Request and Share then Assign (RASTA): Task Assignment for Networked Multi-Robot Teams

Samuel Friedman and Qi Han
Department of Computer Science
Colorado School of Mines,Golden, CO, USA
samfrieds@gmail.com, qhan@mines.edu

***Abstract*—In this paper, we propose an improvement of the Hungarian method to optimally solve the task assignment problem for a multi-robot team. Our proposed method involves all robots collaboratively working together to disseminate cost information and then individually computing an assignment that optimizes a particular global goal. Through theoretical analysis, we show that our approach is able to produce a common optimal assignment, sending significantly fewer messages and resulting in faster convergence than other approaches based on the Hungarian method. Our experimental results back up this claim, demonstrating that, even in the worst case, our approach sends a fraction of the messages required by other assignment methods and as a result scales better as team size increases.**

***Index Terms*—swarm robots, task allocation**

I. Introduction

In future pervasive computing environments, it will be common practice to deploy multiple robots for different applications such as search and rescue, environmental monitoring. These robots need to collaborate with each other to accomplish designated missions. Task assignment is a crucial component in the coordination of large multi-robot teams and can serve as a building block for more complex problems. Often times, task assignment in a multi-robot scenario is concerned with finding a one-to-one matching between robots and available tasks, while either minimizing some assignment cost or maximizing an assignment utility. This one-to-one matching ensures that the robots in a team are not performing redundant tasks (i.e., tasks already completed by other team members), and can contribute to the overall mission's goals in the most effective way possible. Task assignment can also serve as a means for balancing the workload of a multi-robot team and making the most out of limited resources.

Formally, the task assignment problem falls under the category of linear sum assignment problems (LSAPs) [4], where the Hungarian method was the first centralized algorithm developed to produce an optimal solution in finite time [7]. Other methods, such as auction based algorithms [2], [3], have since been shown to also produce optimal assignments for LSAPs in finite time. Although existing methods are able to produce optimal assignments in finite time, as the number of robots that need to be assigned to tasks increases, the need for global information regarding each robot's state (e.g., costs, utility, *etc.*) can become highly prohibitive. The impact of a large number of robots becomes especially problematic in centralized approaches. Furthermore, in cases where robots have limited onboard resources (e.g., energy) and a limited communication range, relying on a single robot to collect the needed information and compute the assignment increases the chance of critical failures occurring, as well as poses many other issues (e.g., leader selection, workload balance). Many existing distributed algorithms to solve the task assignment problem attempt to combat the issues facing centralized approaches by having all robots work together to collaboratively determine an assignment rather than relying on a single coordinator. However, decentralized approaches must also be concerned with ensuring that all robots converge on the same assignment which can be a challenge in itself.

A large variety of distributed approaches have been proposed to solve the task assignment problem, ranging from game-theory based approaches [9] to consensus based approaches. Two of the most popular approaches, however, are auction based methods and distributed versions of the Hungarian method. Although these approaches have shown that they can produce a common optimal assignment in finite time, they are not without their problems. One issue with many of the existing distributed algorithms is that they characterize their performance based on the average computational load of each robot or the total number of communication rounds performed before the robots converge on an assignment. However, in the case of resource limited robots, a more useful measure of performance would be the total number of messages or bytes sent before an assignment is reached. Characterizing the performance of task assignment algorithms for multi-robot teams using the total number of messages or bytes sent, links the performance much closer to the actual amount of energy that would be used by the robots when determining an assignment. This is because the energy required to transmit and receive messages from other robots within the team can be orders of magnitude higher than that required to perform onboard processing. It is also important to keep in mind that this difference in energy consumption between processing and communication grows as the communication distance increases.

Our main contribution in this paper is the proposal of RASTA (Request and Send Then Assign), an approach for task assignment utilizing the Hungarian method that leverages cooperation between robots in a team to disseminate needed information in such a way that each robot can individually compute its assigned task, and such that the overall assignment

is optimal. By focusing first on disseminating the information needed for each robot to compute an assignment, and then computing the assignment locally, RASTA effectively reduces the total number of messages and data needed to be sent during the assignment procedure. Furthermore, we prove through theoretical analysis and simulation that RASTA requires significantly less messages (and data) to be sent compared to other similar distributed task assignment algorithms based on the Hungarian method.

The remainder of this paper is organized as follows. In Section II, we briefly review related work focused on solving the task assignment problem in a distributed fashion. In Section III, we formally define the distributed the task assignment problem. In Section IV-A we briefly discuss an existing distributed task assignment algorithm. In Section IV we discuss our proposed algorithm, and in Section V we provide a theoretical analysis of its message complexity, focusing on the total number of messages sent by the robots and the size of each message. In Section VI we provide an overview of the experiments that we performed and discuss the results of our simulations. Finally, in Section VII we make concluding remarks regarding our results.

II. Related Work

Distributed task assignment has been addressed using two main approaches: the Hungarian method based algorithms and auction based algorithms.

