Assignment Algorithms for Modeling Resource Contention in Multi-Robot Task Allocation

Changjoo Nam, *Student Member, IEEE,* and Dylan A. Shell, *Member, IEEE*

***Abstract*—We consider a multi-robot task allocation (MRTA) problem where costs of performing tasks are interrelated so that the overall performance of the team need not be a standard sum-of-costs (or utilities) model. The generalized optimization form we introduce allows for additional costs incurred by resource contention to be treated straightforwardly. In this variant, a team of networked robots may choose one of a set of shared resources to perform a task (e.g., several routes to reach a destination, or use of other shared resources), and interference may be modeled as occurring when multiple robots use the same resource. We show that the general problem is an NP-hard optimization problem, and investigate specialized sub-instances where the interrelations between costs that are linear or convex functions.**

We propose an exact algorithm for the general problem and, turning to the more specialized sub-instances, introduce an optimal polynomial-time algorithm and an approximation polynomial-time algorithm for the others. The exact algorithm finds an optimal assignment in a reasonable time on small instances. The other two algorithms find an optimal assignment in a short time even if a problem is of considerable size (e.g., in the linear case, 0.5786 sec for 100 robots) and a high-quality solution quickly (e.g., in the convex case, 0.8462 sec), respectively. In contrast to conventional approximation methods, our algorithm provides the performance guarantee.

***Note to Practitioners*—Practical operation of a team of robots requires that one address an idealization made in the vast majority of the literature on task allocation: namely the presumption of task independence. In reality, tasks are not performed in perfect isolation and this paper shows that computing task costs independently, although a prevalent modeling simplification, may be detrimental. Whenever robots use shared resources (e.g., narrow passages, limited communication bandwidth), resource contention and physical interference may cause performance to degrade. These aspects can be thought of as interrelationships between tasks costs and this article introduces an augmented model that expresses such interrelationships by capturing resource-based interactions among robots that change task execution costs. The model is open-ended so that the better a particular deployment of robots is understood, the greater practical domain knowledge can be brought to bear in constructing a precise model of task costs and their interdependencies. This paper describes optimization methods which incorporate the additional costs incurred by resource contention, allowing different types of model (e.g., linear or convex) giving the practitioner flexibility in selecting the model most suited for their specific application. Generally, the algorithms described are fast enough to be applied to real-time applications, but the experimental data also enable an understanding of modelling complexity vs. running-time.**

***Index Terms*—Multi-robot task allocation, assignment algorithm, resource contention, interference**

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.
E-mail: {cjnam, dshell} at cse.tamu.edu

Manuscript received August 1, 2014.

I. Introduction

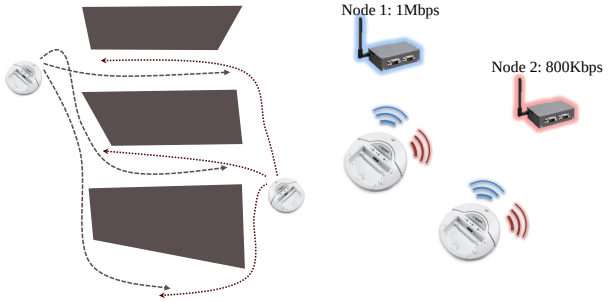
MULTI-ROBOT systems are becoming popular in real-world applications owing in part to the advances in computation power and sensor/communication technology. Representative applications, in which multiple networked robots have demonstrated their advantages over single robots, include environmental monitoring [2], object clustering [3], search and rescue [4], warehouse automation (e.g., Kiva systems), and aerial vehicle delivery (e.g., Amazon). Many problems are identified in various applications such as multi-robot control, human-robot/swarm interaction, communication protocol, and task allocation. Among them, multi-robot task allocation (MRTA) is one of the fundamental problems to be solved in multi-robot coordination, which is independent from the domains where applications are situated.

MRTA addresses optimization of collective performance by reasoning about which robots in a team should perform which tasks. Even starting with the classical work, many different approaches have been proposed, such as behavior-based [5], [6] and market-based [7], [8], [9] task allocation. Although resource contention and physical interference have long been known to limit performance [10], [11], [12], the vast majority of MRTA work considers settings for which interference is treated as negligible (*cf.* review in [13]). This limits the applicability of these methods and computing a task assignment under assumptions of noninterference may produce suboptimal behavior even if the algorithm solves the assignment problem optimally. Several authors have proposed task allocation approaches that model or avoid interference (usually physical interference), see for example, [14], [15], [16], [17] (a summary is shown in Table I.). These works, however, do not set out to achieve global optimality, or understand the computational consequences of a model of interference.

In this paper, following the lead of early and practical work, we assume a networked system in which all information is known by at least one robot that is responsible for optimizing task allocation. In practice, this robot can be dynamically elected robot from amongst the team. We also assume that a robot can perform only one task at a time, each task requires only one robot to execute it, and that the allocation of tasks to robots need consider only current (instantaneously available) information and need not hedge against future plans. This problem falls into single-task robots (ST), single-robot tasks (SR) and instantaneous assignment (IA) axes [13]. The ST-SR-IA MRTA problem can be posed as an Optimal Assignment Problem (OAP), which is well-studied, and can be cast as an integral linear program which is in complexity class P. This conventional MRTA problem does not specify how robots use resources so it is unable for it to account for

TABLE I: A summary of algorithms that consider interference among robots.

Authors & paper	Way to deal with interference	Poly-time?	Optimal?
Dahl et al. [14]	Use reinforcement learning to distribute resources to robots	No	No
Guerrero and Oliver [15]	Include the effect of interference in the utility function of the auction method	No (deadline)	No
Choi et al. [16]	Use a market-based distributed agreement protocol that guarantees a conflict-free assignment	Yes	No (provably good solution)
Pini et al. [17]	Spatially partition tasks to reduce interference	No	No



(a) Physical space (b) Communication bandwidth

Fig. 1: Two examples of resources with limited capacities that must be shared in most practical contexts. Both communication and space contention cause performance to scale sub-linearly with the number of robots.

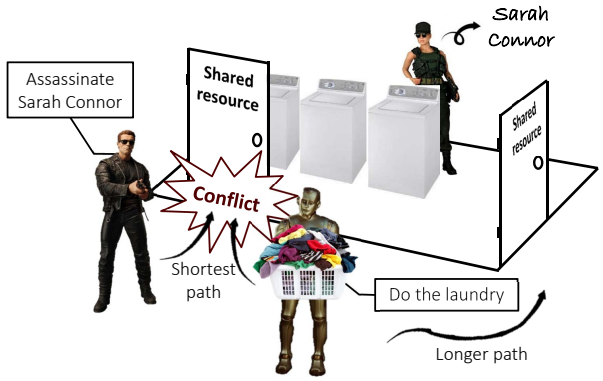


Fig. 2: A specific example of resource contention: two robots choose the shortest path to perform their tasks, they should compute their paths to avoid interaction with each other. When the right robot chooses the longer path via the door on the far right, the sum of distances is larger, but it minimizes cost when resource contention is considered.

interference incurred by sharing resources. Instead, it assumes that resources are individually allocated to robots or, if shared, that they impose no limits.

In our problem, however, robots may have to choose between resources used to perform tasks (e.g., several routes to reach a destination), as shown in Fig. 1, and the costs of performing the tasks may vary depending on the choice. If several robots use the same resource (reflected in a relationship between their choices), we allow interference between them to be modeled. Inter-agent interference (as described in Fig. 2) is treated mathematically as a penalization to the cost of performing that task. In this manner, we can model shared resources and generalize the conventional MRTA problem formulation to include resource contention. The result is an optimization problem for finding the minimum-cost solution including the interference induced penalization cost. We term this the multiple-choice assignment problem with penalization (mAPwP). The model we introduce allows a robot to make a selection from among multiple means by which it could perform a task. Naturally, the penalization depends on the particular selection.

In general, there are many ways penalization costs could be estimated. When evaluation of the interference is polynomial-time computable, we call this the mAPwP problem with polynomial-time computable penalization function (P-type mAPwP). Even with a cheaply computable penalization function, we show that the P-type problem is NP-hard[1]. We also investigate two other problems that have particular forms of penalization functions: linear and general convex penalization functions. We show that the two problems are in P and NP-hard, respectively. We provide an exact algorithm and two polynomial-time algorithms for the problems. The algorithms are domain-independent so that it can be used for many multi-agent scenarios that have quantifiable interference between agents.

The remainder of this paper is organized as follows. Section II discusses the related literature on optimization methods for MRTA. Section III defines the problem mathematically, and Section IV describes the NP-hardness results. Section V presents algorithms, and Section VI extends the suggested modeling method of resource contention to another interrelated costs. Section VII describes experiments, and the final section concludes.

II. Related Work

Recent studies dealing with limited shared resources in multi-robot systems are mainly focused on the multi-robot path planning (MPP) problem: Alonso-Mora et al. [18] employ a mixed-integer quadratic programming methods to optimize trajectories of robots while avoiding collisions; Yu and LaValle [19] propose an integer linear programming method to find collision-free paths for multiple robots; He and van den Berg [20] suggest an MPP algorithm that consists of macro-, micro-, and meso-scale planners. Their meso-scale planner considers groups of other robots as a coherent moving obstacle while the micro-scale planner locally avoids individual obstacles. Those methods quickly find high-quality solutions. However, their approaches are domain-specific so not appropriate for general problems where robots contend for arbitrary shared resources, not necessarily only physical

[1]Adding the notion of multiple choices does not change the complexity class, which is P. However, introducing the penalization function makes the problem hard.

space. Moreover, resources modeled in [19] are able to accommodate only one robot at each time step, which is restrictive to model real-world applications. In [20], the micro-scale collision avoidance is based on local observations, and it does not achieve global optimality. We are not aware of previous hardness results with respect to resource sharing in multi-robot systems.

The equivalence of the classical assignment problem by a network flow problem has been well known for decades. This may lead to the suggestion that one can prevent interference by imposing additional constraints in the form of capacity constraints in the flow formulation. This can be solved by a centralized manner [21] or a distributed manner [22], [23]. However, that approach models interference as a binary penalization, which is zero or infinite, whereas incurred by resource contention are more widely applicable if the interference is modeled as a continuous function that increases proportionally to the amount of interference. (See, for example, our use of published and validated traffic models in Section VII.)

The approach of imposing constraints to restrict robots from using shared resources is used in many MRTA algorithms such as [24], [25], [26], [27], [28], [29]. This approach is widely used because of its simplicity since the constraints can be constructed once restrictions on resources are identified (e.g., the maximum number of robots using a shared resource). However, such constraints satisfy some models of shared resource, but the models are not adequately rich to describe the problem precisely. For example, if a capacity constraint is imposed for a shared resource, an allocation that violates the constraint cannot be considered at all. However, a shared resource can be used without a capacity limit but with some additional costs as more robots use the resource. Inversely, there could be additional costs even though the number of robots using a shared resource is less than a capacity. In addition, [30], [31] also consider MRTA problems where tasks have dependencies. The inter-task dependencies are caused by precedence or deadline of tasks. The dependencies are handled by imposing constraints. Again, this approach may not be describes some problems precisely. For example, a shipping task could miss its deadline if a penalty is paid for not fulfilling the due date.

