Optimal Market-based Multi-Robot Task Allocation via Strategic Pricing

Lantao Liu
Dept. of Computer Science and Engineering
Texas A&M University, College Station, TX, USA
Email: lantao@cse.tamu.edu

Dylan A. Shell
Dept. of Computer Science and Engineering
Texas A&M University, College Station, TX, USA
Email: dshell@cse.tamu.edu

***Abstract*—Auction and market-based mechanisms are among the most popular methods for distributed task allocation in multi-robot systems. Most of these mechanisms were designed in a heuristic way and analysis of the quality of the resulting assignment solution is rare. This paper presents a new market-based multi-robot task allocation algorithm that produces optimal assignments. Rather than adopting a buyer's "selfish" bidding perspective as in previous auction/market-based approaches, the proposed method approaches auctioning from a merchant's point of view, producing a pricing policy that responds to cliques of customers. The algorithm uses price escalation to clear a market of all its goods, producing a state of equilibrium that satisfies both the merchant and customers. The proposed method can be used as a general assignment algorithm as it has a time complexity ($O(n^3 \lg n)$) close to the fastest state-of-the-art algorithms ($O(n^3)$) but is extremely easy to implement. As in previous research, the economic model reflects the distributed nature of markets inherently: in this paper it leads directly to a decentralized method ideally suited for distributed multi-robot systems.**

I. Introduction

Task allocation and assignment methods have become an important paradigm for coordinating teams of robots. Task allocation, as we study here, is a process that involves each robot estimating the utility expected for performing available tasks, and computing an assignment of tasks to robots in order to maximize the collective benefit. Oftentimes, these matchings are computed repeatedly to allow the robot team to adapt collectively as circumstances change. In such cases both running-time and quality of the resulting robot-to-task matching are important determinants of the team's efficiency.

Most classic optimal assignment algorithms were originally developed by operations researchers and are not well suited to multi-robot task allocation. One typically desires decentralization to enhance the system's tolerance of individual failures and to improve the practical running time. Among the popular optimal assignment algorithms developed by operations researchers, only one algorithm can be said to be conveniently distributable: Bertsekas's famous optimal AUCTION algorithm [2]. The economic interpretation of *decentralized* auctioning and bidding procedures has been favored by many roboticists who, in the last three decades, have developed numerous decentralized assignment approaches by employing auction or auction-like mechanisms. Although the auctioning idea is favored by roboticists, Bertsekas's AUCTION algorithm itself is not widely used in robotics. We believe that this is because it suffers from several important drawbacks, which we analyze and address in this paper. Instead, roboticists have designed their own auction/market-based strategies in order to adapt to various multi-robot architectures. Unfortunately most of the existing techniques fail to achieve optimality (see review in [7]) and few quantify the quality of the resulting allocation.

Nevertheless, the AUCTION algorithm provides a good framework for designing distributable assignment algorithms. We present a new algorithm which inherits the features and merits of the AUCTION algorithm while overcoming some of the drawbacks that impede wider adoption in robotics. The new method improves on the AUCTION algorithm in that the redesign removes the intrinsic data dependence. It reaches the global optimum with a time complexity close to the state-of-the-art (even for centralized) methods and, like most market-based approaches, coordination involves only sub-teams.

The new method, instead of auctioning via a series of "selfish" bids from customers, is a mechanism from the perspective of a merchant. It produces a pricing policy that incentivizes coordination between cliques of agents. This paper shows that it is possible to employ a market-based method suited to distributed implementation for multi-robot systems, and still attain global optimality, and do so with strongly polynomial running time (and iterations).

II. Related Work

We address the classic task assignment problem where robots and tasks are matched by forming a one-to-one mapping. This has been termed the *single-task robots, single-robot tasks, instantaneous assignment problem (ST-SR-IA)* [11].

A. Optimal Assignment Algorithms and Decentralization

Many optimal assignment algorithms have been developed (see reviews [4, 24]). The majority are *primal-dual* methods (details of primal and dual formulations are presented in sections that follow). Important examples include the well-known Hungarian algorithm [15], which solves the matching problem by manipulating a matching bipartite graph. Several shortest augmenting path algorithms (*e.g.*, see [5]) were inspired by the Hungarian algorithm and are also primal-dual methods themselves. These algorithms are capable of solving the problem with $O(n^3)$ time complexity when certain searching techniques and/or auxiliary data structures are employed. But one drawback is the difficulty that arises in producing decentralized variants of these algorithms. This is because the construction of an alternating tree for seeking (shortest) augmenting paths may require searching and labeling the entire graph. Some recent progress in solving the assignment in a distributed way has been made (*e.g.*, see work of [12]) although information from all individual agents is still required even when the computation is decentralized. This defect limits the applicability of these algorithms in distributed settings.

Another important instance is the AUCTION algorithm developed by Bertsekas from the late 70s to early 90s [2]. This algorithm's basic computational primitives (*viz.*, bidding and auctioning) are highly localized rather than relying on queries of global information; this led to its wide-spread recognition as a naturally distributable assignment algorithm. Yet the algorithm itself has not been widely adopted for multi-robot problems. There are several possible reasons. The initial version of the AUCTION algorithm [1] can be naturally decentralized, but it has running time that is not bounded for arbitrary data. Applicability to the robotic domain is hindered by the large and uncertain time (and communication) complexity. Later versions employ some techniques (*e.g.*, ϵ-*scaling* [2]) to remove the data dependence but at the cost of undermining the decentralized nature. For instance, ϵ-*scaling* requires multiple runs of the whole algorithm, causing prohibitive communication. The later versions lost the connection with the original auctioning primitives, forfeiting the economic interpretation. These extensions all increase the implementation complexity significantly. Recently [26] extended AUCTION to use local information over a multi-hop network; unsurprisingly it was based on the initial version.

Another important optimal assignment algorithm is the Dinic-Kronrod algorithm [8]. This algorithm was the first $O(n^3)$ optimal assignment algorithm but is independent of the primal-dual theories. Although Dinic-Kronrod's algorithm is not distributable, its source-sink concept and manipulation (see details in [4]) inspired our process. Although our method has an inferior time complexity ($O(n^3 \lg n)$), it is distributable—amongst other features.

