Robots in the huddle: Upfront computation to reduce global communication at run-time in multi-robot task allocation

Changjoo Nam, and Dylan A. Shell[‡]

September 25, 2019

Abstract

We study multi-robot task allocation problems where task costs vary. The variation may be, for example, due to the revelation of new information or other dynamic circumstances. As robots update their cost estimates, typically they will update task assignments to reflect the new information using additional communication and computation. In dynamic settings, the robots are continually repairing the optimality of the system's task assignments, which can incur substantial communication and computation.

We investigate how one can reduce communication and centralized computation expense during execution by using a prior model of how costs may change and performing upfront computation of possible robot–task assignments. First, we develop an algorithm that partitions a team of robots into several independent sub-teams that are able to maintain global optimality by communicating entirely amongst themselves. Second, we propose a method for computing the worst-case cost sub-optimality if robots persist with the initial assignment and perform no further communication and computation. Lastly, we introduce an algorithm to assess whether cost changes affect the optimality of the current assignment through a succession of local communication exchanges. Experimental results show that the proposed methods are helpful in reducing the degree of centralization needed by a multi-robot system (e.g., the third method gave at least 45% reduction of global communication across all scenarios studied). The methods are valuable in transitioning multi-robot techniques which have met with success in structured applications (like factories and warehouses) to the broader, wilder world.

[*]This work was supported in part by National Science Foundation (Awards #IIS-1453652 and #ECCS-1637889) and the KIST Institutional Program (project number 2E28670, Task planning and decision making for multi-robot teams interacting with humans).

[†]C. Nam is with Robotics and Media Institute at Korea Institute of Science and Technology and D. Shell is with Department of Computer Science and Engineering at Texas A&M University.

[‡]Manuscript received Dec 19, 2018; revised Aug 20, 2019.

1 Introduction

Currently multi-robot task allocation (MRTA) methods are among the best established ways for coordinating teams of robots. Task assignment and task allocation algorithms fall under the broad banner of task planning methods, and address the specific concern of which robots (i.e., who) should do which tasks (i.e., what). They seek to maximize some notion of collective performance. Generally, each robot maintains an estimate of the cost of performing the available tasks. The robots share their estimates over a communication network and, depending on the approach involved, this information might be used locally or aggregated centrally. In the most common case, a centralized system, an optimal assignment is computed by a server or a distinguished robot: approaches to compute this include the Hungarian method [14], an auctioneer [7], or a linear programming framework. Despite being optimal on the basis of the initial information, the robot–task assignment may turn out to be unfavorable while executing the tasks. For example, the environment may change, robots may fail, or a variety of other unexpected situations may arise. One way to retain optimality is to compute new assignments to reflect the most recent cost estimates. Regular re-computation may be necessary to ensure fluidity, but this incurs computational and communication expense proportional to the desired responsiveness.

In this paper, we consider the MRTA problem of finding the optimal assignment of a set of tasks to a team of robots when the associated costs may vary at run-time. We propose a cost representation which incorporates uncertainty by generalizing a single cost value to a range of possible values. Take for example the robot in Fig. 1: it is able to estimate its shortest and longest driving times to a destination by considering information about its route. The lower bound would be merely the time spent on driving (distance divided by the maximum speed); adding the maximum waiting time for traffic signals gives an upper

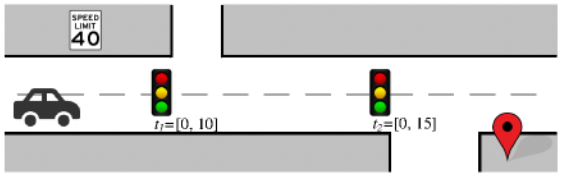
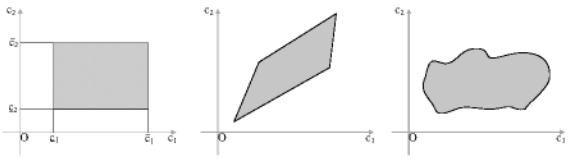


Figure 1: A simple example where task costs are not precisely known to the robot beforehand. The driving time c to the destination varies depending on the traffic signals. A lower bound $\underline{c}$ is $\frac{d}{v_{\max}}$ when $t_1 = t_2 = 0$, and an upper bound $\bar{c}$ is $\frac{d}{v_{\max}} + 25$ (assuming the robot drives with the maximum speed) where d is the distance to the destination and $v_{\max}$ is the maximum speed.



(a) A cost region defined by an upper and a lower bound. Costs are uncertain but independent.

(b) A linear convex boundary showing both uncertainty and a linear interrelationship.

(c) A nonlinear nonconvex boundary, for complex uncertainty and interrelationships.

Figure 2: Illustrations of region-based cost representations. Costs that are uncertain and interrelated are represented with boundaries by treating the set of possible costs as regions.

bound.[1] Then, conceptually, all permissible values for costs fall in a (high-dimensional) region as illustrated in Fig. 2(a). Less trivial interrelationships between the permissible costs can be described by regions more complex than an axis-aligned box (*cf.* Fig. 2b and Fig. 2c).

With this region-based model, the approach we describe performs additional computation upfront in order to lessen later communication and computational burdens. Once execution begins, even though uncertainty may be reduced, the robots are more likely to have constraints and limits on their ability to communicate. Stone and Veloso [29] first studied circumstances where prior computation can aid robots in their subsequent coordination. They introduced the term *locker-room agreement*, motivated by study of prior arrangements (e.g., of switching formations) of soccer robots. Our upfront computation serves a similar purpose, though the technical approach differs: we employ Sensitivity Analysis (SA) of optimal assignments to reason about appropriate behavior in parts of the cost region.

SA has been studied in Operations Research to delineate the forms of perturbations to costs that do not alter the optimal solution [10]. Thus far, SA has found limited applicability to MRTA problems: the analysis assumes that the decision maker is able to access all information offline and has control all over the constituents without communication constraints. The physically distributed nature of multi-robot systems and their limited communication and computational resources pose challenges to the direct application of classical SA. We use SA in an initial phase, before robots are dispatched, to make the robots aware of all possible arrangements of optimal assignments within the cost region. Based on this information, we develop methods that answer questions pertinent to robots as their cost estimates change during the second phase, task execution. These methods can reduce global communication and centralized computation, or quantify the optimality trade-offs if communication is avoided altogether.

To clarify the scope of the paper, next we summarize the setting studied, along with our assumptions and limitations. This paper addresses the question of what we call the *need* to communicate at run-time in three ways. First, by providing a policy for when agents ought to broadcast estimates and re-compute, if they wish to retain global optimality. Secondly, if the operator has robots that adopt the policy of never communicating, we compute a bound on how much their optimality will be affected (so that the operator may make this decision contingent on limited loss of efficiency, for instance). And, thirdly, if communication is used to regain optimality, then we provide a means to escalate the communication so as to involve only those agents who are needed. These three elements reduce communication—as the paper's title alludes to—in three ways, either (1) avoiding communication because it is unnecessary as it will not change the task assignment; (2) avoiding it because the communication costs do not justify the savings in efficiency; or (3) limiting the participants involved in messaging. These reflect savings for most messaging modalities, though the precise value of the third case depends on the particular communication technology involved. As already outlined, the approach employs an initial (compute intense) phase of planning before mission execution. Our assumption is that, at this point, resources (time, memory, communication) are abundant, so we employ a centralized algorithm. Then, in the second phase, namely during execution, these requirements are relaxed. The domain knowledge used to model changes in costs is not expected to be predictive of cost dynamics but only to delineate the limits of the costs

[1]For simplicity, we assume an absence of congestion and acceleration/deceleration here.

could feasibly change.

The following are contributions of this paper:

- We propose a region-based cost representation that captures the uncertainty in the states of robots, tasks, or the environment. This representation is rather rich: it does not make the simplifying assumption that costs of different tasks are independent, and it is capable of modeling tightly interrelated costs (Sec. 4).

- We develop an algorithm that analyzes the cost structure for a given assignment to seek sub-teams within the overall team. It partitions the group into sub-teams that are able to work independently, forgoing global communication by communicating only amongst the members of the sub-teams, but retaining optimality (Sec. 6.3).

- We consider the problem of deciding whether it is beneficial to persist with the current assignment even if cost changes mean that it is no longer optimal. We develop a method for computing the worst-case cost sum if the robots retain their current assignment, allowing one to decide whether to persist with the current assignment because the computational/communication expense needed for reassignment is prohibitive (Sec. 6.4).

- We examine how, once costs change, the robots can determine whether the current task assignments are sub-optimal with minimal communication. Each robot may compute an interval of cost within which any cost variation does not affect optimality. But even if a cost violates these bounds, other costs may have changed too, and optimality may still be retained when the cost changes have been considered too. We introduce a method that incrementally increases the dimensionality of the bounding region, growing the number of costs considered by communicating with additional robots (Sec. 6.5).

2 Related Work

Some researchers have proposed re-optimization schemes for multi-robot systems, allowing updated assignments to be computed more efficiently than a naïve re-computation from scratch. Mills-Tettey et al. [19] describe a dynamic (or incremental) Hungarian method that repairs initial optimal assignment to obtain a new optimal assignment when costs have changed. Shen and Salemi [28] give a decentralized dynamic task allocation algorithm that uses an heuristic search. These methods still use computational resources for those cost modifications which end up with the same assignment.

Parker et al. [23] proposed a decentralized algorithm to minimize the maximum cost where costs change over time. They represent a cost as a monotonically increasing function as time passes (e.g., fire spreading). Each agent assigned to a task decreases the cost (i.e., performs the task) with a fixed rate. They propose a modified MAX-SUM algorithm which optimizes a global utility function greedily, where the modification of the original algorithm is made to incorporate uncertainty of the global utility. They model varying costs precisely and add Gaussian noise to the costs where the noise represents the errors in the cost function modeling or empirically evaluated parameters. Since their method is intended only for circumstances where costs change with a constant rate, it is not applicable for costs which cannot be modeled by a linear function.

In [24, 25], nonlinear models of changing cost are proposed where the growth of cost is monotonically accelerating or decelerating. Although these models provide wider applicability for the cases where the evolution of costs is not simple as linear, constructing such models requires additional domain-specific information. Also, such models represented as functions may be outdated so cannot describe costs accurately in dynamic environments where the models should be updated to reflect changes in the environments.

Another model of changing costs is proposed in [1] for formation control problems of multiple robots. The scaled goal formation problem considers a transition from an original formation to a goal formation where the goal formation is scaled from the initial formation. The model of costs is described by nonlinear functions of a scaling factor (i.e., ratio) between the original and the goal formation changes. To find the global optimal assignment, the proposed method finds intervals of the scaling factor in which each interval has one optimal assignment. In other words, each optimal assignment remains optimal within its corresponding interval of the scaling factor. This approach is similar to our method employing sensitivity analysis developed in Sec. 6.2 that finds the ranges of costs in which any cost value in each range produces the same optimal assignments. However, the model of costs could be outdated in dynamically changing environments, so the intervals found from upfront computation using the model could be invalid during execution owing to new run-time changes that may make the model obsolete.

In [21], the present authors consider multi-robot teams operating in probabilistic domains. In that work we repre-

sented costs as random variables where distributions express uncertainty in the environment and which also incorporate inter-robot couplings as the probabilistic representation does not assume that costs are statistically independent. Although that representation is richer than the interval-based model in dealing with a variety of forms of uncertainty, the algorithm proposed in that work neither takes dynamic changes in costs into account nor considers the system overhead caused in handling uncertain costs.

