Analyzing Uncertainties in Cost: Mitigating Centralization in Multi-Robot Task Allocation[*]

Changjoo Nam and Dylan A. Shell[†]

February 8, 2015

Abstract

We consider the problem of finding the optimal assignment of tasks to a team of robots when the costs associated with the tasks may vary. This arises often because robots typically update their cost estimates and re-compute task assignments to deal with dynamic situations (e.g., the addition of new robots to the team, arrival of new tasks, or the revelation of new information). This paper describes a way to compute a sensitivity analysis that characterizes how costs may alter from current estimates before the optimality of the present assignment is violated. Using this analysis, robots are able to avoid unnecessary re-assignment computations.

By exploiting this analysis, we propose multiple methods that help reduce global communication and centralized computations. First, given a model of how costs may evolve, we develop an algorithm that partitions a team of robots into several independent cliques, which can maintain global optimality by communicating only 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 develop an algorithm that assesses whether cost changes affect the optimality

[*]This paper is an extended version of [1].

[†]Both authors are with the Department of Computer Science and Engineering at Texas A&M University, College Station, Texas, USA. {`cjnam,dshell`} `at cse.tamu.edu`

of the current assignment through a succession of local checks. Experimental results show that our proposed methods reduce the degree of centralization needed by a multi-robot system.

1 Introduction

Multi-robot task allocation (MRTA) addresses optimization of collective performance of a robot team by reasoning about which robots should perform which tasks. In general, each robot estimates the costs of performing each task, then these estimates are shared by robots over a communication network. Most often an optimal assignment is computed by a central computation unit (e.g., most approaches using the Hungarian method [2], an auctioneer [3], or a linear programming framework). But while robots are executing their assigned task, the assumptions under which costs were calculated may turn out to be invalid: the environment may change, robots may fail, or a variety of other unexpected situations may emerge. One solution which ensures a fluid response to these contingencies, is to periodically re-calculate costs and re-compute the optimal task assignments. This solution incurs computational and communication expense proportional to the desired recency.

We are interested in the MRTA problem where an instantaneous scalar estimate of a cost may be inappropriate or invalid. This arises naturally when there is uncertainty in some state used in computing the costs, or when the costs evolve as tasks are performed and the estimates are out of date. In this paper, we model costs for a task as varying within some prescribed range. For example, the autonomous robot in Fig. 1 is able to estimate its shortest and longest driving times (which may be used as cost measures) to a destination by considering information about its route. The lower bound of the time would be merely the time spent on driving (distance over the maximum speed), and adding the maximum waiting time for traffic signals yields the upper bound[1].

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 [4], [5], [6]. Analysis of an optimal assignment must compute a region where costs within that region preserve the current optimal assignment. However, SA has found limited applicabil-

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

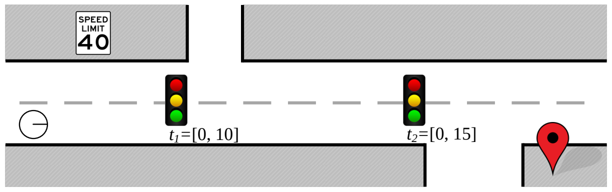


Figure 1: A simple example where task costs which are not precisely known to the robot beforehand. The driving time c to the destination will vary depending on the traffic signals. A lower bound $\underline{c}$ is $\frac{d}{v_{\max}}$ when $t_1 = t_2 = 0$, and an upper bound $\bar{c}$ is $\frac{d}{v_{\max}}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

ity to multi-robot task-allocation problems: the analysis assumes that the decision maker (its counterpart in multi-robot systems is the central computation unit) is able to access all information off-line 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 ideas from SA to develop methods that explore questions pertinent to resource limitations in multi-robot systems. For a given problem instance, these methods reduce global communication and centralized computation, or quantify the optimality trade-offs if communication is avoided. This paper makes the following contributions:

- We develop an algorithm that analyzes the cost structure for a given assignment. It seeks cliques in the team, factorizing the group into sub-teams that are able to work independently, communicating only among themselves, forgoing global communication but without sacrificing global optimality.

- 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 re-assignment is prohibitive.

- 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 a safe (one-dimensional) interval 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 are considered together. We introduce a method that incrementally increases the dimensionality of the bounding region, growing the number of costs considered by communicating with adjacent robots. Global communication may be required in the worst case but oftentimes local computation can reach the conclusion that the assignment is still optimal.

2 Related Work

Some authors have proposed reoptimization schemes for multi-robot systems, allowing updated assignments to be computed efficiently. Mills-Tettey et al. [7] describe a dynamic (or incremental) Hungarian method that repairs initial optimal assignment to obtain a new optimal assignment when costs have changed. Also, Shen and Salemi [8] present a decentralized dynamic task allocation algorithm that uses a heuristic searching method. These algorithms still use computational resources for those cost modifications which end up with the same assignment.

Liu and Shell [9] propose the interval Hungarian method (IHM) to manage uncertainties in costs. Given an optimal assignment, the algorithm computes the maximum interval of each cost in which costs within the interval do not change the current optimal assignment. Thus, robots are able to decide how a cost change affects the optimality of the current solution. However, the algorithm treats the problem of multiple 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 propose a sparsification and partitioning method to distribute the assignment problem to reduce global communication and re-assignment [10]. The method coarsens the utility matrix by using locality and sparsity of tasks. Once the matrix is partitioned into several clusters, each cluster is able to compute an assignment independently. Inspired by that work, we propose a factorization method for problems where mere single time-step sparsity is not enough.