A distributed Hungarian method [5] has been developed in which robots share their states with their neighbors. These states consist of the robot's local information denoted by a bipartite weighted graph, a vertex labeling function, and a counter value. Upon receiving messages containing state information from each of their neighbors, robots then update their own state and perform a step of the Hungarian method to produce a new state that brings them closer to finding an assignment. Robots continue to send messages containing state information to their neighbors until an assignment is found and propagated to all the other robots. The authors have shown that for n robots, a common optimal assignment is found in $\mathcal{O}(n^3)$ communication rounds, and that during each communication round $((2n) * (4 + [\frac{1}{4}log_2(n)]) - 2)$ bytes are sent out by each robot. A different approach [6], based on the Hungarian method has also been taken. The robots maintain a forest of all the alternating trees rooted in free task vertices as well as a forest of the alternating trees in the admissible bipartite graph. The forests determine a robot's changing role, where each robot is either sending messages to other robots in the network or performing calculations during each iteration of the algorithm. The algorithm converges to an optimal assignment with a computational payload for each robot of $\mathcal{O}(n^2)$, and with $\mathcal{O}(n^3)$ total messages being sent. However, there is no analysis of the size of the sent messages.

Auction based approaches typically involve robots bidding for tasks in such a way as to maximize their own profit. These approaches often require an auction coordinator, however, and depending on the way that the auctions are set up they can produce suboptimal assignments. For example, a distributed auction based method [9] utilizes only local information that is available to the robots in order to determine an assignment. Even without global information the assignment produced by this method is within a linear approximation of the optimal one.

III. Problem Statement

We envision a situation in which a large team of networked robots has already been deployed in an environment and is informed of a new set of tasks that must be assigned and undertaken. The network topology of the robotic team is represented by the communication graph $G = (\mathcal{V}, \mathcal{E})$ where each robot is mapped to a vertex $v_i \in \mathcal{V}$, and communication links between robot r_i and robot r_j are represented by an edge, $e_{ij} \in \mathcal{E}$. i.e., if robot r_i is able to communicate with another robot r_j in one hop, then the edge $e_{(v_i,v_j)}$ will exist in G and these two robots are neighbors of each other. Furthermore, the robots are able to dynamically compute their associated costs for each of the tasks, using some metric such as distance, residual energy, or other factors. These costs will remain static during the assignment procedure. For example, robots may take into consideration their distance from a particular task, residual energy, and other factors to compute their cost values. However, robots do not initially know the cost values associated with any of the others. Additionally, we will assume that each of the robots is equipped with specific equipment and is only eligible to complete a subset of the total number of tasks that require that particular hardware. If a robot is ineligible for a particular task it will assign a predetermined fixed cost (e.g., a maximum penalty) to that task.

To formalize this problem mathematically, let us suppose that we are given a set of robots R and a set of tasks T, in which we assume that every robot $i \in R$ initially knows only its own cost information c_{ij} for each of the tasks $j \in T$. Therefore, we are interested in assigning the robots to tasks in such a way that the overall cost of the assignment is minimized. Furthermore, each robot can be assigned to one task, and each task can only be performed by a single robot. To represent a robot's assignment, let $x_{ij} = 1$ if robot i is assigned to task j, otherwise $x_{ij} = 0$. Thus, the distributed assignment problem requires all robots to assign themselves to a unique task such that the overall assignment is optimal.

The assignment problem can be expressed as:

$$\min \sum_{i,j} c_{ij} x_{ij} \tag{1}$$

subject to

$$\sum_{i=1}^{n} x_{ij} = 1, \quad \forall j, \tag{2}$$

and

$$\sum_{j=1}^{n} x_{ij} = 1, \quad \forall i, \tag{3}$$

$$x_{ij} \in \{0, 1\}. \tag{4}$$

Although in this formulation we are looking to minimize the cost of the task assignment, we could equivalently assign tasks in such a way as to maximize an assignment utility. All that would be required to solve such a maximization problem would be to perform a simple transformation of the utility values.

To further focus our efforts, we will make the following assumptions about the problem including assumptions about each robot's prior knowledge and the network topology. Our assumptions are as follows:

1) There are an equal number of tasks and robots (If not, then fictional tasks or robots can be created in such a way that overall cost of the assignment is not impacted by their introduction). This reduces the problem to finding a minimum cost bipartite matching.
2) Each robot is assigned a unique ID and has prior knowledge of what sensors all of the other robots have. This allows each robot to reason about what other robots will be competing for the same subset of tasks as it it.
3) Messages are sent as a broadcast to all of a robot's neighbors.
4) The communication graph is strongly connected and static over the course of the assignment procedure.
5) Cost values determined by a robot do not change during the assignment procedure. Since robots may be performing task assignment multiple times throughout a mission, they must be able to converge to an assignment quickly. We argue that in the short time it should take for robots to perform an assignment their cost information should not have changed.

IV. Proposed Algorithm: RASTA

Our approach RASTA uses the Distributed Hungarian Method for Task Assignment (DHM) [5] for comparison. We next briefly describe DHM, discuss the design principle behind RASTA, and then present the details of RASTA.

A. Preliminary: Distributed Hungarian Method for Task Assignment (DHM)

In DHM the robots collaboratively compute a common optimal assignment by performing sub-steps of the centralized Hungarian method and then share state information amongst each other. The shared state information consists of a bipartite weighted graph, a vertex labeling function, and a counter value. The DHM algorithm is broken down into synchronous communication rounds in which robots share their state information, process received state information along with their own state to produce a temporary state, and finally perform a substep of the centralized Hungarian method to update their own internal state that will be shared in the next communication round. This process continues until an optimal assignment is found and all robots converge to the found assignment.