Liu and Shell [17] proposed the interval Hungarian method (IHM) to permit uncertainties in costs. Given an optimal assignment, the algorithm computes the maximum interval around each cost in which its perturbation does not affect the current optimal assignment. Thus, the robots can determine how a cost change affects the optimality of the current solution. However, that formulation treats the problem of multiple simultaneous cost modifications, which do occur naturally in multi-robot systems (e.g., a single robot failure affects n costs), in an ad hoc fashion.

The same authors also proposed a sparsification and partitioning method to distribute the assignment problem to reduce global communication and reassignment [18]. That method coarsens the utility matrix by using locality and sparsity of tasks. Once the matrix has been partitioned into several clusters, each cluster is able to compute an assignment independently. This method for decentralizing the work mitigates difficulties of the centralized approach such as maintaining global connectivity and performing heavy computations by a single unit. Inspired by that work, here we propose a partitioning method for problems where single time-step sparsity is insufficient.

Chopra et al. [6] propose a distributed version of the Hungarian method. Robots exchange messages containing state information in a peer-to-peer fashion to update their own states. They run the Hungrian method locally with a (possibly incomplete) graph in order to get close to the state producing an optimal assignment. Once an assignment is found through the repeated local communication and computation, it is propagated to all robots. An earlier work [5] proposes a distributed simplex method solving an MRTA problem but computationally more expensive than [6]. Although the both methods and ours aim to reduce global communication and computation, there are several distinctions. First, the previous methods do not consider any central processor whereas our methods aim to reduce computations when centralized methods are used. Second, the previous methods consider costs that are deterministic while our costs could change during execution. Third, the methods assume a strongly connected communication network. A relaxation is suggested in [6] that assumes only a jointly strongly connected network over some time period. However, the necessary condition of our upfront analyses is maintaining just a connected network (not necessarily strong) before the robots are dispatched while the upfront computation is performed. During execution, our methods may need a connected network (still not strongly) but disconnected networks could be allowed based on the result of the analyses.

A preliminary result of this present paper has published in [20]. We extend the prior study to include (i) a richer representation of the region-based cost that models interrelationships between costs (Sec. 4), (ii) an additional experiment that shows the benefit from using the richer representation (Sec. 7), (iii) a new randomized algorithm (Alg. 1) that runs faster than the previous one in [20], (iv) an additional experiment showing the running time of the new randomized algorithm (Sec. 7), (v) an additional proof that shows the computational complexity of Alg. 2 if the cost region is nonlinear and convex (Theorem 5.2), (vi) an appendix that helps understand Theorem 5.2, and (vii) a major revision for the organization, detailed descriptions of the problem, the algorithms, the experiments, and the future work.

3 Preliminaries

This section provides a mathematical formulation of the MRTA problem and introduces sensitivity analysis of an optimal assignment. The analysis computes a region of costs where changes within the region preserve the optimality of the current assignment.

3.1 Multi-robot task allocation

The MRTA problem can be posed as an Optimal Assignment Problem (OAP). For n robots and m tasks, we assume we are given costs $c_{ij} \in \mathbb{R}^{\geq 0}$ that represent the cost of the i^{th} robot R_i performing the j^{th} task T_j for $i \in \{1, \cdots, n\}$ and $j \in \{1, \cdots, m\}$. The robots should be allocated to tasks with the minimum cost sum. Let x_{ij} be a binary variable that equals to 0 or 1, where $x_{ij} = 1$ indicates that the R_i performs T_j. Otherwise, $x_{ij} = 0$. For simplicity here we have assumed that $n = m$. (This is without loss of generality, since if $n \neq m$, dummy robots or tasks would be inserted to make $n = m$.) Then a mathematical description of the MRTA problem is

$$\min \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij} \tag{1}$$

subject to

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

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

$$0 \leq x_{ij} \leq 1 \qquad \forall \{i,j\}, \tag{4}$$

$$x_{ij} \in \mathbb{Z}^{+} \qquad \forall \{i,j\}. \tag{5}$$

We make use of matrix representations $\mathbf{C}$ and $\mathbf{X}^*$ that are $n \times n$ matrices representing a cost matrix and an optimal assignment of the problem, respectively. Matrix $\mathbf{X}^*$ is one among a larger set of *matchings*, which are all matrices satisfying (2)–(5).

3.2 Sensitivity analysis of optimal assignments

Sensitivity analysis (SA) has been studied for several decades in Operations Research to assess the robustness of optima for an optimization problem to perturbations in the input specification [10, 16, 30]. Analysis of an optimal assignment must compute a region where costs within that region preserve the optimality of the current assignment.

The OAP can be relaxed to a linear programming problem (LP) by removing the integral constraint. The LP formulation of MRTA may make use of SA of an optimal assignment to yield a safe region of costs where the assignment remains optimal if all costs stay in the region. We provide a brief interpretation of the analysis for the MRTA problems, based on a comprehensive study of [30].

An LP problem corresponding to an MRTA problem can have more than one feasible solution. For each feasible solution, the decision variables x_{ij} ($i, j \in \{1, \cdots, n\}$) can be divided into basic variables and nonbasic variables where a variable is basic if it corresponds to one of the vectors in the basis, given a feasible basis to a linear-programming problem. If k is an index of a feasible solution, then for each k, critical region CR_k, a set of costs where an MRTA problem has the same optimal assignment for any cost $c \in CR_k$. Thus,

$$CR_k = \{c \in \mathbb{R}^{(n^2)} : \mathbf{c}_{N_k} - \mathbf{c}_{J_k}\mathbf{B}_k^{-1}\mathbf{A}_{N_k} \geq 0\}, \tag{6}$$

where J_k and N_k indicate basic and nonbasic variables of the k^{th} feasible solution, respectively. In other words, any costs within CR_k do not alter the feasible (optimal) solution of k. The matrices $\mathbf{B}_k$ and $\mathbf{A}_{N_k}$ are the constraints of basic variables and nonbasic variables.[2] Here $\mathbf{c}_{J_k}$ and $\mathbf{c}_{N_k}$ are cost vectors of basic and nonbasic variables. The critical region CR_k is formed by linear boundaries with nonempty interiors.

However, there is an additional source of complexity because the MRTA problem is degenerate. One easy way to understand degeneracy is using a polytope defined by the constraints of the optimization problem (2) and (3). In non-degenerate cases, an extreme point of a polytope corresponds to one feasible solution. In degenerate cases, one extreme point corresponds to many different degenerate solutions [11, 13].

Consequently, the critical region CR_k of one feasible solution k is not a complete description of the region that preserves optimality. The complete set is

$$\theta(\mathbf{X}^*) = \bigcup_{k \in H} CR_k, \tag{7}$$

which is the union of critical regions of all feasible solutions where $H = \{k : \mathbf{X}^*_{J_k} = \mathbf{B}_k^{-1}, \mathbf{X}^*_{N_k} = 0\}$, which is the set of indices of feasible solutions. Note that $\theta(\mathbf{X}^*)$ is also a polyhedral set [30, Theorem 17] consisting of linear boundaries that cross the origin.

An $n \times n$ MRTA problem has $2n-1$ basic variables and $(n-1)^2$ nonbasic variables. To compute (7), we must identify the basic and nonbasic variables of the k^{th} feasible solution. The n variables corresponding to costs in the optimal assignment are basic variables, but the degeneracy means that the remaining $n-1$ basic variables (i.e., degenerate basic variables) cannot be identified directly. We choose the $n-1$ basic variables from the remaining n^2-n variables, yielding a total of $\binom{n^2-n}{n-1}$ choices. Not all $\binom{n^2-n}{n-1}$ sets of variables can be feasible solutions because the set H indicates $\mathbf{X}^*_{J_k} = \mathbf{B}_k^{-1}$ which means that $\mathbf{B}_k$ must be nonsingular. If $\mathbf{B}_k$ with the set of those n basic variables and chosen $n-1$ variables is of full rank, then the set is one of the feasible solutions in H.

4 The cost representation: bounded regions and interrelated values

In this section, we formalize the cost representation in terms of a bounded region. We also show how the model of interrelationships gives a better understanding of an assignment problem subject to uncertain costs.

Suppose that the costs belong to a finite region $\mathbb{C}$ where $\mathbb{C} \subseteq \mathbb{R}^{(n^2)}$, so any particular matrix of costs $\mathbf{C} \in \mathbb{C}$. We

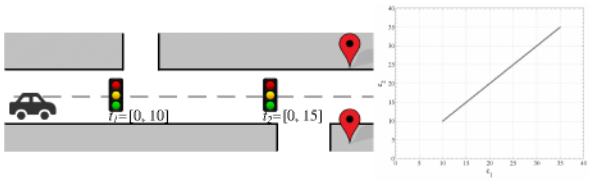
[2] A constraint matrix of an optimization problem consists of coefficients of variables. Here, $\mathbf{B}_k$ is a set of columns corresponding to basic variables. Similarly, $\mathbf{A}_{N_k}$ corresponds to the coefficients of nonbasic variables.

will assume that domain knowledge permits specification of the boundary of $\mathbb{C}$. If only upper and lower bounds of costs are known, we can define the largest cost boundary as in Fig. 2(a), that is, as an n^2-orthotope.[3] In other words, $\mathbf{C} \in \mathbb{C}$ has, for each c_{ij}, a range $\underline{c}_{ij} \leq c_{ij} \leq \bar{c}_{ij}$. While this serves as a concise characterization of uncertain costs, it fails to capture any interrelationships between different costs.

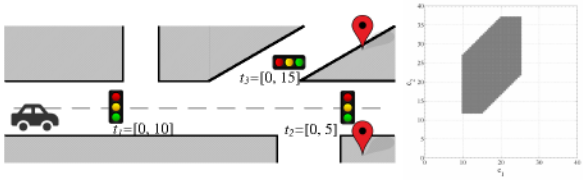
Costs that have interrelationships can be modeled by having a more complex boundary for $\mathbb{C}$ as shown in Fig. 3(a), where the boundary is modeled as a linear function. A slightly richer example appears in Fig. 3(b) where part of a route is shared between two destinations but after a fork in the road there are different waiting times and distances. The resultant cost boundary is a convex polygon.

In practice, costs could have rather more complex boundaries. Fig. 3(c) shows an example of Zermelo's navigation problem [31]. Suppose an underwater vehicle capable of navigating at a certain maximum speed and heading angle moves through a water flowing with some current. The problem is to find the time-optimal path from a position to a destination. If the vehicle solves the problem by steadily aiming at the appropriate fixed angle to the current, the vehicle is able to navigate along a straight path to the destination, which is optimal. The cost of motion changes with changes in the direction of the current. If the current may have any direction, the corresponding boundary of the costs is nonlinear and convex as shown in the right graph in Fig. 3(c).

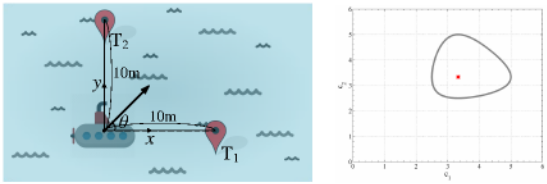
More complex cases might result in cost regions with nonconvex boundaries or interior holes. Although the methods described herein cannot handle such cases exactly, the approach may still have utility if the region is reasonably approximated. For example, a simple way to find such regions is to use the maximum and the minimum points in each dimension of the cost space, or to compute the convex hull. Such over-approximations do not alter the correctness of the algorithms but may cause the robots to incur additional, unnecessary computation/communication because the algorithms' attempts to avoid expending effort will be conservative.



(a) Left: Two destinations have routes that share a common segment. When cost is proportional to driving time, traffic signals (with unknown state) induce a delay which affects both. Right: The line represents the possible costs.



