An Auction Algorithm for Optimal Satellite Refueling

Alexandros Salazar Panagiotis Tsiotras

School of Aerospace Engineering, Georgia Institute of Technology
Atlanta, Georgia, 30332-0150

ABSTRACT

Satellite refueling can extend the lifetime of satellite constellations. Peer-to-peer satellite refueling in particular has the potential to make the most efficient use of the fuel contained in the constellation by redistributing it as on a need-to basis. In this paper we present an alternative implementation of P2P refueling, namely pairing up fuel sufficient satellites with fuel deficient satellites so that they can refuel each other and guarantee that each satellite has more than a certain baseline amount of fuel. To solve this problem we make use of the auction algorithm, which is implemented in a distributed context. Asynchronous bids have been used to illustrate the robustness of the algorithm.

1.0 Introduction

As both government and private groups seek to enhance performance and reduce costs of space systems, satellite constellations are emerging as an attractive alternative to large, multi-million dollar satellites. Their inherent distributed nature provides two advantages: first, a measure of robustness because functions can be implemented redundantly on more than one satellite, and second the ability–in case of failure–of replacing only the satellite that failed, instead of the entire constellation. Moreover, in addition to being able to perform tasks traditionally assigned to a single satellite, such as internet routing,[1] satellite constellations can be used in novel ways to solve otherwise difficult problems, such as altimetry[2] or missile warning and defense.[3]

With this new paradigm comes a whole slew of challenges, including control, deployment, and initialization. Of particular interest both for civilian and for military applications is the refueling problem. It is clear that the ability to refuel a satellite can extend its usefulness. However, in the case of a satellite designed to perform many complex functions, failure in one of the functions can cause the entire satellite to become unusable. In this case, replacement can be very costly, and servicing equally so. Therefore, the usefulness of refueling is limited, since every satellite has a predicted life-cycle, and can at worst be supplied with enough fuel on launch to see it through that time period. By contrast, in the case of a constellation, individual satellites may fail or become obsolete, but they can be replaced without the need to decommission the entire constellation. Since a constellation has potentially unlimited usefulness (subject to periodic upgrades), refueling becomes essential.

The current work on peer-to-peer satellite refueling is mainly found in Refs. 4–9. Therein, the authors addressed optimal scheduling for external refueling,[4] and later peer-to-peer refueling within a constellation.[5,6,9] The latter three papers focused on equalizing the fuel distribution within the constellation by exchanging fuel between the satellites in the constellation. The goal was to minimize the 1-norm of the deviation of the satellites' fuel content from the constellation average fuel. In Refs. 7,8 the authors showed that P2P refueling can be more fuel-efficient than a single vehicle strategy for refueling a large number of satellites.

More recently, the authors of the present paper presented the idea of using an auction algorithm to match satellites in a peer-to-peer refueling scheme whose goal is to guarantee that all satellites have more than a certain baseline amount of fuel, which may be different for each satellite.[10] The auction algorithm is advantageous for two reasons when solving the satellite refueling problem: firstly, it can be implemented in a decentralized framework, and secondly it is robust to asynchronous bids, which can occur as a result of poor communication, temporary failures, or other problems. In this paper, we verify these attractive properties of auctions via simulation results.

Section 2.0 briefly reviews the formulation of the peer-to-peer (P2P) refueling problem as presented in Ref. 10. Section 3.0 presents an asynchronous formulation of the auction algorithm. It is a simplified form the algorithm presented in Ref. 11 which nonetheless is sufficient to model disturbances likely to arise in the space environment. Section 4.0 presents the simulation results for a parallel implementation of the algorithm, both with and without communication problems being modeled. Special attention is being paid to the effect of such communication problems on the speed of convergence of the algorithm. Finally, Section 5.0 summarizes our results.

2.0 The Peer-to-peer Assignment Problem

The formulation we will be using in this paper was put forward in Ref. 10 as a more realistic alternative to the formulation in Refs. 5,6,9. We present the main idea again below, without including the details of the derivation, which can be found elsewhere.[10]

Consider a satellite constellation $\mathcal{C}$ consisting of N satellites $s_1, s_2, \ldots, s_N$. We will call the satellite s_k *fuel sufficient* if

$$f_k^- > \underline{f}_k, \tag{1}$$

where f_k^- denotes the fuel content of the satellite prior to refueling, and $\underline{f}_k$ denotes the minimum fuel required for the satellite to be operational until the next external refueling. Otherwise, the satellite will be called *fuel deficient*. To differentiate fuel deficient satellites from fuel sufficient satellites, we define the index set of fuel deficient satellites, indexed by i, as the set $\mathcal{D} = \{i : i = 1, 2, \ldots, m\}$, and the index set of fuel sufficient satellites, indexed by j as the set $\mathcal{S} = \{j : j = 1, 2, \ldots, n\}$. In what follows, the subscript i will always denote a fuel deficient satellite, and the subscript j will always denote a fuel sufficient satellite.

The peer-to-peer refueling problem consists of finding a refueling strategy that will ensure, for all $s_k \in \mathcal{C}$, that

$$f_k^+ \geq \underline{f}_k, \tag{2}$$

while minimizing fuel consumption during the ensuing rendezvous. Here f_k^+ denotes the fuel content of

satellite s_k after the refueling transaction is completed[a].