3 Problem Description

3.1 MRTA with Changeable Cost

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 performing the j-th task. 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 i-th robot performs the j-th task. Otherwise, $x_{ij} = 0$. Assuming that for each i and j, c_{ij} has a range $\underline{c}_{ij} \leq c_{ij} \leq \bar{c}_{ij}$ in which a cost can have any value within the range. 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

$$\begin{aligned} \sum_{j=1}^{n} x_{ij} &= 1 && \forall i, && (2)\\ \sum_{i=1}^{n} x_{ij} &= 1 && \forall j, && (3)\\ 0 \leq x_{ij} &\leq 1 && \forall \{i, j\}, && (4)\\ x_{ij} &\in \mathbb{Z}^{+} && \forall \{i, j\}. && (5) \end{aligned}$$

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 $n = m$.) We also make use of matrix representations C and X^* that are $n \times n$ matrices representing a cost matrix and an optimal assignment of the problem, respectively. Similarly, $\underline{C}$ and $\bar{C}$ are matrices of $\underline{c}_{ij}$ and $\bar{c}_{ij}$. We note that C is not necessarily a hyper-cuboid bounded by upper and lower bounds. C also can be convex hyper-polytopes if cost changes are bounded by such a region.

3.2 Background: Applying Sensitivity Analysis to MRTA

Since the early work of Gal [4] several decades ago, Sensitivity Analysis (SA) has been recognized as an important method in linear programming (LP). We present a tiny review based on the comprehensive study of

Ward and Wendell [5] to help introduce SA. We will confine the discussion to changes in the objective function coefficients (i.e., costs).

Let k be an index of a basic index set in which a set consists of basic variables[2] of a feasible solution. For each k, critical region R_k is a set of costs where an optimization problem has the same solution X_k for any cost vector $c \in R_k$. More formally,

$$R_k = \{c \in \mathbb{R}^s : c_{N_k} - c_{J_k} B_k^{-1} A_{N_k} \geq 0\} \tag{6}$$

where $s = n^2$, and J_k and N_k indicate basic and nonbasic variables, respectively. B_k and A_{N_k} are constraint matrices of basic variables and nonbasic variables.[3] c_{J_k} and c_{N_k} are costs of basic and nonbasic variables. Note that the critical region is a polyhedral cone with nonempty interiors if $c \in \mathbb{R}^s$ [5].

The MRTA problem can be posed as an Optimal Assignment Problem (OAP), which can be relaxed to a special case of LP, and the LP formulation of MRTA may make use of SA.[4] However, the OAP is degenerate [11] (see Appendix for details of degeneracy) and so needs a special treatment to use the analysis. We introduce the optimal coefficient set

$$\theta(X^*) = \{c \in \mathbb{R}^s : X^* \text{ is optimal for Eq. 1–5}\} \tag{7}$$

in which $c \in \theta(X^*)$ yields the optimal solution X^*. In nondegenerate cases, $R_1 = \theta(X^*)$ where $k = 1$ means the index set of the best solution among feasible solutions. In degenerate cases, one should compute $\theta(X^*)$ as:

$$\theta(X^*) = \bigcup_{k \in H} R_k, \tag{8}$$

where $H = \{k : X^*_{J_k} = B_k^{-1}, X^*_{N_k} = 0\}$, which means the union of critical regions of all degenerate solutions. Note that $\theta(X^*)$ is also a polyhedral set [5, Theorem 17]. Geometrically, all linear boundaries of Eq. 8 cross the origin because Eq. 8 is a union of linear boundaries in Eq. 6 that pass the origin (all zero RHS values). In addition, coefficients of linear boundaries are -1, 0 or 1 because B_k and A_{N_k} are from a totally unimodular coefficient matrix.

[2]A variable is basic if it corresponds to one of the vectors in the basis, given a feasible basis to a linear-programming problem.

[3]From the constraint matrix of an optimization problem, columns corresponding to basic and nonbasic variables form B_k and A_{N_k}, respectively.

[4]Technically, this relaxation requires $c \in \mathbb{Z}^+$. We skip over details of construction of a GCD for the (inevitable) rational representation within a computer.

An $n \times n$ MRTA problem (with n robots and n tasks) has n^2 variables. Owing to degeneracy, the problem has $2n - 1$ basic variables and $(n-1)^2$ nonbasic variables. From Eq. 6, we find that each R_k has $(n-1)^2$ linear boundaries.

3.3 Problem Statement

With a centralized system, computing the results of the SA for the multi-robot task-assignment problem is straightforward. The central unit computes an optimal assignment with the latest costs and Eq. 8. Robots then report cost changes to the central unit and which then checks if they violate $\theta(X^*)$. If the change does not alter the current optimal assignment, the team keeps working as before (no other computation is needed, no other robots need be notified of the cost change).

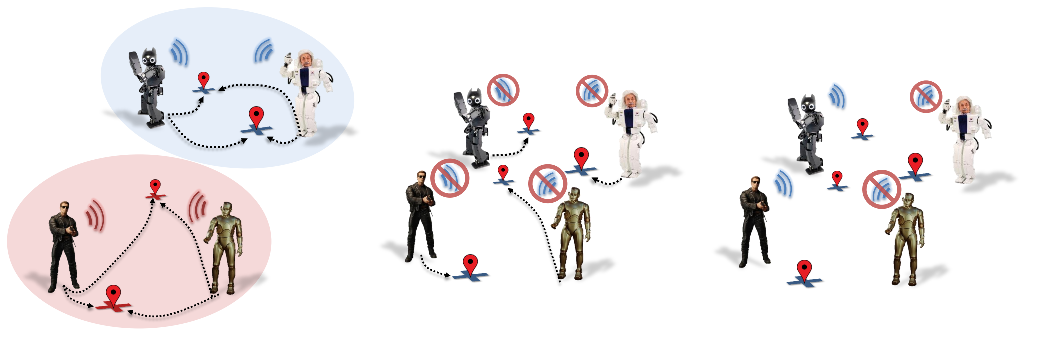
The centralized approach is simple to implement but often undergoes problems arising from the distributed nature of multi-robot systems. Especially, maintaining global connectivity in multi-robot systems is expensive, and the quality of communication changes drastically [12]. We aim to develop methods for distributing the assignment problem to alleviate dependence on the centralized structure: (i) factorizing a team of robots into cliques if such cliques exist (Fig. 2(a)), (ii) computing the cost difference between the worst-case cost sum (if the robots persist their initial assignment) and the best-case cost sum (if they reassign tasks) (Fig. 2(b)), and (iii) communicating locally to decide whether a re-assignment is necessary with cost changes (Fig. 2(c)).