(b) Left: A route forks and each fork has different waiting times and lengths. One traffic signal affects both driving times to the destinations but other signals influence the times independently. Right: The corresponding convex linear cost boundary.



(c) Left: An underwater vehicle plans to navigate to each of two destinations. The vehicle is influenced by water current and adjusts its heading to cancel out the current. The costs depend on the angle θ of the current. Right: The corresponding convex nonlinear cost boundary when $0 \leq \theta < 2\pi$ and the vehicle moves at the maximum speed.

Figure 3: Richer robot navigation scenarios and their corresponding cost boundaries.

5 Problem Description

In a centralized system, a single unit computes an allocation of robots to tasks and propagates the result to the robots. We will assume that any robot is capable of acting as the central unit. This is a common assumption as

[3]The dimensionality of the cost space of an MRTA problem is often extremely high. The 2-D representation in the figure is merely an aid to presentation.

(a) Finding partitions a team to bound communication within the sub-teams (red and blue ovals)

(b) Understanding the cost incurred by forgoing further communication and sticking with the initial assignment (arrows)

(c) Limiting communication to nearby robots to determine if the current assignment is still optimal despite cost changes

Figure 4: Three proposed ways of reducing centralized global communication. They require an upfront computation before the robots are dispatched, but in appropriate circumstances the result of the computation reduces run-time communication significantly.

often leader election algorithms choose the distinguished unit dynamically to ensure that failure of any single robot never debilitates the whole system. To achieve optimal coordination, the central unit should be able to communicate with all robots via a network to (i) collect the most recent cost estimates from the robots and (ii) notify them of task allocations, if the updated costs alter the previous allocation. Maintaining global connectivity in multi-robot systems is expensive, however, and the quality of communication can undergo drastic changes during task execution [22].

It is, therefore, worthwhile to develop methods for distributing the assignment problem in a way which will alleviate the strong and continual dependence on centralization. We assume that robots have some initial opportunity for global communication (e.g., before the robots are dispatched). They use this time to perform additional upfront computation for a *locker-room agreement*, the results of which are used afterward while costs change dynamically. It is worth noting that our focus is on reducing the degree of connectivity of a network and the number of messages communicated in the network irrespective of the mechanism used for maintaining the connectivity.

For a given cost region $\mathbb{C}$, an optimal assignment can be associated to each different cost matrix within the region. If all changes to costs preserve the same assignment, then alterations of the costs, within the bounds of $\mathbb{C}$, have no effect on the optimality of the robot–task pair matching. However, it is more likely that changes *will* effect optimality of the matching: let N be the number of optimal assignments within $\mathbb{C}$. The upfront computation proposed in Sec. 6.2 computes all N optimal assignments $\mathbf{X}_q^*$ for $q \in \{1, \cdots, N\}$ in $\mathbb{C}$ and their $\theta(\mathbf{X}_q^*)$ for each optimal assignment.

Three methods are proposed to make use the result of the upfront computation; each of them helps reduce communication between the central unit and other robots during the task execution (i.e., run-time) phase:

§ 5.1 The team of robots is partitioned into sub-teams (Fig. 4a) so that each sub-team need only communicate internally, if such sub-teams exist.

§ 5.2 The current optimal assignment is compared to other possible assignments within a given cost region. This assesses whether the benefit of communicating actually outweigh that of merely persisting with the current assignment (Fig. 4b).

§ 5.3 Once a robot discovers that its costs have changed, whether these changes have an impact which global or only parochial can be determined by starting with local queries and then gradually escalating their range of communication (Fig. 4c).

5.1 With whom do the robots need to communicate?

By *sub-team* we will mean a proper subset of robots from the entire system, where robots in a sub-team have non-overlapping sets of tasks with robots other sub-teams. Singleton sets of robots satisfy this criterion for a single assignment (the mutual exclusion constraints (2) and (3) ensure this fact). But once the costs may be altered, there are multiple tasks that might be assigned to a robot, and

they depend the precise values involved, the realizations of random variables, and so forth. This makes the notion of a sub-team as well as the preceding criterion both more interesting and more useful.

If such sub-teams exist they need never communicate with robots outside their own sub-team (or even take the other sub-teams into account) when computing their task allocations. [4] Moreover, each sub-teams computation will construct the globally optimal allocation. Partitioning the team of robots into multiple sub-teams yields the benefit reducing communication range and computation load (optimal assignment algorithms have super-linear running-time) when costs could change without sacrificing optimality. Partitioning teams where tasks have spatial locality [26] and sparsity [18] has been studied previously, but we make no assumptions about the particular properties.

Then, when sub-teams exist, the important question becomes one of whether such sub-teams can be found. With cost region $\mathbb{C}$, the possible assignments depend on the cost matrix $\mathbf{C} \in \mathbb{C}$. If all N optimal assignments within the region can be found, one can determine whether the team can be partitioned by basic matrix operations (described in Sec. 6.3). Since costs are nonnegative real numbers, computing assignments for all possible costs is intractable. The problem is to find a set of all possible assignments $\mathbf{X}_q^*$ for $q \in \{1, \cdots, N\}$ quickly so that finding partitions can be done efficiently.

5.2 When should the robots stick with their initial assignment?

Important research has examined how to maintain global connectivity in networks of robots, including, for example, control-based schemes [27] or multi-hop routing protocols [8]. But when is the expense incurred by these methods for maintaining global connectivity prohibitive? Clearly this is a question that can must be answered in a contextually specific way. In the task allocation setting one can assess the difference in assignment quality with and without further communication. Then, with some measure of cost incurred in sending messages, one can make a determination for the particular domain at hand. Specifically, the cost difference between the *worst-case cost sum*—if the robots commit to their initial assignment and use no run-time communication—and the *best-case cost sum*—if they update costs and reallocate—can be used to make the decision. A small cost difference means that the cost reduction obtained by maintaining global connectivity is minor. If the expense for global communication[5]is larger than this reduction, the team does not profit from updating costs and recomputing assignments.

Once optimal assignments $\mathbf{X}_q^*$ and their $\theta(\mathbf{X}_q^*)$ for $q \in \{1, \cdots, N\}$ are computed, the cost difference is computed by finding the minimum cost matrix $\mathbf{C}_{\min_q}$ in each $\theta(\mathbf{X}_q^*)$ and computing

$$\max(\bar{\mathbf{C}} \odot \mathbf{X}_1^* - \mathbf{C}_{\min_q} \odot \mathbf{X}_q^*), \tag{8}$$

for $q \in \{1, \cdots, N\}$ where $\mathbf{X}_1^*$ is the initial (or current) assignment, and the $\odot$ symbol is an operator which denotes the sum of all elements in the Hadamard product of two matrices, that is, $\mathbf{A} \odot \mathbf{B} \equiv \mathbf{1}^T(\mathbf{A} \circ \mathbf{B})\mathbf{1}$. In other words, (8) is the cost difference between the minimum among cost sums (where robots change their assignments) and the maximum cost sum (if robots maintain their initial assignment).

5.3 When costs change, how do you notify only those for whom it matters?

Even if a team or a sub-team of robots decide to respond to every cost change according to the decision made in Sec. 5.2, there are still opportunities to curtail unnecessary communication. Suppose that robots have some knowledge about their own costs (e.g., robot R_a manages a subset of costs c_{ij} for $i = a$, $j \in \{1, \cdots, n\}$) that is a set of tolerable ranges of the costs that preserve the current optimal assignment, *irrespective of how costs of other robots change*. Any cost changes within the ranges incur no communication as those changes are guaranteed not to break global optimality. With this knowledge, the robots can work independently until any of the robots has a cost(s) that deviates from its range.

The robots may have to begin communicating with other robots if a violation occurs. Suppose that the costs of R_a change according to a new information (e.g., a door, which R_a had previously assumed to be open, is found to in fact be locked). This new information may incur violations of the tolerable cost ranges associated with R_a. We describe two cases: (i) only one robot at each time step experiences the violations and (ii) multiple robots could have violations simultaneously. The case (i) could be reasonable in environments with local or infrequent dynamic changes whereas (ii) is more appropriate when environmental changes occur frequently and likely have a

[4]Being in the same sub-team does not necessarily mean that they are always connected directly. Thus, robots in the same sub-team may need multi-hop communication at some time frames during execution. However, finding sub-teams still reduces communication and computation by splitting the whole communication network into multiple small networks.

[5]In this paper, we are not explicit about the expense of communication since it depends on the method used. A common measure might be, e.g., time (with delays incurred by multi-hop routing), or total energy expended.

global influence. In both cases, the first problem is to obtain the knowledge about the tolerance ranges of costs. Geometrically, the cost ranges construct an n^2-orthotope $\mathcal{T}_{ij} = [c_{ij} - \tau_{ij}, c_{ij} + \tau_{ij}]$ for all i and j for the current assignment where a low-dimensional projection of $\mathcal{T}_{ij}$ represents a 1-D interval for each cost. Once the intervals of the costs are calculated, the two cases have different ways of using them to reduce communication.

The robot can assess for itself (without communication) whether the new information incurs cost changes that break global optimality. If the knowledge is not sufficient for the self-assessment, the robot would need additional information about how the costs of other robots have changed. This is the point at which R_a should communicate locally with some other (nearby) robot, say R_b, to know the costs of R_b. Notice that the nearby robot to be included in the local communication could be determined depending on the ease of establishing communication. With R_a and R_b together, it may be the case that the assignment remains optimal. If this is not, then the process should be repeated by bringing some new robot R_c into the fold. There might be some other robots that perceive the locked door and update their costs. Each of those robots could perform the local checking process for itself simultaneously while others do.

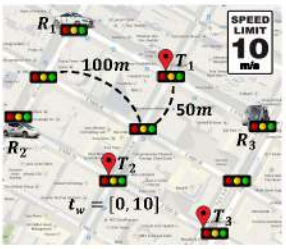
The problem is to obtain the prior knowledge of about the tolerance of the current assignment. Geometrically, it is to find an n^2-orthotope $\mathcal{T}_{ij} = [c_{ij} - \tau_{ij}, c_{ij} + \tau_{ij}]$ for all i and j for the current assignment where a low-dimensional projection of $\mathcal{T}_{ij}$ represents a 1-D interval for each cost. With $\theta(\mathbf{X}_q^*)$, the validity of the current optimal assignment should be checked by incorporating all costs. This is checking whether the current costs, which is a point in the n^2-dim cost space, reside in $\theta(\mathbf{X}_q^*)$. This process requires a collection of current costs through some centralized structure. With the 1-D intervals from $\mathcal{T}_{ij}$, each cost can be checked independently from other costs whether the cost is in its interval. When a robot knows the intervals for its costs, the robot can work independently until any of updated costs leave the intervals. However, distributing the checking process does not come free; the n^2-orthotope $\mathcal{T}_{ij}$ is "conservative" than $\theta(\mathbf{X}_q^*)$ (i.e., $\mathcal{T}_{ij} \subset \theta(\mathbf{X}_q^*)$). If cost changes do not violate any of intervals, nothing need be done since the current assignment is preserved as $\theta(\mathbf{X}_q^*)$ is not violated. Since a violation of $\mathcal{T}_{ij}$ does not necessarily violate $\theta(\mathbf{X}_q^*)$, a robot with costs violating $\mathcal{T}_{ij}$ needs to know the costs of other (nearby) robots to check whether a subspace of $\theta(\mathbf{X}_q^*)$ constructed by the costs of the robots involved is violated. For example, suppose that $\mathbf{C}$ is the cost matrix at an arbitrary time step. In the next step, a cost of one robot changes so $\mathbf{C}$ becomes $\mathbf{C}'violating \notin \theta(\mathbf{X}_q^*)$. It is entirely possible for the cost change of another robot to produce $\mathbf{C}''$ which is in $\theta(\mathbf{X}_q^*)$. If even just these two robots communicate, the current assignment is preserved, and the conclusion has been reached via local communication.