The sets $\mathcal{D}$ and $\mathcal{S}$ induce a natural bi-partition on the set $\mathcal{C}$. Given the index sets $\mathcal{D}$ and $\mathcal{S}$ we may define a bi-partite graph $\mathcal{G} = \{\mathcal{D} \cup \mathcal{S}, \mathcal{E}\}$ over $\mathcal{C}$, where the set of edges $\mathcal{E} = \{(i,j) : i \in \mathcal{D}, j \in \mathcal{S}\}$ in the graph has as vertices pairs of fuel sufficient and fuel deficient satellites. If there are no other operational constraints, $\mathcal{G}$ is a complete bipartite graph.

For each rendezvous between a fuel sufficient satellite and a fuel deficient satellite, we will assume that only one is *active*, namely only one initiates the rendezvous, and returns to its original slot after refueling. The other satellite remains *passive* during the fuel transaction. We can then partition $\mathcal{E}$ into three subsets as follows

$$\mathcal{A} = \{(i,j) \in \mathcal{E} : i \text{ is active}\}, \tag{3a}$$
$$\mathcal{P} = \{(i,j) \in \mathcal{E} : i \text{ is passive}\}, \tag{3b}$$
$$\mathcal{U} = \{(i,j) \in \mathcal{E} : (i,j) \text{ is infeasible}\}, \tag{3c}$$

where an infeasible pair is defined as one for which neither of the two satellites can be active. Note that $\mathcal{A} \cap \mathcal{P} \neq \varnothing$ to allow for the case when both satellites i and j may be active. Define the sets

$$\mathcal{U}_1 = \{(i,j) : p_{ij}^i > f_i^-\} \tag{4}$$
$$\mathcal{U}_2 = \{(i,j) : p_{ij}^j > f_j^-\} \tag{5}$$
$$\mathcal{U}_3 = \{(i,j) : f_i^- + f_j^- - c_{ij} < \underline{f}_i + \underline{f}_j\}, \tag{6}$$

where p_{ij}^i is the fuel for the (active) satellite i to perform the one-way trip to the (passive) satellite j, and similarly for p_{ij}^j, while c_{ij} is the minimum amount of fuel required to perform the rendezvous between satellites i and j. Details for the calculation of these quantities is given in Ref. 10. We simply note that the set of infeasible pairs in (3c) is then given by $\mathcal{U} = \bigcup_{k=1}^{3} \mathcal{U}_k$. Moreover it can be shown that the quantities $c_{ij}, p_{ij}^i, p_{ji}^i$, etc. can be computed explicitly in terms of the problem data as follows:[10]

$$c_{ij} = \min\{p_{ij}^i + p_{ji}^i, p_{ji}^j + p_{ij}^j\}, \tag{7}$$
$$p_{ij}^i = (m_i + f_i^-)\left(1 - e^{-\Delta V_{ij}^i/\sigma_i}\right), \tag{8}$$
$$p_{ji}^i = (m_i + \underline{f}_i)\left(1 - e^{-\Delta V_{ji}^i/\sigma_i}\right) e^{\Delta V_{ji}^i/\sigma_i}, \tag{9}$$
$$p_{ji}^j = (m_j + f_j^-)\left(1 - e^{-\Delta V_{ji}^j/\sigma_j}\right), \tag{10}$$
$$p_{ij}^j = \begin{cases} (m_j + f_j^- - p_{ji}^j - \bar{f}_i + f_i^-)\left(1 - e^{-\Delta V_{ij}^j/\sigma_j}\right), \\ \qquad \text{if} \quad g_{ji}^* = \bar{f}_i - f_i^-, \\ (m_j + \underline{f}_j)\left(1 - e^{-\Delta V_{ij}^j/\sigma_j}\right) e^{\Delta V_{ij}^j/\sigma_j}, \\ \qquad \text{if} \quad g_{ji}^* = f_j^- - \underline{f}_j - p_{ij}^j, \end{cases} \tag{11}$$

[a] A refueling transaction includes the forward and return orbital transfers, as well as the actual exchange of fuel between the two satellites.

where ΔV_{ij}^{i} denotes the velocity impulse required for satellite i to move from its orbital slot to that of satellite j, and similarly for $\Delta V_{ji}^{i}, \Delta V_{ji}^{j}$, and ΔV_{ij}^{j}. The parameter σ_i is given by $\sigma_i = g_0 I_{\text{sp}i}$, with $I_{\text{sp}i}$ being the specific impulse of the satellite and g_0 the acceleration at the Earth's surface. Here g_{ji}^{*} indicates the optimal amount of fuel to transfer from satellite j to satellite i upon rendezvous. It is chosen based on the capacity of the fuel deficient satellite: if satellite i cannot accept all the fuel satellite j can transfer to it, given by $g_{ji}^{*} = f_j^{-} - \underline{f}_j - p_{ij}^{j}$, satellite j will only transfer $g_{ji}^{*} = \bar{f}_i - f_i^{-}$.