B. Multi-robot Task Allocation Methods

Most classic optimal assignment algorithms that reach the global optimal are *centralized*. A review of the literature shows that a large set of *decentralized* multi-robot task allocation methods employ auctioning mechanisms; representative examples include: the distributed market-based paradigm [6, 13, 21], the auction-based MURDOCH model [10, 9], and cooperative auctions [14, 22]. These techniques have been extended to a wide range of multi-robot scenarios, applied to NP-hard problems like routing, planning, scheduling, or used to address problems where partial knowledge is assumed or local information employed [16, 22]. For these reasons, most often the global optimum will not be obtained (see review in [7]); often the resulting allocation quality remains unknown.

Other methods for task assignment operate by partitioning the robots and/or tasks into subgroups and repeating the process recursively [20, 25, 17], as well as behavior-based or role-based strategies [23]. Recently, we proposed an assignment method independent of the market-based framework [18]. That algorithm was shown to be particularly efficient and it is used for comparison with this proposed algorithm in Section VII.

In sum: despite numerous auction/market-based approaches having been developed for multi-robot task allocation, nearly all are sub-optimal methods. Until this paper, the AUCTION algorithm was the only optimal method in this class.

III. PRELIMINARIES

An assignment $\mathcal{A}$ for a multi-robot system consists of a set of robots R and a set of tasks T. (To simplify presentation of the analysis, we assume $|R| = |T| = n$ although the algorithms described in this paper handle the cases $|R| < |T|$ too.) Given utility matrix $U = (u_{ij})_{n\times n}$, where $u_{ij} : R \times T \to \mathbb{R}^+$ denotes the utility of having robot i perform task j, the objective is to find an assignment permutation $\varphi : \{i\} \to \{j\}$ so that the overall utility $u(\varphi) = \sum_{i=1}^{n} u(i, \varphi(i))$ is maximized.

This problem can be formulated equivalently by a pair of linear programs. The first is the *primal* program, which is a maximization formulation:

$$\begin{aligned} \text{maximize } f(\mathbf{x}) &= \sum_i \sum_j u_{ij} x_{ij}, \\ \text{subject to } & \sum_j x_{ij} = 1, \ \forall i, \\ & \sum_i x_{ij} = 1, \ \forall j, \\ & x_{ij} \geq 0 \quad \forall (i,j). \end{aligned} \tag{1}$$

where an optimal solution eventually is an extreme point of its feasible set (*i.e.*, $x_{ij} \in \{0,1\}$). The constraints $\sum_j x_{ij} = 1$ and $\sum_i x_{ij} = 1$ capture the *mutual exclusion* property that restricts each robot to be assigned to exactly one task and each task to be allocated to a unique robot, respectively.

In auctions two roles are employed: each agent expecting a task acts as a *bidder* and, for any round of bidding, an *auctioneer* determines the result. The auctioneer can be any agent (either an honest bidder or an agent excluded from the current bidding). The utility u_{ij} in such scenarios is interpreted as a *budget* that bidder (agent) i has for object (task) j. One may also think of it as an *affordable price* for the object, as it is the value that the bidder thinks the object is worth.

***Definition** 3.1:* The <u>profit margin</u> m_{ij} is defined as the difference between the budget price u_{ij} and the actual price paid p_j:

$$m_{ij} = u_{ij} - p_j. \tag{2}$$

***Definition** 3.2:* A <u>price escalation</u> operation involves raising prices on *individual* or *groups of items*. Raising individual prices involves adding a $\delta \in \mathbb{R}^+$ (associated with a single object j) to increase only its price to $p_j + \delta$. A many-item price raising operation makes the same price adjustment δ to a set of objects S:

$$p_j = \begin{cases} p_j + \delta, & \forall j \in S, \\ p_j, & \text{otherwise.} \end{cases} \tag{3}$$

It is straightforward to prove that raising the prices associated with tasks does not change the assignment problem since all bidders are affected equally by an escalation operation.

***Definition** 3.3:* The <u>*preferred choice*</u> j^* for bidder i is an item with maximal profit margin:

$$j^* = \text{argmax}_{\forall j}\{m_{ij}\}. \tag{4}$$

Given a choice of different objects, a rational bidder would like to choose an object with the largest profit margin. Assessing choices considers only an individual's local row in the utility matrix and can be thought of as reflecting "selfish utility maximizing behavior." Unfortunately merely combining preferred choices of different bidders is likely to violate the mutual exclusion property and produce an infeasible assignment.

***Definition** 3.4:* An agent's <u>*alternate choices*</u> are the items with profit margins identical to the preferred choice, *i.e.*,

$$\{j'\} = \{j \mid m_{ij} = m_{ij^*}\}, \quad \forall j \neq j^*. \tag{5}$$

***Definition** 3.5:* A <u>*preferred choice transfer*</u> $j^* \rightarrowtail j'$ is the operation that transfers from a current preferred choice j^* to an alternate choice j', where $m_{ij^*} = m_{ij'}$ for bidder i. From a profit incentive point of view, the buyer's utility is invariant to these transfer operations; they can be interpreted merely as a difference in the manner in which we break ties.

IV. The auction Algorithm

The AUCTION algorithm, proposed by Bertsekas [1], is a dual-based algorithm for computing an optimal assignment. It has an interpretation in which a team of agents are seen as bidding on a set of tasks. The algorithm can be understood by examining the *dual* of the primal problem (1), as follows:

$$\begin{aligned} \text{minimize } g(\pi, \mathbf{p}) &= \sum_{i=1}^{n} \pi_i + \sum_{j=1}^{n} p_j, \\ \text{subject to } \pi_i + p_j &\geq u_{ij}, \qquad \forall (i,j). \end{aligned} \tag{6}$$

If the set of prices, $\mathbf{p}$, have been determined, then $\pi_i \geq u_{ij} - p_j$, $\forall j$, and $g(\pi, \mathbf{p})$ is minimized if π_i equals the smallest value in its feasible set. This is equivalent to:

$$\pi_i = \max_{\forall j}\{u_{ij} - p_j\} = \max_{\forall j}\{m_{ij}\}. \tag{7}$$

Along with (7), Program (6) becomes unconstrained:

$$g(\mathbf{p}) = \sum_i \max_{\forall j}\{u_{ij} - p_j\} + \sum_j p_j, \tag{8}$$

meaning that price $\mathbf{p}$ simultaneously becomes the primal optimal and the dual optimal if and only if

$$m_{ij^*} = \max_{\forall j}\{u_{ij} - p_j\} \quad \forall i, \tag{9}$$

where m_{ij^*} denotes the largest profit margin for agent i obtained from its preferred choice j^*.