5.4 An illustrative example

We give an example multi-robot navigation problem to show how the proposed methods can be used together. Suppose we have three autonomous robots ($R_{1,2,3}$) and three destinations ($T_{1,2,3}$) as shown in Fig. 5(a). The goal is to have one robot at each destination while minimizing the total sum of traveling times (in seconds). The reader might like to think of it in terms of self-driving taxis picking up customers while the taxi company aims to minimize the total fuel cost (which is proportional to the total traveling time). We assume that the robots drive through the shortest path, and each intersection has a traffic signal. The waiting time at each signal is $t_w \in [0, 10]$. Again, we assume that robots drive at the maximum speed (10 m/s) when they move and, for simplicity, we model neither delays from congestion nor acceleration/deceleration. The corresponding cost matrix is shown in Fig. 5(b).

The initial optimal assignment is $\mathbf{X}_1^* = \left(\begin{smallmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{smallmatrix}\right)$ supposing green lights. First, the robots use the method described in Sec. 5.2 that computes the maximum cost difference (8) to decide whether the robots respond to every cost change. The highest cost sum if the robots keep the initial assignment is 100 (when c_{11}, c_{22}, c_{33} are at the upper bounds). There is no run-time communication and task reassignment with this assignment. If they are committed to update every cost change and recompute the assignment with updated costs, the lowest possible cost sum is 50 (i.e., when $\mathbf{X}^* = \left(\begin{smallmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{smallmatrix}\right)$ and the costs are at the lower bounds). If the expense for global communication is smaller than the difference, the robots keep communicating with the central unit to update cost changes. Otherwise, they simply stick with the initial assignment.

In the case where the robots communicate with each other, the robots try to partition the team into sub-teams using the method described in Sec. 5.1. When such partitions are found, the sub-teams can work independently while not communicating with those robots in other sub-teams. The method for escalating local communication described in Sec. 5.3 can be used within a sub-team or the entire team as and when each robot detects changes in its costs (such as a new speed limit or road closure, in this example).



(a) Multi-robot navigation

	T_1	T_2	T_3
R_1	$[10, 30]$	$[20, 60]$	$[30, 80]$
R_2	$[20, 60]$	$[10, 30]$	$[20, 50]$
R_3	$[10, 30]$	$[20, 60]$	$[10, 40]$

(b) Corresponding cost matrix

Figure 5: An MRTA problem with costs that vary within ranges. The mission objective is to have one of the three robots at each destination, shown in (a), while minimizing the total traveling time. Possible ranges for costs in this instance are collected in the matrix in (b). The methods we describe reduce centralized operation during runtime via upfront computation that is completed before the robots are dispatched.

6 Algorithms

As it is essential to compute (7), in this section we first describe an inexact randomized method for sensitivity analysis that runs much faster than the exact method described in Sec. 3.2. Next, we describe the algorithm employing the randomized (or the exact) sensitivity analysis that computes all N possible assignments $\mathbf{X}_q^*$ and their $\theta(\mathbf{X}_q^*)$ for $q \in \{1, \cdots, N\}$. This algorithm is executed upfront before the robots start performing tasks, then finally, using the result of this prior computation, we propose three algorithms for the problems described in Sec. 5.

6.1 A randomized sensitivity analysis

Any cost $\mathbf{C}$ within the set of linear boundaries $\theta(\mathbf{X}_q^*)$ produces the same optimal assignment $\mathbf{X}_q^*$ for an arbitrary q. The exact method of computing the set is described in Sec. 3.2 which computes (7) by enumerating a factorial number of feasible solutions. Specifically, a feasible solution must include n basic variables (corresponding to the optimal assignment) out of all n^2 variables. Among the remaining $n^2 - n$ variables, $n - 1$ non-degenerate basic variables need to be chosen to complete a feasible solution which consists of $2n - 1$ basic variables. The exact method $\text{SA}(\mathbf{X}_q^*, \mathbf{C}_q)$ enumerates all $k = \binom{n^2-n}{n-1}$ choices. The running time grows exponentially with the input size. Though this computation is done off-line, the method cannot produce a solution in tolerable time unless the instances are small (we show in the section describing experiments that it takes a few minutes for $n < 7$). A faster method is needed for larger instances.

We develop a randomized algorithm, $\text{RANDSA}(\mathbf{X}_q^*, \mathbf{C}_q)$, to facilitate a faster computation of $\theta(\mathbf{X}_q^*)$. Let H' be a partial set of all feasible solutions' indices (recall that H is the complete set). A partial enumeration of the variables brings an incomplete set of linear boundaries, which is $\theta'(\mathbf{X}^*) = \bigcup_{k \in H'} CR_k$, but the incomplete set often covers a large portion of $\theta(\mathbf{X}_q^*)$. From this observation, we implement Alg. 1. With the given n basic variables, the additional $n - 1$ variables are randomly chosen. A feasible solution with the $2n - 1$ basic variables makes a coefficient matrix $\mathbf{B}_k$; one of full rank (i.e., nonsingular) is needed since (6) computes the inverse of $\mathbf{B}_k$ (line 12). The algorithm iterates the while loop (lines 4–15) until the randomly chosen variables complete the basic variables of a feasible solution. If a feasible solution is found, it produces $(n-1)^2$ linear boundaries (line 17). We experimentally verified that even a single feasible solution is enough to produce high-quality solutions. Should a single feasible solution be insufficient, line 4 can be modified to add more feasible solutions to F (along with the addition of a check before line 13 to determine whether J is already in F will avoid duplicating work).

Algorithm 1 RANDSA

Input: The $n \times n$ cost matrix $\mathbf{C}_q$ and its optimal assignment $\mathbf{X}_q^*$ for an arbitrary q
Output: A partial set of linear boundaries $\theta'(\mathbf{X}_q^*)$

1 $F = \varnothing$
2 $S^* = \{s : s \text{ is an index of a variable in } \mathbf{X}_q^* \text{ whose assignment is 1}\}$
3 $S = \{1, 2, \cdots, n^2\} \setminus S^*$
4 **while empty**(F)
5 $S_R = \varnothing$
6 **for** l in 1 to $n-1$ // choose $n-1$ additional basic variables from S
7 $i \leftarrow$ a randomly chosen index from S
8 $S_R = S_R \cup i$
9 $S = S \setminus i$
10 **end for**
11 $J = S^* \cup S_R$ // index set of basic variables in a feasible solution
12 **if** $\mathbf{B}_k$ is full rank . . . . . // $\mathbf{B}_k$ in (6) should be invertible
13 $F = F \cup J$
14 **end if**
15 **end while**
16 $H' = \{1, 2, \cdots, |F|\}$
17 $\theta'(\mathbf{X}^*) = \bigcup_{k \in H'} CR_k$ . . . . . // CR_k computed by (6)
18 **return** $\theta'(\mathbf{X}_q^*)$

6.2 Finding all optimal assignments and $\theta(\mathbf{X}^*)$ in $\mathbb{C}$

A cost region $\mathbb{C}$ may contain multiple optimal assignments. By employing sensitivity analysis (either the exact method or Alg. 1), we develop an algorithm that finds $\theta(\mathbf{X}_q^*)$ for all $\mathbf{X}_q^*$ where $q \in \{1, \cdots, N\}$.

An assignment problem has at least 4-dimensions (two robots and two tasks) so hard to visualize the geometry. A cartoon 2-D representation appears in Fig. 6(a) for pedagogical purposes. One difference from higher dimensional cases is that all linear equations in $\theta(\mathbf{X}^*)$ are greater or equal to zero (see (6)), but the upper boundary of $\theta(\mathbf{X}^*)$ in Fig. 6(a) has the opposite inequality.

We have an initial optimal assignment $\mathbf{X}_1^*$ for an initial cost matrix $\mathbf{C}_1$ and its $\theta(\mathbf{X}_1^*)$ (Alg. 2, lines 2–3 where HUNGARIAN is the standard Hungarian method [14]). Let l be an arbitrary linear boundary in $\theta(\mathbf{X}^*)$. If the objective value is greater than or equal to zero when l is maximized over the shaded area[6] (line 6), l contains the entire cost set (the shaded area $\mathbb{C}$ in Fig. 6a). Otherwise,

[6] All l should be maximized because of the inequalities in (6).

the shaded area is not covered by l (see Fig. 6b) thus a cost on l is perturbed to find a new $\theta(\mathbf{X}^*)$ that includes the remainder of $\mathbb{C}$ (lines 12–22). As the current $\theta(\mathbf{X}^*)$ is expanded by perturbing the costs, newly found $\theta(\mathbf{X}_q^*)$ regions are merged and checked as well. The algorithm terminates if $\theta(\mathbf{X}^*)$ completely includes $\mathbb{C}$. It returns all $\mathbf{X}_q^*$ and $\theta(\mathbf{X}_q^*)$ found.

In Fig. 6(a), the direction of a perturbation is toward $\mathbf{c}'$. The magnitude of the perturbation, ϵ, must be carefully chosen. Too large a value may skip some $\theta(\mathbf{X}_q^*)$. The following describes how we determine the optimal magnitude and the direction of a perturbation.

Lemma 5.1 Let l be an arbitrary linear boundary in $\theta(\mathbf{X}^*)$. A perturbation with magnitude ϵ to perturb an arbitrary point $\mathbf{p}$ on l will not miss any $\theta(\mathbf{X}^*)$ if $0 \leq \epsilon \leq \frac{|\mathbf{p}|}{n}$.

Proof. Let $\mathbf{v}$ be the normal of an arbitrary linear boundary l in $\theta(\mathbf{X}^*)$. From line 6 in Alg. 2, we have $\mathbf{c}'$, which is an extreme point of $\mathbf{C}$ outside of the current $\theta(\mathbf{X}^*)$. The projection of $\mathbf{c}'$ onto $\mathbf{v}$ is

$$\mathbf{p} = \frac{\mathbf{c}' \cdot \mathbf{v}}{|\mathbf{v}|^2}\mathbf{v}.$$

Suppose that $\mathbf{v}_c$ is the normal vector of the nearest boundary to l (other than itself). Let $\mathbf{q}$ be a vector orthogonal to $\mathbf{v}_c$. The direction of movement from $\mathbf{p}$ to $\mathbf{q}$ is along a vector

$$\mathbf{p}_{\text{new}} = \mathbf{p} + d(\mathbf{q} - \mathbf{p})$$

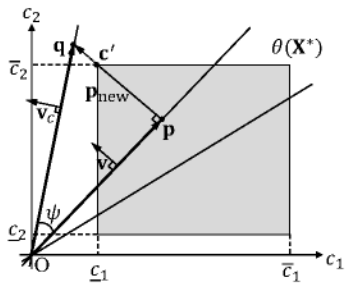
where $d = |\mathbf{p}| \tan\psi$ is the magnitude of the move. We look for the minimum d because $\mathbf{p}_{\text{new}}$ is toward the closest boundary. Since $\tan\psi$ is an increasing function in $(-\frac{\pi}{2}, \frac{\pi}{2})$, d is also minimized if ψ is minimized.

