopping Mall Car Parking System (WMCPS)*. Paper presented at the IEEE Asia-Pacific Services Computing Conference, Hangzhou, China.

LaMorte, W. W. (2017). Mann Whitney U Test (Wilcoxon Rank Sum Test). Retrieved from http://sphweb.bumc.bu.edu/otlt/mph-modules/bs/bs704_nonparametric/BS704_Nonparametric4.html

Liu, Z., & Cai, Y. (2005). *Sweep based multiple ant colonies algorithm for capacitated vehicle routing problem*. Paper presented at the IEEE International Conference on e-Business Engineering, Beijing, China.

Lu, N., Cheng, N., Zhang, N., Shen, X., & Mark, J. W. (2014). Connected Vehicles: Solutions and Challenges. *IEEE Internet of Things Journal, 1*(4), 289-299.

Millonas, M. M. (1994). Swarms, phase transitions, and collective intelligence. In M. Palaniswami, Y. Attikiouzel, R. Marks, D. Fogel, & T. Fukuda (Eds.), *Computational Intelligence: A Dynamic System Perspective* (pp. 137-151): IEEE.

Mimbela, L. E. Y., & Klein, L. A. (2000). *A Summary of Vehicle Detection and Surveillance Technologies used in Intelligent Transportation Systems*. Retrieved from https://www.fhwa.dot.gov/policyinformation/pubs/vdstits2007/vdstits2007.pdf

Nadeem, T., Dashtinezhad, S., Liao, C., & Iftode, L. (2004). TrafficView: Traffic Data Dissemination using Car-to-Car Communication. *Mobile Computing and Communications Review, 8*, 6-19.

Nakrani, S., & Tovey, C. (2007). From honeybees to Internet servers: biomimicry for distributed management of Internet hosting centers. *Bioinspiration & Biomimetics, 2*(4), 182-197.

Pham, D. T., Ghanbarzadeh, A., Koç, E., Otri, S., Rahim, S., & Zaidi, M. (2006). *The Bees Algorithm — A Novel Tool for Complex Optimisation Problems*. Paper presented at the 2nd I*PROMS Virtual International Conference.

Reynolds, C. W. (1987). Flocks, Herds, and Schools: A Distributed Behavioral Model. *Computer Graphics, 21*(4), 25-34.

Rizzoli, A. E., Oliverio, F., Montemanni, R., & Gambardella, L. M. (2007). Ant colony optimization for real-world vehicle routing problems. *Swarm Intelligence, 1*, 135-151.

Sadegh, N. (1993). A perceptron network for functional identification and control of nonlinear systems. *IEEE Transactions on Neural Networks, 4*(6), 982-988.

Sato, T., & Hagiwara, M. (1997). *Bee System: Finding Solution by a Concentrated Search*. Paper presented at the IEEE International Conference On Systems, Man, And Cybernetics, Orlando, FL.

Seeley, T. D., Camazine, S., & Sneyd, J. (1991). Collective Decision-Making in Honey Bees: How Colonies Choose among Nectar Sources. *Behavioral Ecology and Sociobiology, 28*(4), 277-290.

Seely, T. D. (1995). *The wisdom of the hive: the social physiology of honey bee colonies*. Cambridge, Massachusetts: Harvard University Press.

Shoup, D. C. (2005). *The High Cost of Parking*. Chicago: American Planners Association.

Shoup, D. C. (2006). Cruising for parking. *Transport Policy, 13*(6), 479-486.

Sichitiu, M. L., & Kihl, M. (2008). Inter-vehicle communication systems: a survey. *IEEE Communications Surveys & Tutorials, 10*(2).

Srinivasan, D., Loo, W. H., & Cheu, R. L. (2003). *Traffic incident detection using particle swarm optimization*. Paper presented at the Swarm Intelligence Symposium, Indianapolis, IN.

Tang, V. W. S., Zheng, Y., & Cao, J. (2006). *An Intelligent Car Park Management System based on Wireless Sensor Networks*. Paper presented at the First International Symposium on Pervasive Computing and Applications, Urumqi, China.

Tasseron, G., Martens, K., & Heijden, R. v. d. (2015). The Potential Impact of Vehicle-to-Vehicle and Sensor-to-Vehicle Communication in Urban Parking. *IEEE Intelligent Transportation Systems Magazine, 7,* 22-33.

Teodorović, D., & Dell'Orco, M. (2005). Bee colony optimization - A cooperative learning approach to complex transportation problems. *16th Mini-EURO Conference on Advanced OR and AI Methods in Transportation*, 51-60.

Tomar, P., Kaur, G., & Singh, P. (2017). A Prototype of IoT-Based Real Time Smart Street Parking System for Smart Cities *Internet of Things and Big Data Analytics Toward Next-Generation Intelligence* (pp. 243-263): Springer.

Verroios, V., Efstathiou, V., & Delis, A. (2011). *Reaching Available Public Parking Spaces in Urban Environments using Ad-hoc Networking*. Paper presented at the 12th IEEE International Conference on Mobile Data Management, Lulea, Sweden.

Wang, H., & He, W. (2011). *A Reservation-based Smart Parking System*. Paper presented at the IEEE Conference on Computer Communications Workshops, Shanghai, China.

Wilson, E. O. (1975). *Sociobiology: The New Synthesis*. Cambridge, MA: Harvard University Press.

Yang, X., Liu, L., Vaidya, N. H., & Zhao, F. (2004). *A vehicle-to-vehicle communication protocol for cooperative collision warning*. Paper presented at the The First Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services, Boston, MA.

Yau, C. Mann-Whitney-Wilcoxon Test. Retrieved from http://www.r-tutor.com/elementary-statistics/non-parametric-methods/mann-whitney-wilcoxon-test

Zhao, J., Jia, L., Chen, Z., & Wang, X. (2006). *Urban traffic flow forecasting model of double RBF neural network based on PSO*. Paper presented at the Sixth International Conference on Intelligent Systems Design and Applications, Jinan, China.

Zhu, Y., Liu, X., Li, M., & Zhang, Q. (2013). POVA: Traffic Light Sensing with Probe Vehicles. *IEEE Transactions on Parallel and Distributed Systems, 24*(7), 1390-1400.