With the above definitions, we can create a subgraph $\mathcal{G}_f = \{\mathcal{D} \cup \mathcal{S}, \mathcal{E}_f\}$ of $\mathcal{G}$ called the *feasible constellation graph* that consists of the same vertices as $\mathcal{G}$ but only the edges $\mathcal{E}_f$ of $\mathcal{G}$ such that $(i,j) \notin \mathcal{U}$. Clearly, $\mathcal{E}_f = \mathcal{A} \cup \mathcal{P}$. Now, we can weigh each edge $(i,j) \in \mathcal{E}_f$ by $-c_{ij}$, and define $\mathcal{N}(i) = \{j \in \mathcal{S} : (i,j) \in \mathcal{E}_f\}$ as the set of fuel sufficient satellites that can perform a fuel transaction with $i \in \mathcal{D}$. Since we only allow one fuel exchange per satellite, the final result of a P2P refueling strategy is a matching $\mathcal{M} \subset \mathcal{E}_f$ such that the total fuel incurred during the associated rendezvous of the satellite pairs in $\mathcal{M}$ is minimized. By matching here we mean a subset of edges in $\mathcal{E}_f$ such that no two edges share the same vertex. Our problem is then to find the matching that maximizes the weights of the edges contained in it, hence minimizing the total fuel cost, subject to the constraint that each fuel-deficient satellite is matched to exactly one fuel-sufficient satellite.

The problem of finding the optimal matching $\mathcal{M}^{*}$ can then be formulated as a integer program over the bi-partite graph $\mathcal{G}_f$ as follows

$$\text{Maximize} \quad -\sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij} x_{ij} \tag{12}$$

$$\text{Subject to:} \quad \sum_{j=1}^{n} x_{ij} = 1, \quad \forall\, i \in \mathcal{D}, \tag{13}$$

$$\sum_{i=1}^{m} x_{ij} \leq 1, \quad \forall\, j \in \mathcal{S}, \tag{14}$$

$$x_{ij} = 1 \implies j \in \mathcal{N}(i), \quad i \in \mathcal{D}. \tag{15}$$

$$x_{ij} \in \{0,1\}, \quad \forall\, i,j. \tag{16}$$

Inequality (13) enforces the condition that every fuel deficient satellite must be paired with exactly one fuel sufficient satellite. Inequality (14) enforces the condition that every fuel sufficient satellite can be paired with at most one fuel deficient satellite. Equation (15) simply states that all pairs considered are feasible. Note that for a problem to be feasible, necessarily $n \geq m$.

The integrality constraint on x_{ij} is not needed since it is known that assignment problems always have integral solutions.[12] In other words, one may replace the condition (16) by $x_{ij} \geq 0$ without loss of generality, and be left with a linear program rather than an integer program to solve.

3.0 The Asynchronous Auction Algorithm

Problems of the form presented in (12)-(16) are known as *assignment problems*, where persons in one set must be assigned to objects in another set on a one-to-one basis. If the number of persons is equal to

the number of objects, the problem is termed *symmetric*. Otherwise, it is termed *asymmetric*. The P2P refueling problem is an asymmetric assignment problem since, in general, $m \neq n$. The assignment problem is very well known in the literature, and it can be solved either via the standard simplex algorithm,[12] or via more specialized methods such as Kuhn's Hungarian Method (for symmetric problems) that take advantage of the special structure of the problem. Because of their robustness to asynchronous bids when properly formulated, we are especially interested in auction algorithms.[11]

Auction algorithms solve the assignment problem consisting of m persons seeking to be assigned to n objects one a one-to-one basis (assuming $m \leq n$) in which each person bids for the object of its choice, and the objects choose the best suited person from the bidders. Subject to certain conditions on the bids, these algorithms solve the assignment problem efficiently and within arbitrarily given optimality bounds. A thorough discussion of auction algorithms and their applications to transportation problems and network flow problems is given in Ref. 13. The discussion in the latter paper centers around the serial implementation of the auction algorithm. In the current paper, however, we present a parallel and more robust version of the auction algorithm. This algorithm is a simplified version of a general asynchronous distributed algorithm that is known to have excellent performance.[11]

3.1. Overview of the Algorithm

Consider a set of m persons $\mathcal{D} = \{i : i = 1, \ldots, m\}$ (in our case the fuel deficient satellites) whose elements are to be assigned to those of a set of n objects $\mathcal{S} = \{j : j = 1, \ldots, n\}$ (in our case the fuel sufficient satellites) on a one-to-one basis, where $m \leq n$, such that $(i, j) \in \mathcal{E}_f$, where $\mathcal{E}_f$ denotes the set of all allowable pairs. For every person i the set $\mathcal{N}(i)$ consists, as above, of all objects that person i can be assigned to. For each object $j \in \mathcal{N}(i)$ there is a *benefit* a_{ij} for matching person i with object j (in our case the benefit is $-c_{ij}$). The objective is to find the person/object pairs (i, j_i) such that all persons are assigned to exactly one object and that the total benefit $\sum_{i \in \mathcal{D}} a_{ij_i}$ is maximized among all possible person/object pairs.