Normalized vectors of the above vectors are denoted as $\hat{\mathbf{v}}, \hat{\mathbf{v}}_c, \hat{\mathbf{p}}, \hat{\mathbf{p}_{\text{new}}}$, and $\hat{\mathbf{q}}$. Since $\hat{\mathbf{v}} \perp \hat{\mathbf{p}}$ and $\hat{\mathbf{v}}_c \perp \hat{\mathbf{q}}$, the angle between $\hat{\mathbf{v}}$ and $\hat{\mathbf{v}}_c$ is ψ as well. Therefore, $\hat{\mathbf{v}} \cdot \hat{\mathbf{v}}_c = \cos\psi$. Since $\psi = \arccos \hat{\mathbf{v}} \cdot \hat{\mathbf{v}}_c$, ψ is minimum at maximum $\hat{\mathbf{v}} \cdot \hat{\mathbf{v}}_c$.

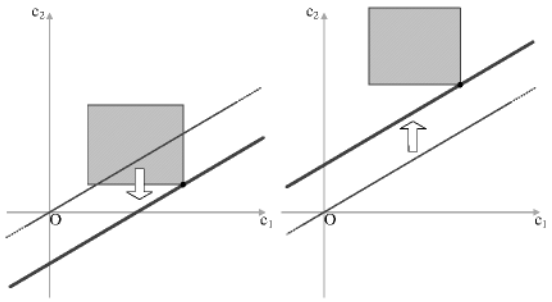
The coefficients (normals) of linear boundaries in $\theta(\mathbf{X}^*)$ are $-1, 0$, or 1 since $\mathbf{B}_k$ and $\mathbf{A}_{N_k}$ are from a totally unimodular coefficient matrix. To maximize the dot product of two normals of boundaries, the boundaries should have the maximum number of 1's (or the maximum number of -1's). The vector $\hat{\mathbf{1}}$ ensures this case, but two boundaries should be distinct; the other boundary must have a single 0. Thus, the product of $\frac{1}{\sqrt{n^2}}[1\,1\,1\cdots1\,1]$ and $\frac{1}{\sqrt{n^2-1}}[1\,1\,1\cdots1\,0]$ is the maximum, that is $\frac{n^2-1}{\sqrt{n^2}\sqrt{n^2-1}}$ where n is the dimension of the cost space.

Now we have $\psi_{\min} = \arccos \frac{n^2-1}{\sqrt{n^2}\sqrt{n^2-1}}$. Therefore,

$$d_{\min} = |\mathbf{p}| \tan\left(\arccos \frac{n^2-1}{\sqrt{n^2}\sqrt{n^2-1}}\right) = \frac{|\mathbf{p}|}{\sqrt{n^2-1}},$$



(a) The perturbation process.



(b) Objective values for two cases.

Figure 6: A 2-D representation of the cost space. (a) Bold lines represent linear boundaries (hyperplanes) of $\theta(\mathbf{X}^*)$ and the shaded area represents a cost region $\mathbb{C}$ bounded by $\underline{\mathbf{C}}$ and $\overline{\mathbf{C}}$. (b) If a boundary does not cover all shaded area, the objective value of maximization over the area is negative (left). Otherwise, the value is nonnegative (right).

which is the distance to the closest boundary along $\mathbf{p}_{\text{new}}$. A safe magnitude for perturbation $0 \leq \epsilon \leq \frac{|\mathbf{p}|}{n} < d_{\min}$, ensures we do not skip any $\theta(\mathbf{X}_q^*)$. □

If Alg. 2 employs Alg. 1 to speedup computation, a partial set $\theta'(\mathbf{X}^*)$ will cover a smaller area than the full $\theta(\mathbf{X}^*)$. Thus, a perturbation with ϵ still never skips any $\theta(\mathbf{X}_q^*)$. However, a single perturbation may not be enough to find a new optimal assignment if the perturbed cost is in between the actual area and the underestimated area. In this case, perturbing multiple times will eventually find a new assignment. When $\theta'(\mathbf{X}^*)$ is close to $\theta(\mathbf{X}^*)$ there are few repetitions.

Note that the shape of cost regions relates to line 6 in Alg. 2, in which the feasible region of the optimization problem is the cost region. If the cost region is linearly constrained convex (i.e., a hyper-polytope such as Fig. 3b), one is able to find the vertices of the cost region that maximize the objective value with linear programming in polynomial time.

On the other hand, the problem becomes the optimiza-

Algorithm 2 FINDTHETA

Input: An $n \times n$ cost matrix $\mathbf{C}_1$, $\underline{\mathbf{C}}$, and $\bar{\mathbf{C}}$
Output: A set of assignments $\mathbf{X}_q^*$ and $\theta(\mathbf{X}_q^*)$ for $q \in \{1, \cdots, N\}$

1 $i = 0, q = 2$
2 $\mathbf{X}_1^* = \text{HUNGARIAN}(\mathbf{C}_1)$
3 $\theta(\mathbf{X}_1^*) = \text{SA}(\mathbf{X}_1^*, \mathbf{C}_1)$ // compute (7)
4 $\theta(\mathbf{X}^*) = \theta(\mathbf{X}_1^*)$
5 **do forever**
6 $(\mathbf{c'}_i, obj_i) = \text{LINPROG}(l_i, \underline{\mathbf{C}}, \bar{\mathbf{C}}, \max)$ // max l_i over the bounds // where l_i is the i^{th} linear boundary in $\theta(\mathbf{X}^*)$
7 **if** $obj_i \geq 0$ // if $\mathbb{C}$ does not satisfy $l_i \geq 0$
8 $i = i + 1$
9 **if** $i = |\theta(\mathbf{X}^*)|$ // if all linear boundaries in $\theta(\mathbf{X}^*)$ checked
10 **break**
11 **end if**
12 **else** // perturb a point on l_i toward $\mathbf{c}'$ to find a new $\mathbf{X}^*$ and $\theta(\mathbf{X}^*)$
13 $\mathbf{p} = \frac{\mathbf{c'} \cdot \mathbf{v}_i}{|\mathbf{v}_i|^2} \mathbf{v}_i$ // $\mathbf{p}$ is a projection of $\mathbf{X}_i$ onto l_i
14 $\epsilon = \frac{|\mathbf{p}|}{n}$
15 $\mathbf{p}_{\text{new}} = \mathbf{p} + \epsilon(\mathbf{c'} - \mathbf{p})$
16 $\mathbf{C}_q = \text{RESHAPE}(\mathbf{p}_{\text{new}}, n)$ // reshape vector into $n \times n$ matrix
17 $\mathbf{X}_q^* = \text{HUNGARIAN}(\mathbf{C}_q)$
18 $\theta(\mathbf{X}_q^*) = \text{SA}(\mathbf{X}_q^*, \mathbf{C}_q)$
19 $\theta(\mathbf{X}^*) = \theta(\mathbf{X}^*) \cup \theta(\mathbf{X}_q^*)$
20 $i = 0$
21 $q = q + 1$
22 **end if**
23 **end do**
24 **return** $\{\mathbf{X}_1^*, \cdots, \mathbf{X}_N^*\}$ and $\{\theta(\mathbf{X}_1^*), \cdots, \theta(\mathbf{X}_N^*)\}$

tion of a linear objective function subject to a nonlinear convex constraint set (which we call LONC) if the feasible region is convex but nonlinear (as in the example in Fig. 3c). To show that Alg. 2 still has the same time complexity with the convex linear feasible region case, we prove that LONC is in $\mathcal{P}$. We show a polynomial-time reduction from LONC to the optimization of a nonlinear convex objective function subject to a linear convex set (NOLC), which is proven to be in $\mathcal{P}$ [3].

Theorem 5.2 LONC is in $\mathcal{P}$.

Claim. If LONC $\leq_{\text{P}}$ NOLC, LONC is in $\mathcal{P}$ since NOLC is in $\mathcal{P}$.

Proof. In line 6 of Alg. 2, the objective function of the linear programming is separable. If both the objective function and the set of constraints are separable, LONC is easily transformed to NOLC in polynomial time by variable substitution (an example of the substitution is given in the Appendix). If the constraint set is represented by nonseparable functions, some known methods in Table 13.1 of [4] can transform nonseparable functions to separable in polynomial time. Therefore, LONC $\leq_{\text{P}}$ NOLC. □

However, the problem becomes $\mathcal{NP}$-hard if the feasible region is nonconvex [2]. In this case, we can find a minimum convex region that includes the nonconvex region as we discussed briefly in Sec. 4. Alternatively, we can replace the linear programming solver (line 6 in Alg. 2) by a solver appropriate to the shape of the feasible region, using a suitable polynomial-time approximation algorithm for nonconvex optimization problems (see the review of optimization solvers in [15]). For the cost regions with k isolated convex regions, one may still use Alg. 2 applying it k times for each sub-region. Each run produces the result for that sub-region, and the last step involves eliminating duplicated pieces among the results.

The preceding discussion of hardness results show that the analysis of the assignment optimality can include interrelated costs essentially for free if they are linear, but are tractable even for costs related to problems like that of Zermelo's (Fig. 3c). This is important for experimental results in Sec. 7.6 which will show that treating interrelated costs as independent significantly decreases the value of the analysis.

6.3 Partitioning the team of robots

Partitioning a team of robots into sub-teams, as discussed in Sec. 5.1, is performed via elementary matrix operations on all the assignment matrices computed from Alg. 2. First, the sum of the assignments $\mathbf{X}_C = \sum_{q=1}^{N} \mathbf{X}_q^*$ is computed. Zero elements in $\mathbf{X}_C$ mean that the associated robot–task pairs will be never assigned. The matrix

columns and rows are exchanged to find a block diagonal matrix, where a polynomial time method exists. For example, a method converts $\mathbf{X}_C$ to a 0-1 matrix (nonzero values of $\mathbf{X}_C$ becomes 1 while zero values do not change), which takes $O(n^2)$, and uses a $O(n \log n)$ quicksort twice (row-wise and column-wise) using the binary values of rows and columns. Then the time complexity of the method is $O(n^2)$. If a block diagonal matrix can be found, the main diagonal blocks represent sub-teams. Given such an input matrix, clearly communication and computation localized to within each sub-team suffices for the robots to achieve global optimality.

We show an example of finding sub-teams from the output of Alg. 2. Suppose Alg. 2 returns three assignments whose sum is

$$\mathbf{X}_C = \begin{pmatrix} 1&0&0&0\\0&0&1&0\\0&1&0&0\\0&0&0&1 \end{pmatrix} + \begin{pmatrix} 0&1&0&0\\0&0&1&0\\1&0&0&0\\0&0&0&1 \end{pmatrix} + \begin{pmatrix} 0&1&0&0\\0&0&0&1\\1&0&0&0\\0&0&1&0 \end{pmatrix} = \begin{pmatrix} 1&2&0&0\\0&0&2&1\\2&1&0&0\\0&0&1&2 \end{pmatrix}.$$

Then $\mathbf{X}_C$ is converted to a 0-1 matrix where each row (and column) has a binary string. Sorting the rows in a descending order followed by a column-wise sorting yields a block diagonal matrix as shown below.

$$\begin{pmatrix} 1&2&0&0\\0&0&2&1\\2&1&0&0\\0&0&1&2 \end{pmatrix} \xrightarrow{\text{conversion}} \begin{pmatrix} 1&1&0&0\\0&0&1&1\\1&1&0&0\\0&0&1&1 \end{pmatrix} \xrightarrow{\text{sort}} \begin{pmatrix} 1&1&0&0\\1&1&0&0\\0&0&1&1\\0&0&1&1 \end{pmatrix}$$

Practically, the diagonalization is simply done using a MATLAB function `blkdiag`. The two diagonal blocks represent two sub-teams because the two robots corresponding to the first two rows do not share any tasks with the other two robots.

