From Selfish Auctioning to Incentivized Marketing

Lantao Liu[1], Dylan A. Shell[2] and Nathan Michael[1]

[1]The Robotics Institute, Carnegie Mellon University
[2]Dept. of Computer Science and Engineering, Texas A&M University

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 an 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 and their preferences. The algorithm uses price escalation to clear a market of all its items, producing a state of equilibrium that satisfies both the merchant and customers. This effectively assigns all robots to their tasks. 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.

1 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 (Bertsekas, 1990). 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 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 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 (communication among them is assumed).

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

*The main algorithm was first presented in the 2013 Robotics: Science and Systems conference (Liu and Shell, 2013).

is possible to employ a market-based method suited to decentralized implementation for multi-robot systems, and still attain global optimality, and do so with strongly polynomial running time (and iterations).

2 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)* (Gerkey and Matarić, 2004).

2.1 Optimal Assignment Algorithms and Decentralization

Many optimal assignment algorithms have been developed (see reviews (Burkard *et al.*, 2009; Pentico, 2007)). 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 (Kuhn, 1955), which solves the matching problem by manipulating a matching bipartite graph. Several shortest augmenting path algorithms (*e.g.*, see Derigs (1985)) 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 Giordani *et al.* (2010)) 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.

An important instance is the AUCTION algorithm developed by Bertsekas from the late 70s to early 90s (Bertsekas, 1990). 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 (Bertsekas, 1979) 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-$scaling$ (Bertsekas, 1990)) 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 Zavlanos *et al.* (2008) 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 (Dinic and Kronrod, 1969). This algorithm was the first $O(n^3)$ optimal assignment algorithm but is independent of the primal-dual theories. However, this fact was only discovered many years after its invention due to translation problems (it was developed and published in Russian during the Soviet Era). Dinic-Kronrod's method operates on cost matrix with an initial infeasible assignment; it starts from an arbitrary unassigned column, and strategically manipulates values of other columns (without changing the assignment problem) until a new assignment configuration is found such that the starting unassigned column is assigned to some row while maintaining that other assigned columns are still assigned. In such way, the problem is solved once no unassigned column remains. Although Dinic-Kronrod's algorithm is not distributable, its source-sink concept and manipulation (see details in Burkard *et al.* (2009)) inspired our algorithmic designing process. Despite our method has an inferior time complexity ($O(n^3 \lg n)$), it is distributable — amongst other features.

Table 1 details the comparison of important features among the aforementioned task allocation/assignment methods.

2.2 Multi-robot Task Allocation Methods

Most classic optimal assignment algorithms that reach the global optimality are *centralized*. A centralized assignment algorithm can be efficient in addressing the task allocation of small-scale systems; they can be straightforward to implement because they exploit the fact that the every robot's information is globally accessible. The disadvantages become important for larger scale systems (involving tens, hundreds, even thousands of robots); significant drawbacks include, for example, (1.) imbalance in computation and communication: the central controller is responsible for information collection and dissemination as well as data processing for the whole system. A heavy workload may result in low efficiency for the central

Table 1: FEATURES COMPARISON

Methods \ Features	Optimal	Polynomial	Distributable
The proposed	Yes	Yes	Yes
AUCTION	Yes	Weakly	Yes
Market-based	No	Yes	Yes
Hungarian variants	Yes	Yes	No
Dinic-Kronrod's	Yes	Yes	No

controller, which yields in the low efficiency for the entire system; (2.) poor robustness against failure owing to the dependence on the single central controller: the failure (*e.g.*, a drained battery) of central controller may paralyse the entire system. Many existing multi-robot task allocation methods focus on distributing the communication and computation to sub-teams or even at individuals, and have been successfully demonstrated in a variety of scenarios such as formation control (Agmon *et al.*, 2010), mapping and exploration (Vincent *et al.*, 2008), path planning and navigation (Turpin *et al.*, 2012), *etc*. Other representative approaches for task allocation operate by partitioning the robots and/or tasks into subgroups and may repeat the process recursively which eases decentralization (McLurkin and Yamins, 2005; Smith and Bullo, 2007a), as well as behavior-based or role-based strategies (Parker, 1998; Vail and Veloso, 2003).