The standard synchronous auction algorithm works by assigning a *price* π_j to every object, with the price potentially changing at each iteration, and then having each person bid on the object that it finds most desirable by *value*, which is defined as the difference between benefit and price. MA person i bids on an object j_i if

$$a_{ij_i} - \pi_{j_i} = \max_{j \in \mathcal{N}(i)} \{a_{ij} - \pi_j\}, \tag{17}$$

which is called the *complementary slackness* or CS condition. Objects will then take the bids they receive from each person, select the highest bidder, and be assigned to that person until a better bid comes along. When all persons are assigned, the algorithm terminates.

The brief overview given above is a naive implementation of the algorithm. Because of the tight CS condition, the algorithm may cycle. In addition, it assumes perfect information knowledge on the part of each satellite regarding the price vector $\pi = [\pi_1, \pi_2, \ldots, \pi_n]$. In order to prevent cycling and model communication problems, we discuss next the more generalized *asynchronous auction algorithm* which was first presented in Ref. 11. The algorithm presented therein is very general, and we simplify the discussion in our presentation. Specifically, we assume that all satellites that are unassigned are able to bid every time a bid can be made.

We assume that time is discretized by bids: events (changes in the state of the assignments) occur only at the end of the bidding and assignment calculations. The calculations themselves are referred to as iterations. At the beginning of an iteration, we denote by $\mathcal{V}(t) \subseteq \mathcal{D}$ the set of persons that are unassigned at time t. We assume that all persons in $\mathcal{V}(t)$ are capable of bidding at each iteration, but we do not assume that the members of $\mathcal{V}(t)$ have up-to-date pricing information. Instead, we denote the pricing information at time t as $\pi(t)$, a vector of length n, where $\pi_j(t)$ is the price associated with object $j \in \mathcal{S}$ at time t, and we assume that each person $i \in \mathcal{V}(t)$ is aware of a price vector $\pi(\tau_i(t))$ for some $\tau_i(t) \leq t$ which has the property that $\tau_i(t) \to \infty$ as $t \to \infty$. Finally, we define $r_j(t)$ as the index of the person that object j is assigned to at time t.

It is a standard result in linear programming[12] that an optimal assignment satisfies the CS condition. However, as mentioned earlier, strict application of the CS condition in the auction algorithm can lead to cycling,[13] so a relaxation of the CS condition, namely the *ε-complementary slackness* condition, or ε-CS condition for short, is enforced. Specifically, for each pair (i, j_i) in an assignment at time t, the following holds:[11]

$$a_{ij_i} - \pi_{j_i}(t) \geq \max_{j \in \mathcal{N}(i)} \{a_{ij} - \pi_j(t)\} - \varepsilon, \tag{18}$$

where $\varepsilon > 0$. We now state the algorithm, in an exposition that closely follows that of Ref. 11.

INITIALIZATION: Set $t = 0$. At the beginning of the algorithm, no persons are assigned to any objects, so $\mathcal{V}(t) = 0$, and $\pi(0) = 0$. At each subsequent iteration, all assigned pairs satisfy (18).

TERMINATION: The algorithm terminates when $\mathcal{V}(t) = \varnothing$.

BIDDING: Each person $i \in \mathcal{V}(t)$ calculates the maximum value among all objects $j \in \mathcal{N}(i)$,

$$v_i(t) = \max_{j \in \mathcal{N}(i)} \{a_{ij} - \pi_j(\tau_i(t))\}, \tag{19}$$

an object that yields that maximum value,

$$j_i(t) = \arg \max_{j \in \mathcal{N}(i)} \{a_{ij} - \pi_j(\tau_i(t))\}, \tag{20}$$

the second best value among all objects $j \in \mathcal{N}(i)$,

$$w_i(t) = \max_{\substack{j \in \mathcal{N}(i) \\ j \neq j_i(t)}} \{a_{ij} - \pi_j(\tau_i(t))\}, \tag{21}$$

and a bid for object j_i

$$\beta_i(t) = a_{ij_i(t)} - w_i(t) + \varepsilon. \tag{22}$$

Each person then submits their bid to the appropriate object. If $j_i(t)$ is the only object in $\mathcal{N}(i)$, then we define $w_i(t) = -\infty$, and subsequently $\beta_i(t) = +\infty$.

ASSIGNMENT: Each object receives bids from a (possibly empty) set of persons $B_j(t) = \{i \in \mathcal{V}(t) : j_i(t) = j\}$. If $B_j(t) \neq \varnothing$, each object j determines the highest bid

$$b_i(t) = \max_{i \in B_j(t)} \beta_i(t), \tag{23}$$

and a person submitting such a bid,

$$i_j(t) = \arg \max_{i \in B_j(t)} \beta_i(t). \tag{24}$$

Object j then updates the pair $(\pi_j(t), r_j(t))$ according to the following rule:

$$(\pi_j(t+1), r_j(t+1)) = \begin{cases} (b_j(t), i_j(t)) & \text{if } b_j(t) \geq \pi_j(t) + \varepsilon \\ (\pi_j(t), r_j(t)) & \text{otherwise.} \end{cases} \tag{25}$$

Since the above algorithm is a special case of the totally asynchronous algorithm discussed in Ref. 11, the results shown therein hold. In particular:

1. The algorithm is guaranteed to terminate if the problem is feasible.
2. The final assignment benefit is guaranteed to be within $m\varepsilon$ of the optimal assignment benefit, where $m = |\mathcal{D}|$.