B. RASTA Design Rationale

To provide motivation to our approach we can observe that when performing task assignment, only robots equipped with the same sensors or actuators would be competing amongst each other for a particular subset of tasks.

Theorem 1. *Breaking up the job of assigning tasks to the whole swarm into multiple smaller assignment problems that focus only on robots equipped with the same sensors or actuators and the tasks requiring that equipment, still produces an optimal overall assignment.*

Proof. Since all robots award the same predetermined fixed cost (e.g., a maximum penalty) to tasks for which they are ineligible, each group of robots equipped with the same sensors or actuators have a maximal cost equal to the predetermined fixed cost for the subset of tasks for which they are all eligible to complete, and the same predetermined fixed cost for all other tasks. Therefore, within the smaller assignment problem, no matter what task from the eligible subset a robot gets assigned, that cost will be less than or equal to the cost associated with all of the other tasks that were not considered in the assignment (i.e., those in the ineligible subset). Furthermore, swapping the assignment of any two robots with different equipment will always result in the overall cost of the assignment either increasing or remaining the same. The only way in which the overall assignment's cost could possibly decrease in a suboptimal solution would be to swap the assignments of two robots that are both eligible for the same subset of tasks. Therefore, the cost of the assignment produced by breaking the work up into smaller assignment problems, which only consider robots with the same sensors or actuators, is equivalent to that produced by considering all of the robots together. □

We can also observe that in existing distributed methods based on the Hungarian method, such as those proposed in [5] and [6], the messaging complexity of their algorithms are linked directly to the time complexity of the Hungarian method. This is because in those methods the robots are passing messages amongst themselves to compute steps of the Hungarian method itself. From a message conservation perspective, it would be much more efficient to simply disseminate the needed cost information to each robot (where in the worst case, cost information can get from one robot to another in at most $n-1$ hops for a connected communication graph) and then perform the Hungarian method rather than trying to come up with an assignment as a whole group.

With these observations in mind we propose a method in which each robot gathers from its neighbors the needed cost information to compute its own assignment, while also assisting in the sharing of any cost information that it knows. Once the robots have collected all the needed cost information to compute an assignment they are then able to locally perform the Hungarian method to produce an assignment. This method attempts to reduce the total number of messages and data sent when performing a task assignment procedure by unlinking the messaging complexity from the time complexity of the Hungarian method, and instead shifting it to the complexity of disseminating the needed data.

C. RASTA Details

1) Message Formation: To ensure that each robot is able to collect the information that is needed to compute its own assignment, we propose the use of two special types of messages: request messages and score messages.

The purpose of a **request message** is twofold. Each request message serves as a way for a robot to let their neighbors know what cost information they are personally interested in, as well as allows each robot to update their own interests regarding cost information to better assist their neighbors in the gathering of needed information. Each request message contains an ID field that lets the receiver know which robot the request is from, and an interest field that contains a bit for each of the tasks to represent whether or not the sender is interested in receiving cost information for that particular task.

Score messages are designed to contain the cost information of a particular robot. Similar to the request messages these messages contain an ID field that lets the receiver know which robot the message was sent from, but in addition to the sender ID field, a second ID field is used to convey which robot's cost information the message contains. A third field is then used to store the cost information. To reduce the size of this third field, a robot only needs to share its costs for each of the tasks that it is eligible for. Since all other tasks would have been assigned a predetermined fixed score, there is no need to share this information since it is implicitly known.

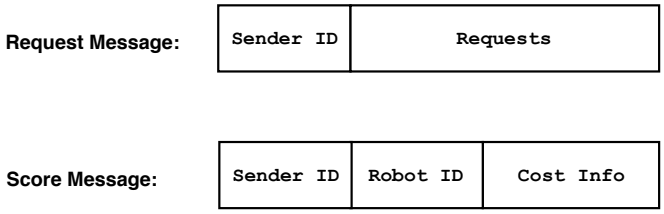

Fig. 1. Structure of the request and score messages.

2) Information Dissemination: For ease of understanding, we will present our proposed method for information dissemination throughout the multi-robot team as a synchronous procedure, however, the steps of this algorithm could also be performed by each robot asynchronously and still lead to the same outcome. Fig. 2 shows the flow chart of the information dissemination process followed by each robot.

The task assignment procedure starts after all robots receive a new set of tasks that must be assigned. First, each robot computes its own cost information and determines what other robots it needs to receive cost information from in order to compute an assignment. Let us call the set of all other robots, from which the robot must receive cost information from to compute an assignment, l_{need}, and let a second set, l_{known}, contain a list of all of the members of the team that a robot currently has cost information for. Finally, let $l_{request}$ be the set of other robots from which a robot would like to receive cost information from.