Market-based task allocation is a resource redistribution methodology that borrows the market/economy wisdom of human beings, and aims at pursuing a maximal benefit for the whole system in a *decentralized* fashion. Per the definition of Dias *et al.* (2006), a market-based task allocation method can be any strategy that applies for market mechanisms in order to allocate limited resources to certain agents whilst optimizing a team's common objective. Particular auctioning procedures provide solutions to important issues (Wolfstetter, 1994): which information (*i.e.,* the outcome of comparative item evaluations) ought to be revealed? To whom should it be revealed? How should the comparison be done in an impartial way so that the decision making does not favor a particular individual? How can this be done efficiently so that the process is terminated quickly in sale? The precise way auction mechanisms addressing these problems has lead auctions to be the most commonly used mechanisms in the market-based paradigm.

Auction mechanisms can be further categorized into two subsets: *single-item* auctions and *combinatorial* auctions. The single-item auction is the classic auctioning method that allows each bidder to submit one bid, and an honest auctioneer makes decisions to award the item to the highest bidder. Mathematical models (*e.g.*, Bertsekas (1979) or Wolfstetter (1994)) show that an optimal solution can be obtained in such scenarios. In contrast, the combinatorial action is more complex in terms of diverse auctioning policies. Also, the determination of winners is NP-complete and inapproximable (Sandholm, 2002). Specifically, combinatorial auctions allows each bidder to submit bids for a set/bundle of items, such that a bid can be expressed as a synergy of evaluations with regard to differing items (usually evaluations on different items are not additive). The presented work is built on the single-item auction, making it conceivable that an efficient algorithm may also produce the optimal solution.

A review of the literature shows that a large set of decentralized multi-robot task allocation methods employ market-based or auction-based mechanisms; representative examples include: the distributed market-based framework (Dias *et al.*, 2002; Goldberg *et al.*, 2003; Michael *et al.*, 2008), the auction-based MURDOCH model (Gerkey and Matarić, 2002, 2000), and cooperative auctions (Koenig *et al.*, 2010; Nanjanath and Gini, 2006). 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 (Lagoudakis *et al.*, 2005; Nanjanath and Gini, 2006). Note that, even for the single-item auctions, existing methods for solving multi-robot task allocation problems are typically designed on the basis of an intuition and may be presented without any rigorous algorithmic analysis. For these reasons, most often the global optimum will not be obtained (see review in Dias *et al.* (2006)); oftentimes the resulting allocation quality remains unknown. Until this paper, the AUCTION algorithm was the only optimal method in this class.

Complex task scenarios may also impose various constraints such as time (Luo *et al.*, 2013) and resources (Zhang and Parker, 2013). A more common constraint is the communication (Smith and Bullo, 2007b; Zavlanos *et al.*, 2008), where each robot has a limited communication range and is able to exchange information with only neighbors in the vicinity. Information accessibility affects assignment solution quality, and sub-optimality can be obtained only if local (incomplete) information is available. Note, the proposed work assumes the availability of requested information, *i.e.*, communication among agents in

the involved sub-teams are assumed to be reliable (multi-hop message relay might be required for communication between a pair of distant agents).

Recently, we proposed an assignment method (Liu and Shell, 2012a) that is also optimal and distributable. It utilizes a *task-swap* mechanism, where a certain subset of robots may form a clique in which tasks are redistributed. Upon the completion of each redistribution, each robot involved obtains a new task. Throughout execution of the algorithm, every robot (no matter if it is involved in task swaps or not) is still assigned a task, meaning that the algorithm is an anytime assignment method. The task-swap based algorithm has some philosophical similaries to the presented method: both require only subset of robots to be involved at each stage, and both algorithms have a time complexity of $O(n^3 \lg n)$. However, the two algorithms are fundamentally different in terms of both underlying theoretical models (the task-swap method is *primal*-based whereas this proposed work is *dual*-based), and their resulting mechanisms and interpretations (market or auction based methods are independent of task swap based frameworks). The task swap approach is used for comparison with this proposed algorithm in Section 7.

3 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 $u\colon R \times T \to \mathbb{R}^+$ 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:

$$\begin{aligned} \text{maximize } f(\mathbf{x}) = & \sum_{i=1}^{n}\sum_{j=1}^{n} u_{ij}x_{ij}, \\ \text{subject to } & \sum_{j=1}^{n} x_{ij} = 1, \quad i = 1, \cdots, n, \\ & \sum_{i=1}^{n} x_{ij} = 1, \quad j = 1, \cdots, n, \\ & x_{ij} \geq 0, \quad i, j = 1, \cdots, n, \end{aligned} \tag{1}$$

where an optimal solution eventually is an extreme point of its feasible set (*i.e.*, $x_{ij} \in \{0, 1\}$). The constraints $\sum_{j=1}^{n} x_{ij} = 1$ and $\sum_{i=1}^{n} x_{ij} = 1$ 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. We assume that both roles are truthful. The auctioneer can be any agent (either an honest bidder or an agent excluded from the current bidding).