The only issue remaining is that of feasibility. Sadly, there is no *a priori* guarantee that a given problem is feasible. If the problem is infeasible and there are no methods for detecting it, the algorithm will go on indefinitely, since the set $\mathcal{V}(t)$ will never be empty. Fortunately, several ways of detecting infeasibility exist, all of which are easy to implement assuming *some* foreknowledge by the satellites of the fuel capacity of the whole constellation. For example, there is a minimum bound on $v_i(t)$ for any feasible problem given by

$$v_i(t) \geq -(2n-1)C - (n-1)\varepsilon, \tag{26}$$

for all $i \in \mathcal{D}$ and for all $t \geq 0$, in case the algorithm is initialized with $\pi_j(0) = 0$, where $C = \max_{(i,j) \in \mathcal{E}_f} |a_{ij}|$. Since for an infeasible problem, at least one object will receive an infinite number of bids, at least one $v_i(t)$ will drop below this lower bound. Substituting C with the total fuel capacity of the constellation $\sum_k \bar{f}_k$ (where $\bar{f}_k$ is the maximum fuel capacity of the k-th satellite in constellation), we have a condition that can easily be assumed known *a priori* by each satellite and moreover is an upper bound on C, since no feasible cost can be greater than the total fuel content of the constellation.

4.0 Parallel and Asynchronous Implementation

In the above presentation, we assumed that all satellites in $\mathcal{V}(t)$ bid at every iteration. Borrowing terminology from the theory of nonlinear numerical optimization,[14] this is known as the Jacobi implementation. The serial algorithm, where only *one* of the satellites in $\mathcal{V}(t)$ bids at any given time, is said to be of Gauss-Seidel type. Intermediate versions, where a subset of $\mathcal{V}(t)$ bids each time, are known as block-Gauss-Seidel.[11,14]

The particular instance of the asynchronous auction algorithm described above is not the most general asynchronous auction algorithm. The version in Ref. 11 allows for only subsets of $\mathcal{V}(t)$ to bid (in other words, it allows block-Gauss-Seidel implementations), as well as for the price data for computing the maximum value and that for computing the second maximum value to be different (say $\tau_{i_1}(t)$ and $\tau_{i_2}(t)$).

For the purposes of validating the use of the auction algorithm, we used a Jacobi implementation, where every unassigned person bids at every iteration. The main variable was the updating of the prices: in order to simulate slow or defective communications, the algorithm stores the time history of the price vector $\pi(t)$. At each step, there is a fixed probability that every satellite will get the latest vector, implemented in the form of a random number with a threshold. If the random number is below the threshold, person (satellite) i obtains the latest price vector. If it is above the threshold, person i keeps its current price vector. This guarantees that $\tau_i(t) \to \infty$ as $t \to \infty$, while simulating random communications problems amongst the satellites. We also note that with probability zero, no updating happens for the prices, and the algorithm will never terminate, while with probability one, we have a perfect information parallel algorithm.

It is implicit in our implementation that there is one (or more) satellites that are organizing the data and time the request for bids. This satellite is the one that generates the events by which we discretized time in Section 3.0, and has the additional job of keeping track of how many objects have been assigned to a person. Since once an object is assigned, it remains assigned, this "central" satellite will declare the algorithm terminated when m objects are assigned.

4.1. Validation of the Asynchronous Algorithm

In order to validate the proposed algorithm, we compared it to the problem we solved in Ref. 10, which consisted of twenty evenly spaced satellites in the same circular orbit. The fuel content data for the constellation is reproduced in Table 1. The matching produced by the Gauss-Seidel type algorithm used in Ref. 10 is shown in Figure 1. The algorithm took 108 iterations to converge to a near optimal solution with a cost of 107.565 units of fuel, where the units are normalized as described in Ref. 10. The relaxation factor was chosen as $\varepsilon = 1/(|\mathcal{D}|+1)$, which guarantees that the matching will be within 1 fuel unit of the optimal cost.

Table 1. Satellite fuel specifics for the constellation in Ref.[10]

Satellite	f_k^-	$\underline{f}_k$	Satellite	f_k^-	$\underline{f}_k$
1	12	30	11	44	30
2	9	30	12	75	30
3	26	30	13	52	30
4	0	30	14	97	30
5	23	30	15	80	30
6	29	30	16	58	30
7	29	30	17	82	30
8	23	30	18	48	30
9	13	30	19	60	30
10	14	30	20	95	30