An alternative is for the P-type problem can be cast as a linearly constrained 0-1 programming problem, with the penalization function incorporated into the objective function with the cost sum. The objective function is optimized over a polytope defined by the mutual exclusion and integral constraints. The results in this paper suggest that one can have an optimal solution in polynomial time if the penalization function is linear. When the penalization is more complex, a common method to solve the problem is enumeration, for example using the branch-and-bound method, but its time complexity in the worst case is as bad as that of an exhaustive search; rather more insight is gained by employing the method we introduce in this paper. Many practical algorithms [32], [33], [34] are suggested in the literature, but they also have exponential running time in the worst case. Linearizing the complex penalization function could be an alternative to have polynomial running time but has no performance guarantee.

TABLE II: Nomenclature.

$G(R,T,E)$	a bipartite multigraph consisting of two disjoining sets R and T and a collection of edges E;
L	the bit length of input variables of an instance;
N_Π	the number of all assignments;
$Q(\cdot)$	the penalization function;
Q_l	the penalization function of the l-th resource;
Q^s	the penalization of s-th assignment;
X^*	the optimal assignment;
$X^{s-/+}$	the s-th assignment before/after penalization;
c_{ijk}	the cost of performing the j-th task by the i-th robot in the k-th manner;
c^*	the cost sum of the optimal assignment;
$c^{s-/+}$	the cost sum of the s-th assignment before/after penalization;
d	the length of a road;
i	the index of vertices in R (robots);
j	the index of vertices in T (tasks);
k	the index of edges in E (choice);
n	the number of vertices in R (robots);
n_l	the number of robots on the l-th resource;
m	the number of vertices in T (tasks);
p_{ij}	the number of choices between $r_i \in R$ and $t_j \in T$;
x_{ijk}	the binary variable that indicates that the i-th robot performs the j-th task in the k-th manner;
s	the s-th best assignment in terms of optimality;
v_{f}	the traffic flow speed of a road;
β	the coefficients of a penalization function;
η	the ratio of an approximated solution to an optimal solution ($\eta = c'^*/c^*$);
λ	the slope of the headway-speed curve;
ρ	the traffic density of a road;
ρ_j	the jam density of a traffic road;

Lastly, Roughgarden [35] introduces noncooperative routing games in which each agent chooses a complete route between a source and a sink in a network in congestion-sensitive manner. Routing games have the objective of minimizing the sum of traffic costs including additional costs from congestion, which is same with the multi-vehicle traffic problem used in the experiments (Section VII-C). It is interesting that selfish agents are able to find an optimal set of routes. However, routing games confine their applications to routing problems on physical resources (e.g., roads) so they are limited to deal with general resource contention.

III. Problem Formulation

A. Bipartite Multigraph

The mAPwP problem can be expressed as a bipartite multigraph. Let $G = (R, T, E)$ be a bipartite multigraph consisting of two independent sets of vertices R and T, where $|R| = n$ and $|T| = m$, and a collection of edges E. An edge is a set of two distinct vertices denoted (i, j) and incident to i and j. Each edge in G is incident to both a vertex in R and a vertex in T, and p_{ij} is the number of edges between two vertices. The vertices in R and T can be interpreted as n robots and m tasks, respectively. An edge is a way in which a robot may use resources, for which it expected to select one among p_{ij} choices for a given task. The precise interaction between resources is modeled via penalization function, described next.

B. Multiple-Choice Assignment Problem with Penalization (mAPwP)

Given n robots and m tasks, the robots should be allocated to tasks with the minimum cost. Each allocation of a robot to a task can be done via one of the p_{ij} choices where i and j are indices of the robots and the tasks, respectively. Each of the p_{ij} choices represents some set of resources used by a robot to achieve a task. The multiple choices indicate the resources can be used in many ways. We assume we are given c_{ijk}, the interference-free cost of the i-th robot performing the j-th task through the k-th choice. Let x_{ijk} be a binary variable that equals to 0 or 1, where $x_{ijk} = 1$ indicates that the i-th robot performs the j-th task in the k-th manner. Otherwise, $x_{ijk} = 0$.

In problem domains where multiple robots share resources, use of the same limited resource will typically incur a cost. We model this via a function which corrects the interference-free assignment cost (i.e., the linear sum of costs) by including the additional cost of the effects of resource contention ($Q(\cdot)$ in Eq. 1)[2]. We assume that the cost and the penalization are nonnegative real numbers. We also permit the cost to positive infinity when interference is catastrophic (or, for example, only one robot is permitted to use the resource). We assume $n = m$. If $n \neq m$, dummy robots or tasks would be inserted to make $n = m$. Then a mathematical description of the mAPwP problem is

$$\min \sum_{i=1}^{n}\sum_{j=1}^{m}\sum_{k=1}^{p_{ij}} x_{ijk}c_{ijk} + Q(x_{111}, x_{112}, \ldots, x_{11p_{11}}, \ldots, x_{nmp_{nm}}), \tag{1}$$

subject to

$$\sum_{j=1}^{m}\sum_{k=1}^{p_{ij}} x_{ijk} = 1 \qquad \forall i, \tag{2}$$

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

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

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

We note that Eq. 5 is superfluous if no penalization function is considered or $Q(\cdot)$ is linear, because the constraint matrix satisfies the property of totally unimodular (TU) matrix. Specifically, an optimization problem with a linear objective function has only integer solutions if its constraint matrix satisfies totally unimodularity [36], so the integral constraint is not necessary.[3]

C. Penalization

The penalization function maps a particular assignment to the additional cost associated with the interference. In the formulation of mAPwP earlier, $Q(\cdot)$ denotes the penalization function in most general terms. If the mAPwP is with a polynomial-time computable $Q(\cdot)$, it is the P-type problem. The input domain for Q has $\sim O(\max\{n, m\}! \cdot (\max\{p_{ij}\})^{\min\{n,m\}})$ elements; in most cases a penalization function is more conveniently written in some factorized form. One example is if one is concerned only with the number of robots using a resource, not precisely the identities of the robots that are. If $Q_l(n_l)$ is the penalization function of the l-th choice where n_l is the number of robots for that choice, then the total penalization could be written as:

$$\begin{aligned} &Q(x_{111}, x_{112}, \ldots, x_{11p_{11}}, \ldots, x_{nmp_{nm}}) \\ &= Q_1(n_1) + Q_2(n_2) + \ldots + Q_q(n_q) \\ &= \sum_{l=1}^{q} Q_l(n_l). \end{aligned} \tag{6}$$

where q is the total number of choices in an environment. If the robots are homogeneous, n_l is the same as the number of robots on the l-th choice. Otherwise, each robot has a weight that represents the occupancy of the robot. The P-type problem is a general problem that $Q(\cdot)$ can be any form of function. If $Q(\cdot)$ is convex, the mAP becomes the mAP with convex penalization function (C-type mAPwP). Especially, it comes to be the mAP with linear penalization function (L-type mAPwP) if $Q(\cdot)$ is linear. The descriptions of the problems are summarized in Table III.

TABLE III: A summary of the mAP problems.

Problem	Description
mAPwP	The multi-choice assignment problem with penalization
P-type	The mAPwP problem with any penalization functions that are polynomial-time computable
DP-type	The decision version of the P-type problem
C-type	The mAPwP problem with convex penalization functions
L-type	The mAPwP problem with linear penalization functions

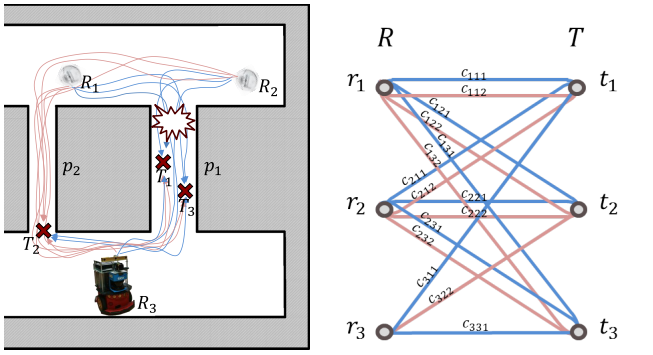
D. Examples

An example of the mAPwP is shown in Fig. 3(a). The goal is to minimize the total traveling time by distributing robots (R_1, R_2 and R_3) to three destinations (T_1, T_2 and T_3). R_1 and R_2 can use all the paths, but R_3 cannot use the passage p_2 because R_3 is wider than the passage. A weighted bipartite multigraph that is equivalent to the example is shown in Fig. 3(b). The graph has $|R| = |T| = 3$ vertices, and every pair of vertices has 2 edges except for $p_{31} = p_{32} = p_{33} = 1$. There will be interference, for example, if both R_1 and R_2 try to reach destinations on p_1, so a time delay is incurred which must be added to the total traveling time.

Types of shared resources need not be limited to physical space. A family of cooperative information collecting missions could have resource contention on shared communication channels. The mission is collecting information, such as pictures, depth information, or audio source, from environments and transmitting them to a central repository while minimizing the sum of completion time. Each robot is required to choose one of the locations in an environment and transmit collected

[2]The formal definition of $Q(\cdot)$ will be shown in Section III-C.

[3]The standard treatment of the Optimal Assignment problem without a penalization factor for task allocation (e.g., in [13]) considers only a bipartite graph (i.e., $\forall_i \forall_j p_{ij} = 1$). Although TU is well-known for the problem, we believe this to be the first recognition of this fact for the problem above.



(a) An example of the mAPwP.

(b) The equivalent graph representation.

Fig. 3: An example of the mAPwP and its graph representation. (a) Robots have a choice between routes to reach their destinations, but interference will occur if a passageway is shared (e.g., if both R_1 and R_2 try to reach destinations via p_1.) (b) A weighted bipartite multigraph representation for this example. An edge between r_i and t_j represents the use of a resource to perform the j-th task by the i-th robot, and its weight (c_{ijk}) is a cost associated with performing the task by the robot. x_{ijk} is a binary variable that indicates allocation of a robot to a task through a resource (the variables are omitted for clarity).

data through one of several private wireless networks that have different bandwidth.[4] Data transmission time depends on the size of the chosen channel's throughput and the data size, but additional transmission time occurs if the traffic exceeds the bandwidth. This example of network congestion can be formulated similarly with the physical space case.