In (i), we find all N possible assignments with a cost matrix C, which is changeable, to factorize a team of robots without locality and sparsity of tasks. The challenge is how to find N assignments. A brute-force method is computing all assignments for all costs in the hyper-cuboid, but it is impossible because there is an infinite number of $c_{ij} \in [\underline{c}_{ij}, \bar{c}_{ij}]$.

Once optimal assignments X_q^* and their $\theta(X_q^*)$ for $q = 0, \cdots, N$ are computed by resolving the challenge in (i), (ii) can be solved by finding $C_{\min_q}$ in each $\theta(X_q^*)$ and compute

$$\max(\bar{C}X_0^* - C_{\min_q}X_q^*) \tag{9}$$

for $q = 0, \cdots, N$ where X_0^* indicates the initial optimal assignment. In other words, Eq. 9 is the cost difference between the minimum among cost sums, where robots change their assignments for the recent cost updates, and the maximum cost sum if robots maintain their initial assignment.



(a) Factorizing a team into smaller cliques.

(b) Persisting an initial assignment.

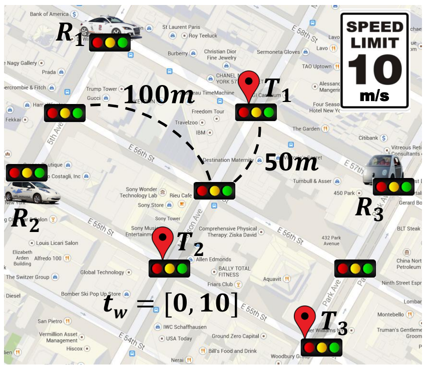
(c) Local communication to check optimality violations of costs.

Figure 2: The methods to mitigate centralization in MRTA. (a) Cliques could be found by analyzing $\theta(X_q^*)$ for $q = 0, \cdots, N$, where N is the number of all possible assignments with given C. (b) The maximum cost loss is computed for which robots do not have communication and persist an initial assignment even with cost changes. (c) Robots have local communication to check whether their changed costs violate the current $\theta(X^*)$.

If robots have one-dimensional intervals of their costs in which any cost change within its interval does not alter the current assignment regardless of other cost changes, the robots can work independently until any of their own intervals is violated. In (iii), such intervals should be computed and distributed to robots. If a robot finds one of its intervals is violated, the robot checks itself whether $\theta(X_q^*)$ is violated by looking at its all other costs. If other cost changes countervail the violation, the current assignment is preserved. Otherwise, the robot communicates with an adjacent robot to consider more cost changes. Global communication may be required in the worst case, but local communication is often enough.

3.4 An Example

We show a simple scenario and how the proposed methods can be used concurrently. We consider a multi-robot navigation problem, which is a multi-robot version of the example shown in Fig. 1. Suppose we have three autonomous robots ($R_{1,2,3}$) and three destinations ($T_{1,2,3}$) as shown in Fig. 3. Time is the measure of cost. The goal is to have one robot at each destination while minimizing the total sum of traveling times. We assume that the robots drive through the shortest path, and each intersection has a traffic



(a) A multi-robot navigation example.

	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) The cost matrix corresponding to (a).

Figure 3: An example of an MRTA problem with changeable costs. We have three robots ($R_{1,2,3}$) and three destinations ($T_{1,2,3}$). The goal is to have one robot at each destination while minimizing the total sum of traveling time. Since the costs could vary within the ranges in (b), there are multiple assignments possible. The proposed methods can be used concurrently to analyze the assignments and to have less centralized operations.

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 we do not model delays from congestion and acceleration/deceleration. The corresponding cost matrix is shown in Fig. 3(b).

The initial optimal assignment is $X_0^* = \left(\begin{smallmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{smallmatrix}\right)$. If global communication and computation are reliable and not prohibitively expensive, the robots may use SA directly. If not, the central unit computes the maximum cost difference (ii) to decide whether the robots respond to changes. The worst cost sum when the robots keep the initial assignment is 100, and the best cost sum when they consider cost changes is 50 (i.e., when $X_1^* = \left(\begin{smallmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{smallmatrix}\right)$. If the central unit decides that the difference (loss) 50 is too large, it tries to find cliques (i) but there is no such clique in this example. Therefore, the robots finally cope with cost changes in an incremental fashion (iii) from local communication.

4 Algorithms

In this section, we briefly describe our implementations of computing Eq. 8. Then we propose three methods for the problems stated in the previous section.

4.1 Computing $\theta(X^*)$

4.1.1 An exact method

A feasible solution must include n optimal variables (correspond to optimal assignments). Among the remaining $n^2 - n$ variables, $n - 1$ variables need to be chosen to complete a feasible solution which consists of $2n - 1$ basic variables. Our implementation enumerates all $|k| = \binom{n^2-n}{n-1}$ feasible solutions to compute Eq. 8. The running time grows factorially as the input size increases. However, the interiors of R_k may overlap and $\theta(X^*)$ could be covered by a subset of R_k. A method such as [13] can be used to find nonoverlapping subsets of $\theta(X^*)$ which requires less effort than enumerating all feasible solution sets.

4.1.2 An anytime algorithm

We develop an anytime algorithm to facilitate a faster computation of $\theta(X^*)$. A partial enumeration of degenerate solutions bring an incomplete $\theta(X^*)$, but an incomplete set often takes a large portion of the complete $\theta(X^*)$. From this observation, we implement an anytime algorithm that enumerates degenerate solutions and computes corresponding critical regions (Eq. 6) as much as possible with given time.