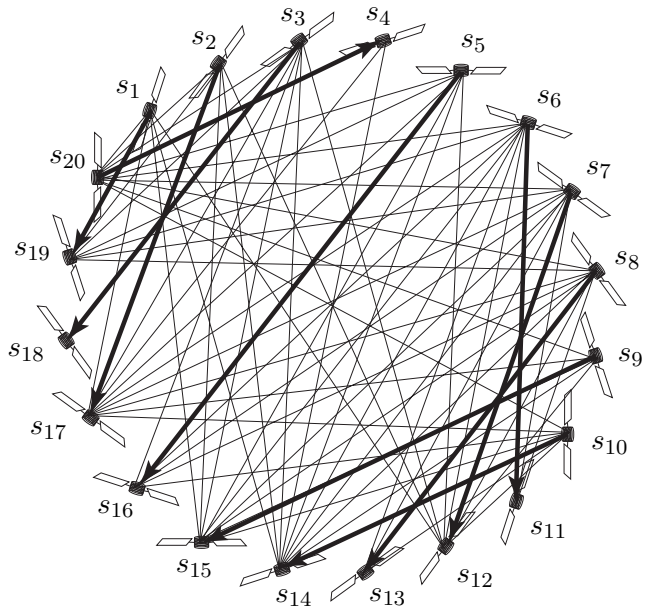

Figure 1. The constellation graph for the example from Ref. 10. The bold lines indicate the final optimal pairings. The thin lines indicate all feasible pairs. The arrows point from the active to the passive satellite. Serial implementation of the algorithm. No information loss.

Running the code in parallel yielded a bit of a surprise: after taking only 40 iterations, the code converged to a different matching. As shown in Figure 2, the satellites assigned to satellites 9 and 10 were switched, as were those for satellites 6 and 7. This matching had a cost of 107.5772, which, though a little higher than that of the first algorithm, is still well within the tolerance guaranteed by the algorithm. Moreover, running the asynchronous algorithm with different randomly generated rates of update (from 10% to 90% probability of data loss) yielded additional near-optimal matching matchings, all within tolerance. The best matching obtained was the one shown in Figure 3, with only the assignments for 9 and 10 being switched relative to the Gauss-Seidel type solution. The cost of this matching was 107.5523, and the matching was obtained after 48 iterations, with a price update probability of 0.9.

In addition to confirming that the auction algorithm converges reliably to a near-optimum solution, the above tests show that there is no *a priori* matching that the algorithm will converge to, and that randomization of the price update laws may affect the final result. However, the optimal cost will always be within $m\varepsilon$ of the optimal one. This is evident in our results since all the costs were within 0.1 fuel units of each other, which is significantly less than the 1 fuel unit of tolerance guaranteed by the theory. This is significant from a practical standpoint, since the algorithm is very fast even for a medium-sized constellation of 20 satellites: it can be run several times, simulating communication problems even if there are none, and then choose the best matching from those calculated. It is even possible that some matchings may have advantages extraneous to the cost (such as proximity) that will make it useful to know alternative matchings that are guaranteed to be optimal within a given fuel tolerance.

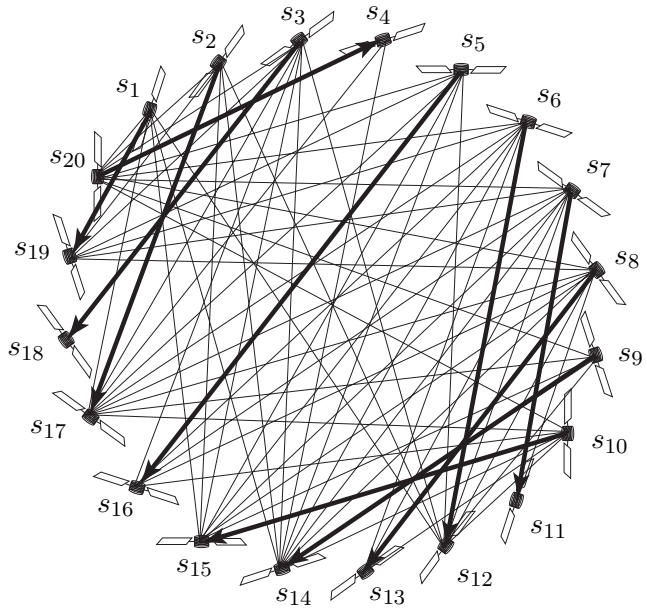

Figure 2. The constellation graph with the matching from the parallel version of the algorithm.

4.2. Performance Results of Asynchronous Bids

To get a better idea of the response of the auction algorithm to asynchronous bids, we ran two separate tests. First, we ran a test with the same base case as above and averaged the number of iterations for 1000 runs of the algorithm at $0.1, 0.2, \ldots, 0.9$ probability of price vector update. Mathematically, consider the following definition of $\tau_i(t)$:

$$\tau_i(t+1) = \begin{cases} t+1 & \text{if } X(t) \leq P_i(t), \\ \tau_i(t) & \text{if } X(t) > P_i(t), \end{cases} \tag{27}$$

where $X(t)$ is a uniformly distributed random variable on $[0, 1]$, and $P_i(t)$ is the probability that satellite i will get the new price vector $\pi(t+1)$. We ran the algorithm on the problem using the same probability for all satellites, $P_i(t) = P$ for $P = \{0.1, \ldots, 1.0\}$ and for 1000 runs at each probability level. The average results are given in Table 2.

Table 2. Average number of iterations to termination with respect to update probability.