IV. NP-Hardness of mAPwP Problems

In this section, we show the P-type and C-type problems are NP-hard optimization problems, and the L-type problem is in P. We prove the corresponding decision version of the P-type (DP-type) is NP-complete to prove the P-type problem is an NP-hard optimization problem [37]. Then we briefly describe the L-type problem is in P and show the C-type problem is NP-hard.

A. The P-type problem is NP-hard

Theorem 4.1 The DP-type problem is in NP.

Proof. The DP-type problem simply asks whether an assignment has cost less than a given threshold.
Input: n robots, m tasks, p_{ij} choices, a polynomial-time computable penalization function Q, and costs of edges c_{ijk}, a constant α.
Question: Is the penalized cost of a given assignment less than α?
Certificate: An arbitrary assignment x_{ijk}.
Algorithm:

1 Check whether the assignment violates any constraints
2 Calculate the total cost of the assignment
3 Penalize the cost by the penalization function
4 Check whether the penalized cost is less than α

[4]To simplify the problem, we assume that the time for approaching to a location and transmitting data dominates the time for other tasks such as data acquisition. We also assume that physical space is enough to perform tasks without interference among robots.

This is polynomial-time checkable so that the DP-type problem is in NP. □

Theorem 4.2 The DP-type problem is NP-hard.
Claim. The proof is based on relation to the classic boolean satisfiability problem. The 3-CNF-SAT problem asks whether a given 3-CNF formula is satisfiable or not. It is a well-known NP-complete problem. If 3-CNF-SAT $\leq_{\mathrm{P}}$ DP-type, then the DP-type problem is NP-hard.

Proof. The reduction algorithm begins with an instance of 3-CNF-SAT. Let $\Phi = C_1 \wedge C_2 \wedge ... \wedge C_k$ be a 3-CNF boolean formula with k clauses over n variables, and each clause has exactly three distinct literals. We shall construct an instance of the DP-type problem where $p_{ij} = 1$ ($i = 1, ..., n$ and $j = 1, ..., 2n$) such that Φ is satisfiable if and only if the solution of the instance of DP-type problem has cost less than a constant α.

We construct a bipartite multigraph $G = (R, T, E)$ as follows. We place n nodes $r_1, r_2, ..., r_n \in R$ for n variables and $2n$ nodes $t_1, f_1, t_2, f_2, ..., t_n, f_n \in T$ for truth values (true and false) of the variables. For $i = 1, ..., n$ and $j = 1, ..., 2n$, we put edges $(r_i, t_i) \in E$ and $(r_i, f_i) \in E$ where t_i and $f_i \in T$. The costs of the edges are given by c_{ij}. In addition, we construct an assignment by assigning vertex i in R to vertex j in T only when $x_{ij} = 1$ for $i = 1, ..., n$ and $j = 1, ..., 2n$. (Note that $x_{ij} \in \{0, 1\}$.)

Now, we construct a function Φ_{J} as follows. Each clause in Φ is transformed to a sum of terms in parentheses so that the terms correspond to the three literals in the clause. For a positive literal, we put x_{ij} where i is equal to the index of the literal and $j = 2i - 1$ whereas $j = 2i$ for a negative literal. Disjunctions of clauses are transformed to multiplications. A penalization of an assignment is defined as

$$Q = \begin{cases} 0 & \Phi_{\mathrm{J}} > 0 \\ \mathrm{N} & \text{otherwise,} \end{cases} \tag{7}$$

where N is a large number. If Φ_{J} has a solution which makes $\Phi_{\mathrm{J}} > 0$, the penalization is zero. Therefore, the cost of the assignment is $\sum_{i,j} c_{ij} x_{ij}$ and $Q = 0$ so the assignment has the total cost $\sum_{i,j} c_{ij} x_{ij}$. Otherwise, it will have a large nonzero penalization such as N. We can easily construct Q from Φ in polynomial time.

As an example, consider the construction if we have

$$\begin{aligned} \Phi = (x_1 \vee x_2 \vee \neg x_4) \wedge (x_2 \vee x_4 \vee \neg x_5) \\ \wedge (x_3 \vee \neg x_1 \vee \neg x_2), \end{aligned} \tag{8}$$

then the transformation is shown in Fig. 4. Φ has five variables so five nodes and ten nodes are placed in R and T, respectively. The nodes in R and T which have the same subscripts are connected. We produce function:

$$\begin{aligned} \Phi_J = &(x_{11} + x_{23} + x_{48}) \cdot (x_{23} + x_{47} + x_{5 \cdot 10}) \\ &\cdot (x_{35} + x_{12} + x_{24}), \end{aligned} \tag{9}$$

and its penalization will be 0 or N depending on the assignment.

We show that this transformation is a reduction in a little more detail. First, suppose that Φ has a satisfying assignment. Then each clause contains at least one literal that true is

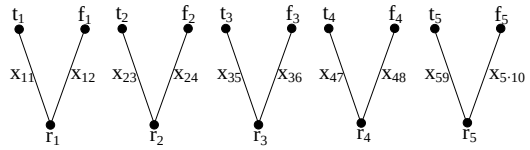


Fig. 4: The DP-type problem derived from the 3-CNF formula $\Phi = (x_1 \vee x_2 \vee \neg x_4) \wedge (x_2 \vee x_4 \vee \neg x_5) \wedge (x_3 \vee \neg x_1 \vee \neg x_2)$. A satisfying assignment of Φ has $x_1 = 1, x_2 = 1, x_3 = 1$, and x_4, x_5 either 0 or 1. Corresponding assignment is that $x_{11} = 1, x_{12} = 0, x_{23} = 1, x_{24} = 0, x_{35} = 1, x_{36} = 0$. The values of other elements do not affect the satisfiability of Φ. This assignment makes $\Phi_{\text{J}} > 0$.

assigned, and each such literal corresponds to a matching of r_i and t_i. On the contrary, a literal assigned false corresponds to a matching of r_i and f_i. Thus, assigning truth values to the literals to make Φ satisfied yields matchings between R and T. We claim that the matchings are an assignment which makes $\Phi_{\text{J}} > 0$. The assignment makes each sum of three terms (in parentheses) at least 1 so that Φ_{J}, a multiplication of the parenthesized terms, is greater than or equal to 1. Therefore, by the construction, we can get the total cost of the assignment and answer whether the cost is less than α.

Conversely, suppose that the DP-type problem has an assignment that makes $\Phi_{\text{J}} > 0$. We can assign truth assignments to the literals corresponding to the matchings between R and T so that each clause has at least one variable which is true. Since each clause is satisfied, Φ is satisfied. Therefore, 3-CNF-SAT $\leq_{\text{P}}$ DP-type.[5] □

In the example of Fig. 4, a satisfying assignment of Φ has $x_1 = 1$, $x_2 = 1$, $x_3 = 1$, and x_4, x_5 either 0 or 1. Corresponding matchings in DP-type are that r_1 and t_1, r_2 and t_2, r_3 and t_3 while r_4 is matched to either t_4 or f_4. Also, r_5 is matched to either t_5 or f_5. Therefore, the assignment is $x_{11} = 1, x_{12} = 0, x_{23} = 1, x_{24} = 0, x_{35} = 1, x_{36} = 0$. The values of other elements do not affect the satisfiability of Φ. This assignment makes $\Phi_{\text{J}} > 0$.

Corollary 4.3 By Theorem 4.1 and 4.2, the DP-type problem is NP-complete. Therefore, the P-type problem is an NP-hard optimization problem.

B. The L-type problem is in P

Mathematically, the L-type problem can be cast as an integer linear programming problem whose constraint matrix satisfies the property of totally unimodularity. This problem can be solved in polynomial time as described in [36, Corollary 2.2]. Therefore, the L-type problem is in P.

C. The C-type problem is NP-hard

The mAP with a convex quadratic penalization function (CQ-type) is a proper subset of the C-type problem, and is a natural next step after examining L-type problem. The CQ-type problem has the form

$$\min \{x^T H x + cx \colon Ax \leq b, x \in \{0,1\}\} \tag{10}$$

[5]There can be a simplification of the reduction. We can construct a function Φ_{J} from any of the 3-CNF-SAT problem Φ. We define a polynomial-time penalization function Q as Eq. 7. Then solving the DP-type problem solves the corresponding instance of the 3-CNF-SAT problem.

where H is positive semidefinite and symmetric, c is nonnegative, A is TU, and b is integer.

The following binary quadratic programming (BQP) is an NP-hard problem [38, Theorem 4.1]. The BQP problem is

$$\min \{y^T M y + dy \colon A'y \leq b', y \in \{0,1\}\} \tag{11}$$

where $M = L^T D L$, $D = I$, $d = 0$, L is TU and nonsingular, A' is TU, and b' is integer.

Theorem 4.4 The CQ-type problem is NP-hard.

Claim. If M is symmetric and positive semidefinite, we can reduce any BQP to an instance of the CQ-type problem. Namely, BQP $\leq_{\text{P}}$ CQ-type.

Proof. Since $D = I$, $M = L^T L$. Then $(L^T L)^T = (L)^T (L^T)^T = (L^T L)$. Thus, M is symmetric.

For any column vector v, $v^T L^T L v = (Lv)^T Lv = (Lv) \cdot (Lv) \geq 0$. Thus, $L^T L$ is positive semidefinite. Therefore, BQP $\leq_{\text{P}}$ CQ-type as we claimed. □

Lemma 4.5 CQ-type $\subsetneq$ C-type.

Corollary 4.6 By Theorem 4.4 and Lemma 4.5, the C-type problem is NP-hard.

D. A Polynomial-time Solvable Class of the C-type problem

The C-type problem is a nonseparable convex optimization problem. If a C-type problem can be converted to a separable convex optimization problem without breaking the totally unimodularity of the constraint matrix, the problem is solvable in polynomial time [39] and Alg. 4.2 in [39] is an optimal algorithm for these cases. Note that a nonseparable problem can be transformed to a separable problem by substituting non-separable dependent polynomials with additional independent variables and imposing additional constraints[6] (see [40, Table 13.1] and the appendix for more detail and an example). Since the cost sum part of the objective function in Eq. 1 is separable in itself, only the penalization function is subject to conversion. A constraint matrix after the conversion is

$$A_{\text{SP}} = \begin{bmatrix} A & 0 \\ A_{\text{N}} & -I \end{bmatrix} \tag{12}$$

where A is the original TU constraint matrix, and $[A_{\text{N}} - I]$ are newly imposed constraints. A is $n \times mp$, A_{N} is $w \times mp$, 0 is $n \times w$, and I is $w \times w$ matrix where w is the number of newly added variables. According to the properties of TU matrix, we have the following properties of A_{SP}:

- $[A\ 0]$ is TU.
- If A_{N} is TU, $[A_{\text{N}}\ -I]$ is also TU.
- Joining two arbitrary TU matrices is not guaranteed to make a TU matrix. Thus, A_{SP} may not be TU although both of $[A\ 0]$ and $[A_{\text{N}}\ -I]$ are TU.
- If A_{N} is not TU, then A_{SP} is not TU.