6.4 Choosing to persist with the initial assignment

The idea proposed in Sec. 5.2 to analyze the impact of totally forgoing further communication despite cost changes is implemented in Alg. 3. All assignments $\mathbf{X}_q^*$ and $\theta(\mathbf{X}_q^*)$ for $q \in \{1, \cdots, N\}$ are given by Alg. 2 (line 1). For each $\theta(\mathbf{X}_q^*)$, we seek the minimum costs $\mathbf{C}_q$ over $\theta(\mathbf{X}_q^*)$ within the bounds $\underline{\mathbf{C}}$, and $\bar{\mathbf{C}}$ (line 3). For each q, we have an assignment $\mathbf{X}_q^*$ and the minimum cost $\mathbf{C}_{\min_q}$. In line 6, $\min(\mathbf{C}_{\min_q} \odot \mathbf{X}_q^*)$ returns the minimum cost sum among the N assignments. On the other hand, we can compute the maximum cost sum if robots never change from their initial assignment with $\bar{\mathbf{C}} \odot \mathbf{X}_1^*$. Thus, line 6 gives the maximum cost loss when robots persist with the initial assignment and opt to avoid communication and reassignment.

As described in Sec. 5.4, one can decide with c_{worst} whether to persist with the initial assignment by considering the communication/computational expense of a reassignment.

Algorithm 3 MAXLOSS

Input: An $n \times n$ cost matrix $\mathbf{C}_1$, $\underline{\mathbf{C}}$, and $\bar{\mathbf{C}}$
Output: A maximum cost difference c_{worst}

1 $(\mathbf{X}_q^*, \theta(\mathbf{X}_q^*)) =$ FINDTHETA$(\mathbf{C}_1, \underline{\mathbf{C}}, \bar{\mathbf{C}})$ for $q \in \{1, \cdots, N\}$
2 **for** $q = 1$ to N
3 $\quad (\mathbf{c'}_q, obj_q) =$ LINPROG$(\mathbf{c}, \theta(\mathbf{X}_q^*), \underline{\mathbf{C}}, \bar{\mathbf{C}}, \min)$ // minimize costs over $\theta(\mathbf{X}_q^*)$ with the bounds
4 $\quad \mathbf{C}_q =$ RESHAPE$(\mathbf{c'}_q, n)$
5 **end if**
6 $c_{\text{worst}} = \max\{\bar{\mathbf{C}} \odot \mathbf{X}_1^* - \min(\mathbf{C}_{\min_q} \odot \mathbf{X}_q^*)\}$
7 **return** c_{worst}

6.5 Incremental communication

We develop an algorithm for determine whether cost changes destroy the optimality the current assignment through escalation of the communication neighborhood, the idea discussed in Sec. 5.3. A set of boundaries $\theta(\mathbf{X}^*)$ can be summarized in different ways to show relevant information about the effect of cost changes. A one-dimensional cut provides a lower and an upper bound for each cost. This interval is valid if all other costs remain unchanged. The generalization, α-dimensional cuts, allow simultaneous changes of α costs, but $n^2 - \alpha$ costs must remain unchanged. Tolerance approaches, such as that of [9], find a single value (*tolerance percentage*) to represent the maximum cost perturbation which can be applied simultaneously and independently without affecting optimality. The current cost matrix $\mathbf{C}$ is expanded in all dimensions to have the margins as much as the tolerance percentage. It produces a tolerance cost region, essentially an n^2-orthotope in which each dimension is bounded by an interval $c_{ij} - \tau_{ij} \leq c_{ij} \leq c_{ij} + \tau_{ij}$ where $\tau_{ij} \in \mathbb{R}^{\geq 0}$.

These intervals (call them τ-intervals to distinguish them) are not larger than the 1-D cuts of $\theta(\mathbf{X}^*)$, but their validity is completely independent of changes in other costs. This is attractive in multi-robot systems because communication is not needed for cost changes inside the associated τ-intervals. On the other hand, even when a robot's cost has violated its τ-interval, other costs may have changed to counter-balance the effect so that the optimality of the present assignment is retained, i.e, remaining inside $\theta(\mathbf{X}^*)$. The algorithm shown in Alg. 4 checks for a violation incrementally, starting from the robot itself and then, if necessitated by the results of the earlier checks, by growing to incorporate information from other robots.

The set of boundaries $\theta(\mathbf{X}^*)$ and τ-intervals of the initial assignment are computed and distributed to robots (lines 2–5). Then the following procedure runs on each

robot R_i concurrently. If c_{ij} violates its τ-interval, the costs are collected in a set C_{v_i} (lines 6–11). Robot R_i checks $c_{ij} \in C_{v_i}$ altogether whether they satisfy $\theta(\mathbf{X}^*)$. The checking returns $V_i \in \{0, 1\}$ where $V_i = 0$ means that the cost changes turn out not to violate $\theta(\mathbf{X}^*)$ and otherwise $V_i = 1$. This is done by substituting cost variables in $\theta(\mathbf{X}^*)$ for the initial and changed costs (line 13). If $V_i = 0$, the algorithm terminates. Otherwise, R_i finds another robot, initiates communication to it, first requesting and then receiving changed cost set C_{v_a}. We assume there is at least one robot in communication range. If there is no such robot, R_i can increase transmit power, navigate, or wait until a robot is found. If any R_i ultimately returns $V_i = 1$, global communication is necessary; if none of the R_i return $V_i = 1$, their cost changes do not alter the current assignment, and this fact has been determined without requiring global communication.

Algorithm 4 INCREMENTALCOMM

Input: The current $n \times n$ cost matrix $\mathbf{C}$, $\underline{\mathbf{C}}$, and $\bar{\mathbf{C}}$
Output: Indicator variables $\{V_1, \cdots, V_n\}$

1 $l = 1, V_1, \cdots, V_n = 0, C_{v_i} = \varnothing$
2 $\mathbf{X}^* = \text{HUNGARIAN}(\mathbf{C})$
3 $\theta(\mathbf{X}^*) = \text{SA}(\mathbf{X}^*, \mathbf{C})$
4 $\mathcal{T} = \text{TA}(\theta(\mathbf{X}^*), \mathbf{C})$ // compute τ-intervals: // $\mathcal{T}_{ij} = [c_{ij} - \tau_{ij}, c_{ij} + \tau_{ij}]$
5 Distribute $\theta(\mathbf{X}^*)$ and $\mathcal{T}_{ij}$ to corresponding R_i
// Below runs on each robot R_i concurrently //
6 **for** $j = 1$ to n // i is fixed to each robot's index
7 **if** $\underline{c}_{ij} < c_{ij} - \tau_{ij}$ and $c_{ij} + \tau_{ij} > \bar{c}_{ij}$ // if $\mathcal{T}_{ij}$ is violated
8 $V_i = 1$ // there is at least one violation in R_i's cost
9 $C_{v_i} = C_{v_i} \cup c_{ij}$ // collect violated costs
10 **end if**
11 **end for**
12 **while** $|C_{v_i}| \leq n^2$ // while not all costs are included
13 $V_i = \text{CHECK}(\theta(\mathbf{X}^*), C_{v_i}, \mathbf{C})$ // check C_{v_i} altogether
14 **if** $V_i = 0$
15 **break**
16 **end if**
17 $(R_a, C_{v_a}) = \text{FINDADJACENT}(R_i)$ // R_a is an adjacent robot
18 $C_{v_i} = C_{v_i} \cup C_{v_a}$
19 **end while**
20 **return** V_i // $V_i = 1$ if global comm. needed, otherwise $V_i = 0$

6.6 Complexity analysis

Computing $\theta(\mathbf{X}^*)$ via the exact SA has $O(kn^2)$ time complexity where k is the number of feasible solutions. Each feasible solution has $(n-1)^2$ linear boundaries so there are at most $k(n-1)^2$ boundaries. Alg. 1 computes the boundaries for one feasible solution thus it has $O(n^2)$ time complexity. Alg. 2 is dominated by the SA computation (i.e., determining $\theta(\mathbf{X}^*)$) which is repeated in the while loop (line 18). Iteration continues as new $\theta(\mathbf{X}^*)$s are found. The time complexity is, thus,, thus,, thus,, thus, $O((kn^2)^N)$ where N is the number of possible assignments in $\mathbb{C}$.

The time complexity of the partitioning method depends on the number of row/column exchanges in the $n \times n$ matrix (which is the sum of all N assignments). In the worst case, all rows and columns are exchanged so the complexity is $O(n^2)$. Alg. 3 executes Alg. 2 first, and an $O(n^3)$ LP runs N times, giving $O((kn^2)^N + Nn^3) = O((kn^2)^N)$. If the output of Alg. 2 has already been computed, the complexity of Alg. 3 is $O(Nn^3)$. Alg. 4 includes SA so is dominated by it, but the remainder of the procedure, which runs on each robot has $O(n)$ time complexity.

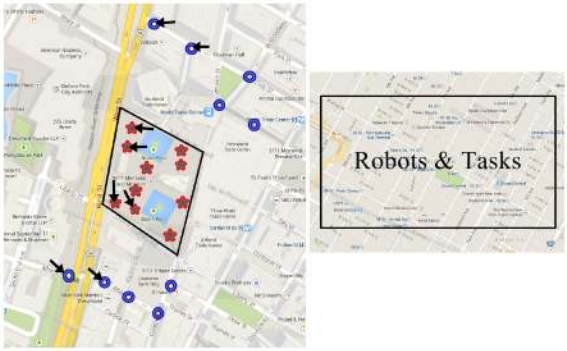
Except for Alg. 1, all the algorithms do not have polynomial time complexity because they employ SA. However, RANDSA (Alg. 1) runs fast while producing high-quality solutions (which will be shown in Sec. 7.1) so it serves as a useful replacement for SA. Even with SA, the algorithms work well for small instances since the costs are incurred prior to task execution. So long as the robots have a few minutes in the "huddle" they are able to complete the computation.

7 Experiments

We consider two scenarios where cost—traveling time—would change depending on the situation. Both employ the same assumptions as the example in Sec. 5.4. The first one is a rescue scenario shown in Fig. 7(a). In a city center, 10 victims (red stars) are inside a disaster site (black polygon) and 10 robots (blue circles) are outside. The robots navigate into the site while pushing debris. The robots move with 1 m/s speed and meet debris every 10 m. The time to push one object is $t_w \in [0, 1]$. The second scenario is a multi-robot navigation problem in an urban area shown in Fig. 7(b), where 30 Robots and 30 tasks are uniformly distributed in a bounded area. The robots move at 10 m/s and encounter a traffic signal every 300 m. The waiting time for the signal is $t_w \in [0, 30]$. Distances from the robots to the tasks are collected using the Google API [12]. The raw data are in meters but converted to time (sec) by considering the moving speeds of the robots. The system is with Intel Core i5, 8G RAM and MATLAB.

Table 1: The running time of Alg. 1 (10 repetitions).

n	5	10	15	20	25	30	35	40
Time	0.0390	0.1710	0.9940	2.614	11.47	42.70	62.03	136.1
Std. Dev.	0.0208	0.0628	1.092	1.431	9.795	29.88	36.63	91.39



(a) A rescue scenario. Stars are victims and circles are robots.

(b) A navigation scenario. Randomly distributed robots and tasks inside the box.

Figure 7: Experimental setup for the multi-robot navigation problem. The marked robots and tasks in (a) are specially chosen for Sec. 7.3.

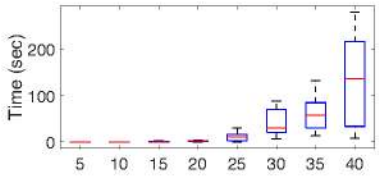


Figure 8: The box plot of the running time of Alg. 1 (10 repetitions).

