# When to do your own thing: Analysis of cost uncertainties in multi-robot task allocation at run-time

Changjoo Nam and Dylan A. Shell

***Abstract*— We address the problem of finding the optimal assignment of tasks to a team of robots when the associated costs may vary, which arises when robots deal with uncertain or dynamic situations. We detail how to compute a sensitivity analysis that characterizes how much costs may change before optimality is violated. Using this analysis, robots are able to avoid unnecessary re-assignment computations and reduce global communication. First, given a model of how costs may evolve, we develop an algorithm to partition the robots into independent cliques, each of which maintains global optimality by communicating only amongst themselves. Second, we propose a method for computing the worst-case sub-optimality if robots persist with the initial assignment, performing no further communication/computation. Lastly, we develop an algorithm that assesses whether cost changes affect the optimality through an escalating succession of local checks. Experiments show that the methods reduce the degree of centralization needed by a multi-robot system.**

## I. 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., using the Hungarian method [1], an auctioneer [2], or a linear programming framework). But while robots are executing their assigned task, the assumptions under which costs were calculated may be invalidated: the environment may change, robots may fail, *etc.*. One solution, ensuring a fluid response to these contingencies, is to periodically re-calculate costs and re-compute the assignments, but it incurs computational/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in some state estimates used in computing the costs, or when the costs evolve as tasks are performed and are out of date. This paper models costs for a task as varying within some prescribed range. An example is shown in Fig. 1, where a robot estimates its shortest and longest driving times.

Sensitivity analysis (SA) has been studied for several decades in Operations Research to assess the robustness of an optimization problem's optima to perturbations in the input specification [3], [4], [5]. Analysis of an optimal assignment must compute a region where costs within that region preserve the current optimal assignment. However, SA has found limited applicability 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.

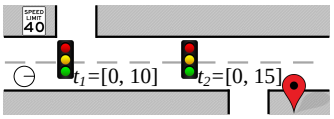



Fig. 1: An example where task costs which are not precisely known. The driving time $c$ to the destination varies depending on the traffic signals. A lower bound $\underline{c}$ is $\frac{d}{v_{\max}}$ when $t_1 = t_2 = 0$, and an upper bound $\bar{c}$ is $\frac{d}{v_{\max}} + 25$ where $d$ is the distance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

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 factorizes a group of robots into sub-teams that are able to work independently, communicating only internally, 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 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.

This work was partially supported by the NSF via IIS-1302393.

Both authors are with the Department of Computer Science and Engineering at Texas A&M University, College Station, Texas, USA. E-mail: cjnam@cse.tamu.edu

## II. Related Work

Several authors have proposed reoptimization schemes for multi-robot systems, allowing modified assignments to be computed efficiently. Mills-Tettey et al. [6] describe a dynamic (or incremental) Hungarian method to repair an earlier optimal assignment when costs change. Shen and Salemi [7] present a decentralized dynamic task allocation algorithm using a heuristic searching method. Both these algorithms still employ computation even when cost modifications do not affect the resultant assignment.

Liu and Shell [8] propose the interval Hungarian Method (iHM) that, given an optimal assignment, computes the maximum interval around each cost which preserve optimally. Robots are, thus, able to determine whether a new solution should be computed. The algorithm tackles multiple cost modifications, which occur naturally in multi-robot systems (e.g., a single robot failure affects $n$ costs), in an *ad hoc* fashion. Those authors also propose a method that uses locality and sparsity to partition the cost matrix [9]. The partitioned matrix yields clusters, each cluster is able to compute an assignment independently, thereby reducing global communication and re-assignment. Inspired by that work, we propose a factorization method for problems where mere single time-step sparsity is not enough.

## III. Problem Description

### A. 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 variable that equals either 0 or 1, where $x_{ij} = 1$ indicates that the $i$-th robot performs the $j$-th task. Otherwise, $x_{ij} = 0$. Additionally, we assuming that for each $i$ and $j$, we know that $c_{ij}$ has a range $\underline{c}_{ij} \leq c_{ij} \leq \bar{c}_{ij}$. 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

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

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

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

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

For simplicity we have assumed that $n = m$ here, but this is without loss of generality. 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 $\bar{c}_{ij}$. $C$ is not necessarily a hyper-cuboid bounded by upper and lower bounds but can be convex hyper-polytopes if cost changes are bounded by such a region.