Since a TU A_{N} could make A_{SP} either TU or non–TU, the totally unimodularity of A_{SP} should be checked. The definition of TU (i.e., the determinant of every square submatrix has value -1, 0, or 1) or a necessary and sufficient condition

[6]Theoretically, any optimization problem can be restated as a separable program, but this is of limited practically as the number of the additional variables and constraints is large [40].

described in [41] can be used to check the totally unimodularity. However, using those methods for the entire A_{SP} could be computationally expensive for large-sized problems. We suggest a preliminary test to see if a transformation breaks the totally unimodularity of the original problem.

Theorem 4.7 Separable convex integer optimization problem, whose constraint matrix is not TU, is NP-hard.

Proof. Separable integer linear programming (ILP) problem is a special case of the separable convex integer programming problem. An ILP, whose constraint matrix is not TU, is NP-hard [42] regardless of whether it is separable or not. Therefore, the separable convex integer optimization problem, whose constraint matrix is not TU, is NP-hard. □

Thus, a non–TU A_{N} makes the C-type problem NP-hard by Theorem 4.7. A preliminary test that is checking the totally unimodularity of A_{N} before checking the entire A_{SP} may save time, because A_{SP} needs not to be checked if A_{N} is not TU. However, A_{SP} shall be checked if A_{N} is TU since a non–TU A_{N} is a sufficient condition, but not a necessary condition, to make non–TU A_{SP}.

E. Remark on the hardness results

There is a significance in the NP-completeness (not merely the NP-hardness) result of the DP-type problem when the penalization function is polynomial-time computable. The problem with polynomial-time solvable penalization functions has a spectrum of the hardness from P to NP-complete; the upper bound is perhaps surprising. Many MRTA problems become NP-hard when richer and more precise descriptions (e.g., additional constraints) are added to the problem formulation [13]. While the problem is not expected to be polynomial-time solvable, even if some polynomial-time algorithms did solve all NP-complete problems, the NP-hard ones might remain. If a problem has a non-polynomial-time-solvable penalization function, the problem becomes NP-hard. The spectrum is a concise visualization of understanding how hard the problem is depending on the form of the penalization function.

V. Algorithms for mAP Problems

In this section, we devise algorithms for mAP problems. The exact algorithm for the P-type problem recursively enumerates unpenalized assignments and their costs from the best assignment in terms of optimality, by calling a combinatorial optimization algorithm for each iteration. However, no enumeration and optimization algorithm exists for multigraphs, so we must extend Murty's ranking algorithm [43] and the Hungarian method [44] to the weighted bipartite multigraphs. The extension does not change the complexity class of the problem since the problem's coefficient matrix is still totally unimodular even with a bipartite multigraph [45]. We term the algorithms the Multiple-Choice (MC) Hungarian and Multiple-Choice (MC) Murty's ranking algorithm.

Then we suggest polynomial-time algorithms for the L-type and C-type problems. For brevity, we denote them by the (optimal) L-type algorithm and the (approximate) C-type algorithm, respectively. The algorithms consist of two phases: the optimization phase and the rounding phase. In the first phase, we relax the integral constraint Eq. 5 so that a solution can be obtained in polynomial time, but it can be fractional. Thus, the second phase rounds a fractional solution to ensure the integrality of the assignment. We use an interior point method (IPM) in the first phase and the MC Hungarian method in the second phase. The L-type algorithm is optimal, and the C-type algorithm is near-optimal. We provide the performance guarantee of the C-type algorithm.

A. The Multiple-Choice Hungarian Method

We generalize the Hungarian method to allow multiple choices of performing tasks. Fig. 5 shows the differences in input and output between the original Hungarian method and the MC Hungarian method. For implementation, we modify the labeling operations (the initialization and the update operations) and the path augmentation from the original Hungarian method. The labeling operations include all p_{ij} edges incident to i and j. In the path augmentation step, the minimum-weighted edge among p_{ij} is selected as the path between i and j. The pseudocode is given in Alg. 1. The time complexity of this algorithm is $O(p^2(\max\{n,m\})^3)$.

Algorithm 1 The Multiple-Choice (MC) Hungarian method

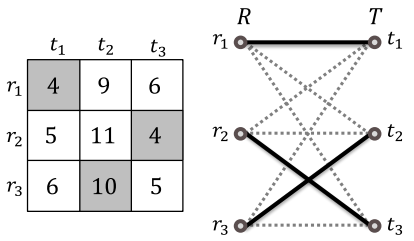
Input: An $n \times mp$ cost matrix which is equivalent to a weighted bipartite multigraph $G = (R, T, E)$ where $|R| = n, |T| = m$ and $p_{ij} = p, \forall\{i, j\}$.
Output: An optimal assignment M^* and its cost c^*.

1. Generate initial labeling $l(i) = \min_{1\le j\le m}\{c_{ijk}\}$, $\forall i \in [1, n]$ and $l(j) = 0, \forall j \in [1, m]$ and matching M.
2. If M perfect, stop. Otherwise, pick an unmatched vertex $r \in R$. Set $A = \{r\}$, $B = \varnothing$.
3. If $N(A) = B$, update labels by
$$l(r) = l(r) - \delta \quad r \in A$$
$$l(t) = l(t) + \delta \quad t \in B$$
where $\delta = \max_{r\in R, t\in T-B}\{l(r) + l(t) + c_{ijk}\}$.
4. If $N(A) \neq B$, pick $t \in N(A) \setminus B$.
   4a. If t unmatched, $u \to t$ is an augmenting path, then augment M and go to step 2.
   4b. If t is matched to z, extend alternating tree by $A = A \bigcup \{z\}$, $B = B \bigcup \{t\}$, and go to step 3.

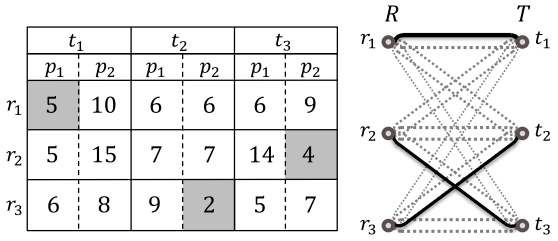
Note: $N(r) = \{t | (r, t) \in G_{\text{e}}\}$, where G_{e} is the equality graph, and $N(A) = \bigcup_{\forall r\in A} N(r)$.

B. The Multiple-Choice Murty's Ranking Algorithm

We modify the partitioning part of the original ranking algorithm. The set of all matchings is partitioned into subsets by removing each vertex and edges of s-th matching. After finding an optimal solution of each subset by Alg. 1, the vertices and the edges of the optimal solution are recovered. In the removing and recovering procedures, p_{ij} edges are removed and recovered all together. The other parts are same as the original version. The time complexity of this algorithm is $O(sp^2(\max\{n,m\})^4)$.



(a) The Hungarian method solves the problems that have only single choice.



(b) The MC Hungarian method allows multiple choice of performing tasks.

Fig. 5: A comparison between the Hungarian and the MC Hungarian methods. Their input cost matrices with output assignments (shaded squares) and corresponding graphs are shown. A bold line indicates an allocation of a robot to a task. The second summation in Eq. 2 and Eq. 3 ensures a task to be performed through only one resource if $p_{ij} > 1$.

C. Exact Algorithm for the P-type problem

1) Algorithm Description: The pseudocode is given in Alg. 2. We denote the s-th assignment before/after penalization as $X^{s-/+}$ and its cost is $c^{s-/+}$. Similarly, Q^s refers the penalization of the s-th assignment. In the first iteration (i.e., $s = 1$), the algorithm computes the best assignment without penalization (c^{1-}). The penalization of the best assignment (Q^1) is computed and added to the cost of the best assignment ($c^{1+} = c^{1-} + Q^1$). Then, the algorithm computes the next-best assignment and compares its unpenalized cost (c^{s-}) with the minimum penalized cost to the previous step ($\min\{c^{1+}, ..., c^{(s-1)+}\}$). The MC Murty's ranking algorithm enables recursive computation of the next-best assignment (line 3). The algorithm repeats each iteration until either of the following conditions are met: when an unpenalized cost is greater or equal to the minimum penalized cost so that $\min\{c^{1+}, ..., c^{(s-1)+}\} \leq c^{s-}$, or the algorithm has enumerated all assignments ($N_\Pi = {}_mP_n \times \Pi_{i,j}^{n,m} p_{ij}$).

Fig. 6 illustrates the terminating condition of the algorithm. The algorithm computes the best ($s = 1$) assignment and its unpenalized cost. Once an assignment is determined, its penalization is computed and added to the unpenalized cost. The algorithm enumerates assignments iteratively and terminates when an unpenalized cost is larger than the current minimum cost including penalization. In the figure, the fourth assignment has a larger unpenalized cost than the cost of the second (current minimum) assignment. Thus, the algorithm terminates after it computes the unpenalized cost of the fourth assignment. Even without penalizations, all subsequent assignments have larger unpenalized costs than the minimum cost. The exact algorithm guarantees optimality but has potentially impractical running-time, as it may enumerate factorial numbers (N_Π) of iterations in the worst case.

D. Optimal Algorithm for the L-type problem

The first phase uses an interior point method (IPM) for linear programming (LP). LP has the optimal solution on a vertex of a polytope. All vertices of a polytope defined by a TU matrix are integer. However, an IPM may produce a fractional solution in which a problem has multiple optimal solutions [36]. In this case, all optimal solutions form an

Algorithm 2 Exact algorithm

Input: An $n \times mp$ cost matrix which is equivalent to a weighted bipartite multigraph $G = (R, T, E)$ where $|R| = n, |T| = m$ and $p_{ij} = p, \forall\{i, j\}$, and penalization functions Q_l for all l.
Output: An optimal assignment X^* and its cost c^*.