Each iteration of the AUCTION algorithm consists of two phases: the bidding phase and the assignment phase. To guarantee termination of the algorithm in finite steps, a mechanism called ϵ-*complementary slackness* (ϵ-CS) is used to perturb the problem when a potential stall occurs. This happens if the problem deadlocks because the bidding price increment becomes zero (*e.g.,* as a group of bidders have equal preferences for a set of items).

***Definition** 4.1:* The <u>ϵ-*complementary slackness*</u> condition is defined as:

$$m_{ij^*} \geq \max_{\forall j}\{u_{ij} - p_j\} - \epsilon. \tag{10}$$

In other words, an extra amount of $\epsilon > 0$ is sacrificed from the maximal profit margin in order to beat out other bidders and win the preferred choice.

Bidding phase:
If agent i has a preferred choice that conflicts with other agents (assume they form a set $\mathbb{P}(j)$ for a common item j), it is then engaged in a bidding competition and computes the bidding increment (*i.e.*, bidding prices that this agent would like to further add)

$$\gamma_i = v_i - w_i + \epsilon, \tag{11}$$

where v_i is the profit margin of preferred choice and w_i is the profit margin of a second preferred choice, more formally,

$$v_i = \max_{\forall j}\{m_{ij}\}, \quad w_i = \max_{\forall j \neq j^*}\{m_{ij}\}. \tag{12}$$

Assignment phase:
A single winner will be determined as it will be the one with highest bidding increment

$$i^* = \text{argmax}_{i \in \mathbb{P}(j)} \gamma_i. \tag{13}$$

The price of object j is raised by $\max_{i \in \mathbb{P}(j)} \gamma_i$ and published to all bidders $i \in \mathbb{P}(j)$. Agents that do not win the item, then alter their preferred choices to select other items. ◁

These bidding and assignment phases repeat until no conflict between preferred choices remain. Once this occurs, the result solves the assignment problem.

Note that ϵ must be selected with care to ensure that optimality will not be violated. An appropriate value is obtained by $\epsilon < \frac{1}{n}$ when all u_{ij} are integers. The time complexity of the resulting scheme is then $O(n^3 \lg(\frac{nC}{\epsilon}))$, where $C = \max_{(i,j)}|u_{ij}|$. This shows that AUCTION is a *pseudo-polynomial* algorithm because the running time is influenced by the input data. A polynomial version of the algorithm has been described by using a multi-trial method called ϵ-*scaling*. Unfortunately multi-trial methods undermine the decentralized nature of the approach and, hence, its appropriateness for distributed robot applications — so far as we are aware, the technique has never been used in the robot task assignment literature. Thus, we do not discuss them further in this work; for more detail see [2].

V. The Proposed Algorithm

The pricing mechanism in AUCTION provides a good starting point to develop a market-based optimal assignment algorithm that is naturally distributable but also has strongly polynomial running time. In the AUCTION algorithm, the optimization steps can be interpreted from the perspective of bidders, each behaving selfishly to eliminate competitors from their preferred tasks. In contrast, the proposed algorithm adopts the merchant's point of view. The merchant seeks to clear the market of goods through strategic selection of prices. In a sense, the algorithm has the merchant steer purchasing behavior of the customers.

A. Algorithm Description

To view the proposed algorithm from an economic perspective, suppose that the tasks, $j \in [1, n]$, are articles in a market, only one instance of each article is available, and the agents, $i \in [1, n]$, are customers visiting the market. The utility value u_{ij} still represents the budget of customer i for article j. Assume that each customer is allowed to buy one item in this market and the merchant wants to clear the market (*i.e.,* sell all the articles whilst ensuring each customer obtains an article). Once this is achieved, the market reaches a form of equilibrium.

Just like standard bidders in an auction, the customers make their own independent choices that reflect their local preferences. A merchant is selected in the same way as selecting an auctioneer. Specifically, for articles preferred by multiple customers, the merchant will raise prices by an amount so that the articles are no longer the most profitable choice for some customers. This is in contrast to the AUCTION algorithm in which bidders are responsible for raising the prices. As the merchant instigates escalation, the customers are motivated to consider other articles. This way of guiding the customer's choices reflects the broader philosophy of altering the behavior of people via economic incentivization, as is widely used by policy-makers in the real world [19].

Algorithm V.1 MARKET_STAGE (t)

```
 1: set $\check{T} := \{t\}$, $\check{R} := \check{R} \cup \{i \mid \varphi(i) = t\}$
 2: $sink := 0$, $\bar{i} := 0$, $\bar{j} := 0$; record vectors $\xi := \{0\}$, $\mathbf{\Omega} := \{\varnothing\}$
 3: while $sink = 0$ do
 4:     $\delta := \infty$
 5:     for each $i$ in set $\check{R}$ do
 6:         $l := \text{argmax}_{j \in T \setminus \check{T}}\{u_{ij} - p_j\}$
 7:         $v := u_{i\varphi(i)} - p_{\varphi(i)}$,  $w := u_{il} - p_l$
 8:         if $v - w < \delta$ then
 9:             $\delta := v - w$
10:             $\bar{i} := i$,  $\bar{j} := l$
11:     for each $j$ in set $\check{T}$ do
12:         $p_j := p_j + \delta$
13:     $\Omega_{\bar{i}} := \Omega_{\bar{i}} \cup \{\bar{j}\}$,  $\xi_{\bar{j}} := \bar{i}$
14:     if $\{i \mid \varphi(i) = sink\} = \varnothing$ then
15:         $sink := \bar{j}$
16:     else
17:         $\check{T} := \check{T} \cup \{\bar{j}\}$, $\check{R} := \check{R} \cup \{i \mid \varphi(i) = \bar{j}\}$

18: $j_\varphi := 0$, $j_p := sink$
19: while $j_\varphi \neq t$ do
20:     $j_\varphi := \varphi(\bar{i})$, $\varphi(\bar{i}) := j_p$
21:     if $j_\varphi \neq t$ then
22:         $\bar{i} := \xi_{j_\varphi}$
23:         $j_p := j_\varphi$
```