Probability	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
Iterations	222.2	152.4	111.9	86.4	70.7	60.5	53.2	47.7	44.0

We note that even in a situation where there is only a ten percent probability that updated price data is received, the algorithm, on average, runs only four and a half times slower than the fully parallel algorithm.

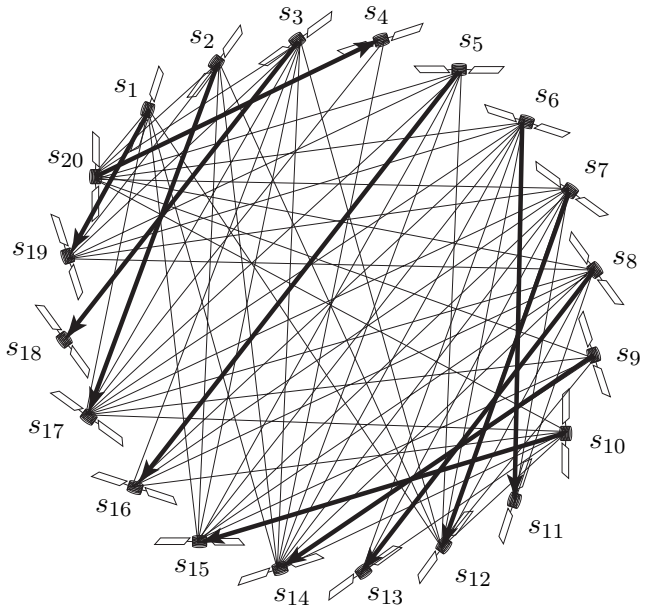

Figure 3. The constellation graph with the best matching obtained with the asynchronous parallel version of the algorithm.

This is very good performance, but may be skewed by the fact that we modeled information *loss* rather than information *delay*. In other words, there is no situation where $\tau_i(t+1) = \tau_i(t-q)$ for some integer $q \geq 1$. This means that when the satellite receives an update, the update is the newest price vector, $\pi(t)$. The effects of receiving an older price vector, say $\pi(t-q)$ for some $q \geq 1$, are not included in the current simulation.

In order to check that this promising performance is not an aberration, we ran three more examples of randomly generated constellations. The configuration of the constellations was identical to the previous one, but the fuel content for each satellite was randomly chosen so that each satellite had a 0.4 probability of being fuel deficient. Mathematically, we the fuel content was determined as follows:

$$f_k^- = Y(k)\bar{f}_k, \quad \underline{f}_k = 0.4\bar{f}_k \tag{28}$$

where $Y(k)$ is a uniformly distributed random variable on $[0, 1]$, and $\bar{f}_k = 100$ units of fuel.

The results for the run are given in Table 3.

Table 3. Average number of iterations to termination with respect to update probability.

P	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
Constellation 1	37.4	28.2	22.4	18.9	16.2	14.5	13.5	12.8	12.3	12.0
Constellation 2	18.1	13.4	10.4	8.1	7.0	6.0	5.2	4.8	4.3	4.0
Constellation 3	6.5	5.0	4.2	3.5	3.0	2.6	2.4	2.2	2.1	2.0

Here again we note the same pattern as before, where lower update probability results in longer running times, but the increase is not excessive, even for low update probabilities.

Finally, to illustrate the behavior of the algorithm over a more complete range of probabilities, we ran one randomly generated constellation at 1000 runs for probability P on the interval $[0.1, 1]$. The results are presented in Figure 4, where it is clear that the run time increases as the probability of update goes to zero.

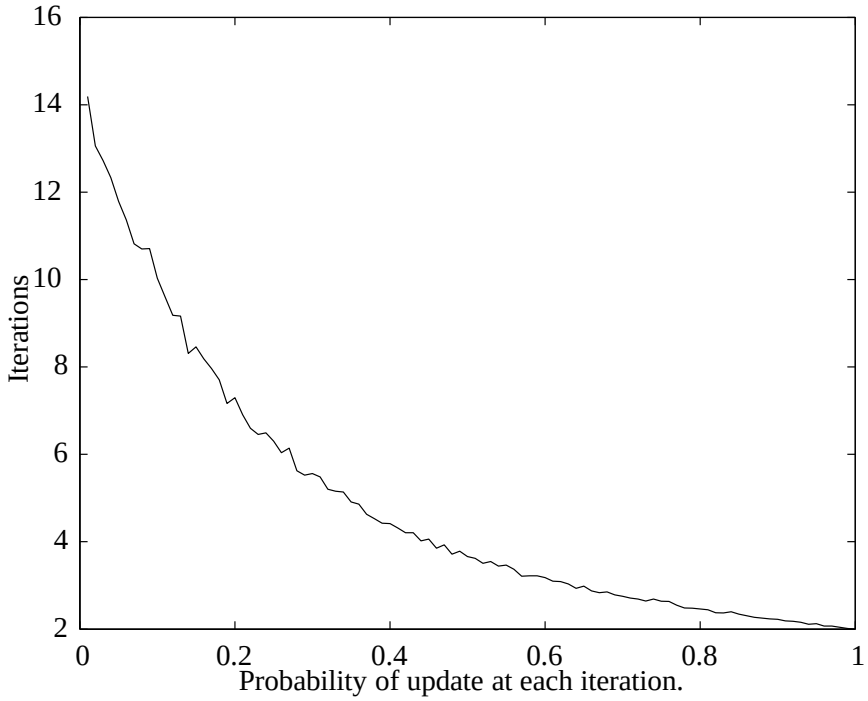

Figure 4. The number of iterations compared with the probability of each element of the set $\mathcal{D}$ having updated price information after each iteration.