7.1 Computing $\theta(\mathbf{X}^*)$

The running times of the exact method SA are 0.0180, 0.0410, 0.4370, 53.19 sec for $n = 3, 4, 5, 6$, respectively (standard deviations are 0.0063, 0.0185, 0.0607, 4.5368, and 10 repetitions). The exact method would not be appropriate if the problem size is large but can be used for small problems if few minutes of upfront computation time are allowed before robots are dispatched. With large sizes, Alg. 1 can help decrease running-time. Practically, the algorithm produces near-optimal solutions quickly. Fig. 8 and Table. 1 show the running time of Alg. 1 with 10 repetitions. The standard deviation increases as the problem size grows because there are more cases that the set of variables with randomly chosen ones cannot pass the rank test, which incur repetitions of the random sampling.

As discussed in Sec. 6.1, Alg. 1 computes a partial set $\theta'(\mathbf{X}^*)$ of the complete set $\theta(\mathbf{X}^*)$ (by finding $H' \subset H$). The solution quality can be measured by the ratio of the partial set to the complete set. However, the running time for computing complete sets is prohibitive for large instances, so we use a Monte Carlo method to check solution quality. An $n \times n$ cost matrix $\mathbf{C}$ is sampled from a uniform distribution $\mathcal{U}(10, 20)$ and the optimal assignment $\mathbf{X}^*$ for $\mathbf{C}$ is computed by HUNGARIAN. Then we sample an $n \times n$ perturbation matrix from $\mathcal{U}(-10, 100)$. It is added to $\mathbf{C}$ so we get a perturbed cost matrix (which represents changed costs within $\mathbb{C}$). Let $\mathbf{C}_\mathrm{p}$ and $\mathbf{X}^*_\mathrm{p}$ be a perturbed cost matrix and its optimal assignment (also computed by HUNGARIAN), respectively. If $\theta'(\mathbf{X}^*)$ does not contain $\mathbf{C}_\mathrm{p}$ (i.e., the changed costs violate the set of boundaries) but $\mathbf{X}^*_\mathrm{p}$ is the same with $\mathbf{X}^*$, it means that $\mathbf{C}_\mathrm{p}$ is in between $\theta(\mathbf{X}^*)$ and $\theta'(\mathbf{X}^*)$, which is the region of the complete set that the partial set does not cover. This case represents a false positive. We repeat the random perturbation 10,000 times and count false positives. It is shown that the solution qualities exceed 0.99 in average (10 repetitions).

It is worth noting that the high-quality could be exaggerated since the area which the random perturbed matrices occupy would not sufficiently cover all of the underestimated part of the exact area covered by $\theta(\mathbf{X}^*)$. For example, the perturbed costs would stay around their original cost. They likely result in the solution qualities close to optimal since the perturbed costs only cover the part where true positives dominate. But practically, our choice of the distribution of perturbations ($\mathcal{U}(-10, 100)$) is reasonable in the sense that it produces relatively large perturbations (cost changes) to the original costs which are uniformly distributed over $[10, 20]$.

Even though Alg. 1 shows a good performance, we use the exact method in the following experiments to find a theoretically complete region since the topic of this paper is not about improving running time *per se*.

7.2 Reducing futile effort

If the robots can establish global connection, $\theta(\mathbf{X}^*)$ can be used in a very useful way. Robots report cost changes to the central unit which then checks if the updated costs violate $\theta(\mathbf{X}^*)$. If they do not, the current optimal assign-

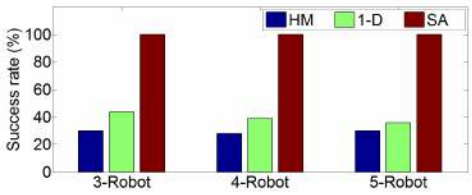
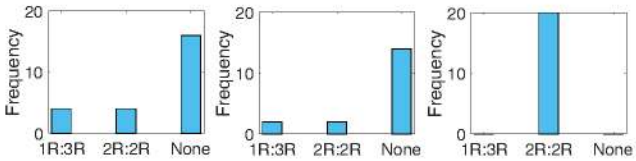


Figure 9: Comparisons of different approaches with respect to cost changes.



(a) The rescue scenario. (b) The navigation scenario. (c) Tasks with spacial locality.

Figure 10: The result of partitioning. Frequency means the number of sub-teams found in 20 trials.

ment is preserved, and the team keeps working as before (no other computation is needed, no other robots need be notified of the cost change).

We compare systems with HUNGARIAN, 1-D intervals, and SA to see how efficiently they deal with cost changes. Suppose that there are multiple consecutive updates to costs. A system using HUNGARIAN must execute the algorithm at every update to ascertain whether the updated costs alter the current assignment. Some re-computations find new assignments, but the others would return the same assignment as before. The methods using a 1-D interval for each cost (e.g., the IHM discussed in Sec. 2) save some re-computation, attempting a new assignment when any of the intervals are violated. Nevertheless some re-computation would still be in vain because the method fails to consider simultaneous cost changes. Lastly, a system with SA does not recompute an assignment unless changed costs actually alter the current assignment. It only executes the *valid* re-computations. We measure the number of valid re-computations with the same cost changes. We compare HUNGARIAN and the 1-D cuts of $\theta(\mathbf{X}^*)$, which are equivalent to IHM, and $\theta(\mathbf{X}^*)$.

Given an arbitrary $n \times n$ cost matrix (for $n = 3, 4, 5$) uniformly sampled from $[0, 1]$, an optimal assignment was computed. Then 50 random perturbation matrices, uniformly sampled between $[0, 2]$, are added to the cost matrix. The result is shown in Fig. 9 and Table 2. The success rate is computed by $(\#$ of valid re-computations$/\#$ of re-computations$) \times 100$. The result clearly shows that SA reduces unnecessary computations and communication. We note that the amount of upfront computation is not compared here. Although SA has the smallest amount of computation at run-time, it may need the largest computational resources at planning time because SA is the most expensive in terms of computation. HUNGARIAN needs no upfront computation as it does not plan ahead. 1-D is cheaper than SA as there is a variant [17] runs in polynomial time. However, communication and computational resources at run-time are much expensive than those at planning time, so the benefit of using SA is still obvious.

7.3 Partitioning a team of robots

For each scenario, we randomly choose four robots and four tasks from the data collected from Google API. For each chosen problem instance, we check whether the team has partitions and find sub-teams if it has (proposed in Sec. 5.1). The result is shown in Fig. 10 and Table 3. In Table 3, we show the results from 20 runs in each scenario. The frequency indicates the number of runs that have the particular sub-teams as labeled in the leftmost column. A label such as 2R:2R, for example, should be interpreted as there being two sub-teams, each of two robots. Even though the scenarios do not have obvious spatial sparsity and/or locality, the algorithm is able to detect sub-teams when a team has such an underlying structure. The average running times of two scenarios are 1.337 sec and 1.905 sec ($\sigma = 0.0951$, $\sigma = 0.1448$ for 20 repetitions), respectively. We also report results of partitioning when tasks do have strong spatial locality (Fig. 10c). Here, the spatial locality means that two robots and two tasks are located as the marked robots and tasks in Fig. 7(a).

7.4 Persisting with an initial assignment

We test Alg. 3 that computes the maximum cost loss when robots persist with the initial assignment and make any further effort for cost changes. Table 4 shows examples of maximum cost losses for $n = 2, 3, 4$. The first column (PERSIST) shows the largest possible cost sum if the robots persist with the initial assignment. The second column (CHANGE) shows the smallest possible cost sum if they are committed to respond to any cost changes with communication and re-computation. The difference between two columns gives the third column (MAX LOSS) which is the maximum loss of costs if robots persist with the initial assignment without paying no effort on run-time cost changes. One (the central unit or an operator) can decide whether to execute the initial assignment without having any communication and computation using the

Table 2: Comparisons of different approaches with respect to cost changes. (HUNGARIAN, 1-D intervals, SA.)

	$n = 3$			$n = 4$			$n = 5$		
Method	Attempts	Success	Rate	Attempts	Success	Rate	Attempts	Success	Rate
HUNGARIAN	50	15	30.00%	50	14	28.00%	50	15	30.00%
1-D	36	15	43.59%	36	14	38.88%	42	15	35.71%
SA	15	15	100%	14	14	100%	15	15	100%

Table 3: The result of partitioning. Frequency means the number of sub-teams found in 20 trials.

Sub-team size	Frequency		
	Rescue	Navigation	Locality
1R's only	0	0	0
1R:3R	4	2	0
1R:1R:2R	0	0	0
2R:2R	4	2	20
No sub-team	12	16	0

Table 4: The maximum loss of persisting with assignment. Some examples of execution results are shown (navigation scenario). The three columns shows cost sums (sec).

Size	PERSIST	CHANGE	**MAX LOSS**
3R	970.2	424.7	545.5
4R	1385	730.2	655.4
5R	716.8	402.3	314.5

cost loss information. If the communication/computation expenses are prohibitive, it would be beneficial to persist with the initial assignment. For example, if the total execution time of tasks including the time for additional communication and computation for dealing with cost changes takes longer than the suboptimal execution time of the initial assignment, it is more beneficial for the robots not to use their time for updating the optimal assignment. The the average running times of the algorithm (including Alg. 2 with the exact SA) are 0.6140, 9.830, and 262.6 sec (20 repetitions) for $n = 3, 4, 5$ (standard deviations are 0.2941, 2.811, and 97.75, respectively). The large standard deviations result from the varied N which is the number of possible assignments in the cost region. Since we choose the initial cost randomly, some repetitions could have many possible assignments whereas others do not. The range of N is $[1, n!]$ so the variance becomes larger as n increases.

7.5 Incremental communication

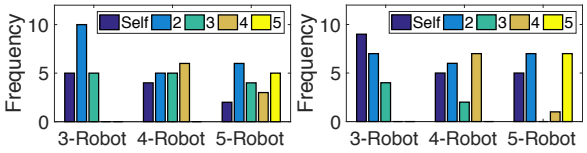
Finally, we show how few communication messages are actually needed to detect whether optimality has been violated by cost changes. For each scenario, we compute the τ-intervals and distribute them to the robots. Each robot independently performs its task unless its costs violate the τ-intervals. Once a violation is detected, the robot runs the individual procedure in Alg. 4. For each changed set of costs, each robot needs to check the violation by self-assessment (i.e., checking its own costs for all tasks) or with nearby robots. We record how many robots are involved with this process for each robot and how many times such the process occurs until tasks are completed. A team may have several local communications, but one robot may require global communication. To understand such cases, we record the size of the largest neighborhood needed for communication among the robots. For example, the previous case needs global communication even though only one robot needs it while other robots do local checks. Note that we ensure every robot has at least one violation so all robots execute Alg. 4. We randomly choose robots and tasks from the data sets. Our experiment varies the team size from three to five. Fig. 11 and Table 5 show the results (for 20 repetitions). In the following, we give an example interpretation of the result in the navigation scenario (the leftmost bars in Fig. 11b) with three robots. Among all 20 trials, the self-assessment is enough until the end of the mission in nine trials, 2-robot communication is the maximum range in seven trials, and global communication is needed in the rest four trials. As the result shows, local communication suffices in many trials (bold numbers in Table 5). The reported running time includes the computation time for the τ-intervals and Alg. 2 with SA.

7.6 Interrelated costs

Since we propose the region-based cost that can represent interrelationships between costs, we examine the benefit of using the richer model compared to a simple bounded cost (like Fig. 2a). We assume that costs have linear relationships and, like Fig. 3(a), the robots share one re-

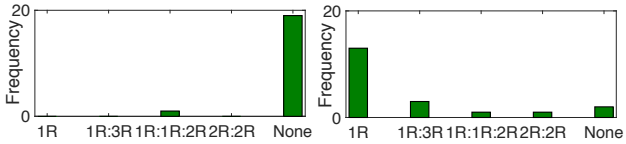
Table 5: Frequency of communication ranges. The bold numbers indicate the frequencies of local communications. For example, in the rescue scenario with three robots (3R), six trials out of 20 only require self-assessment (no inter-robot communication) until tasks are completed.