Algorithm V.2 MARKET_MAIN

```
1: for $j := 1$ to $n$ do
2:     while $|\{i \mid \varphi(i) = j, \ \forall i\}| > 1$ do
3:         $MARKET\_STAGE(j)$
```

The essence of the proposed algorithm is that the merchant employs a conservative means to stimulate the market so that, after each stage, one previously ignored item garners sufficient attention to be sold owing to the appropriate adjustments in price. For an arbitrary item t, if multiple customers select it as their preferred choice, these customers form a clique $\check{R} = \mathbb{P}(t)$ with a common preference conflict for article t. In the case where there are multiple conflicted articles, they form a set $\check{T} = \{t\}$. A virtual merchant is randomly selected from either $\check{R}$ or $R \setminus \check{R}$. Customers $i \in \check{R}$ describe their *budget margins* $v_i - w_i$ to the merchant, where $v_i = \max_{j \in \check{T}}\{m_{ij}\}$ is the maximal profit margin for articles currently conflicted (in set $\check{T}$), and $w_i = \max_{j \in T \setminus \check{T}}\{m_{ij}\}$ is the maximal profit margin for articles that are not currently conflicted (not in set $\check{T}$). If the prices of currently conflicted articles escalate more than a customer's budget margin, that customer will lose interest in the articles and switch his preference to others. Having obtained budget margin information, the merchant computes the minimum price increment $\delta = \min_{i \in \check{R}}\{v_i - w_i\}$ and increases the prices of all articles in $\check{T}$.

After this operation, at least one customer must have new alternate choices $\{j'\}$ from $T \setminus \check{T}$. If any of these alternate choices are only preferred by single (non-conflicted) customers, they can be immediately sold to them. (We call these available non-conflicted articles "sinks" in the allocation.) Otherwise, set $\{j'\}$ are now conflicting, $\check{T} \leftarrow \check{T} \cup \{j'\}$, and the clique grows $\check{R} \leftarrow \check{R} \cup \{i \mid \varphi(i) = j', \ \forall j'\}$, where $\varphi(\centerdot)$ denotes the assignment mapping as defined before. The merchant will adjust the prices again and this process repeats until no customer is involved in a conflict. In Algorithm V.1, the first while loop (Lines 3—17) is the process of finding the minimum price increment which expands sets $\check{R}$ and $\check{T}$ for each iteration. The second while loop (Lines 19—23) is the assignment process, which tracks back from the sink (newly sold article) to the source t (original conflicted article) with the help of vectors ($\mathbf{\Omega}$ and ξ) recording alternate choices, and updates the assignment via preferred choice transfers. Algorithm V.2 calls the individual stages until all customers obtain unique items.

Fig. 1 gives a simple assignment problem as an example that demonstrates the progress of the algorithm. The horizontal array across the top of each utility matrix (shown in red) represents the prices for the items; the values in the matrices are profit margins ($u_{ij} - p_j$). Shaded cells and dashed cells denote the preferred choices and alternate choices for customers, respectively. The diamonds to the left of some of the rows denote competing customers, while triangles along the bottom point to the items that are conflicting. Fig. 2 gives this same example from an equilibrium flow perspective.

B. Analysis: Global Optimality and Time Complexity

The proposed algorithm produces globally optimal assignment solutions. Proof of this (in common with the manner of proof for most primal-dual methods including the AUCTION algorithm) depends on examining the values of dual variables once the market equilibrium is reached. The customers' profit margins and articles' prices are the dual variables. In each iteration customers are only interested in purchasing their preferred choices (which is also the reason for the conflicting articles and hence violation of the mutual exclusion property).

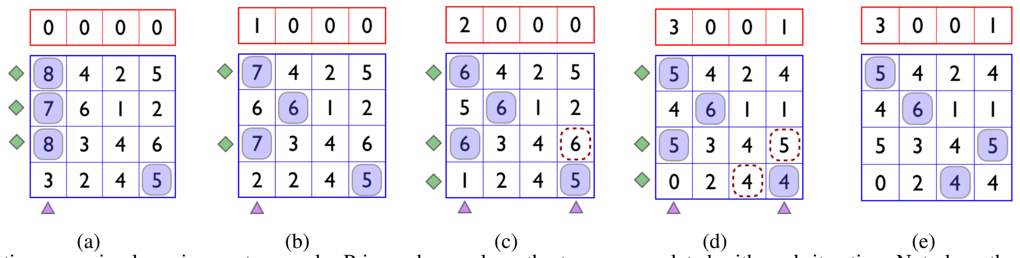


Fig. 1. Demonstration on a simple assignment example. Prices, shown along the top, are escalated with each iteration. Note how the initially independent fourth robot only becomes involved when another robot's purchasing preferences bring them into conflict in (c).