1 Initialize $s = 1$
2 **while** $s < N_\Pi$
3 Compute X^s and c^{s-} /∗MC Murty's ranking algorithm∗/
4 **if** $s = 1$
5 Compute Q^s and $c^{s+} = c^{s-} + Q^s$
6 $s = s + 1$
7 **else**
8 **if** $(c^{s-} \geq \min\{c^{1+}, ..., c^{(s-1)+}\})$
9 $X^* = X^{s-1}$ and $c^* = \min\{c^{1+}, ..., c^{(s-1)+}\}$
10 **return** X^*, c^*
11 **else**
12 Compute Q^s and $c^{s+} = c^{s-} + Q^s$
13 $s = s + 1$
14 **end if**
15 **end if**
16 **end while**
17 $X^* = X^s$ and $c^* = \min\{c^{1+}, ..., c^{s+}\}$
18 **return** X^*, c^*

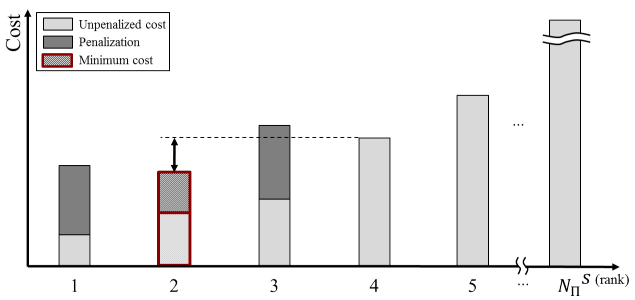


Fig. 6: An illustration of the exact algorithm's terminating condition. When an unpenalized cost is larger than the current minimum cost (including a penalization), at $s = 4$, the algorithm terminates because all subsequent assignments cost more than the minimum cost even without penalizations.

optimal face of the polytope [46]. It is then likely that an IPM converges to an interior point of this optimal face, which is not integer. By using an IPM, we obtain a polynomial running time[7] but lose the integrality of the solution.

If the solution from the first phase is fractional, we use the MC Hungarian method to choose one of the multiple optimal solutions which is integer. The fractional matrix from the first phase is doubly stochastic: the sum of each value in a row and a column is equal to one (e.g., $\left(\begin{smallmatrix}0.9 & 0.1\\ 0.1 & 0.9\end{smallmatrix}\right)$). A doubly stochastic matrix must be produced because the assignment satisfies the mutual exclusion constraint (Eq. 2 - 3), which is same with the definition of the doubly stochastic matrix. Owing to the combinatorial structure of the fractional assignment matrix, each value of the assignment variables can be interpreted as a weight of the likelihood where the variable has the value of one. We use the MC Hungarian method where the input matrix (i.e., cost matrix) is the fractional assignment matrix. The MC Hungarian method for rounding outputs an integer assignment matrix (satisfying the mutual exclusion and the integral constraints) whose cost sum is the maximum. Therefore, the fractional matrix is combinatorially rounded (e.g., $\left(\begin{smallmatrix}1 & 0\\ 0 & 1\end{smallmatrix}\right)$). The pseudocode of the L-type algorithm is not given due to the space limit but same with Alg. 3 except Line 1: it uses an IPM for LP.

The time complexity of the IPM for LP that we used is $O((\max\{2n, nmp\})^3 L)$ [47][8] where L is the bit length of input variables. The Multiple-Choice Hungarian method has $O(p^2(\max\{n, m\})^3)$ complexity. Thus, the overall time complexity is $O((\max\{2n, nmp\})^3 L)$. We use MOSEK optimization toolbox for MATLAB [49], particularly `msklopt` function.

E. Approximation Algorithm for the C-type problem

The pseudocode is given in Alg. 3. The first phase uses an IPM for a convex optimization problem. The objective function must be twice differentiable to use the IPM. In convex programming, the solution could be fractional because not only are there multiple optimal solutions but also it is the unique optimal fractional solution. We also use the MC Hungarian method to round fractional solutions. Since the rounded solution may not be an optimal integer assignment, we provide its performance guarantee.

Theorem 5.1 The performance guarantee of the C-type algorithm is $\max\{Q_{1,\ldots,N_\Pi}\} - \min\{Q_{1,\ldots,N_\Pi}\}$.

Proof. Let $P_{1,\ldots,N_\Pi}$ be assignments of an C-type problem instance and $Q_{1,\ldots,N_\Pi}$ be the penalizations of the assignments. Let K be the upper bound of unpenalized costs, so all assignments can have their unpenalized costs up to K. Without loss of generality, all $P_{1,\ldots,N_\Pi}$ have the unpenalized cost K because there can be multiple assignments that have the same cost sum. Let J be the largest integer solution among $P_{1,\ldots,N_{\Pi-1}}$ and P_{N_Π} be the optimal assignment whose cost is J^*. Then

$$J = \mathrm{K} + \max\{Q_{1,\ldots,N_{\Pi-1}}\},$$

and we define

$$J^* = \mathrm{K} + \epsilon + Q_{N_\Pi}$$

where ϵ is a nonnegative real number.
Since $J \geq J^*$, $\max\{Q_{1,\ldots,N_{\Pi-1}}\} \geq Q_{N_\Pi}$ which means $Q_{N_\Pi} = \min\{Q_{1,\ldots,N_\Pi}\}$. Then

$$\begin{aligned} &J - J^* = J - (\mathrm{K} + \epsilon + Q_{N_\Pi}) = J - \mathrm{K} - \epsilon - Q_{N_\Pi} \\ &\leq J - \mathrm{K} - Q_{N_\Pi} = \max\{Q_{1,\ldots,N_{\Pi-1}}\} - Q_{N_\Pi} \\ &= \max\{Q_{1,\ldots,N_{\Pi-1}}\} - \min\{Q_{1,\ldots,N_\Pi}\}. \end{aligned}$$

Since $\max\{Q_{1,\ldots,N_\Pi}\} \geq \max\{Q_{1,\ldots,N_{\Pi-1}}\}$,

$$J - J^* \leq \max\{Q_{1,\ldots,N_\Pi}\} - \min\{Q_{1,\ldots,N_\Pi}\}.$$

□

The significance of the performance guarantee can differ depending on the viewpoint. Clearly, the importance of the performance guarantee depends on the precise form (and the value) of the penalization function. Thus, the guarantee is less important when its particular value is very large. However, the guarantee is significant in the sense of its independence from assignments. Regardless of which assignment is computed, the guarantee solely depends on the penalization function. We have seen that the hardness of the problem has a spectrum depending on the form of the penalization function (in Section IV-E). Likewise, the importance of the guarantee also has a spectrum depending on the form of the penalization function. If a penalization function has bounded changes with respect to its input, the difference between an approximation and an optimal value would not be very large, so the (practical) importance of the guarantee is more significant. We may construct a penalization function that satisfies a certain condition (e.g., limiting the function's change), and increase the significance of the performance guarantee.

The time complexity of an IPM for a convex optimization problem is $O((\max\{2n, nmp\})^{3.5} L)$ [50]. Thus, the overall time complexity is $O((\max\{2n, nmp\})^{3.5} L)$. We use MOSEK `mskscopt` function for the optimization phase. Table IV summarizes the problems and algorithms.

Algorithm 3 The C-type algorithm

Input: An $n \times mp$ cost matrix which is equivalent to a weighted bipartite multigraph $G = (R, T, E)$ where $|R| = n, |T| = m$, $p_{ij} = p, \forall\{i, j\}$, and convex penalization functions Q_l for all l.
Output: An optimal assignment X^* and its cost c^*.

1 Compute $X^*_{\mathbb{R}^+}$ and $c^*_{\mathbb{R}^+}$ /* IPM for CP */
2 Compute $\hat{X}^*_{\mathbb{Z}^+}$ and $\hat{c}^*_{\mathbb{Z}^+}$ /* MC Hungarian method */
3 $X^* = \hat{X}^*_{\mathbb{Z}^+}$, $c^* = \hat{c}^*_{\mathbb{Z}^+}$
4 **return** X^*, c^*

VI. Extension: Modeling Synergies

The modeling method presented in this paper also can be applied to modeling negative penalization, namely synergies.

[7]The simplex method does not produce a fractional solution because it visits only vertices which lie on integer points. However, the simplex method is not a polynomial-time algorithm because it could visit exponentially many vertices.

[8]The state of the art is [48] whose complexity is $O(\frac{\max(2n,nmp)^3}{\ln\max(2n,nmp)}L)$.

TABLE IV: A summary of the problems and algorithms.

Problem:		P-type	C-type	L-type
Objective function		Polynomial-time computable	Convex	Linear
Complexity class		NP-hard	NP-hard	P
Algorithm	Step I	Iterative method (Ranking alg. + MC Hungarian)	Linear programming (IPM)	Convex optimization (IPM)
	Step II		Rounding (MC Hungarian method)	
Overall complexity		$O({}_mP_n \times \Pi_{i,j}^{n,m} p_{ij})$	$O((\max\{2n, nmp\})^{3.5}L)$	$O((\max\{2n, nmp\})^{3}L)$
Performance guarantee		Optimal	$\max\{Q_{1,\ldots,N_\Pi}\} - \min\{Q_{1,\ldots,N_\Pi}\}$	Optimal

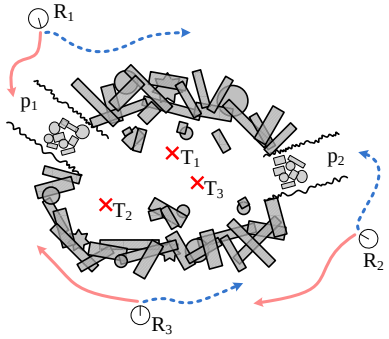


Fig. 7: An example in which both synergy and resource contention occur while robots perform individual tasks. Tasks are inside of the cluttered disaster site, and each robot can choose one path between p_1 and p_2 to reach the tasks. The robots who choose the same path become to push debris together.

Some synergistic effects make interrelations among costs and can be modeled by a concise representation like $Q(\cdot)$ in Eq. 1. Fig. 7 shows an example. Robots are located outside of a cluttered disaster site (e.g., a collapsed building or an explosion site). The robots have individual tasks (e.g., monitoring survivors until assistance arrives) inside of the site. There are multiple paths to reach the tasks, but they are covered with debris. Robots should push their ways through the debris. The goal is to minimize the total traveling time to reach their destinations. Even though the robots perform their own tasks, there can be collaborations among robots on the same resource such as pushing debris together on the same path. Robots are neither tightly coupled to make coalitions nor required to have pre-coordination. Collaborations make synergistic effects, but more robots on the same resource produce more resource contention so the number of robots on shared resources is needed be determined optimally. Note that the exact algorithm is not applicable when negative costs may be incurred by synergistic effects. In this case, the negative costs could make any subsequent penalized cost c^{s+} less than the current optimal assignment c^*, so the algorithm is not able to decide whether to terminate before it enumerates all assignments. However, the other two algorithms are still applicable when synergistic effects are modeled as a linear or a convex function.

Applications are not limited to physical interactions. When robots explore to search for their individual targets, one robot can connect to a network and tell other robots in the network what it detects if the detected target is not the robot's target. The collaboration reduces searching time and its effectiveness is amplified as more robots participate by connecting to the same network. However, this synergistic effect would be deteriorated as more robots connect to the same network because the communication bandwidth is limited.