4.2 Factorizing a Team of Robots

Factorization can be done by analyzing all possible assignments X_q^* for $q = 0, \cdots, N$ computed by Alg. 1. An assignment problem has at least 4-dimension (two robots and two tasks) so it is hard to visualize geometry of the problem. Thus, we describe a figurative 2-D representation in Fig. 4(a). One difference of the 2-D representation from higher dimensional cases is that all linear equations in $\theta(X^*)$ are greater or equal to zero (see Eq. 6), but the upper boundary of $\theta(X^*)$ in Fig. 4(a) has the opposite inequality.

We have an initial optimal assignment X_0^* and its $\theta(X_0^*)$ (Alg. 1, line 2-3). l is an arbitrary linear boundary in $\theta(X^*)$. If the objective value is greater or equal to zero when l is maximized over the shaded area[5] (line 6), l contains the entire cost set (the shaded area C in Fig. 4a). Otherwise, the shaded

[5]All l should be maximized because of the inequalities of them (see Eq. 6).

area is not covered by l (see Fig. 4b) thus a cost on l is perturbed to find a new $\theta(X^*)$ that includes the rest of C (line 12-22). Once the current $\theta(X^*)$ is expanded by perturbing points (i.e., costs), newly found $\theta(X_q^*)$ are merged and checked also. The algorithm terminates if $\theta(X^*)$ completely includes C. It returns all X_q^* and $\theta(X_q^*)$ found.

In Fig. 4(a), the direction of a perturbation is toward $\mathbf{c}'$. The magnitude ϵ of the perturbation should be carefully chosen because a large ϵ may skip some $\theta(X_q^*)$ on the way. We describe how we determine the optimal magnitude and the direction of a perturbation.

Theorem 4.1. Let l be an arbitrary linear boundary in $\theta(X^*)$. The magnitude of a perturbation ϵ to perturb an arbitrary point $\mathbf{p}$ on l of $\frac{|\mathbf{p}|}{n}$ will not miss any $\theta(X^*)$.

Proof. Let $\mathbf{l}$ be the normal of an arbitrary linear boundary l in $\theta(X^*)$. From line 6 in Alg. 1, we have $\mathbf{c}'$, which is an extreme point of C outside of the current $\theta(X^*)$. The projection of $\mathbf{c}'$ onto $\mathbf{l}$ is

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

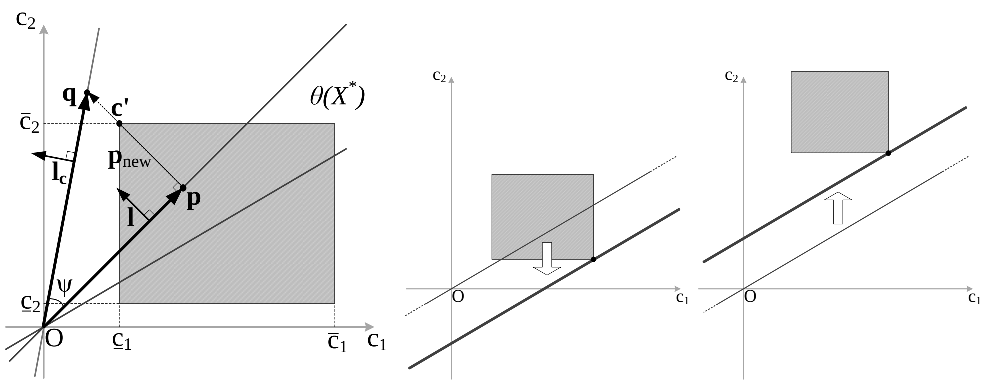
Suppose that $\mathbf{l}_c$ is the normal vector of the nearest boundary to l (other than itself). Let $\mathbf{q}$ be a vector orthogonal to $\mathbf{l}_c$. The direction of movement from $\mathbf{p}$ to $\mathbf{q}$ is along a vector

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

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

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

As we discussed in Section 3.2, coefficients (normals) of linear boundaries in $\theta(X^*)$ are $-1, 0$, or 1. To make a dot product of two normals of boundaries maximum, the boundaries should have the maximum number of 1's (or the maximum number of -1's). $\hat{\mathbf{1}}$ is the case but two boundaries should be distinct. Therefore, the product of $\frac{1}{\sqrt{n^2}}[1\,1\,1 \cdots 1\,1]$ and $\frac{1}{\sqrt{n^2-1}}[1\,1\,1 \cdots 1\,0]$ 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.



(a) A 2-D figurative picture. (b) Objective values of two cases.

Figure 4: A 2-D figurative representation of cost space. (a) Bold lines represent linear boundaries (hyperplanes) of $\theta(X^*)$ and the shaded area represents a cost matrix C bounded by $\underline{C}$ and $\bar{C}$. (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).

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

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

which is the distance to the closest boundary along $\mathbf{p}_{new}$. We set a safe magnitude of perturbation $\epsilon = \frac{|\mathbf{p}|}{n}$, which does not skip any $\theta(X^*)$. $\square$

Cliques can be found easily by summing the assignments found: $X_C = \sum_{q=0}^{N} X_q^*$. If there are elements with zero in X_C, the robot-task pairs are never assigned. If a block diagonal matrix can be found, the main diagonal blocks represent cliques. With local communication and computation within each clique only, the robots achieve global optimality.