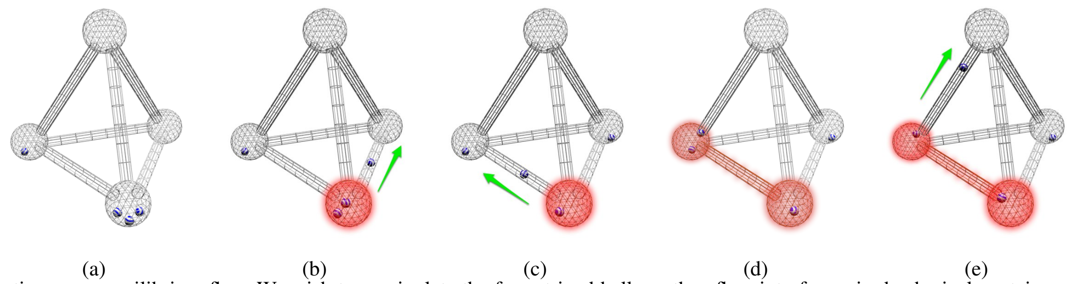


Fig. 2. Visualization as an equilibrium flow. We wish to manipulate the four striped balls so they flow into four wired spherical containers that are connected with tunnels among each other. The way to remove balls from a wired container is to create a perturbation, *i.e.*, add energy to a container to make the enclosed balls unstable (imagine the balls behave like heated microscopic particles). The essence of the algorithm is that it always adds the *minimum* energy required to push exactly one ball out of a crowded container; and it guarantees that a ball will never flow back into an already heated sphere (see Fig. 2(d)). A stage finishes when one ball flows into a previously empty sphere (see Fig. 2(b) and 2(e)).

Each customer i always maximizes his profit margin

$$\max\ m_i = \max_{\forall j}\{u_{ij} - p_j\}. \tag{14}$$

Since the relationship expressed in Eq. (7)—(9) is an if and only if, we can begin with Eq. (9) (as equivalently expressed in Eq. (14) and determine that

$$m_i + p_j \geq u_{ij}, \quad \forall(i,j). \tag{15}$$

which is essentially the constraint of the dual program (6). Thus, the dual program always maintains its feasibility. Moreover, by plugging Eq. (14) into the objective function of the dual program, the new objective function is unconstrained and in every iteration is maintained at a local optimum by each customer:

$$g(\mathbf{p}) = \sum_i \max_{\forall j}\{u_{ij} - p_j\} + \sum_j p_j. \tag{16}$$

Besides, the algorithm terminates at precisely the moment when each customer obtains a preferred choice and no conflicting articles remain. Or put another way, the moment when the primal program (1) also becomes feasible. Feasibility of both primal and dual as well as the optimality of the dual yields the global optimal solution for both programs, as stated in the *duality theorems* [3]. Therefore, when all articles are sold out, the global optimal solution must be produced. This proves the condition of global optimality.

The proposed algorithm has a time complexity of $O(n^3 \lg n)$. During each stage, the most costly operations are the computation of budget margin $v_i - w_i$ for rows $i \in \check{R}$. The v_i are always the maximal profit margins in $\check{T}$, while w_i are always the maximal profit margins in $T \setminus \check{T}$. The sets of $\check{T}$ (and by implication $T \setminus \check{T}$) change dynamically with each iteration, so efficiently maintaining the largest values requires a data structure like a priority queue or balanced tree. By employing one of these data structures, $O(\lg n)$ time is needed to extract the maximum value, and there are at most n^2 entries, so at most n^2 such operations, requiring a worst case $O(n^2 \lg n)$ per stage. A single stage will assign one non-conflicting task and there are at most $n-1$ non-conflicting tasks to be assigned, giving the total time complexity of $O(n^3 \lg n)$.

C. Analysis: An Economic Perspective

Intuitively, the larger the price escalation, the greater the number of customers who—though showing interest in purchasing those goods previously—are likely to be discouraged. To quantify the effects of escalation, the price increment δ_j for article j is denoted by*:

$$\min_{i\in\mathbb{P}(j)}\{v_i - w_i\} \leq \delta_j \leq \max_{i\in\mathbb{P}(j)}\{v_i - w_i\}, \tag{17}$$

where $\mathbb{P}(\centerdot)$ has the same meaning as before. Thus, the price increment must be between the largest profit margin, from the most willing customer, and the smallest profit margin, from the customer with weakest purchasing incentive but who is still intending to buy the good.

***Definition** 5.1:* Given a price increment δ_j for article j, we use parameter $\Gamma \in \mathbb{N}$ where $1 \leq \Gamma \leq |\mathbb{P}(j)|$ to quantify this price adjustment's incentive.

$$\Gamma = |\{i \mid v_i - w_i \leq \delta_j, \forall i \in \mathbb{P}(j)\}|, \tag{18}$$

This measures the effectiveness of the incentive's ability to alter customers' preferences, *i.e.*, a larger incentive indicates a greater alteration of the market's behavior, and vice versa.

***Proposition** 5.1:* In the AUCTION algorithm, an auctioning operation on an arbitrary item j always has the **largest** market incentive: $\Gamma = |\mathbb{P}(j)|$.

This follows because in the AUCTION algorithm,

$$\delta_j = \max_{i\in\mathbb{P}(j)}\{v_i - w_i\} + \epsilon, \tag{19}$$

*Note that the word *pricing* is also used in the linear programming literature, in particular for branch-and-bound and cutting plane methods. They use the term to describe the dynamic introduction of new variables. Their use is distinct and should not be confused with the economic use in this paper.

with $\epsilon > 0$, showing that the preferred choice of this item is changed for all the involved bidders. However, the bidder with sacrifice equal to ϵ becomes the winner (as interpreted in [2], he takes "risk" for winning this item) and the other $\Gamma - 1$ bidders are forced to alter their preferences.

***Proposition** 5.2:* The proposed algorithm always has the **smallest** incentive for (individual or group) price escalation.

In the case of *individual* price escalation for an arbitrary item j, we compute the price increment via $\delta_j = \min_{i\in\mathbb{P}(j)}\{v_i - w_i\}$. From (18), this is equivalent to

$$\Gamma = \left|\{i \mid v_i - w_i = \min_{i\in\mathbb{P}(j)}\{v_i - w_i\},\ \forall i \in \mathbb{P}(j)\}\right|. \quad (20)$$

In the case of price escalation for a *group of items*, a common price increment of $\delta_{\check{T}} = \min_{i\in\bigcup\mathbb{P}(j)}\{v_i - w_i\}, \forall j \in \check{T}$ is used on all items in set $\check{T}$. Let $\check{R} = \bigcup_{j\in\check{T}}\mathbb{P}(j)$, then the cumulative incentives from these items are

$$\Gamma^{'} = \sum_{j\in\check{T}}\left|\{i \mid v_i - w_i = \min_{i\in\check{R}}\{v_i - w_i\},\ \forall i \in \check{R}\}\right|. \quad (21)$$

The merchant always attempts the smallest price increment that is sufficient to alter a minimum number (but at least one) customers' purchasing preferences. This is a critical distinction from the AUCTION algorithm, and it is precisely the reason that global optimality is obtained with strongly polynomial time.