VII. Experiments

We demonstrate that the exact algorithm works well and returns a result in reasonable time for practically sized cost matrices. The L-type and C-type algorithms produce solutions quickly for even larger matrices. We implemented all the algorithms in MATLAB. The solution quality is measured by a ratio of an approximated solution to an optimal solution $\eta = \frac{c'^*}{c^*} \geq 1$. We assume that $n = m$ and $p_{ij} = p, \forall\{i.j\}$ for all the experiments. If the p_{ij}s are not identical, then we add dummy edges with infinite cost. As we detail next, both randomly generated problem instances and instances based on real-world scenarios were used to validate the algorithms. Also, they are demonstrated with physical robots in small-scale experiments in our laboratory. First, we provide some detail on the particular penalization models used.

A. Penalization Functions

A penalization function models the interference incurred in a particular environment, and should consider the specific aspects of the robots and environment. Simple examples based on a factorization that adds costs as a function of the number of robots utilizing a resource, include models in the form of linear and an convex quadratic functions. Following the form in Eq. 6, let those penalization models for use of the l-th resource be

$$Q_l(n_l) = \begin{cases} \beta_{\mathrm{L}} n_l + \beta'_{\mathrm{L}} & n_l \geq 1 \\ 0 & \text{otherwise,} \end{cases} \tag{13}$$

and

$$Q_l(n_l) = \begin{cases} \beta_{\mathrm{C}} n_l^2 + \beta'_{\mathrm{C}} n_l + \beta''_{\mathrm{C}} & n_l \geq 1 \\ 0 & \text{otherwise,} \end{cases} \tag{14}$$

where β_{L}, β'_{L}, β_{C}, β'_{C}, and β''_{C} are constants.

For the multi-vehicle transportation scenario, we used the classic flow model developed by domain experts to quantify traffic congestion [51]. Many models have been proposed in the literature that compute a traffic speed v (m/s) according to traffic density ρ (vehicle/m). We let $\rho = n_l$ because we

only consider an instantaneous assignment problem. We use an exponential model for our application that travel time (sec) is used as cost

$$Q_l(n_l) = \begin{cases} \frac{d_l}{v_{\mathrm{f}}\left[1-\exp\left\{-\frac{\lambda}{v_{\mathrm{f}}}\left(\frac{1}{n_l}-\frac{1}{\rho_j}\right)\right\}\right]} & \rho_j \geq n_l \\ +\infty & \text{otherwise,} \end{cases} \quad (15)$$

where v_{f} is the free flow speed (when $n_l = 0$), ρ_j is the jam density, and λ is the slope of the headway-speed curve[9] at $v = 0$, and d_l is the length of a resource that could be shared with other robots such as a passage.

We also suggest an idea of designing wireless network congestion models. Throughput of per client (Mbps) with respect to the number of client (n) is shown in [52, Figure 4 and Table 4]. From the data, a quadratic convex function is fitted where the function represents the relationship between the number of clients (robots) and the unit transmission time (reciprocal of throughput). This is a rough modeling based on the data sheet, and domain experts may build more accurate models. The network congestion example suggested in Section III-D can use this model to minimize the total completion time including data transmission time.

B. Random Problem Instances

A uniform cost distribution ($\mathcal{U}(0, 60)$) is used to test the algorithms. The penalization function Eq. 14 is used for the exact and C-type algorithms, and Eq. 13 is used for the L-type algorithm ($\beta_{\mathrm{L}} = \beta_{\mathrm{C}} = 1$ and $\beta'_{\mathrm{L}} = \beta'_{\mathrm{C}} = \beta''_{\mathrm{C}} = 0$). With fixed $p = 5$, the size of the cost matrix (n) increases from 5 to 100 at intervals of 5 (10 iterations for each n). Fig. 8 and Table V show running times and solution qualities of our algorithms. We also compare them with conventional methods such as branch and bound (BB) and randomized rounding. [10] We do not report results for the BB method on the C-type because the running-time is impractical and prohibitive even when the instance size is small (e.g., $n = 10$). We display results in multiples of 10, owing to the space limitations.

The exact algorithm finds an optimal assignment in a reasonable time on small instances ($n \leq 8$); even a small problem has a huge search space (e.g., $N_\Pi = 375,000$ when $n = 5$ and $p = 5$). The L-type and C-type algorithms quickly find solutions even if n is large. Our methods are faster than the BB methods and similar to the randomized rounding methods. However, the methods we propose are the only to have polynomial running time. The solution qualities of the C-type is better than the BB method (also the proposed algorithms have a performance guarantee). The BB methods find an optimal solution but have exponential worst-case time complexity. These properties are shown in the results: the running time is longer but the solution quality is optimal ($\eta = 1$). The randomized rounding methods are faster because they have a single random-rounding step, but the randomness makes their solution quality bad.

[9]The ratio of (infinitesimal) velocity change over (infinitesimal) headway change.

[10]If a penalization function is linear, network flow algorithms can have a better bound (e.g., $O(pn^3)$) than the L-type algorithm. For convex penalization functions, network flow algorithms have no way to deal with nonlinear objective functions [53]. One of the possible ways that leads to global optimality for the problems with nonlinear objective functions is to use a nonlinear programming formulation. However, the formulation may not produce integer solutions so BB and randomized rounding are used.

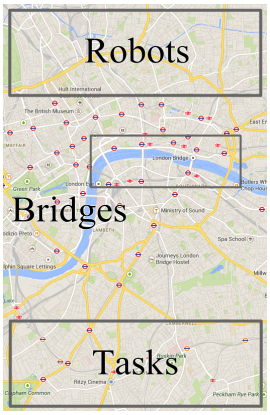


Fig. 9: Robots and tasks are located across five bridges. n robots and tasks are uniformly distributed in the upper and lower boxes, respectively.

C. Multi-Vehicle Traffic Transportation Problems

A multi-vehicle transportation problem is used as a representative real-world application for our algorithms. We assume that n homogeneous robots and n tasks are distributed across p bridges in an urban area as shown in Fig. 9. The robots and the tasks are uniformly distributed within the boundaries. Distances from the robots to the tasks though the bridges are collected by using the Google Directions API [54]. The raw data are in meters (m) but converted to time (sec) according to v_{f}. Thus, the cost is travel time without congestion and penalized by the increased time owing to congestion. With fixed $p = 5$, n increases from 5 to 50 (3 to 9 for the exact algorithm). Other parameters are set as follows:

$$\begin{aligned} v_{\mathrm{f}} &= 16.67 \text{ m/s}, \\ d_l &= \{500,\ 300,\ 250,\ 400,\ 200\} \text{ m}, \\ \rho_{jl} &= \{120,\ 80,\ 70,\ 90,\ 80\} \text{ robot/choice} \\ \lambda_l &= \{0.1389,\ 0.1667,\ 0.1528,\ 0.1944,\ 0.1389\} \text{ s}^{-1} \end{aligned}$$

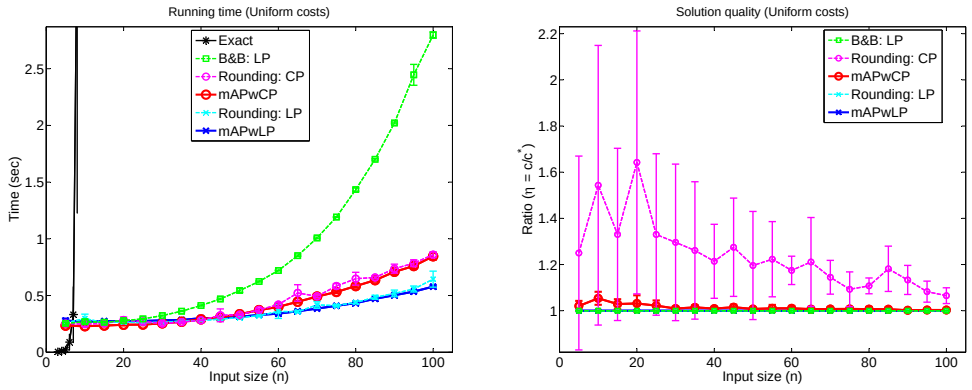
where $l = 1, ..., 5$. The parameters reflect the characteristics of the real-world multi-vehicle transportation problem.

We use Eq. 15 for the exact algorithm. However, our implementations for Alg. 3 do not allow a complex exponential objective function like Eq. 15. Thus, we approximate Eq. 15 with a linear and a convex quadratic function such as Eq. 13 and 14. An example of the approximations is shown in Fig. 10(a). The quality of the approximation is measured by the sum of squared residuals. Table VI shows the approximation results of all penalization functions of the five bridges.

TABLE VI: Penalization function approximation results of the five bridges.

	Residuals				
Fitting type	Bridge 1	Bridge 2	Bridge 3	Bridge 4	Bridge 5
Linear	301.2	408.8	531.1	389.6	283.8
Convex	22.92	70.61	124.1	52.28	49.21

The Fig. 10(b) shows solution qualities when the approximated functions are used. For each instance, we compute an optimal assignment with the exact algorithm when the original



(a) Running times (b) Solution qualities

Fig. 8: Running time and solution quality of random instance. (a) The L-type and C-type algorithms are slightly faster than the rounding method whose worst case running time is exponential. (b) The C-type has better solution quality than the rounding method.

TABLE V: Running time and solution quality of random instances.

(a) The exact algorithm.

	n	3	4	5	6	7	8	9
Running time (sec)	Mean	0.0041	0.0115	0.0208	0.0894	0.3324	3.8327	95.1580
	Std. dev.	0.0043	0.0047	0.0144	0.0721	0.2467	2.6072	93.7848

(b) The L-type and C-type algorithms and existing methods.

		n	10	20	30	40	50	60	70	80	90	100
L-type	Running time (sec)	Mean	0.2689	0.2743	0.2812	0.3003	0.3127	0.3406	0.3848	0.4333	0.5029	0.5786
		Std. dev.	0.0040	0.0091	0.0037	0.0064	0.0062	0.0046	0.0059	0.0081	0.0091	0.0067
	Quality (η)	Mean	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
		Std. dev.	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
C-type	Running time (sec)	Mean	0.2296	0.2421	0.2546	0.2898	0.3354	0.4005	0.4908	0.5835	0.7094	0.8462
		Std. dev.	0.0051	0.0068	0.0055	0.0044	0.0071	0.0116	0.0101	0.0150	0.0187	0.0311
	Quality (η)	Mean η	1.0537	1.0314	1.0093	1.0092	1.0076	1.0104	1.0074	1.0067	1.0020	1.0030
		Std. dev.	0.0282	0.0353	0.0086	0.0078	0.0042	0.0105	0.0089	0.0100	0.0016	0.0020
B&B: LP	Running time (sec)	Mean	0.2688	0.2787	0.3235	0.4121	0.5442	0.7207	1.0080	1.4342	2.0219	2.7971
		Std. dev.	0.0160	0.0084	0.0040	0.0054	0.0162	0.0037	0.0070	0.0137	0.0080	0.0277
	Quality (η)	Mean η	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
		Std. dev.	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
Rounding: LP	Running time (sec)	Mean	0.2972	0.2662	0.2644	0.2797	0.3051	0.3598	0.4173	0.4392	0.5186	0.6446
		Std. dev.	0.0386	0.0146	0.0072	0.0053	0.0067	0.0619	0.0305	0.0121	0.0295	0.0706
	Quality (η)	Mean	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
		Std. dev.	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
Rounding: CP	Running time (sec)	Mean	0.2477	0.2841	0.2538	0.2816	0.3310	0.4203	0.4930	0.6485	0.7368	0.8602
		Std. dev.	0.0208	0.0266	0.0150	0.0041	0.0075	0.0164	0.0104	0.0564	0.0398	0.0249
	Quality (η)	Mean η	1.5435	1.6424	1.2960	1.2140	1.1957	1.1746	1.1448	1.1075	1.1332	1.0651
		Std. dev.	0.6057	0.5695	0.3391	0.1603	0.2342	0.0614	0.0731	0.0342	0.0630	0.0342

model is used. Then we compare it to the assignments when the approximated functions are used. As a result, the solution qualities are good (less than 1.024) so those approximations are acceptable.