4.3 Choosing to Persist with the Initial Assignment

The pseudocode is shown in Alg. 2. All assignments X_q^* and $\theta(X_q^*)$ for $q = 0, \cdots, N$ are given by Alg. 1 (line 1). For each $\theta(X_q^*)$, find the minimum costs C_q over $\theta(X_q^*)$ with the bounds $\underline{C}$, and $\bar{C}$ (line 3). For each q, we have an assignment X_q^* and the minimum cost $C_{\min_q}$. In line 6, $\min C_{\min_q} X_q^*$ returns the minimum cost sum of N assignments. On the other hand, we can compute the maximum cost sum if robots do not change their initial assignment, by computing $\bar{C}X_0^*$. Therefore, line 6 gives the maximum cost loss when

Algorithm 1 FindTheta

Input: An $n \times n$ cost matrix C_0, $\underline{C}$, and $\bar{C}$

Output: A set of assignments X_q^* and $\theta(X_q^*)$ for $q = 0, \cdots, N$

```
1  $i = 0, q = 1$
2  $X_0^* = \text{Hungarian}(C_0)$
3  $\theta_0(X^*) = \text{SA}(X_0^*, C_0)$ // compute Eq. 8.
4  $\theta(X^*) = \theta_0(X^*)$
5  while (1)
6     $(\mathbf{c}'_i, obj_i) = \text{linprog}(l_i, \underline{C}, \bar{C}, \max)$ // max $l_i$ over the bounds.
                                                  // $l_i$: $i$-th linear boundary in $\theta(X^*)$
7     if $obj_i \geq 0$ // if $C$ does not satisfy $l_i \geq 0$,
8        $i = i + 1$
9        if $i = |\theta(X^*)|$ // if all linear boundaries in $\theta(X^*)$ are checked,
10          break
11       end if
12    else // perturb a point $p$ on $l_i$ toward $\mathbf{c}'$ to find a new $X^*$ and $\theta(X^*)$.
13       $\mathbf{p} = \frac{\mathbf{c}' \cdot \mathbf{l}_i}{|\mathbf{l}_i|^2} \mathbf{l}_i$ // $\mathbf{p}$ 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       $C_q = \text{reshape}(\mathbf{p}_{\text{new}}, n)$ // reshape a vector to an $n \times n$ matrix.
17       $X_q^* = \text{Hungarian}(C_q)$
18       $\theta(X_q^*) = \text{SA}(X_q^*, C_q)$
19       $\theta(X^*) = \theta(X^*) \cup \theta(X_q^*)$
20       $i = 0$
21       $q = q + 1$
22    end if
23 end while
24 return $\{X_0^*, \cdots, X_N^*\}$ and $\{\theta(X_0^*), \cdots, \theta(X_N^*)\}$
```

robots persist the initial assignment while having no communication and re-assignment.

Algorithm 2 MaxLoss

```
Input: An $n \times n$ cost matrix $C_0$, $\underline{C}$, and $\bar{C}$
Output: A maximum cost difference $c_{\text{worst}}$

1 $(\{X_0^*, \cdots, X_N^*\}, \{\theta(X_0^*), \cdots, \theta(X_N^*)\}) = \text{FindTheta}(C_0, \underline{C}, \bar{C})$
2 for $q = 0$ to $N$
3     $(\mathbf{c}'_q, obj_q) = \text{linprog}(c, \theta(X_q^*), \underline{C}, \bar{C}, \min)$
                                        // minimize costs over $\theta(X_q^*)$ with the bounds.
4     $C_q = \text{reshape}(\vec{c'}_q, n)$
5 end if
6 $c_{\text{worst}} = \max\{\bar{C}X_0^* - \min(C_{\min_q} X_q^*)\}$
7 return $c_{\text{worst}}$
```

One can decide with c_{worst} whether to persist with the initial assignment by considering the computational/communication expense of a re-assignment.

4.4 Incremental Communication

$\theta(X^*)$ can be summarized in different ways to show relevant information about the effect of cost changes. A one-dimensional cut yields a lower and an upper bound for each cost. This interval is valid if all other costs remain unchanged. But α-dimensional cuts allow simultaneous changes of the α costs, but $n^2 - \alpha$ costs must remain unchanged. The tolerance approach finds the maximum tolerance percentage of the costs that finally gives a tolerance region. The region is a hyper-cuboid in which each dimension is bounded by an interval $c_{ij} - \tau_{ij} \leq c_{ij} \leq c_{ij} + \tau_{ij}$ where $\tau_{ij} \in \mathbb{R}^{\geq 0}$. (See [5] and the more recent advance by Filippi [14] for details.)

This intervals (call this τ-interval for distinction) is not larger than the 1-D cuts of $\theta(X^*)$, but they are independent from other cost changes. It is attractive in multi-robot systems because robots do not need any communication for cost changes, unless their own intervals are violated by cost changes. On the other hand, even though one of a robot's costs violates its τ-interval, other cost changes countervail the violation. For example, an increase of a cost violates the interval, but a decrease of another cost could retain the optimal assignment. We develop an algorithm shown in Alg. 3 that incrementally checks a violation from a robot itself to adjacent robots.

$\theta(X_0^*)$ and τ-intervals of the initial assignment are computed and distributed to robots (line 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 C_{v_i} (line 6-11). R_i checks $c_{ij} \in C_{v_i}$ altogether whether they satisfy $\theta(X_0^*)$. (Use c_{ij} of C_0 for $c_{ij} \notin C_{v_i}$.[6]) The checking returns $V_i \in \{0, 1\}$ where $V_i = 0$ means that the cost changes turn out not to violate $\theta(X_0^*)$ and otherwise $V_i = 1$. It can be done simply by substituting cost variables in $\theta(X_0^*)$ by the initial and changed costs (line 13). If $V_i = 0$, the algorithm terminates and return V_i to the central unit. Otherwise, R_i finds an adjacent robot and receives its changed cost set C_{v_a}.[7] If any of R_i finally returns $V_i = 1$, the robots need global communication; if none of R_i returns $V_i = 1$, their cost changes do not alter the current assignment, and this fact is checked without global communication.