### B. Background: Applying Sensitivity Analysis to MRTA

Since the early work of Gal [3], Sensitivity Analysis (SA) has been recognized as an important method in linear programming (LP). We present a tiny review based on a comprehensive study in [4] to help introduce SA.

A variable is basic if it corresponds to one of the vectors in the basis, given a feasible basis to a linear-programming problem. Let $k$ be an index of an entry in a set of basic variables 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. $B_k$ and $A_{N_k}$ are constraint matrices of basic variables and nonbasic variables.[1] $c_{J_k}$ and $c_{N_k}$ are costs of basic and nonbasic variables. Note that the critical region is a polyhedral cone with nonempty interiors if $c \in \mathbb{R}^s$ [4].

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 may make use of SA.[2] However, the OAP is degenerate [10] (see Appendix in [11] for details) and needs a special treatment. 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 $H = \{k : X^*_{J_k} = B_k^{-1}, X^*_{N_k} = 0\}$, representing the union of critical regions of all degenerate solutions. $\theta(X^*)$ is also a polyhedral set [4, 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.

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.

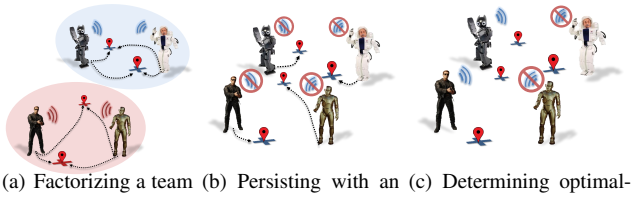
### C. Problem Statement

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 case is conceptually simple but difficult to employ in a real multi-robot system where maintaining global connectivity is expensive, and communication quality variable [12]. To alleviate these issues we describe three methods for distributing the assignment problem: (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 with their assignment) and the best-case cost sum (if they re-assign) (Fig. 2(b)),

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

[2] Technically, this relaxation requires $c \in \mathbb{Z}^{+}$. We skip details computing GCDs for the inevitable rational representation within a digital computer.



(a) Factorizing a team into smaller cliques. (b) Persisting with an initial assignment. (c) Determining optimality violations locally.

Fig. 2: Three methods to reduce MRTA centralization. (a) Cliques are sought by analyzing $\theta(X_q^*)$ with a given $C$. (b) The worst case maximum sub-optimality is computed should robots persist with their assignment. (c) Robots use local communication to check whether the changed costs now violate the current $\theta(X^*)$.

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 $C$ to factorize a team of robots. The challenge is to find $N$ assignments. A brute-force method computing all assignments for all costs in the hyper-cuboid is impossible because there are infinite $c_{ij} \in [\underline{c}_{ij}, \bar{c}_{ij}]$.

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

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

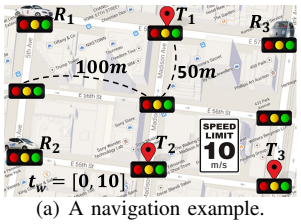
for $q = 0, \cdots, N$ where $X_0^*$ indicates the initial optimal assignment. In other words, Eq. 9 is the cost difference between the minimum among cost sums, where robots change their assignments for the updated costs, and the maximum cost sum if robots maintain their current assignment.

When 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 their own interval is violated. In (iii), these intervals must be computed and distributed to robots. If a robot finds one of its intervals violated, the robot checks whether $\theta(X_q^*)$ is violated by looking at its all other costs. If other cost changes counteract the violation, the current assignment is preserved. Otherwise, the robot communicates with an adjacent robot to increase the set of cost changes considered. Global communication may be result in the worst case, but local communication often suffices.

### D. An Example

We show how the proposed methods can be used concurrently. We consider a multi-robot version of the navigation example shown in Fig. 1. Suppose we have three autonomous robots $R_{1,2,3}$ and three destinations $T_{1,2,3}$ (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 signal. The waiting time at each signal is $t_w \in [0, 10]$. 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. Otherwise, the central unit computes the maximum cost difference (ii) to decide whether the robots reassign in response 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 substantial, it seeks cliques (i) but no such clique exists in this example. Thus, the robots finally adjust to the cost changes incrementally via (iii), i.e., using local communication.




(a) A 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 corresponding cost matrix.

Fig. 3: An example of 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.

## IV. Algorithms

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

### A. Computing $\theta(X^*)$

*1) An exact method:* A feasible solution must include $n$ optimal variables (that correspond to optimal assignments). Among the remaining $n^2 - n$ variables, $n-1$ must 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.