An auction process also requires two types of prices: the first is the *actual price* p_j for an item j, which denotes the price that this item is currently labelled; the other type is the *budge price* that bidder (agent) i has for item (task) j: it is equal to the utility u_{ij}, and one may also think of it as an affordable price that bidder i thinks item j is worth. By default, the *price* refers to the actual price.

Definition 3.1 *The profit margin 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 price escalation operation involves raising prices on* individual *or* groups of items*. Raising individual prices involves adding a $\delta \in \mathbb{R}^+$ (associated with a single item 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 items S:*

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

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 preferred choice j^* for bidder i is an item with maximal profit margin:*

$$j^* = \underset{j=1,\cdots,n}{\operatorname{argmax}}\{m_{ij}\}. \tag{4}$$

Given a choice of different items, a rational bidder would like to choose an item 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 alternate choices 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 preferred choice transfer $j^* \rightarrowtail j'$ 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.*

4 The AUCTION Algorithm

The AUCTION algorithm, proposed by Bertsekas (1979), 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:

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

If the set of prices, $\mathbf{p} = \{p_j\}, \forall j$, have been determined and are invariant, then the feasibility of constraint $\pi_i \geq u_{ij} - p_j$ and the minimization of $g(\pi, \mathbf{p})$ imply that π_i equals the supremum of the closed set $\{u_{ij} - p_j\}$. This is equivalent to:

$$\pi_i = \max_{j=1,\cdots,n}\{u_{ij} - p_j\} = \max_{j=1,\cdots,n}\{m_{ij}\}. \tag{7}$$

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

$$g(\mathbf{p}) = \sum_{i=1}^{n} \max_{j}\{u_{ij} - p_j\} + \sum_{j=1}^{n} p_j, \tag{8}$$

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

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

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 ϵ-complementary slackness condition is defined as:*

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

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,

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

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

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

The price of item 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. $\triangleleft$

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. However, the number of bidding rounds for one auctioning stage is proportional to the value of $\frac{\max_{(i,j)} u_{ij}}{\epsilon}$, and the algorithm has an overall time complexity of $O(n^2 \frac{\max_{(i,j)} u_{ij}}{\epsilon})$ (Bertsekas, 1992). If we choose ϵ to be slightly less than $\frac{1}{n}$, the complexity becomes $O(n^3 \max_{(i,j)} u_{ij})$. In other words, the running time is *weakly polynomial* since it also depends on the value of input data besides the number of agents n. (Note also u_{ij} are integers so that they cannot be scaled arbitrarily small.) The time complexity is later improved by using a multi-trial method called ϵ-$scaling$. Unfortunately multiple trials undermine the decentralized nature of the approach and, hence, its appropriateness for distributed robot applications especially when the communication cost is prohibitive. So far as we are aware, the ϵ-$scaling$ version has not been used in the robot task assignment literature. Thus, we do not discuss them further in this work; for more detail see Bertsekas (1990, 1992).

5 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 items through strategic selection of prices. In a sense, the algorithm has the merchant steer purchasing behavior of the customers.

5.1 Algorithm Description

To view the proposed algorithm from an economic perspective, suppose that the tasks, $j = 1, \cdots, n$, are items in a market, only one instance of each item is available, and the agents, $i = 1, \ldots, n$, are customers visiting the market. The utility value u_{ij} still represents the budget of customer i for item 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 items whilst ensuring each customer obtains an item). 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 items preferred by multiple customers, the merchant will raise prices by an amount so that the items 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 items. 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 (Mankiw, 2011).