VI. Algorithm Decentralization

The new algorithm is suitable for decentralization the same way as the AUCTION algorithm: both computation and communication become localized to involve sub-teams of robots. We assume that customers (robots) have knowledge of all articles (tasks) and are able to locate all conflicted customers. This can be achieved in different ways depending on the communication model involved (a typical example is via broadcast over a publisher/subscriber mechanism as used in MURDOCH [9, 10]). If communication has limited range, conflicted customers can locate and communicate with each other via multi-hop message passing (see [26]); additional bookkeeping may be required if connectivity is not maintained or the topology changes during execution of the algorithm.

Only those customers with preferences which cause conflicts over the same articles need be involved in resolving the conflict. In other words, only the robots with preferences failing to satisfy the mutual exclusion property need communicate. For agents with non-conflicting preferences, price escalation on other articles will never increase their profit margins and, thus, their current preferred choices remain unaltered. They only become involved when they enter the current stage's clique, which occurs when they conflict with the alternate choices of some agent(s) in the clique, (Step 17 in Algorithm V.1). Otherwise, they are excluded from current stage's consideration. The decentralized algorithm involves two roles: a virtual *merchant* is randomly selected who is then responsible for operations during the current stage, and *customers* are formed from robots still looking to be assigned tasks. Each customer keeps listening to the merchant's messages, including price increment information, new items in $\check{T}$, and the alternate choice transfer operations.

Algorithm VI.1 Customer[i]

```
1: initiate $\check{T} := \varnothing$, $\Omega(i) := \varnothing$
2: ▷ upon receiving price increment $\delta$
3: update profit margins $m_{ij}$, $\forall j \in \check{T}$
4: if new alternate choice $t_a \in T \setminus \check{T}$ exists then
5:     $\Omega(i) := \Omega(i) \cup \{t_a\}$
6:     send newly exploited $t_a$ to merchant of current stage
7: ▷ upon receiving updated $\check{T}$:
8: $j^{'} := \text{argmax}_{j\in T\setminus\check{T}}\{u_{ij} - p_j\}$
9: $v_i - w_i := (u_{i\varphi(i)} - p_{\varphi(i)}) - (u_{ij'} - p_{j'})$
10: submit bid $v_i - w_i$ to merchant
11: ▷ upon receiving assignment $\varphi(i)$
12: update assignment (the preferred choice and an alternate choice
    swaps)
13: select a conflicting customer as new merchant, go to next stage
```

Algorithm VI.2 Merchant

```
1: initiate: $\check{T} := \{\varphi(i)\}$, send $\check{T}$ to associated conflicting cus-
   tomers $i \in \check{R}$
2: ▷ upon recv. newly exploited $t_a$ from customer $i$
3: $\xi(t_a) := i$
4: if $t_a$ is still conflicting with preferred customers, say, $H$ then
5:     $\check{T} := \check{T} \cup \{t_a\}$, $\check{R} := \check{R} \cup H$
6:     send $\check{T}$ to customers $i \in \check{R}$
7: else
8:     $sink := t_a$, query $\Omega(i)$ from customers $i \in \check{R}$
9:     compute new assignment among $\check{R}$ with alternate choice
       records of $\Omega$ and $\xi$
10:    broadcast assignment result $\varphi(i)$ to customers $i \in \check{R}$
11: ▷ upon receiving bids $v_i - w_i$ from $i \in \check{R}$
12: minimal price increment $\delta := \min_{i\in\check{R}}\{v_i - w_i\}$
13: broadcast $\delta$ to customers $i \in \check{R}$
```

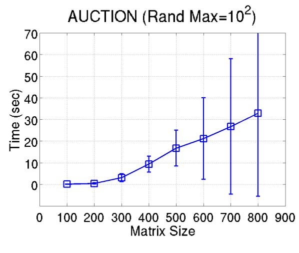
After receiving this information, computations are performed locally and the results sent to the merchant. The merchant listens to the information from customers, including the newly discovered alternate choices, choice margin information, and information for updating the customers involved. High-level pseudo-code for the customers and merchant roles are given in Algorithms VI.1 and VI.2, respectively. Detailed computations are analogous to the centralized version.

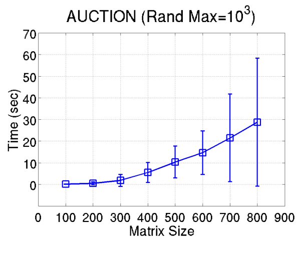
VII. Experiments

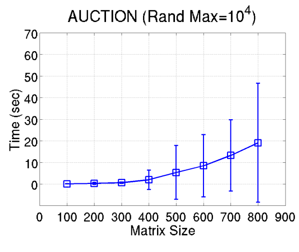
Besides the experimental results shown in Bertsekas' original papers, reported results on the AUCTION algorithm's performance are surprisingly limited. We implemented all aforementioned algorithms in C++, and ran the experiments on a Thinkpad laptop with 1.60GHz CPU and 2GB of memory. All results are obtained from data of 50 trials.

A. Performance as a Centralized Algorithm

Since the proposed algorithm has a strongly polynomial time complexity and can be used as an optimal assignment algorithm in a centralized fashion, we compared it with the Hungarian method (the classical benchmark), the AUCTION algorithm, and our previous *task swap-based* method (another distributable optimal algorithm [18] with time complexity of $O(n^3 \lg n)$ but not using any market-based notions).


(a)


(b)


(c)

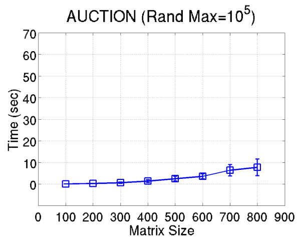

(d)

Fig. 3. Practical running times for the AUCTION algorithm.