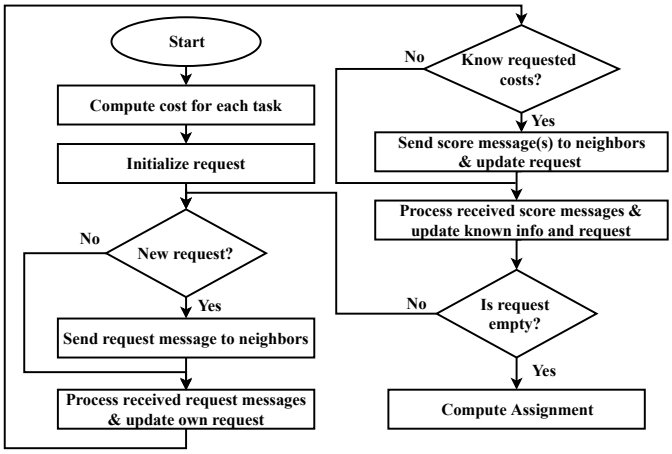

Fig. 2. Flowchart showing the information dissemination process followed by each robot.

In the first round of the algorithm, each robot initializes the set l_{need} to contain the IDs of all other robots equipped with the same sensors or actuators as itself, and initializes l_{known} to include its own ID. Each robot then initializes $l_{request}$ to contain all of the same elements as l_{need}. Using the $l_{request}$ set, each robot then creates an initial request message and broadcasts it to its neighbors.

Upon receiving a request message from a neighbor during the first communication round, a robot adds any IDs to $l_{request}$ that were requested by its neighbors and not currently present in the set (i.e., effectively merging each of its neighbors $l_{request}$ sets with its own). The robot then stores each of its neighbors' requests locally to maintain a database of which robots requested which information.

In subsequent rounds updating $l_{request}$ based on the received request messages becomes a little more complicated. A robot compares each of its newly received request messages to the ones it has stored. If a message indicates that a neighboring robot is no longer requesting a particular set of information that it previously had, and none of the other neighbors or the robot itself are requesting it either, then the associated ID can be removed from $l_{request}$. If a new piece of information is requested that was not requested prior then it is added to $l_{request}$. The robot then replaces the old request message it had stored with the new one. Additionally, a robot only needs to send a new request message during the start of a communication round if $l_{request}$ has changed since the last time a request message was sent by the robot.

After receiving and processing all of the received request messages, each robot looks at l_{known} to see if it has any of the cost information requested by one of its neighbors. This is done by checking the intersection of l_{known} and $l_{request}$. If a robot knows any of the cost information requested by a neighbor (i.e., $l_{known} \cap l_{request} \neq \emptyset$), it forms a score message containing the requested information and broadcasts it out. In the first round this can only occur if a neighbor requests the robot's own cost information. However, in later rounds

a robot may send out multiple score messages containing cost information associated with different robots. The sender then removes the ID associated with the score message from $l_{request}$ and modifies each of its stored neighbor's requests so that they no longer reflect a desire for that particular cost information.

If a robot receives a score message from a neighbor, it saves the cost information locally (if it is not already known) and updates l_{known} to reflect the newly acquired information. If the robot received cost information regarding a robot in $l_{request}$, it removes the associated element from $l_{request}$ if and only if none of its neighbors (excluding the robot that sent the message) are requesting that particular cost information. Finally, the robot modifies the stored request of the robot who sent the score message to reflect that it does not need the information that it just sent.

The process of sending request messages and score messages continues until the $l_{request}$ set of each robot is empty and l_{known} contains all of the elements of l_{need}. When these conditions are met then neither a robot nor any of its neighbors need additional information to compute an assignment. Each robot is then free to utilize the Hungarian method locally using only the cost information related to l_{need} to compute its own assignment. Furthermore, since all robots equipped with the same sensors or actuators will be using the same cost information to compute an assignment, and assuming they store it in the same order (e.g., by robot ID), then they are guaranteed to come to the same solution and there will be no conflicting assignments for the entire system.

Since any information request is able to reach every robot in the team in at most $n-1$ communication rounds, even if a robot and its immediate neighbors do not require any additional cost information before $n-1$ communication rounds have completed, each robot must wait to compute its assignment in case it needs to service other incoming requests. After waiting a predetermined amount of time or participating in at least $n-1$ communication rounds, every robot is guaranteed to have received request information (although indirectly) from every other robot. Once these requests have been serviced, and a robot has all its needed information, which may take at most another $n-1$ communication rounds (i.e., for a total of $2n-2$ communication rounds), a robot is free to compute its own assignment.

Figure 3 shows the finite state machine of each robot's information dissemination process and Algorithm 1 is the pseudocode.

V. Convergence Analysis

Since our algorithm is focused on disseminating the information necessary for each robot to locally calculate their own assignment the messaging complexity of our algorithm is tied to the network topology. Therefore, we will provide analysis of several special scenarios to characterize the performance of our algorithm.

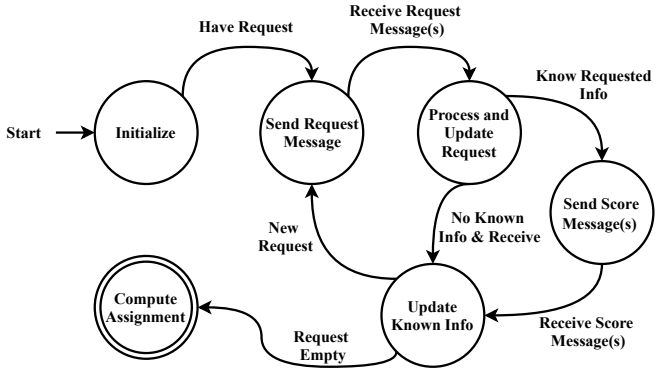

Fig. 3. High-level Finite State Machine of each robot's information dissemination process.