Algorithm 5.1 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: $\quad \delta := \infty$
5: $\quad$ **for each** i in set $\check{R}$ **do**
6: $\quad\quad l := \text{argmax}_{j \in T \setminus \check{T}} \{u_{ij} - p_j\}$
7: $\quad\quad v := u_{i\varphi(i)} - p_{\varphi(i)},\ w := u_{il} - p_l$
8: $\quad\quad$ **if** $v - w < \delta$ **then**
9: $\quad\quad\quad \delta := v - w$
10: $\quad\quad\quad \bar{i} := i,\ \bar{j} := l$
11: $\quad$ **for each** j in set $\check{T}$ **do**
12: $\quad\quad p_j := p_j + \delta$
13: $\quad \Omega_{\bar{i}} := \Omega_{\bar{i}} \cup \{\bar{j}\},\ \xi_{\bar{j}} := \bar{i}$
14: $\quad$ **if** $\{i \mid \varphi(i) = sink\} = \varnothing$ **then**
15: $\quad\quad sink := \bar{j}$
16: $\quad$ **else**
17: $\quad\quad \check{T} := \check{T} \cup \{\bar{j}\}, \check{R} := \check{R} \cup \{i \mid \varphi(i) = \bar{j}\}$
18: $j_\varphi := 0, j_p := sink$
19: **while** $j_\varphi \neq t$ **do**
20: $\quad j_\varphi := \varphi(\bar{i}), \varphi(\bar{i}) := j_p$
21: $\quad$ **if** $j_\varphi \neq t$ **then**
22: $\quad\quad \bar{i} := \xi_{j_\varphi}$
23: $\quad\quad j_p := j_\varphi$

Algorithm 5.2 MARKET_MAIN

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

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 item t. In the case where there are multiple conflicted items, 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 margin*s $v_i - w_i$ to the merchant, where $v_i = \max_{j \in \check{T}} \{m_{ij}\}$ is the maximal profit margin for items currently conflicted (in set $\check{T}$), and $w_i = \max_{j \in T \setminus \check{T}} \{m_{ij}\}$ is the maximal profit margin for items that are not currently conflicted (not in set $\check{T}$). If the prices of currently conflicted items escalate more than a customer's budget margin, that customer will lose interest in the items 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 items 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 items "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 5.1, 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 item) to the source t (original conflicted item) with the help of vectors ($\mathbf{\Omega}$ and ξ) recording alternate choices, and updates the assignment via

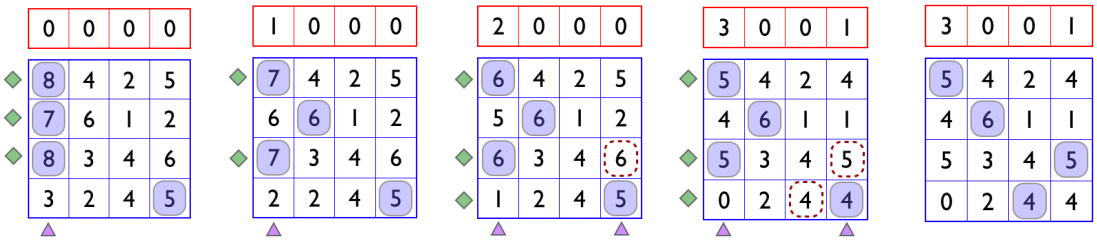


Figure 1: Demonstration on a simple assignment example. Prices, shown along the top, are escalated with each iteration. Note how the initially independent fourth robot only becomes involved when another robot's purchasing preferences bring them into conflict in (c).

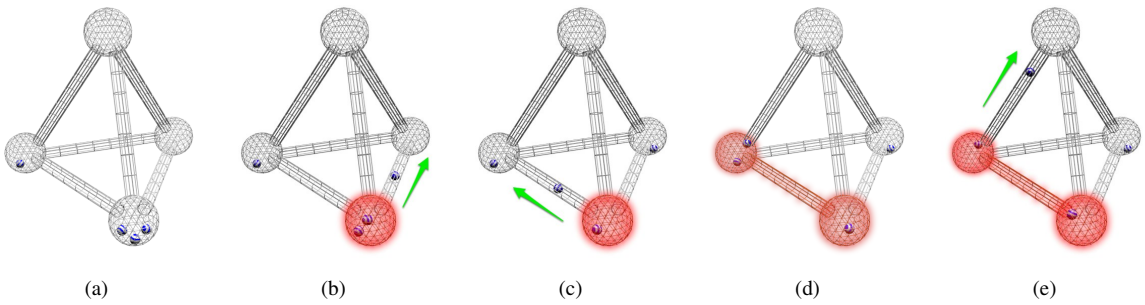


Figure 2: Visualization as an equilibrium flow. We wish to manipulate the four striped balls so they flow into four wired spherical containers that are connected with tunnels among each other. The way to remove balls from a wired container is to create a perturbation, *i.e.*, add energy to a container 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)).

preferred choice transfers. Algorithm 5.2 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.