As can be seen from these results, there is good indication that the the auction algorithm is very well suited for this particular application, since even high rates of data loss do not detract from relatively fast convergence. However, it must be noted that several instances of randomly generated constellations fell prey to price wars. While this problem can be alleviated by the use of forward-reverse auctions, it is not entirely avoided. In addition, the ε-scaling method,[13] the preferred method of avoiding price wars, is not readily applicable to asymmetric auctions as far as we know. However, every time the straightforward auction algorithm terminated within a reasonable time, the running times associated with the asynchronous algorithm were of the same order of magnitude even under unfavorable communication conditions.

5.0 Summary

This paper presents simulation results to validate the robustness of the auction algorithm to solve the P2P refueling problem. The purpose of the P2P problem is to yield refueling transfer and pairs that ensure that each satellite has a certain minimum level of fuel that will ideally allow it to remain operational until the next outside refueling. The simulation results show that the auction algorithm displays two important properties:

1. It terminates with fewer iterations in a distributed context.
2. Poor price information updating has a benign effect on convergence. Specifically, convergence rate degrades gracefully as the probability of out-of-date price information increases.

The ability of the algorithm to perform well even under poor communication circumstances is one of its main attractive characteristics. The main concern is termination time, since it is guaranteed that when it terminates, the result will be within the desired optimality tolerance. There is good indication, however, that this is not an issue, since the asynchronous implementation converges quickly, despite the occasional occurrence of price wars.

6.0 References

[1]Narvaeez, P., Clerget, A., and Dabbous, W., "Internet Routing Over LEO Satellite Constellations," *Third AMC/IEEE Conference on Satellite-Based Information Services (WOSBIS '98)*, October 1998.

[2]Raney, R. K. and Porter, D., "WITTEX: An Innovative Multi-Satellite Radar Altimeter Constellation," *Report of the High-Resolution Ocean Topography Science Working Group Meeting*, edited by D. Chelton, Oregon State University, Cornvallis, 2001.

[3]Andreas, N., "Space-Based Infrared System (SBIRS) System of Systems," *IEEE Aerospace Conference Proceedings*, Vol. 4, Aspen, CO, February 1997, pp. 429–438.

[4]Shen, H. and Tsiotras, P., "Optimal Scheduling for Servicing Multiple Satellites in a Circular Constellation," *AIAA Guidance, Navigation and Control Conference*, AIAA-2002-4844, Monterey, CA, August 2002.

[5]Shen, H. and Tsiotras, P., "Peer-to-peer Refuelling Within a Satellite Constellation Part I: The Zero-Cost Rendezvous Case," *Proceedings of the 42nd IEEE Conference on Decision and Control*, Vol. 4, Maui, HW, 2003, 2003, pp. 4435–4350.

[6]Shen, H. and Tsiotras, P., "Peer-to-peer Refuelling Within a Satellite Constellation Part II: The Nonzero-Cost Rendezvous Case," *Proceedings of the 42nd IEEE Conference on Decision and Control*, Vol. 4, Maui HW, 2003, pp. 4363–4268.

[7]Dutta, A. and Tsiotras, P., "Asynchronous Optimal Mixed P2P Satellite Refueling Strategies," *Malcom D. Shuster Astronautics Symposium*, Buffalo, NY, June 13–15 2005, AAS Paper 2005-474.

[8]Tsiotras, P. and de Neilly, A., "Comparison Between Peer-to-Peer and Single-Spacecraft Refueling Strategies for Spacecraft in Circular Orbits," *Infotech at Aerospace*, Sept. 26–29 2005, Crystal City, DC.

[9]Shen, H. and Tsiotras, P., "Peer-to-Peer Refueling for Circular Satellite Constellations," Journal of Guidance, Control, and Dynamics, 2005, to appear.

[10]Salazar, A. and Tsiotras, P., "An Auction Algorithm for Allocating Fuel in Satellite Constellations Using Peer-to-Peer Refueling," *American Control Conference*, June 2006 (submitted).

[11]Bertsekas, D. P. and Castañon, D. A., "Parallel Synchronous and Asynchronous Implementations of the Auction Algorithm," Parallel Computing, Vol. 17, 1991, pp. 707–732.

[12]Bertsimas, D. and Tsitsiklis, J. N., *Introduction to Linear Optimization*, Athena Scientific, Belmont, MA, 2nd ed., 1997.

[13]Bertsekas, D. P., "Auction Algorithms for Network Flow Problems: A Tutorial Introduction," Computational Optimization and Applications, Vol. 1, No. 1, 1992, pp. 7–66.

[14]Ortega, J. and Rheinboldt, W., *Iterative Solution of Nonlinear Equations in Several Variables*, Academic Press, New York, 1970.