We compare our method with the optimal assignment problem formulation that does not include additional costs incurred by resource contention (but the additional costs occur when robots perform tasks). We use Alg. 3 with the convex quadratic penalization function that we approximated from Eq. 15. For the comparison, we use the Hungarian method to compute an optimal assignment without considering penalization. Once an assignment is obtained, we use the same convex quadratic penalization function to the additional costs associated with the assignment. Table VII show the results (10 iterations for each n). The results show that incorporating resource contention into this transportation problem is crucial to achieve global optimality.

Next, we compare our algorithms with the existing methods that use our models. Fig. 11 and Table VIII show the results (10 iterations for each n). The results are similar to the random instance case. This experiment shows that our algorithms can model realistic scenarios of robotic applications.

TABLE VII: Differences of the total cost sums between the C-type and the Hungarian method in the multi-vehicle transportation problem.

n	5	10	15	20	25	30	35	40	45	50
Cost difference (sec)	5.5311	15.7429	14.0122	20.4954	21.3051	40.0226	45.8332	49.9823	61.9301	83.9979

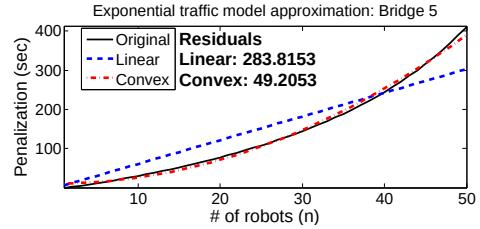
TABLE VIII: Running time and solution quality of the multi-vehicle transportation problem.

(a) The exact algorithm.

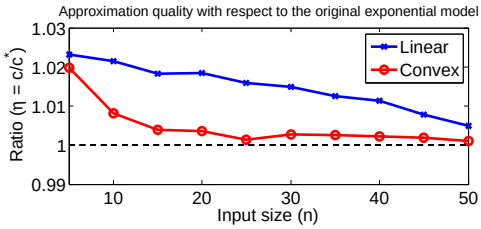
	n	3	4	5	6	7	8	9
Running time (sec)	Mean	0.0044	0.0088	0.0395	0.1298	0.5913	4.2897	126.0774
	Std. dev.	0.0015	0.0045	0.0290	0.1171	0.8772	6.8811	202.1757

(b) The L-type and C-type algorithms.

		n	5	10	15	20	25	30	35	40	45	50
L-type	Running time (sec)	Mean	0.2634	0.2627	0.2678	0.2710	0.2753	0.2855	0.2846	0.2935	0.3017	0.3118
		Std. dev.	0.0069	0.0054	0.0054	0.0073	0.0032	0.0108	0.0072	0.0037	0.0024	0.0033
	Quality (η)	Mean	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
		Std. dev.	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
C-type	Running time (sec)	Mean	0.2293	0.2232	0.2307	0.2417	0.2496	0.2589	0.2779	0.2976	0.3170	0.3515
		Std. dev.	0.0159	0.0034	0.0059	0.0058	0.0081	0.0056	0.0076	0.0133	0.0045	0.0040
	Quality (η)	Mean	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
		Std. dev.	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000

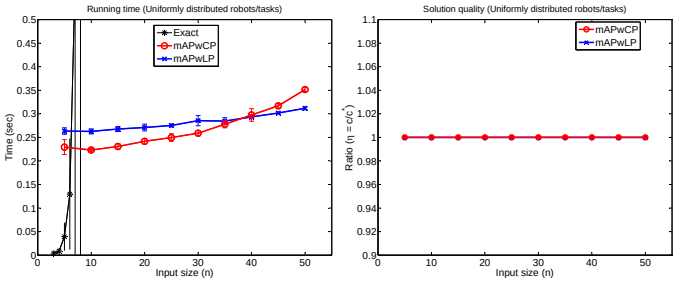


(a) An example of approximations



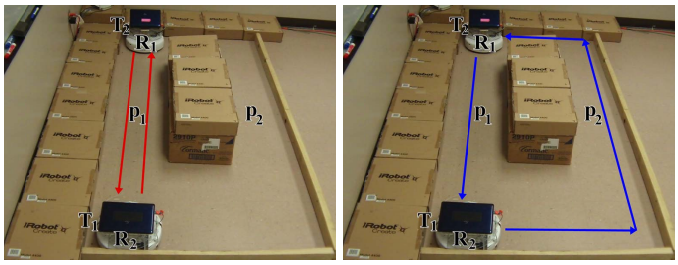
(b) Solution quality

Fig. 10: Approximations of a complex nonlinear function to simple functions for practical implementations. (a) We approximate a complex exponential function with a linear and a convex quadratic function. (b) Solution qualities when the approximated functions are used for all five bridges.



(a) Running times (b) Solution qualities

Fig. 11: Running time and solution quality of the multi-vehicle transportation problem. (a) The L-type and C-type algorithms quickly produce solutions. (b) The qualities are close to one for both algorithms.



(a) Both R_1 and R_2 use p_1. (b) R_1 uses p_1, and R_2 uses p_2.

Fig. 12: Two cases of resource use by two mobile robots. (a) Robots use the same resource so that interference is occurred. (b) Robots use different resources to avoid the interference.

D. Physical Robot Experiment

We demonstrate that our method achieves global optimality even interference is not negligible. Fig. 12 shows the experimental setting. Two iRobot Creates (R_1 and R_2) have tasks of visiting the other robot's position on the opposite side of environment (T_1 and T_2). There are two passages to reach their destinations (shown as p_1 and p_2 in the figure). We use travel time as the cost and Eq. 15 as the penalization function. We compute the assignment with Alg. 2. We assume that R_1 and R_2 are identical. Space constraints and the data from the previous experiments forced us to omit reporting quantitative results.

When the robots move through the shortest path to the destination to attain the minimum travel time, they choose the same passage p_1 (Fig. 12a). However, this choice incurs interference between the robots. When the assignment is penalized, the best assignment is changed to the other assignment: R_1 uses p_1 and R_2 uses p_2 (Fig. 12b). When the robots use the same resource p_1, it takes 102 seconds to complete the tasks whereas the interference-free assignment takes 87 seconds.

VIII. Conclusion

In this paper, we define the mAPwP problems and show that the P-type and C-type problems are NP-hard, and the L-type problem is in P. We develop the Multiple-Choice Hungarian method, which is a generalization of the original method, to allow multiple choices of performing tasks. We present an exact algorithm that generalizes Murty's ranking algorithm to solve the multiple-choice problem, which employs the MC Hungarian method as a subroutine. In addition, we propose two polynomial-time algorithms for the L-type and C-type problems. The L-type algorithm produces an optimal assignment, and the C-type algorithm computes a solution with bounded quality. In the experiments, we model interference among robots by introducing several penalization functions; the results show that the exact algorithm finds an optimal solution, and the L-type and C-type algorithms produce an optimal and a high-quality solution quickly. We also conduct physical robot experiments to show how resource contention aggravates optimality in practice and that the proposed algorithm achieves global optimality when an interference model is included.

Since the algorithms use the centralized approach where a central unit computes an optimal assignment and distributes the assignment to other robots, their use could be restrictive if central computation and global communication are not possible or the expense of central computation and global communication is prohibitive. We are interested in developing decentralized versions of the algorithms along with modeling contention on other types of (e.g., non-physical) resources.

APPENDIX

Here, we show the transformation of a nonseparable C-type problem to a separable problem, and show how to check the totally unimodularity of the transformed problem.

Suppose that $n = m = 3$, $p_{ij} = 2, \forall\{i, j\}$, and a convex quadratic penalization function $Q(\cdot)$ in Eq. 1 is

$$\begin{aligned} &Q(x_{111}, x_{112}, \ldots, x_{331}, x_{332}) \\ &= (x_{111} + x_{121} + x_{131} + x_{211} + x_{221} + x_{231} \\ &\quad + x_{311} + x_{321} + x_{331})^2 + (x_{112} + x_{122} + x_{132} \\ &\quad + x_{212} + x_{222} + x_{232} + x_{312} + x_{322} + x_{332})^2 \end{aligned} \tag{16}$$

which is Eq. 14 where $\beta_{\mathrm{C}} = 1$ and $\beta'_{\mathrm{C}} = \beta''_{\mathrm{C}} = 0$. Eq. 16 can be written as

$$Q(\cdot) = {y_1}^2 + {y_2}^2 \tag{17}$$

where

$$\begin{aligned} y_1 &= x_{111} + x_{121} + \ldots + x_{321} + x_{331}, \\ y_2 &= x_{112} + x_{122} + \ldots + x_{322} + x_{332}. \end{aligned}$$

Thus, additional constraints

$$\begin{aligned} x_{111} + x_{121} + \ldots + x_{321} + x_{331} - y_1 = 0, \\ x_{112} + x_{122} + \ldots + x_{322} + x_{332} - y_2 = 0 \end{aligned}$$

are added to Eq. 2–5. Therefore, A_{N} in Eq. 12 is

$$A_{\mathrm{N}} = \left[\begin{array}{ccccccccc} 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \end{array} \right.$$

$$\left. \begin{array}{ccccccccc} 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \end{array} \right],$$

and this is TU by definition. Since a TU A_{N} does not always guarantee a TU A_{SP}, A_{SP} also has to be checked. A_{SP} is not TU because at least one of its submatrix has a determinant other than -1, 0, or 1 (e.g., $\det([1\ 1\ 0; 1\ 0\ 1; 0\ 1\ 1]) = -2$). Therefore, this C-type problem instance does not belong to the polynomial-time solvable class of the C-type problem.

Acknowledgment

This work was supported by National Science Foundation (award numbers CMMI-1100579 and IIS-1302393).