5.2 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 items' 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 items and hence violation of the mutual exclusion property). Each customer i always maximizes his profit margin

$$\max m_i = \max_j \{u_{ij} - p_j\}. \tag{14}$$

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

$$m_i + p_j \geq u_{ij}, \quad i, j = 1, \cdots, n. \tag{15}$$

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:

$$g(\mathbf{p}) = \sum_{i=1}^{n} \max_j \{u_{ij} - p_j\} + \sum_{j=1}^{n} p_j. \tag{16}$$

Besides, the algorithm terminates at precisely the moment when each customer obtains a preferred choice and no conflicting items 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* (Bradley *et al.*, 1977). Therefore, when all items 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 $O(n^3 \lg n)$.

5.3 Analysis: An Economic Perspective

To distinguish the essence of our method from the AUCTION algorithm, we provide analysis from the perspective of economic incentives.

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

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

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

where $\mathbb{P}(\cdot)$ 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 item.

Definition 5.1 *Given a price increment δ_j for item 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,

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

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 Bertsekas (1990), 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

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

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

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

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 somewhat analogous to applying a gentle hand on the price policy tiller to influence economic behavior.

One critical difference between our method and the AUCTION algorithm also lies in the manner of addressing a tie, *i.e.*, a group of customers/bidders have equal or similar preferences over the same items. In our method, a tie can always be resolved with a bounded number of price escalations (each operation eliminates one conflicted item); in contrast, even the selfish bidding always attempts the largest market incentive, yet price increments may be too small to beat out competitors quickly (in the extreme case of equal preferences, a price escalation cannot be more than $1/n$).

6 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 items (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 (Gerkey and Matarić, 2000, 2002)). If communication has limited range, conflicted customers can locate and communicate with each other via multi-hop message passing (see Zavlanos *et al.* (2008)); 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 items need to be involved in resolving the conflict. In other words, only the robots with preferences failing to satisfy the mutual exclusion property need to communicate. For agents with non-conflicting preferences, price escalation on other items 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 5.1). 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 the involved conflicted robots 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 6.1 Customer[i]

1: initiate $\check{T} := \varnothing$, $\Omega(i) := \varnothing$
2: ▷ `upon receiving price increment` δ
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: $\quad \Omega(i) := \Omega(i) \cup \{t_a\}$
6: $\quad$ 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 6.2 Merchant

1: initiate: $\check{T} := \{\varphi(i)\}$, send $\check{T}$ to associated conflicting customers $i \in \check{R}$
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: $\quad \check{T} := \check{T} \cup \{t_a\}$, $\check{R} := \check{R} \cup H$
6: $\quad$ send $\check{T}$ to customers $i \in \check{R}$
7: **else**
8: $\quad sink := t_a$, query $\Omega(i)$ from customers $i \in \check{R}$
9: $\quad$ compute new assignment among $\check{R}$ with alternate choice records of Ω and ξ
10: $\quad$ 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 δ 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. After each stage which sells exactly one ignored item, the size of conflicted customers is reduced. So long as the conflict still exists, a merchant is required and if the current merchant is still involved in conflicts, it can continue to play the role; otherwise, the role will pass to any of the currently conflicted customers. Conflicts over disjoint sets of items also allow multiple merchants to exist simultaneously.

High-level pseudo-code for the customers and merchant roles are given in Algorithms 6.1 and 6.2, respectively. Detailed computations are analogous to the centralized version.

7 Experiments

Besides the experimental results shown in Bertsekas' original papers, reported results on the AUCTION algorithm's performance are surprisingly limited. We implemented the 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.