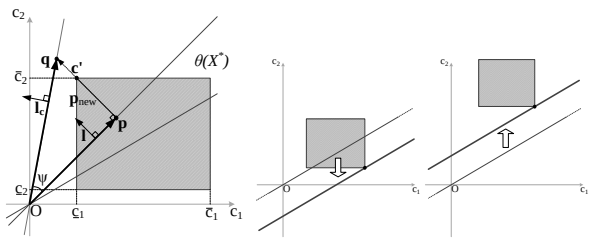
*2) An anytime algorithm:* We develop an anytime algorithm to facilitate a faster computation of $\theta(X^*)$. A partial enumeration of degenerate solutions bring an incomplete $\theta(X^*)$, but an incomplete set often takes a large portion of the complete $\theta(X^*)$. From this observation, we enumerate degenerate solutions and compute as many corresponding critical regions (Eq. 6) as possible for the given time.

### B. Factorizing a Team of Robots

Factorization can be achieved by analyzing all possible assignments $X_q^*$ for $q = 0, \cdots, N$ computed by Alg. 1. We give an illustrative 2-D representation in Fig. 4(a). One difference between 2-D and higher dimensions is that all linear equations in $\theta(X^*)$ are $\geq 0$ (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 $\geq 0$ when $l$ is maximized over the shaded area[3] (line 6), $l$ contains the entire cost set (the shaded area $C$ in Fig. 4a). Otherwise, the shaded area is not covered by $l$ (see Fig. 4b) thus a cost on $l$ is perturbed to find a

[3]All $l$ should be maximized because of the inequalities in Eq. 6.




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

Fig. 4: A 2-D figurative representation of cost space. (a) Bold lines represent linear boundaries 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).

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 $C$.

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

**Theorem** 4.1. If $l$ is an arbitrary boundary in $\theta(X^*)$. The magnitude of a perturbation $\epsilon$ to perturb an arbitrary point $\mathbf{p}$ on $l$ of $\frac{|\mathbf{p}|}{n}$ will not miss any $\theta(X^*)$. (Proof appears in [11].)

Cliques can be found 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.

### C. Choosing to maintain with the Initial Assignment

Pseudocode appears 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 minimum cost $C_{\min_q}$. In line 6, $\min C_{\min_q} X_q^*$ returns the minimum cost sum of $N$ assignments. One can compute the maximum cost sum if robots maintain the assignment by computing $\bar{C}X_0^*$, i.e., line 6.

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

### D. Incremental Communication

$\theta(X^*)$ can be summarized to show relevant information about the effect of cost changes in different ways. 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 $\alpha$-dimensional cuts allow simultaneous changes of the $\alpha$ 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 [4], [14] for details.)

These intervals (call these $\tau$-intervals) are not larger than the 1-D cuts of $\theta(X^*)$, but they are independent from other cost changes. This is useful in multi-robot systems because robots need no communication to assess cost changes, unless their own intervals are violated. On the other hand, even though one of a robot's costs violates its $\tau$-interval, other cost changes may offset the violation. We develop an algorithm (Alg. 3) that incrementally checks a violation from a robot itself to adjacent robots.

**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 $\{\theta(X_0^*), \cdots, \theta(X_N^*)\}$

**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 **return** $c_{\text{worst}} = \max\{\bar{C}X_0^* - \min(C_{\min_q} X_q^*)\}$

$\theta(X_0^*)$ and $\tau$-intervals of the initial assignment are computed and distributed to robots (lines 2-5). Then the procedure that followings runs on each robot $R_i$ concurrently. If $c_{ij}$ violates its $\tau$-interval, costs are collected in $C_{v_i}$ (line 6-11). $R_i$ checks $c_{ij} \in C_{v_i}$ whether they together satisfy $\theta(X_0^*)$. (Use $c_{ij}$ of $C_0$ for $c_{ij} \notin C_{v_i}$.[4]) The check 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 $V_i = 1$. This is achieved by substituting cost variables in $\theta(X_0^*)$ with the initial and changed costs (line 13). If $V_i = 0$, the algorithm terminates and returns $V_i$ to the central unit. Otherwise, $R_i$ finds an adjacent robot and receives its changed cost set $C_{v_a}$.[5] If any of $R_i$ finally returns $V_i = 1$, the robots will need global communication; if none of $R_i$ returns $V_i = 1$, their cost changes do not alter the current assignment, and this fact has been determined soundly without global communication.