References

[1] C. Nam and D. A. Shell, "Assignment algorithms for modeling resource contention and interference in multi-robot task-allocation," in *Proc. of IEEE Int. Conf. on Robotics and Automation*, 2014.

[2] K. Low, J. Dolan, and P. Khosla, "Adaptive multi-robot wide-area exploration and mapping," in *Proc. of Int. Joint Conf. on Autonomous Agents and Multiagent Systems-Volume 1*, 2008, pp. 23–30.

[3] S. Kazadi, A. Abdul-Khaliq, and R. Goodman, "On the convergence of puck clustering systems," *Robotics and Autonomous Systems*, vol. 38, no. 2, pp. 93–117, 2002.

[4] B. Duncan and R. Murphy, "Autonomous capabilities for small unmanned aerial systems conducting radiological response: Findings from a high-fidelity discovery experiment," *Journal of Field Robotics*, 2014.

[5] L. Parker, "Alliance: an architecture for fault tolerant multirobot cooperation," *IEEE Trans. on Robotics*, vol. 14, pp. 220–240, 1998.

[6] B. Werger and M. Matarić, "Broadcast of local eligibility for multi-target observation," *Dist. Autonomous Robotic Syst. 4*, pp. 347–356, 2001.

[7] S. Botelho and R. Alami, "M+: a scheme for multirobot cooperation through negotiated task allocation and achievement," in *Proc. of IEEE Int. Conf. on Robotics and Automation*, 1999, pp. 1234–1239.

[8] M. B. Dias, "Traderbots: A new paradigm for robust and efficient multirobot coordination in dynamic environments," Ph.D. dissertation, Carnegie Mellon University, 2004.

[9] B. Gerkey and M. Matarić, "Sold!: Auction methods for multi-robot coordination," *IEEE Trans. on Robotics*, vol. 18, pp. 758–768, 2002.

[10] D. Goldberg and M. Matarić, "Interference as a Tool for Designing and Evaluating Multi-Robot Controllers," in *Proc. of AAAI Nat. Conf. on Artificial Intell.*, 1997, pp. 637–642.

[11] D. Goldberg, "Evaluating the dynamics of agent-environment interaction," Ph.D. dissertation, University of Southern California, 2001.

[12] D. Shell and M. Matarić, "On foraging strategies for large-scale multi-robot systems," in *Proc. of IEEE/RSJ Int. Conf. on Intelligent Robots and Syst.*, 2006, pp. 2717–2723.

[13] B. Gerkey and M. Matarić, "A formal analysis and taxonomy of task allocation in multi-robot systems," *Int. J. of Robotics Research*, vol. 23, pp. 939–954, Sept. 2004.

[14] T. Dahl, M. Matarić, and G. Sukhatme, "Multi-robot task-allocation through vacancy chains," in *Proc. of IEEE Int. Conf. on Robotics and Automation*, 2003, pp. 2293–2298.

[15] J. Guerrero and G. Oliver, "Physical interference impact in multi-robot task allocation auction methods," in *Proc. of IEEE Workshop on Dist. Intell. Syst.: Collective Intell. and Its Apps.*, 2006, pp. 19–24.

[16] H. Choi, L. Brunet, and J. How, "Consensus-based decentralized auctions for robust task allocation," *IEEE Trans. on Robotics*, vol. 25, no. 4, pp. 912–926, 2009.

[17] G. Pini, A. Brutschy, M. Birattari, and M. Dorigo, "Interference reduction through task partitioning in a robotic swarm," in *In Proc. of Int. Conf. on Informatics in Control, Automation and Robotics*, 2009, pp. 52–59.

[18] J. Alonso-Mora, M. Rufli, R. Siegwart, and P. Beardsley, "Collision avoidance for multiple agents with joint utility maximization," in *Proc. of IEEE Int. Conf. on Robotics and Automation*, 2013, pp. 2833–2838.

[19] J. Yu and S. LaValle, "Planning optimal paths for multiple robots on graphs," in *Proc. of IEEE Int. Conf. on Robotics and Automation*, 2013, pp. 3612–3617.

[20] L. He and J. van den Berg, "Meso-scale planning for multi-agent navigation," in *Proc. of IEEE Int. Conf. on Robotics and Automation*, 2013, pp. 2839–2844.

[21] T. Du, E. Li, and A.-P. Chang, "Mobile agents in distributed network management," *Commun. of the ACM*, vol. 46, pp. 127–132, 2003.

[22] A. Kumar, B. Faltings, and A. Petcu, "Distributed constraint optimization with structured resource constraints," in *Proc. of Int. Conf. on Autonomous Agents and Multiagent Syst.*, 2009, pp. 923–930.

[23] T. Matsui, H. Matsuo, M. Silaghi, K. Hirayama, and M. Yokoo, "Resource constrained distributed constraint optimization with virtual variables," in *Proc. of AAAI Conf. on Artificial Intell.*, 2008, pp. 120–125.

[24] F. Tang and L. Parker, "A complete methodology for generating multi-robot task solutions using asymtre-d and market-based task allocation," in *Proc. of IEEE Int. Conf. on Robotics and Automation*.

[25] S. Sariel, T. Balch, and N. Erdogan, "Incremental multi-robot task selection for resource constrained and interrelated tasks," in *Proc. of IEEE/RSJ Int. Conf. on Intelligent Robots and Syst.*

[26] P. Shiroma and M. F. M. Campos, "Comutar: A framework for multi-robot coordination and task allocation," in *Proc. of IEEE/RSJ Int. Conf. on Intelligent Robots and Syst.*

[27] Y. Zhang and L. Parker, "Considering inter-task resource constraints in task allocation," *Autonomous Agents and Multi-Agent Systems*, vol. 26, pp. 389–419, 2013.

[28] S. Chien, A. Barrett, T. Estlin, and G. Rabideau, "A comparison of coordinated planning methods for cooperating rovers," in *the International Conference on Autonomous Agents*, 2000, pp. 100–101.

[29] S. A. Hong and G. Gordon, "Decomposition-based optimal market-based planning for multi-agent systems with shared resources," in *International Conference on Artificial Intelligence and Statistics*, vol. 15, 2011, pp. 351–360.

[30] L. Luo, N. Chakraborty, and K. Sycara, "Multi-robot assignment algorithm for tasks with set precedence constraints," in *Proc. of IEEE Int. Conf. on Robotics and Automation*.

[31] ——, "Distributed algorithm design for multi-robot task assignment with deadlines for tasks," in *Proc. of IEEE Int. Conf. on Robotics and Automation*, 2013, pp. 3007–3013.

[32] H. Lenstra, "Integer programming with a fixed number of variables," *Math. of Operations Research*, pp. 538–548, 1983.

[33] R. Kannan, "Minkowski's convex body theorem and integer programming," *Math. of Operations Research*, vol. 12, no. 3, pp. 415–440, 1987.

[34] D. Dadush, C. Peikert, and S. Vempala, "Enumerative lattice algorithms in any norm via m-ellipsoid coverings," in *Proc. of IEEE Annu. Symp. on Found. of Comput. Sci.*, 2011, pp. 580–589.

[35] T. Roughgarden, "Routing games," *Algorithmic game theory*, vol. 18, 2007.

[36] T. Dey, A. Hirani, and B. Krishnamoorthy, "Optimal homologous cycles, total unimodularity, and linear programming," *SIAM J. on Computing*, vol. 40, no. 4, pp. 1026–1044, 2011.

[37] V. Kann, "On the approximability of np-complete optimization problems," Ph.D. dissertation, Royal Institute of Technology, 1992.

[38] R. Baldick, "A unified approach to polynomially solvable cases of integer "non-separable" quadratic optimization," *Discrete Appl. Math.*, vol. 61, no. 3, pp. 195–212, 1995.

[39] D. Hochbaum and J. Shanthikumar, "Convex separable optimization is not much harder than linear optimization," *J. of the ACM*, vol. 37, no. 4, pp. 843–862, 1990.

[40] S. Bradley, A. Hax, and T. Magnanti, *Applied mathematical programming*. Addison Wesley, 1977.

[41] P. Camion, "Characterization of totally unimodular matrices," *Proc. of American Mathematical Society*, vol. 16, pp. 1068–1073, 1965.

[42] R. Karp, *Reducibility among combinatorial problems*. Springer, 1972.

[43] K. Murty, "An algorithm for ranking all the assignments in order of increasing cost," *Operations Research*, vol. 16, pp. 682–687, 1968.

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

[45] R. Brualdi and H. Ryser, *Combinatorial matrix theory*. Cambridge University Press, 1991, vol. 39.

[46] L.-H. Zhang, W. Yang, and L.-Z. Liao, "On an efficient implementation of the face algorithm for linear programming," *J. of Computational Math.*, vol. 31, no. 4, pp. 335–354, 2013.

[47] C. Gonzaga, *An algorithm for solving linear programming problems in $O(n^3L)$ operations*. Springer, 1989.

[48] K. Anstreicher, "Linear programming in $o([\frac{n^3}{\ln n}]l)$ operations," *SIAM J. on Optimization*, vol. 9, no. 4, pp. 803–812, 1999.

[49] Mosek, "The mosek optimization software version 6," *Online at http://www.mosek.com*, 2009.

[50] Y. Nesterov, A. Nemirovskii, and Y. Ye, *Interior-point polynomial algorithms in convex programming*. SIAM, 1994, vol. 13.

[51] G. Newell, "Nonlinear effects in the dynamics of car following," *Operations Research*, vol. 9, no. 2, pp. 209–229, 1961.

[52] J. Florwick, J. Whiteaker, A. Amrod, and J. Woodhams, "Wireless lan design guide for high density client environments in higher education," in *Design guide*. Cisco Systems, 2013.

[53] M. S. Bazaraa, J. J. Jarvis, and H. D. Sherali, *Linear programming and network flows*. John Wiley & Sons, 2011.

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

Changjoo Nam is a Ph.D. student in the Department of Computer Science and Engineering at Texas A&M University since Fall 2011. He received his M.S. and B.S. in Electrical Engineering from Korea University. Before starting his study at Texas A&M, he worked at Korea Institute of Science and Technology as a researcher. His current research interest includes multi-robot task allocation (MRTA) problems in dynamic and uncertain environments, as well as mobile robot navigation.

Dylan A. Shell is an assistant professor in the Department of Computer Science and Engineering at Texas A&M University in College Station, Texas. He received his B.Sc. degree in computational & applied mathematics and computer science from the University of the Witwatersrand, South Africa, and his M.S. and Ph.D. in Computer Science from the University of Southern California. He took a position as Postdoctoral Research Associate in the USC Interaction lab in 2008, before joining Texas A&M. His research aims to synthesize and analyze complex, intelligent behavior in distributed systems that exploit their physical embedding to interact with the physical world.