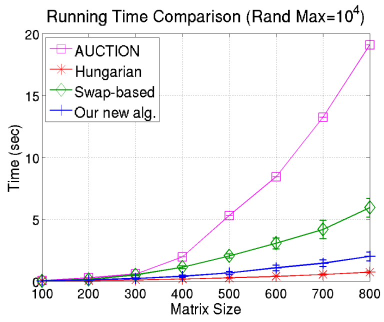
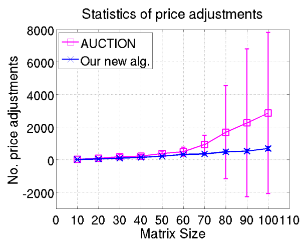


Fig. 4. A comparison of running times. (Standard deviations for AUCTION algorithm are large and thus omitted.)


(a)

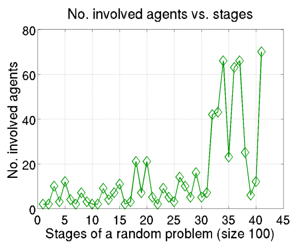

(b)

Fig. 5. (a) The number of price adjustments in our algorithm compared with bids in the AUCTION algorithm; (b) An example showing the number of involved robots along the evolution of stages.

We first tested the AUCTION algorithm in a variety of settings, and the results highlight its extreme sensitivity to the input data. Fig. 3 shows the running times under different sized matrices and randomized data from different ranges (*i.e.*, $[0, Max]$). From Fig. 3(a) to Fig. 3(d) we can see an improvement in running times of the AUCTION algorithm, indicating that the algorithm works the best when data are scattered with larger differences (as this produces larger bidding margins). Table I is a detailed comparison of the AUCTION algorithm and our method for randomly generated values in $[0, 10^3]$, which reveals obvious advantages of the proposed algorithm (both in terms of the mean time and the standard deviations).

TABLE I
RUNNING TIME PERFORMANCE: STATISTICS

	Matrix size	200	400	600	800
AUCTION	Average	0.4362	5.4619	14.615	32.868
	Std. Dev.	0.6745	4.6023	10.055	27.529
	Maximum	2.3801	16.401	38.336	105.37
	Minimum	0.0517	0.2885	0.4599	12.300
Our algo.	Average	0.0862	0.3956	1.0907	1.9998
	Std. Dev.	0.0243	0.0907	0.2184	0.3597
	Maximum	0.1730	0.5658	1.7649	2.6351
	Minimum	0.0638	0.1981	0.7942	1.3584

Fig. 4 summarizes results of the difference between the proposed algorithm and other popular methods mentioned in this paper. Compared with the benchmark (centralized) Hungarian algorithm, our algorithm is slightly inferior, but much faster than both the AUCTION algorithm and the swap-based method. The AUCTION algorithm performs the worst among the four. Note that, the swap-based method [18] has a time complexity of $O(n^3 \lg n)$ which is the same as this proposed algorithm, however, this presented algorithm seems to perform much better in practice. This may imply that, although our approach has a worst case of $O(n^3 \lg n)$, the actual amortized time complexity could be $\Theta(n^3)$. (In our algorithm, for both the centralized and decentralized versions, only a partial matrix need be dynamically computed per round; costly "ranking" data structures are created only as needed.)

B. Performance as a Decentralized Method

We have shown in Section VI that our method is as naturally distributable as AUCTION. Here we compare the method in detail with two other distributable optimal algorithms: AUCTION and the task swap-based method.

We compared our algorithm and the AUCTION method by investigating the total number of price escalations. This is because each price escalation represents one auction operation in the AUCTION algorithm or one market adjustment in our market-based method. The results, shown in Fig. 5(a), show that the proposed algorithm requires far fewer such steps, suggesting a great reduction in communication. We also compared the total number of stages (each stage requires an auctioneer or a merchant) between the two methods. With 100 agents, the results shows that our method needs on average only 40 stages, whereas the AUCTION algorithm requires almost 2000 stages to complete. This implies that our algorithm has eliminated the notoriously long iterations inherent in the AUCTION algorithm. Fig. 5(b) is a typical example (matrix size 100×100) showing the evolution of the stages, along with the number of robots involved in each.

We also compared this algorithm with the recent swap-based method, which approaches the optima by searching for a series of task exchanges between robots. First we investigated the total number of stages required for completion. Fig. 6(a) shows that both methods need very similar numbers of stages, and that the swap-based method generally performs slightly better. We then compared the average number of agents involved in each stage, which reflects the amount of communication. Fig. 6(b) shows that the proposed method always requires fewer agents to be involved. When the number of agents is large, the superiority is clear (for a size of 100, this algorithm involves only half number of that of the swap-based method).

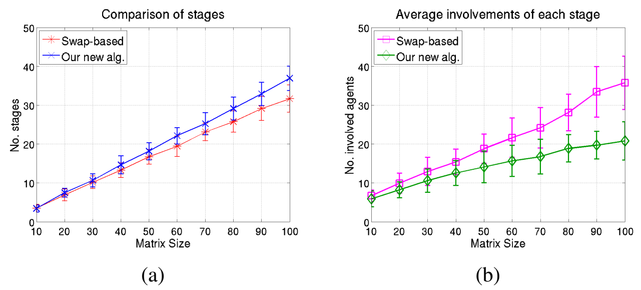


Fig. 6. (a) Total number of stages required; (b) Average number of involved agents in each stage.

VIII. Discussions and Future Work

There are several other things worth mentioning:
(1) The proposed algorithm works for unbalanced robots and tasks ($\#tasks > \#robots$, otherwise utility matrix needs to be transposed). The algorithm terminates when all conflicted customers have been resolved, and available articles are chosen in a greedy manner so that optimality is always guaranteed. This feature is also important since if a robot has failed, it can be simply removed from the current market, and the individual failure does not undermine the whole system;
(2) Conflicts can also be resolved concurrently. This means that every conflicted article can initiate a resolution in a decentralized fashion. However, if a robot is subsequently involved in multiple conflicts, later ones have to pause and wait until the earliest one is resolved and the preferred article's price has been escalated (this may change the robot's status in other conflict involvements though).