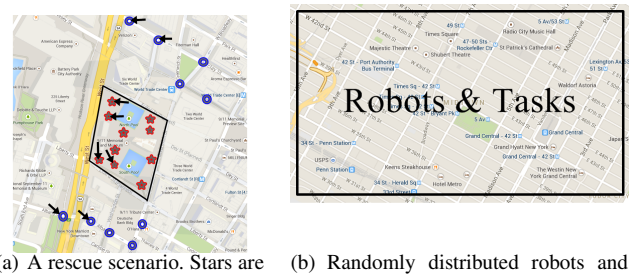
### E. Complexity Analysis

Computing $\theta(X^*)$ takes $O(|k|n^2)$ time 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 $\theta(X^*)$ was

[4] This needs an assumption that $c_{ij} \notin C_{v_i}$ remain unchanged.
[5] We assume it can reach at least one robot.

**Algorithm 3** IncrementalComm

**Input:** An $n \times n$ cost matrix $C_0$, $\underline{C}$, and $\bar{C}$
**Output:** Indicator variables $\{V_1, \cdots, V_n\}$
$l = 1$, $V_1, \cdots, V_n = 0$, $C_{v_i} = \emptyset$
$X_0^* = \text{Hungarian}(C_0)$
$\theta(X_0^*) = \text{SA}(X_0^*, C_0)$
$\mathcal{T} = \text{TA}(\theta(X_0^*), C_0)$ // compute $\tau$-intervals: $\mathcal{T}_{ij} = [c_{ij} - \tau_{ij}, c_{ij} + \tau_{ij}]$
Distribute $\theta(X_0^*)$ and $\mathcal{T}_{ij}$ to corresponding $R_i$
// Below lines run on each robot $R_i$ concurrently.
**for** $j = 1$ to $n$ // $i$ is fixed to each robot's index.
  **if** $\underline{c}_{ij} < c_{ij} - \tau_{ij}$ and $c_{ij} + \tau_{ij} > \bar{c}_{ij}$ // if $\mathcal{T}_{ij}$ is violated,
    $V_i = 1$ // there is at least one violation in $R_i$'s cost.
    $C_{v_i} = C_{v_i} \cup c_{ij}$ // collect violated costs
  **end if**
**end for**
**while** $|C_{v_i}| \leq n^2$ // while not all costs are included
  $V_i = \text{Check}(\theta(X_0^*), C_{v_i}, C_0)$ // check $C_{v_i}$ altogether
  **if** $V_i = 0$
    **break**
  **end if**
  $(R_a, C_{v_a}) = \text{FindAdjacent}(R_i)$ // $R_a$ is an adjacent robot
  $C_{v_i} = C_{v_i} \cup C_{v_a}$
**end while**
**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.

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

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, the remainder of the procedure, which runs on each robot, has $O(n)$ time complexity. 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 in 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 usually 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.

## V. Experiments

We consider two scenarios where cost is traveling time, both employ the same assumptions as the example in Section III-D. The first one is a rescue scenario (Fig. 5a) where 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 at 1 m/s and meet debris at every 10 m. The time to push one object is $t_w \in [0, 1]$. The second scenario is a navigation problem (Fig. 5b), where 30 Robots and 30 tasks are uniformly distributed in an area. The robots move at 10 m/s and encounter a traffic signal at every 300 m whose waiting time is $t_w \in [0, 30]$. Distances between robots and tasks are collected using the Google Directions API. The raw data (m) are converted to time (sec) using robots' speeds.

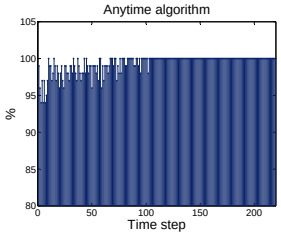



Fig. 6: The performance of the anytime algorithm. It quickly approaches to 100% which is the result from the exact method.