Algorithm 1 Information Dissemination & Task Assignment

Each robot i performs the following tasks:
Initialization:
$l_{known} \leftarrow \{i\}$
$l_{request} \leftarrow l_{need}$
$l_{prev} \leftarrow \emptyset$
Information Dissemination:
while $l_{request} \neq \emptyset$ **do**
  if $l_{request} \neq l_{prev}$ **then**
    Send $l_{request}$ to neighbors
    $l_{prev} \leftarrow l_{request}$
  end if
  Update $l_{request}$ based on received requests
  for $j \in l_{request}$ **do**
    if $j \in l_{known}$ **then**
      Send score message for robot j to neighbors
      $l_{request} \leftarrow l_{request} \setminus \{j\}$
    end if
  end for
  Update l_{known} and $l_{request}$ based on received score messages
end while
Assignment:
Compute assignment locally

A. Message Complexity: Number of Messages

The total number of messages sent by the robots before each robot has the necessary information to compute its assignment is dependent on the network topology. Therefore, we have provided an analysis of a number of different communication graphs (representing different network topologies). Our analysis shows that in the worst case the number of messages sent before each robot has the information necessary to converge to a common assignment is $\mathcal{O}(n^2)$, while in the best case it is $\mathcal{O}(n)$.

1) Line Communication Graph: Intuitively, a line communication graph is the worst case scenario for a connected graph when trying to disseminate the information needed for task assignment. If we assume that one of the end nodes needs

information from the other end node then it will take at most $n-1$ communication rounds for a request message to reach the other end node and at most another $n-1$ score messages to be sent for the first end node to receive the requested information. If we assume that all other robots in the network are a similar number of communication hops away from the information they need, then it follows that the messaging complexity for a line communication graph is on the order of $\mathcal{O}(n^2)$.

2) Complete Communication Graph: In contrast to the worst case scenario of a line communication graph, a complete graph in which every robot is able to communicate with every other robot directly represents the best case scenario. In this case, the analysis of the total number of messages sent is straight forward. In the first round, each robot sends a request message that reaches every other robot. Each robot then would transmit their own cost information in a single score message that also reaches every other robot. Finally, having received all of the information they needed, each robot would transmit one final request message indicating that they no longer needed any more cost information. In this case, a total of exactly $2n$ request messages and n score messages would be transmitted before an assignment could be calculated. Thus, the messaging complexity for a complete communication graph is on the order of $\mathcal{O}(n)$.

3) Star Communication Graph: A similar analysis of a star communication graph in which a single robot is able to communicate with all others reveals that the messaging complexity in that case is also $\mathcal{O}(n)$. To provide intuition as to why this claim is true, consider that in a star communication graph a robot's information requests or score information can reach any other robot in the network in at most two communication hops. For completeness, however, we will provide an in-depth analysis of this particular situation.

In the first round each robot sends a request message. The robot acting as the central hub of the star communication graph receives the request messages from every other robot, while all the other robot's receive the central robot's request. After receiving and processing the requests, each robot would send at most a single score message containing their own cost information. In the second round, the central robot would send a request message that reflects all of the robot's requests, while the other robots may or may not send a request message themselves. At this point every robot would send their own cost information to the central robot, that would then be able to share it with all the others that need it, sending at most $n-1$ score messages in the next round. All robots would then have the cost information necessary to compute their own assignments, and no more messages would need to be sent. Therefore, it is clear that the messaging complexity for a star communication graph is $\mathcal{O}(n)$.

Table I summarizes the comparison of message complexity between DHM and RASTA for line, complete, and star communication graphs.

TABLE I
COMPARISON OF MESSAGING COMPLEXITY

	Total Number of Messages		
	Line	**Complete**	**Star**
DHM	$\mathcal{O}(n^3)$	$\mathcal{O}(n^3)$	$\mathcal{O}(n^3)$
RASTA	$\mathcal{O}(n^2)$	$\mathcal{O}(n)$	$\mathcal{O}(n)$

B. Message Complexity: Message size

To perform a more complete analysis of our algorithm, it is also important to take into consideration the message size. Along with the theoretical messaging complexity the message size helps to predict the total amount of data that would need to be transmitted before an assignment can be found.

We will derive the size of both the request and score messages using the following assumptions.

1) Each ID field can be represented by a single number, the size of which is dependent on the total number of robots.
2) Each robot can be represented by a single bit in the request field, where a 1 represents a desire for that particular robot's cost information and a 0 indicates no need for the information.
3) Each value in the cost filed can be stored in a single integer, therefore at most there will be n integers stored in this field in cases where all robots are equipped with the same sensors or actuators and the task assignment problem cannot be broken down. In other cases a fraction of n integers will be stored in this field.

Therefore, using the assumptions laid out above the size of each request message is $\frac{1}{8}(\log_2(n)+n)$ bytes and the maximum size of each score message is $\frac{1}{8}(2\log_2(n))+(4n)$. Table II summarizes the comparison of message size for DHM and RASTA.