4.5 Complexity Analysis

Computing $\theta(X^*)$ has $O(|k|n^2)$ time complexity where $|k|$ is the number of degenerate solution sets. Each k has $(n-1)^2$ linear boundaries so there are at most $|k|(n-1)^2$ boundaries. Alg. 1 is dominated by $\theta(X^*)$ computation in the while loop (lines 5–23). Each loop iterates if a new $\theta(X^*)$ has found. Therefore, the time complexity is $O((|k|n^2)^N)$ where N is the number of possible assignments with cost matrix C. Alg. 2 executes Alg. 1 first, and an $O(n^3)$ LP runs N times. Then it has $O((|k|n^2)^N + Nn^3) = O((|k|n^2)^N)$. Alg. 3 includes Alg. 1 so is dominated by it, but the remainder of the procedure, which runs on each robot has $O(n)$ time complexity (we ignoring the costs of inter-robot communication in this analysis). If Alg. 2 follows after executing Alg. 1, its complexity is $O(Nn^3)$ since it uses the output of Alg. 1.

The worst-case time complexity is not polynomial to input size because N and $|k|$ are on the order of a factorial of n. However, not all $c_{ij} \in \mathbb{R}^{\geq 0}$ are likely to be considered because it is a bounded region, and the number of possible assignments is manageable. Also, $|k|$ can be reduced by using known methods such as that in [13], as discussed.

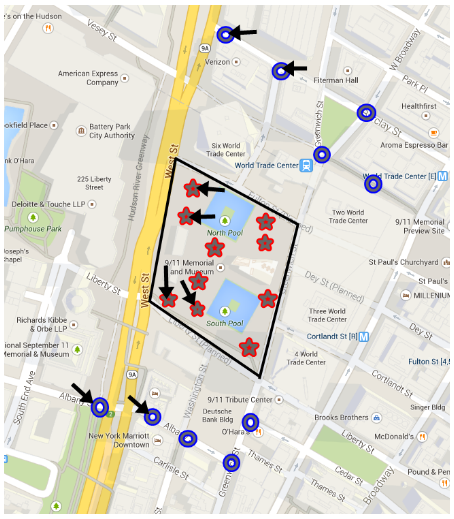
[6]This needs an assumption that $c_{ij} \notin C_{v_i}$ remain unchanged.

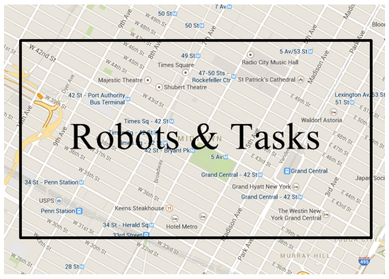
[7]We assume there is at least one robot in range. If there is no such robot, R_i can navigate or wait to have a robot.

Algorithm 3 IncrementalComm

Input: An $n \times n$ cost matrix C_0, $\underline{C}$, and $\bar{C}$
Output: Indicator variables $\{V_1, \cdots, V_n\}$

```
1  $l = 1, V_1, \cdots, V_n = 0, C_{v_i} = \emptyset$
2  $X_0^* = \text{Hungarian}(C_0)$
3  $\theta(X_0^*) = \text{SA}(X_0^*, C_0)$
4  $\mathcal{T} = \text{TA}(\theta(X_0^*), C_0)$ // compute $\tau$-intervals: $\mathcal{T}_{ij} = [c_{ij} - \tau_{ij}, c_{ij} + \tau_{ij}]$
5  Distribute $\theta(X_0^*)$ and $\mathcal{T}_{ij}$ to corresponding $R_i$
   // Below lines run 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    $V_i = \text{Check}(\theta(X_0^*), C_{v_i}, C_0)$ // 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 $V_i = 0$.
```


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



(b) Randomly distributed robots and tasks.

Figure 5: Experimental setup for the multi-robot navigation problem. The marked robots and tasks in (a) are specially chosen for Section 5.3.

5 Experiments

We consider two scenarios based on reality where cost is traveling time. Both employ the same assumptions as the example in Section 3.4. The first one is a rescue scenario shown in Fig. 5(a). Here, 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 at every 10 m. The time to push one object is $t_w \in [0, 1]$. The second scenario is the multi-robot navigation problem shown in Fig. 5(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 at 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 [15]. The raw data are in meters but converted to time (sec) by considering robots' moving speeds.

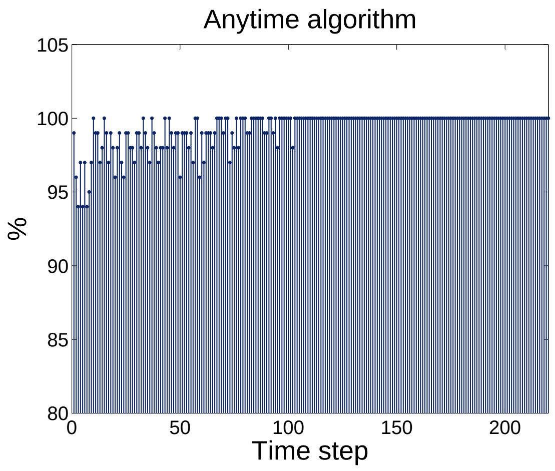


Figure 6: The performance of the anytime algorithm. It quickly approaches to 100% which is the result from the exact method.

5.1 Computing $\theta(X^*)$

5.1.1 An exact method

The running times of the exact method are 0.014, 0.0764, 1.4524, 98.0622 sec for $n = 3, 4, 5, 6$, respectively (variances are 0.001, 0.006, 0.0021, 0.1947, and 10 iterations). Since the number of degenerate solutions $|k|$ has factorial growth as n increases, the running time is not fast for larger problem instances. However, as discussed above, $|k|$ can be reduced. With large sizes, the anytime algorithm we suggested can help decrease running-time.

5.1.2 An anytime algorithm

As discussed, the anytime algorithm computes smaller area $\theta(X^*)$. Fig. 6 shows percentages of the area from the anytime method with respect to the complete $\theta(X^*)$ from the exact method when $n = 3$. For each time step, the percentage is measured by 100 uniform samples over $c_{ij} \in [0, 1000], \forall i, j$. The anytime algorithm quickly approaches to 100% (at the 220-th step, it is same with the exact method). This means that $\theta(X^*)$ from this anytime method is likely good enough to useful in most practical instances. Since the topic of this paper is not about improving running time *per se*, we use the exact method to find a theoretically complete region.