We are also interested in investigating the proposed market-based mechanism further in future work. We believe the major contribution of this work lies in the evidence that a market-based mechanism is capable of reaching the global optimum in a distributed manner with limited number of steps. However, resolving assignment ties require successful communication, which might be a potentially problematic assumption for some robotic scenarios. (The review of literature in Section II illustrates how challenging it is to reach the global optima if each robot can obtain only local information.) Therefore, one direction is embedding this new mechanism in other popular auction/market-based methods (*e.g.*, by modifying or re-designing their bidding or pricing policies) to improve their solution quality even if optimality need be sacrificed.

IX. Conclusions

Like the AUCTION algorithm, which one can regard as a procedure by which bidders with overlapping buying interests resolve their conflicts, we introduce an algorithm that can be regarded as analogous to the process of selecting pricing polices to alter purchasing behavior to clear all inventory. To summarize, the novel algorithm has several attractive features:
i.) Simple primitives and optimality: The technique has an intuitive interpretation which is inherently distributed and leads naturally to a decentralized implementation. Unlike most other market-based methods, global optimality can be achieved.
ii.) Strongly polynomial time complexity: This is an advantage over the classic optimal and distributable AUCTION algorithm, as well as existing decentralized market-based algorithms. Sensitivity to input values is also eliminated.
iii.) Distributed computation and communication: Even in computing the global optimum, typically only a small subset of robots are found to be involved. The decentralized variant of the algorithm require no single privileged global controller.

This new task allocation mechanism has potential to lead to improvements in numerous existing sub-optimal market-based approaches too.

References

[1] D. P. Bertsekas. A distributed algorithm for the assignment problem. *Lab. for Information and Decision Systems Report, MIT*, 1979.
[2] D. P. Bertsekas. The auction algorithm for assignment and other network flow problems: A tutorial. *Interfaces*, 20(4):133–149, 1990.
[3] S. Bradley, A. Hax, and T. Magnanti. *Applied Mathematical Programming*. Addison-Wesley, 1977.
[4] R. E. Burkard, M. Dell'Amico, and S. Martello. *Assignment problems*. Society for Industrial and Applied Mathematics, New York, NY, 2009.
[5] U. Derigs. The Shortest Augmenting Path Method for Solving Assignment Problems–Motivation and Computational Experience. *Annals of Operations Research*, pages 57–102, 1985.
[6] M. B. Dias and A. Stentz. Opportunistic optimization for market-based multirobot control. In *Proc. IROS*, pages 2714–2720, 2002.
[7] M. B. Dias, R. Zlot, N. Kalra, and A. Stentz. Market-Based Multirobot Coordination: A Survey and Analysis. *Proc. of the IEEE*, 94(7), 2006.
[8] E. A. Dinic and M. A. Kronrod. An algorithm for the solution of the assignment problem. *Sov. Math. Dokl.*, pages 1324–1326, 1969.
[9] B. P. Gerkey and M. J. Mataric. Murdoch: Publish/subscribe task allocation for heterogeneous agents. In *Fourth International Conference on Autonomous Agents*, pages 203–204, 2000.
[10] B. P. Gerkey and M. J. Mataric. Sold!: auction methods for multirobot coordination. *IEEE Trans. on Robotics and Autom.*, 18(5), 2002.
[11] B. P. Gerkey and M. J. Matarić. A Formal Analysis and Taxonomy of Task Allocation in Multi-robot Systems. *IJRR 23(9):939–954*, 2004.
[12] S. Giordani, M. Lujak, and F. Martinelli. A Distributed Algorithm for the Multi-Robot Task Allocation Problem. *LNCS: Trends in Applied Intelligent Systems*, 6096:721–730, 2010.
[13] D. Goldberg, V. A. Cicirello, M. B. Dias, R. G. Simmons, S. F. Smith, and A. Stentz. Task allocation using a distributed market-based planning mechanism. In *Proc. AAMAS*, pages 996–997, 2003.
[14] Sven Koenig, Pinar Keskinocak, and Craig A. Tovey. Progress on Agent Coordination with Cooperative Auctions. In *Proc. AAAI*, 2010.
[15] H. W. Kuhn. The Hungarian Method for the Assignment Problem. *Naval Research Logistic Quarterly 2:83–97*, 1955.
[16] M. G. Lagoudakis, E. Markakis, D. Kempe, P. Keskinocak, A. Kleywegt, S. Koenig, C. Tovey, A. Meyerson, and S. Jain. Auction-based multi-robot routing. In *Robotics: Science and Systems*, 2005.
[17] L. Liu and D. Shell. Multi-level partitioning and distribution of the assignment problem for large-scale multi-robot task allocation. In *Robotics: Science and Systems*, 2011.
[18] L. Liu and D. Shell. A distributable and computation-flexible assignment algorithm: From local task swapping to global optimality. In *Proceedings of Robotics: Science and Systems*, 2012.
[19] N.G. Mankiw. *Principles of Economics*. Economics Series. Cengage Learning, 2011.
[20] J. McLurkin and D. Yamins. Dynamic task assignment in robot swarms. In *Proceedings of Robotics: Science and Systems*, 2005.
[21] N. Michael, M. M. Zavlanos, V. Kumar, and G. J. Pappas. Distributed multi-robot task assignment and formation control. In *ICRA*, 2008.
[22] M. Nanjanath and M. Gini. Dynamic task allocation for robots via auctions. In *Proc. ICRA*, pages 2781–2786, 2006.
[23] L. E. Parker. Alliance: An architecture for fault-tolerant multi-robot cooperation. *IEEE Trans. on Robotics and Automation*, 14(2), 1998.
[24] D. W. Pentico. Assignment problems: A golden anniversary survey. *European Journal of Operational Research*, pages 774–793, 2007.
[25] S. L. Smith and F. Bullo. A geometric assignment problem for robotic networks. In *Modeling, Estimation and Control: Festschrift in Honor of Giorgio Picci on the Occasion of his 65 Birthday*, 2007.
[26] M. M. Zavlanos, L. Spesivtsev, and G. J. Pappas. A Distributed Auction Algorithm for the Assignment Problem. In *47th IEEE Conference on Decision and Control*, pages 1212–1217, 2008.