TABLE II
COMPARISON OF MESSAGE SIZE

		Message Size (Bytes)
DHM		$\mathcal{O}(n\log n)$
RASTA	**(Request)**	$\mathcal{O}(n)$
	(Score)	$\mathcal{O}(n)$

TABLE III
COMPARISON OF MESSAGE SIZE

		Message Size (Bytes)
DHM		$((2n)*(4+[\frac{1}{4}log_2(n)])-2)$
RASTA	**(Request)**	$\frac{1}{8}(\log_2(n)+n)$
	(Score)	$\frac{1}{8}(2\log_2(n))+(4n)$

C. Time Complexity: Computation

As for the computational load of each robot, since we are relying on the existing Hungarian method to perform the

assignment, in the worst case the computational complexity is $\mathcal{O}(n^3)$. However, this analysis does not consider the exploitation of a heterogeneous swarm. In Section IV-B we have shown that breaking up the task assignment problem into smaller assignment problems that consider only robots equipped with the same sensor or actuator still results in the convergence to an optimal solution. Therefore, if we are able to break up the assignment into smaller problems, the value of n in these cases can be a fraction of the total size of the entire multi-robot system, resulting in considerable reductions in the computational complexity of running the Hungarian method at each robot, since they only need to compute the assignment for a smaller subset of the whole team. Furthermore, since the Hungarian method is deterministic, if each robot runs the algorithm locally (and possibly with a smaller subset of the cost information focused only on robots with the same hardware and abilities), they will all produce a consistent assignment (i.e., one without conflicts). Upon producing this assignment, there is no need for further communication among the robots to verify that the assignment is consistent since the information dissemination stage ensures that every robot is able to run the Hungarian method on its own and produce the same result.

D. Time Complexity: Communication

Now using our analysis on the total number of messages needed to be sent and the size of each message we are able to produce an upper bound on the time it takes the robots to communicate with each other to converge to an assignment. If we assume that all robots communicate with a fixed data rate, d_{rate}, and assume that at most one robot can send a message at a time (although some communication technologies and topologies may allow for multiple robots to communicate simultaneously without interfering with one another) the communication times shown in Eq. (5) through Eq. (8) can be derived. Eq. (5) shows the time for DHM, while Eq. (6) through Eq. (8) show the time for RASTA with line, complete, and star communication graphs respectfully. A summary of these upper bounds is also shown in Table IV.

$$\frac{((2n^4)(4+[\frac{1}{4}log_2(n)])-2n^3)}{d_{rate}} \tag{5}$$

$$\frac{(2n-3)\frac{1}{8}(\log_2(n)+n)+(n^2-n)(\frac{1}{8}(2\log_2(n))+(4n))}{d_{rate}} \tag{6}$$

$$\frac{\frac{n}{4}(\log_2(n)+n)+\frac{n}{8}(2\log_2(n))+(4n)}{d_{rate}} \tag{7}$$

$$\frac{\frac{n}{2}(\log_2(n)+n)+(2n-1)\frac{1}{8}(2\log_2(n))+(4n)}{d_{rate}} \tag{8}$$

TABLE IV
COMPARISON OF COMMUNICATION TIMES

	Communication Time (s)
DHM	$\mathcal{O}(n^4)$
RASTA (Line)	$\mathcal{O}(n^3)$
RASTA (Complete)	$\mathcal{O}(n^2)$
RASTA (Star)	$\mathcal{O}(n^2)$

It is important to note that the overall convergence time of the two algorithms is going to be the combination of both the computational time complexity and the communication time complexity. That is, the time it will take each algorithm to converge to an assignment will be the sum of the time it takes for the robots to communicate and the time it takes for the robots to perform local computations.

VI. PERFORMANCE EVALUATION

To further characterize the performance of our proposed algorithm under various conditions and verify our theoretical analysis, we implemented both RASTA and DHM in C++.

A. Experimental Setup

We performed simulation experiments on multiple different system sizes and communication graphs. Additionally, we utilized the NS-3 [1] simulation environment to randomly generate each robot's costs and ensured that the same values (i.e., by using the same seed) were used both for trials using RASTA and for those using DHM.

In these simulations we did not consider heterogeneous robot teams. Since DHM is unable to exploit heterogeneity in the same way that RASTA is able to, it would not serve as a far comparison. It should be obvious why exploiting heterogeneity can reduce the total number of messages sent, since now instead of every robot requiring the cost information from every other robot, they just need the cost information from a smaller subset. Even though we did not take advantage of heterogeneity in our simulations, we still are able to show a considerable reduction in the messaging overhead required to perform task assignment when using RASTA when compared with DHM.

In each experiment the total number of messages sent before all the robots converged on a common assignment or had the information necessary to calculate their own assignment was recorded. From this messaging data we were also able to determine the total size of the data transmitted by the robots. Furthermore, we verified that the assignments produced using RASTA and DHM both produced optimal assignments.

In cases in which there was a single optimal assignment, it was observed that both methods produced identical assignments. However, in cases in which more than one optimal assignment existed, such as when two or more robots share the same cost values for multiple tasks, the two methods did not always produce identical assignments. Even though the

results in some cases were not identical, it was verified that the assignments had the same cost and that they were both still optimal.