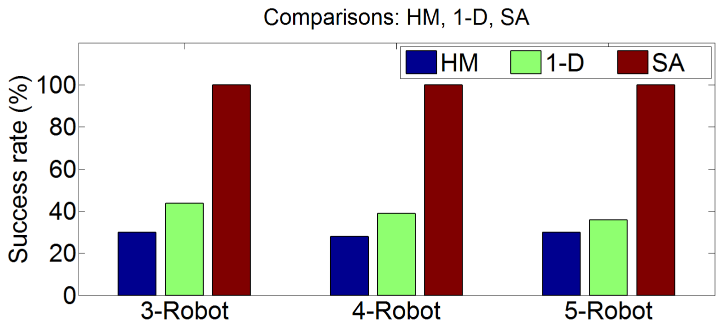


Figure 7: Comparisons of different approaches with respect to cost changes.

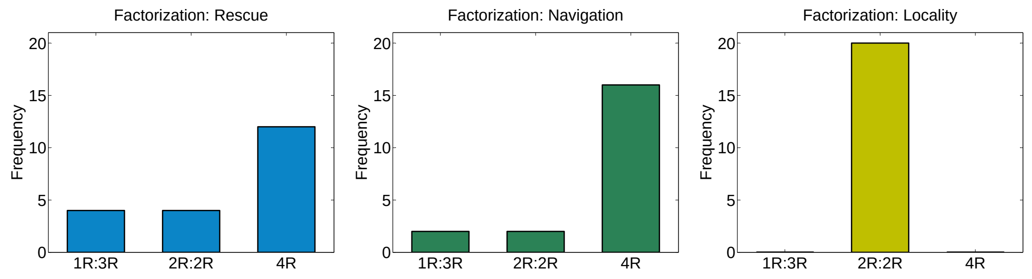
5.2 Reducing Futile Effort

We compare systems with the Hungarian method, 1-D intervals, and SA to see how efficiently they deal with cost changes. Suppose that there are multiple consecutive updates to costs given to the central unit. A system using the standard Hungarian method 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 be fruitless attempts. Employing the 1-D interval method (e.g., iHM) saves 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. We measure the number of effective re-computations with the same cost changes. We compare the standard Hungarian method and the 1-D cuts of $\theta(X^*)$, which are identical to the intervals from the iHM, and $\theta(X^*)$.

Given an arbitrary $n \times n$ cost matrix (for $n = 3, 4, 5$), an optimal assignment was computed. Then 50 random matrices, uniformly sampled between $[0, 2]$, are added to the cost matrix. The result is shown in Fig. 7 and Table 1. The success rate is computed by (# of assignment changes/# of re-computations)$\times$ 100. The result clearly shows that SA reduces unnecessary computations and communication.

Table 1: Comparisons of different approaches with respect to cost changes. (The Hungarian method, 1-D intervals, sensitivity analysis.)

	$n = 3$			$n = 4$			$n = 5$		
Method	Attempts	Success	Rate	Attempts	Success	Rate	Attempts	Success	Rate
HM	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%



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

Figure 8: Factorization results. Frequencies of cliques found (20 iterations).

5.3 Factorizing a Team of Robots

For each scenario, we randomly choose four robots and four tasks from the data collected. For each chosen problem instance, we run Alg. 1. The result is shown in Fig. 8 and Table 2 (2R:2R means that there are two cliques of two robots). Even though the scenarios do not have obvious spatial sparsity and/or locality, the algorithm is able to detect cliques when a team has such structure. The average running times of two scenarios are 2.0210 sec and 2.2768 sec ($\sigma^2 = 0.1144$, $\sigma^2 = 0.1851$ for 20 iterations), respectively. We also report results of factorization when tasks do have strong spatial locality (Fig. 8(c)). Two robots and two tasks are located as the marked robots and tasks in Fig. 8.

5.4 Persisting with an Initial Assignment

Table 3 shows examples of maximum cost losses for different sizes of team. One (the central unit or an operator) can decide whether to execute the initial assignment without having any communication and computation using the cost loss information. If the communication/computation expenses

Table 2: Factorization results. Frequencies of cliques found (20 iterations).

Clique size	Frequency		
	Rescue	Navigation	Locality
1R:3R	4	2	0
2R:2R	4	2	20
4R only	12	16	0

Table 3: The maximum loss of persisting assignment. Some examples of execution results are shown (navigation scenario). The middle three columns shows cost sums (sec), and the last column shows running time (sec).

Size	Persist	Change	Max Loss	Time
2R	867.4	591.0	276.40	0.0624
3R	1036.6	518.2	518.4	0.8424
4R	1885.2	895.7	989.5	16.1305

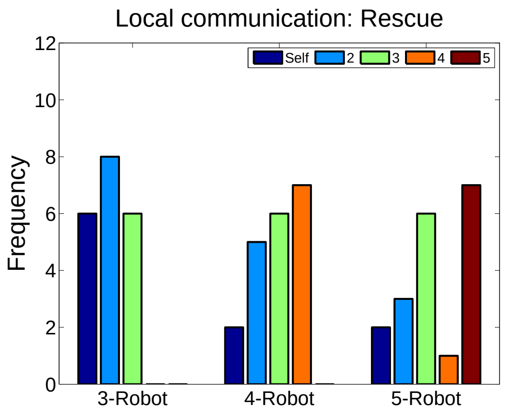
are prohibitive, it would be beneficial to persist the initial assignment.

Table 4: Frequency of communication ranges. The bold numbers indicate the frequencies of local communications. For example, in the rescue scenario, 3-robot team has 6 self checks and 8 two-robot communications.