	Rescue							Navigation						
Range / Team size	Self	2	3	4	5	Time (sec)		Self	2	3	4	5	Time (sec)	
						Mean	Var						Mean	Var
3R	**5**	**10**	5	N/A	N/A	0.0475	0.0023	**9**	**7**	4	N/A	N/A	0.0495	0.0018
4R	**4**	**5**	**5**	6	N/A	0.5710	0.0352	**5**	**6**	**2**	7	N/A	0.4685	0.0273
5R	**2**	**6**	**4**	**3**	5	7.468	11.07	**5**	**7**	**0**	**1**	7	6.635	14.56



(a) The rescue scenario. (b) The navigation scenario.

Figure 11: Frequency of communication ranges. For each team size, the left most bar means self-assessment whereas the right most bar mean global communication. Local communication is more frequent with Alg. 4.



(a) No interrelationship modeled. (b) Interrelationships modeled.

Figure 12: The results of partitioning a team of robots without and with interrelationship modeling (20 trials). Modeling interrelationships reduces false positives in computing all possible assignments within a cost region. Thus, more sub-teams can be found with the smaller number of assignments.

source to perform tasks. Thus, any vicissitude affecting the resource changes all costs by the same amount. For example, if there is a delay on a common route to reach tasks, all traveling times increase accordingly by the same amount. We randomly choose an $n \times n$ cost matrix and a scalar value of delay from $\mathcal{U}(0, 10)$. We model the relationships of costs through a set of linear equations (e.g., $c_{ij} - c_{pq} = t_{ij} - t_{pq}$ for $ij \neq pq$ where t is a nominal cost without delay). We first run Alg. 2 to compute all optimal assignments under the cost boundary modeling interrelationships of costs. We also run the algorithm with a simple bounded region for the same instance, which does not model the interrelationship arising from the shared resource.

By modeling costs more accurately considering interrelationships, the resulting cost region could be smaller than the simple bounded cost. This smaller cost region produces a smaller number of output assignments from Alg. 2. Thus, the richer model prunes away some scenarios (possible assignment alterations) that the robots do not have to prepare for. The results (Fig. 12 and Table 6) show that the number of output assignments is greatly reduced when interrelationships are modeled (averagely from 9.95 to 1.85 for $n = 4$). From this analysis, we are able to find more sub-teams. For example, with the simple bounded cost, there is only one trial where sub-teams are found. In the rest 19 trials, no sub-team is found. On the other hand, the richer cost model enables the partitioning method to find more sub-teams (18 out of 20 trials). In addition, the reduced N by having a smaller cost region decreases the running time of the algorithms that iterate their procedure N times.

8 Conclusion

In this paper, we propose a cost representation that incorporates uncertainty in costs and is also capable to expressing some interrelationships between costs. The representation assumes that costs are bounded by a finite region. We employ a sensitivity analysis approach for multi-robot task allocation and compare it with other methods, showing that is advantageous when costs change. We also proposed three methods that reduce centralization of multi-robot systems alongside the basic routine for computing $\theta(X^*)$ and a fast approximate version, which is a randomized algorithm. We examined our algorithms with two realistic scenarios and data, not merely randomly generated matrices. We also show that modeling interrelationships yields tighter cost regions and hence better predictions for how the optimality of the task allocation under interrelated and uncertain costs.

Table 6: The results when cost interrelationships are modeled (20 repetitions).

(a) The number of assignments from given cost regions.

Interrelationship	Not modeled	Modeled
Mean	9.9500	1.8500
Std. dev.	4.2855	1.0984

(b) Frequency of sub-teams found when $n = 4$.

Partition type	Interrelationship	
	Not modeled	Modeled
1R's only	0	13
1R:3R	0	3
1R:1R:2R	1	1
2R:2R	0	1
No sub-team	19	2

Finally, it is worth noting that the algorithms and results that are reported work even if an overestimation of the region of feasible costs is given. More generally, if robots are permitted to generate global synchronization events, then even an underestimated version of the region could be useful. For example, when some cost variation is found to violate the presumed model, the robot could trigger synchronization with a more pessimistic region. Doing so is analogous to the way most MRTA approaches operate today: they simply replan when things change. With region-based models one can hope to find regions which force replanning less frequently.

Our future work includes the problems occurring if a modeled cost region is no longer valid owing to some unforeseen circumstances. As a result, all the upfront computations done with the outdated cost region might be ineffective. In this case, the cost region should be updated with new information collected from observations, and the upfront computation needs to follow for the updated region. We are interested in developing methods that find and use some reusable pieces from the previous results if such an outdated cost region is detected. Unless there is a holistic change in the environment which is rare, only few of costs need remodeling at one point (of course there may be a series of such cost remodelings as the robots observe more dynamic changes). We will investigate how we can prevent a complete restart of the upfront computation if partial changes in the cost region occur.

A An example transformation from NOLC to LONC

An example of LONC is $\max x_1 + x_3 - x_4$ subject to $x_1^2 + x_3^4 \geq 1$ and $x_2^2 + x_4^2 \geq 1$. This can be transformed to $\max \sqrt{y_1} + \sqrt[4]{y_3} - \sqrt{y_4}$ subject to $y_1 + y_3 \geq 1$ and $y_2 + y_4 \geq 1$ by substituting $y_i = x_i^2$ for $i \in \{1, \cdots, 4\}$. Now the transformed objective function is convex and the constraints are linear, which is NOLC.

References

[1] Srinivas Akella. Assignment algorithms for variable robot formations. In *Proceedings of International Workshop on the Algorithmic Foundations of Robotics*, 2016.

[2] Charles Audet, Pierre Hansen, Brigitte Jaumard, and Gilles Savard. A branch and cut algorithm for nonconvex quadratically constrained quadratic programming. *Mathematical Programming*, 87:131–152, 2000.

[3] Aharon Ben-Tal and Arkadi Nemirovski. *Lectures on modern convex optimization: analysis, algorithms, and engineering applications*. Siam, 2001.

[4] Stephen Bradley, Arnoldo Hax, and Thomas Magnanti. Applied mathematical programming. 1977.

[5] Mathias Bürger, Giuseppe Notarstefano, Francesco Bullo, and Frank Allgöwer. A distributed simplex algorithm for degenerate linear programs and multi-agent assignments. *Automatica*, 48(9):2298–2304, 2012.

[6] Smriti Chopra, Giuseppe Notarstefano, Matthew Rice, and Magnus Egerstedt. A distributed version of the hungarian method for multirobot assignment. *IEEE Transactions on Robotics*, 33(4):932–947, 2017.

[7] Bernardine Dias, Robert Zlot, Nidhi Kalra, and Anthony Stentz. Market-based multirobot coordination: A survey and analysis. *Proceedings of the IEEE*, 94:1257–1270, 2006.

[8] Nakju Lett Doh, Changjoo Nam, Suk-Kyu Lee, and Hwang-Nam Kim. Particle filter-assisted ad hoc routing in a multi-hop wireless ad hoc network for multi-robots. *Journal of Institute of Korean Electrical and Electronics Engineers*, 14(4):312–316, 2010.

[9] Carlo Filippi. A fresh view on the tolerance approach to sensitivity analysis in linear programming. *European Journal of Operational Research*, 167:1–19, 2005.

[10] Tomas Gal. *Postoptimal analyses parametric programming and related topics*. McGraw-Hill, 1979.

[11] Tomas Gal, Hermann-Josef Kruse, and Peter Zörnig. *Survey of solved and open problems in the degeneracy phenomenon*. Springer, 1988.

[12] Google. The Google Directions API. `https://developers.google.com/maps/documentation/directions/`, 2013.

[13] Harvey Greenberg. An analysis of degeneracy. *Naval Research Logistics Quarterly*, 33:635–655, 1986.

[14] Harold Kuhn. The hungarian method for the assignment problem. *Naval Research Logistic Quarterly*, 2(1-2):83–97, 1955.

[15] Sven Leyffer and Ashutosh Mahajan. Nonlinear constrained optimization: methods and software. *Argonne National Laboratory, Illinois*, 2010.

[16] Chi-Jen Lin and Ue-Pyng Wen. Sensitivity analysis of objective function coefficients of the assignment problem. *Asia-Pacific Journal of Operational Research*, 24:203–221, 2007.

[17] Lantao Liu and Dylan Shell. Assessing optimal assignment under uncertainty: An interval-based algorithm. *International Journal of Robotics Research*, 30(7):936–953, 2011.

[18] Lantao Liu and Dylan Shell. Large-scale multi-robot task allocation via dynamic partitioning and distribution. *Autonomous Robots*, 33:291–307, 2012.

[19] Ayorkor Mills-Tettey, Anthony Stentz, and Bernardine Dias. The dynamic hungarian algorithm for the assignment problem with changing costs. Technical Report CMU-RI-TR-07-27, Carnegie Mellon University, 2007.

[20] Changjoo Nam and Dylan Shell. When to do your own thing: Analysis of cost uncertainties in multi-robot task allocation at run-time. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 1249–1254, 2015.

[21] Changjoo Nam and Dylan A Shell. Analyzing the sensitivity of the optimal assignment in probabilistic multi-robot task allocation. *IEEE Robotics and Automation Letters*, 2(1):193–200, 2017.

[22] Michael Otte and Nikolaus Correll. The any-com approach to multi-robot coordination. In *IEEE International Conference on Robotics and Automation: Network Science and Systems Issues in Multi-Robot Autonomy*, 2010.

[23] James Parker, Alessandro Farinelli, and Maria Gini. Decentralized allocation of tasks with costs changing over time. *IJCAI Workshop on Synergies between Multiagent Systems, Machine Learning and Complex Systems*, pages 62–73, 2015.

[24] James Parker and Maria Gini. Tasks with cost growing over time and agent reallocation delays. In *Proceedings of the 2014 international conference on Autonomous agents and multi-agent systems*, pages 381–388. International Foundation for Autonomous Agents and Multiagent Systems, 2014.

[25] James Parker and Maria L Gini. Controlling growing tasks with heterogeneous agents. In *Proceedings of International Joint Conference on Artificial Intelligence*, pages 461–467, 2016.

[26] James Parker, Ernesto Nunes, Julio Godoy, and Maria Gini. Exploiting spatial locality and heterogeneity of agents for search and rescue teamwork. *Journal of Field Robotics*, 2015.

[27] Lorenzo Sabattini, Nikhil Chopra, and Cristian Secchi. Decentralized connectivity maintenance for cooperative control of mobile robotic systems. *The International Journal of Robotics Research*, 32(12):1411–1423, 2013.

[28] Wei-Min Shen and Behnam Salemi. Distributed and dynamic task reallocation in robot organizations. In *Proceedings of IEEE International Conference on Robotics and Automation*, volume 1, pages 1019–1024, 2002.

[29] Peter Stone and Manuela Veloso. The cmunited-97 simulator team. In *RoboCup-97: Robot Soccer World Cup I*, pages 389–397. Springer, 1998.

[30] James Ward and Richard Wendell. Approaches to sensitivity analysis in linear programming. *Annals of Operations Research*, 27:3–38, 1990.

[31] Ernst Zermelo. Über das navigationsproblem bei ruhender oder veränderlicher windverteilung. *ZAMM-Journal of Applied Mathematics and Mechanics/Zeitschrift für Angewandte Mathematik und Mechanik*, 11(2):114–124, 1931.