In addition to testing the impacts of various system sizes on the number of messages and bytes sent to perform task assignment, we also tested the impact of the type of communication graph. We tested three special types of communication graphs including complete communication graphs, star communication graphs, and line communication graphs. Furthermore, we tested randomized communication graphs that were created by randomly assigning each robot a position in a predefined area and using a simple disk communication model to determine their neighbors. Precautions were taken to ensure that these randomly constructed communication graphs were connected.

B. Experimental Results

Figs. 4 through 7 show the average number of messages sent before an assignment was reached for numerous trials of running our simulations on complete, line, star, and randomized communication graphs respectively. For each of the different types of communication graphs, 12 sizes of robot teams were tested (i.e., 12 different values of n), and 30 trials were run for each size.

Looking at the results for both the complete communication graph and star communication graph shown in Figs. 4 and 5 it is evident that our experimental results line up with our theoretical analysis. In Section V we proved that for these two types of communication graphs the total number of messages sent was on the order of $\mathcal{O}(n)$ and the experimental data clearly shows a linear relationship between the total number of messages sent (for both request and score messages) and the number of robots. Similarly, the experimental results for the line communication graph shown in Fig. 6 back up our theoretical analysis, in which we showed that the total number of messages sent would be $\mathcal{O}(n^2)$. This quadratic relationship between the number of robots n and the total number of messages sent can be seen for both message types sent by RASTA in Fig. 6.

To provide further comparison of the performance of RASTA and DHM it is useful to look at some of the experimental results quantitatively. For example, in the worst case scenario (i.e., the line communication communication graph) for a multi-robot team size of $n = 42$ it was found that on average a total of $3,126$ messages were sent by RASTA (i.e., combining both request and score messages), while $25,273$ messages were sent by DHM before an assignment was found. This is an average reduction in the number of messages sent by a factor of over 8. It is important to note that this reduction factor continues to grow as the number of robots increases. In the best case scenario (i.e., a complete communication graph), we see an even larger reduction in the number of messages sent. Again, looking at a team size of $n = 42$, our results show that on average RASTA sends 53.8 times less messages than DHM.

Similarly, Figs. 8 through 11 show the average number of bytes sent before an assignment was reached for numerous trials of running our simulation on complete, line, star, and randomized communication graphs respectfully. Similar results to what was shown in Figures 4 through 7 can be seen, where in all cases, RASTA requires less data to be transmitted before an assignment can be found and is more resilient to increases in the number of robots participating in the task assignment procedure.

As an example of the data savings of RASTA compared to DHM, if we look at the results shown in 9 for $n = 36$, we see that DHM sends on average 6.176 MB of data before an assignment is reached while RASTA only needs to send 0.177 MB on average to compute the same assignment. Again, similar to what was observed with the messages, the discrepancy between the amount of data sent by these two methods only continues to grow as the number of robots increases.

Finally, Fig. 12 shows the time it takes for different sizes of robot teams to converge to a final assignment. For this particular result we assumed that each robot communicated at a rate of 10 Mbits/second, which has been shown to be a reasonable data rate even for distances of up to 10km [8]. Looking at Fig. 12 it is clear that these results are consistent with our theoretical analysis in Section V-D. The convergence rates as the size of the robotic team increases grows strictly slower for RASTA than the convergence rates for DHM.

VII. Conclusion

In this paper we proposed a method for performing task assignment for a multi-robot team, in which needed information is first disseminated to the robots and then the Hungarian method is utilized to calculate an assignment for each robot locally. We have shown that in the worst case a total of $\mathcal{O}(n^2)$ messages need to be sent by the robots before an assignment can be calculated and have backed up our theoretical analysis with simulation results. Other similar methods for task assignment, like DHM, have a messaging complexity on the order of $\mathcal{O}(n^3)$. In our theoretical analysis and simulation, we looked only at static communication graphs. However, it would be interesting to see the impact on performance if these communication graphs were dynamic or if the multi-robot team had intermittent connectivity where the communication graph was not always connected. Also, this work needs to be extended to address the scenarios where robots are not homogeneous, and the number of robots is not the same as the number of tasks.

Acknowledgment

This work is supported in part by NASA SmallSat Technology Partnership (STP) program with grant number 80NSSC18M0048.

References

[1] The NS-3 network simulator. https://www.nsnam.org/. Accessed: 2018-09-4.

[2] Dimitri P. Bertsekas. Auction algorithms for network flow problems: A tutorial introduction. *Computational Optimization and Applications*, 1(1):7–66, 1992.