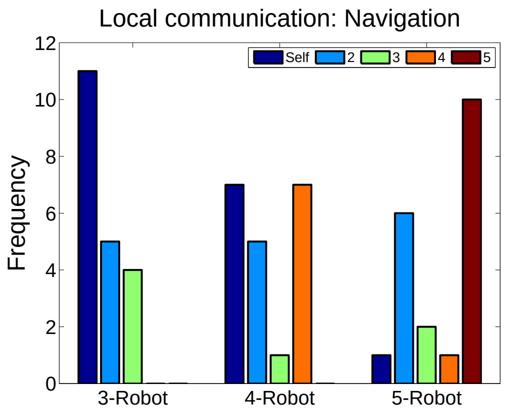
	Rescue							Navigation						
Range / Team size	Self	2	3	4	5	Time (sec)		Self	2	3	4	5	Time (sec)	
						Mean	Var						Mean	Var
3R	**6**	**8**	6	N/A	N/A	0.0769	0.0004	**11**	**5**	4	N/A	N/A	0.0476	0.002
4R	**2**	**5**	**6**	7	N/A	0.5156	0.0471	**7**	**5**	**1**	7	N/A	0.4898	0.373
5R	**2**	**3**	**6**	**1**	7	20.0539	55.4456	**1**	**6**	**2**	**1**	10	19.5001	60.9115

5.5 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 any robot has intervals violated, the robot runs the individual procedure in Alg. 3. For each changed set of costs, we check how many robots are involved in communicating, and record the frequency of occurrence for this number. A team may have several local checks, but one robot may require global communication. In such a case, we record the largest communication needed among the robots. Note that we ensure every robot has at least one



(a) The rescue scenario.



(b) The navigation scenario.

Figure 9: Frequency of communication ranges. For each team size, the left most bar means individual check whereas the right most bar mean global communication. Local communication is more frequent with Alg. 3.

violation so all robots execute Alg. 3. We randomly choose robots and tasks from the data sets. We change the team size from three to five. Fig. 9 shows the results (for 20 iterations). In many executions, purely local communication is enough (bold numbers in Table 4) to see how the costs changes affect optimality of the current assignment. Note that running time includes the central unit's computation time for the τ-interval and $\theta(X^*)$. As the team size increases, the running time increases as there is a combinatorial number of local communications. The variance also increases because an early termination takes very short time while additional local communications take an amount of time related to a combinatorial factor.

6 Conclusion

In this paper, we employed a sensitivity analysis approach for multi-robot task allocation and compared 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 fast approximate version, which is an anytime algorithm. We examined our algorithms with realistic scenarios and data, not merely randomly generated matrices. Our future work will examine other types of cost uncertainty (e.g., time-varying cost functions, stochastically represented costs) and unpredictable situations in multi-robot task allocation.

APPENDIX: Understanding Degeneracy in LP

One easy way to understand degeneracy in LP is using a polytope defined by constraints of an optimization problem. In nondegenerate cases, an extreme point of a polytope corresponds to one feasible solution. In degenerate cases, one extreme point corresponds to many different degenerate solutions. It is worth noting that the stalling and cycling problems in the Simplex method are caused by pivoting between the multiple degenerate solutions on the same extreme point. See [16, 17] for more details.

References

[1] C. Nam and D. Shell, "When to do your own thing: Analysis of cost uncertainties in multi-robot task allocation at run-time," accepted for publication in the *Proc. of IEEE Int. Conf. on Robotics and Automation*, 2015.

[2] H. Kuhn, "The hungarian method for the assignment problem," *Naval Research Logistic Quarterly*, vol. 2, no. 1-2, pp. 83–97, 1955.

[3] B. Dias, R. Zlot, N. Kalra, and A. Stentz, "Market-based multirobot coordination: A survey and analysis," *Proc. of the IEEE*, vol. 94, pp. 1257–1270, 2006.

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

[5] J. Ward and R. Wendell, "Approaches to sensitivity analysis in linear programming," *Annals of Operations Research*, vol. 27, pp. 3–38, 1990.

[6] C.-J. Lin and U.-P. Wen, "Sensitivity analysis of objective function coefficients of the assignment problem," *Asia-Pacific J. of Operational Research*, vol. 24, pp. 203–221, 2007.

[7] A. Mills-Tettey, A. Stentz, and B. Dias, "The dynamic hungarian algorithm for the assignment problem with changing costs," 2007.

[8] W.-M. Shen and B. Salemi, “Distributed and dynamic task reallocation in robot organizations,” in *Proc. of IEEE Int. Conf. on Robotics and Automation*, vol. 1, 2002, pp. 1019–1024.

[9] L. Liu and D. Shell, “Assessing optimal assignment under uncertainty: An interval-based algorithm,” *Int. J. of Robotics Research*, vol. 30, no. 7, pp. 936–953, 2011.

[10] ——, “Large-scale multi-robot task allocation via dynamic partitioning and distribution,” *Autonomous Robots*, vol. 33, pp. 291–307, 2012.

[11] C.-J. Lin and U.-P. Wen, “Sensitivity analysis of the optimal assignment,” *European J. of Operational Research*, vol. 149, pp. 35–46, 2003.

[12] M. Otte and N. Correll, “The any-com approach to multi-robot coordination,” in *IEEE Int. Conf. on Robotics and Automation: Network Science and Systems Issues in Multi-Robot Autonomy*, 2010.

[13] T. Gal and J. Nedoma, “Multiparametric linear programming,” *Management Science*, vol. 18, pp. 406–422, 1972.

[14] C. Filippi, “A fresh view on the tolerance approach to sensitivity analysis in linear programming,” *European J. of Operational Research*, vol. 167, pp. 1–19, 2005.

[15] Google, “The Google Directions API,” https://developers.google.com/maps/documentation/directions/, 2013.

[16] H. Greenberg, “An analysis of degeneracy,” *Naval Research Logistics Quarterly*, vol. 33, pp. 635–655, 1986.

[17] T. Gal, H.-J. Kruse, and P. Zörnig, *Survey of solved and open problems in the degeneracy phenomenon.* Springer, 1988.