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

| | $n = 3$ | | | $n = 4$ | | | $n = 5$ | | |
|---|---|---|---|---|---|---|---|---|---|
| Alg. | Attempts | Success | Rate | Attempts | Success | Rate | Attempts | Success | Rate |
| HM | 50 | 15 | 30.0% | 50 | 14 | 28.0% | 50 | 15 | 30.0% |
| 1-D | 36 | 15 | 43.6% | 36 | 14 | 38.9% | 42 | 15 | 35.7% |
| SA | 15 | 15 | 100% | 14 | 14 | 100% | 15 | 15 | 100% |

### A. Computing $\theta(X^*)$

*1) Exact method:* The running times of the exact method are 0.014, 0.076, 1.452, 98.062 sec for $n = 3, 4, 5, 6$, respectively (variances are 0.001, 0.006, 0.002, 0.195, 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.

*2) Anytime algorithm:* The anytime algorithm computes a smaller area $\theta(X^*)$. Fig. **??** 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 220th step). This means that $\theta(X^*)$ from this anytime method is probably good enough practically for most 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.

### B. 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 are unnecessary. Employing the 1-D interval method (e.g., iHM) saves some re-computation, attempting a new assignment when any of the intervals are violated, but it still involves some wasted effort when multiple costs change simultaneously. 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 under identical cost changes, comparing the standard Hungarian method, 1-D cuts of $\theta(X^*)$ (equivalent to 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 Table I. The success rate is computed by $(\#\text{ of assignment changes}/\#\text{ of re-computations}) \times 100$. The result clearly shows that SA reduces unnecessary computations and communication.

TABLE II: Factorization results. Frequencies of cliques (20 iterations).

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

TABLE III: 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 | 0.062 |
| 3R | 1036.6 | 518.2 | 518.4 | 0.842 |
| 4R | 1885.2 | 895.7 | 989.5 | 16.131 |

### C. 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 ran Alg. 1. The result is shown in Table II (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 2.021 sec and 2.277 sec (variances are 0.114 and 0.185 for 20 iterations), respectively. We also report results of factorization when tasks do have strong spatial locality (Col 3). Two robots and two tasks are located as the marked robots and tasks in Table II.

### D. Maintaining an assignment

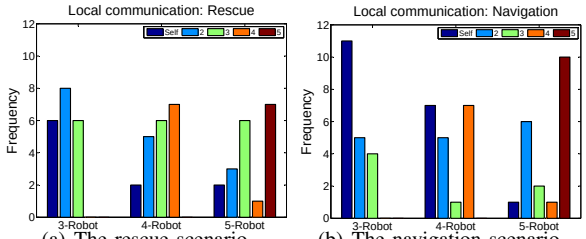
Table III shows examples of maximum cost losses for different sizes of team. One can decide whether to continue with the initial assignment without having any communication and computation using the cost loss information, depending on the relative communication/computation expense.

### E. Incremental Communication

Finally, we show how few communication messages are actually needed to detect whether optimality has been violated. For each scenario, we compute the $\tau$-intervals and distribute them to the robots. Each robot independently performs its task unless its costs violate the $\tau$-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 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. 7 shows the results (for 20 iterations). In many executions, purely local communication is enough (bold numbers in Table IV). Note that running time includes the central unit's computation time for the $\tau$-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.

TABLE IV: 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 | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Team | Range | | | | | Time (sec) | | Range | | | | | Time (sec) | |
| size | Self | 2 | 3 | 4 | 5 | Mean | Var | Self | 2 | 3 | 4 | 5 | Mean | Var |
| 3R | **6** | **8** | 6 | - | - | 0.077 | 0.000 | **11** | **5** | 4 | - | - | 0.048 | 0.002 |
| 4R | **2** | **5** | **6** | 7 | - | 0.516 | 0.047 | **7** | **5** | **1** | 7 | - | 0.490 | 0.373 |
| 5R | **2** | **3** | **6** | **1** | 7 | 20.054 | 55.446 | **1** | **6** | **2** | **1** | 10 | 19.500 | 60.912 |



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

Fig. 7: 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.

## VI. Conclusion

In this paper, we employ a sensitivity analysis for multi-robot task allocation and compared it with other methods, showing that is advantageous when costs change. We 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.

## References


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

[2] 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.

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

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

[5] 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.

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

[7] 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.

[8] 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.

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

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

[11] C. Nam and D. Shell, "Analyzing uncertainties in cost: Mitigating centralization in multi-robot task allocation," http://engineering.tamu.edu/media/2193014/2015-2-1.pdf, CSE Dept., Texas A&M Univ., Tech. Rep., 2015.

[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," *Euro. J. of Operational Research*, vol. 167, pp. 1–19, 2005.