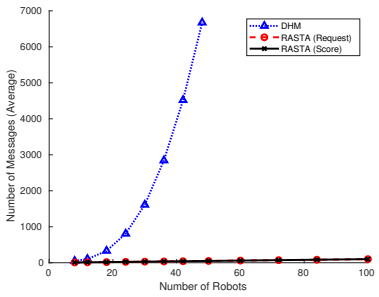

Fig. 4. Impact of team size on average number of messages (complete communication graph)

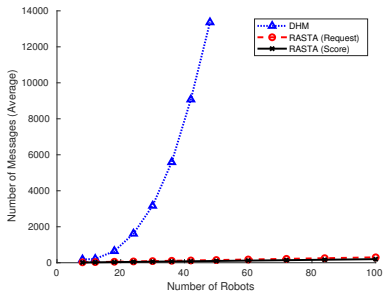

Fig. 5. Impact of team size on average number of messages (star communication graph)

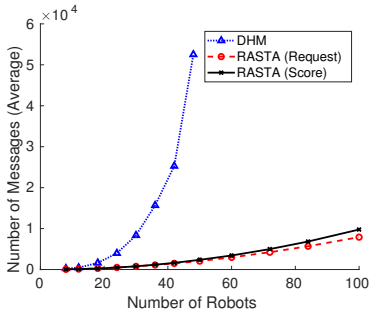

Fig. 6. Impact of team size on average number of messages (line communication graph)

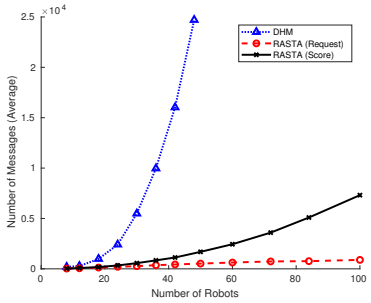

Fig. 7. Impact of number of robots on average number of messages (randomized communication graph)

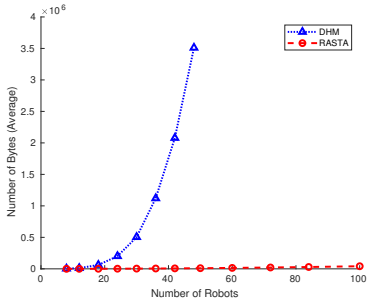

Fig. 8. Impact of number of robots on average number of bytes sent (complete communication graph)

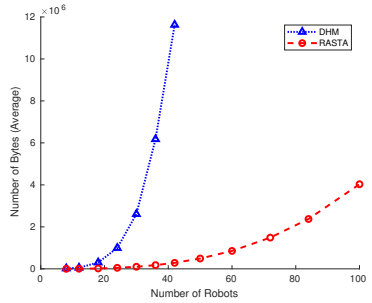

Fig. 9. Impact of number of robots on average number of bytes sent (line communication graph)

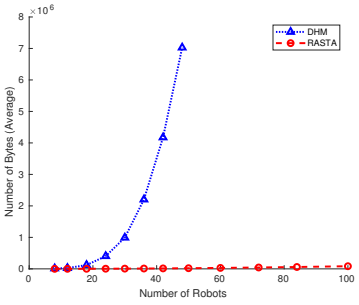

Fig. 10. Impact of number of robots on average number of bytes sent (star communication graph)

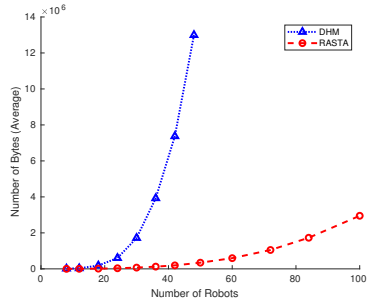

Fig. 11. Impact of number of robots on average number of bytes sent (randomized communication graph)

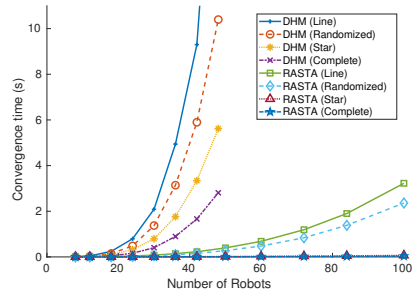

Fig. 12. Convergence time in seconds for different sizes of robotic teams

[3] Dimitri P. Bertsekas and David A. Castaon. Parallel synchronous and asynchronous implementations of the auction algorithm. *Parallel Computing*, 17(6-7):707–732, 1991.

[4] Rainer E. Burkard and Cela Eranda. *Linear assignment problems and extensions*. Universitat Graz/Technische Universitat Graz. SFB F003 - Optimierung und Kontrolle, 1998.

[5] S. Chopra, G. Notarstefano, M. Rice, and M. Egerstedt. A distributed version of the hungarian method for multirobot assignment. *IEEE Transactions on Robotics*, 33(4):932–947, Aug 2017.

[6] Stefano Giordani, Marin Lujak, and Francesco Martinelli. A distributed algorithm for the multi-robot task allocation problem. *Trends in Applied Intelligent Systems Lecture Notes in Computer Science*, pages 721–730, 2010.

[7] H. W. Kuhn. The hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2(12):83–97, 1955.

[8] R. Radhakrishnan, W. W. Edmonson, F. Afghah, R. M. Rodriguez-Osorio, F. Pinto, and S. C. Burleigh. Survey of inter-satellite communication for small satellite systems: Physical layer to network layer view. *IEEE Communications Surveys Tutorials*, 18(4):2442–2473, May 2016.

[9] M. M. Zavlanos, L. Spesivtsev, and G. J. Pappas. A distributed auction algorithm for the assignment problem. In *47th IEEE Conference on Decision and Control*, pages 1212–1217, Dec 2008.