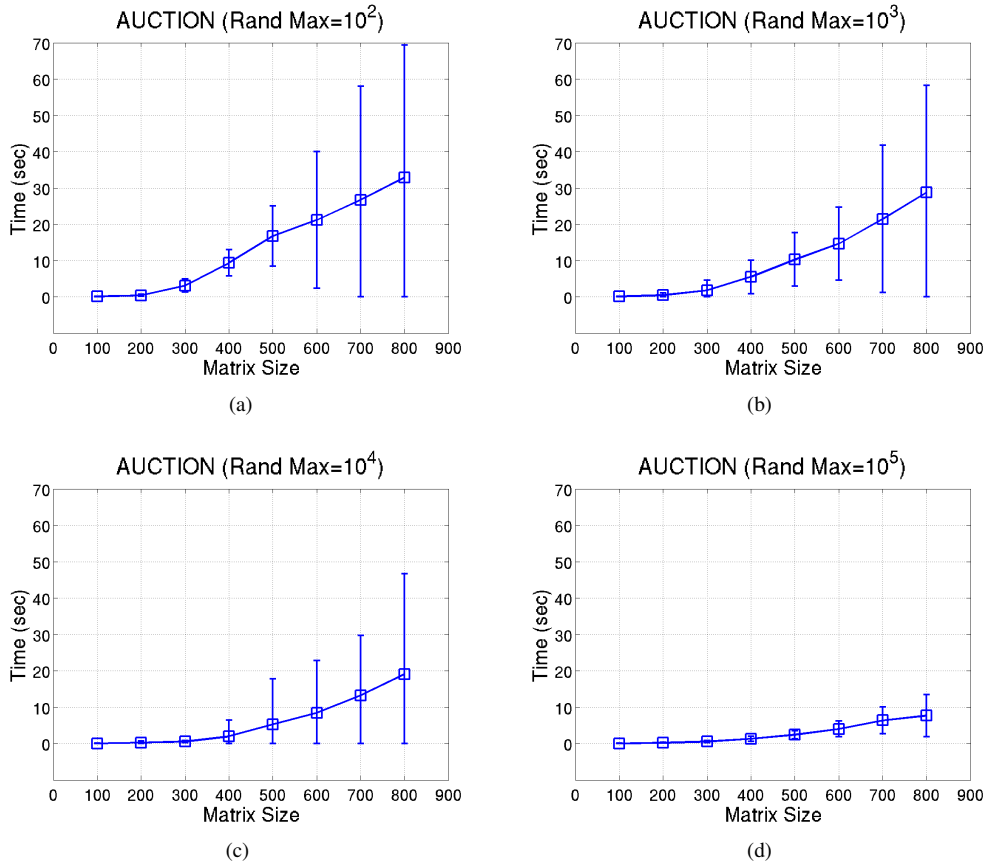


Figure 3: Practical running times for the AUCTION algorithm (vertical bars denote standard deviations).

7.1 Performance as a Centralized Algorithm

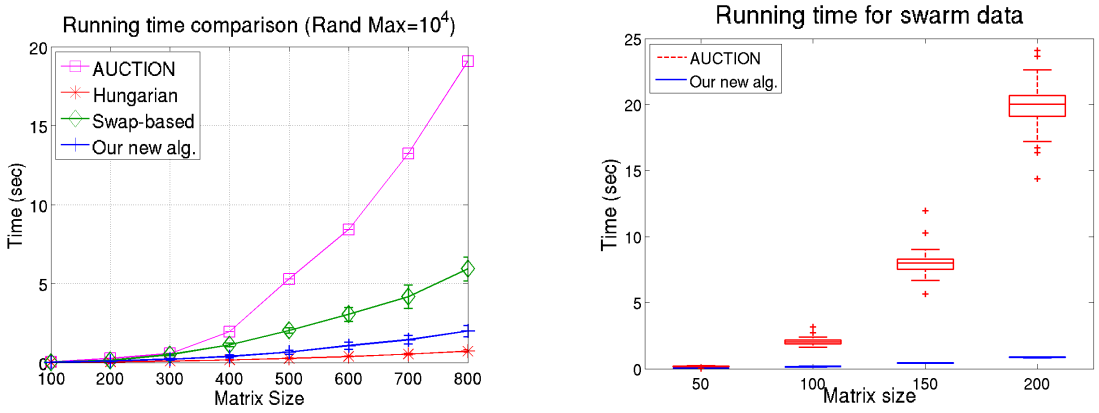
Since the proposed algorithm has a strongly polynomial time complexity and can be used as an optimal assignment algorithm in a centralized fashion, we compared it with the Hungarian method (the classical benchmark), the AUCTION algorithm, and our previous *task swap-based* method (another distributable optimal algorithm (Liu and Shell, 2012a) with time complexity of $O(n^3 \lg n)$ but not using any market-based notions).

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.*, $[0, Max]$). (Note, randomly generated data is utilized at first because we wish to evaluate the algorithm as a general assignment algorithm not limited to robotics scenarios.) 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 2 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).

Fig. 4(a) summarizes results of the difference between the proposed algorithm and other aforementioned methods. 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

Table 2: 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



(a) (b)

Figure 4: A comparison of running times based on (1) randomly generated data (note: standard deviations for AUCTION algorithm are large and thus omitted) and, (b) cost matrix obtained from a simulated swarm system (note: the 1st, 2nd and 3rd quartiles are the bottom, mid-bar and top of the box; whiskers are within 1.5 IQR and outliers are denoted as small "+").

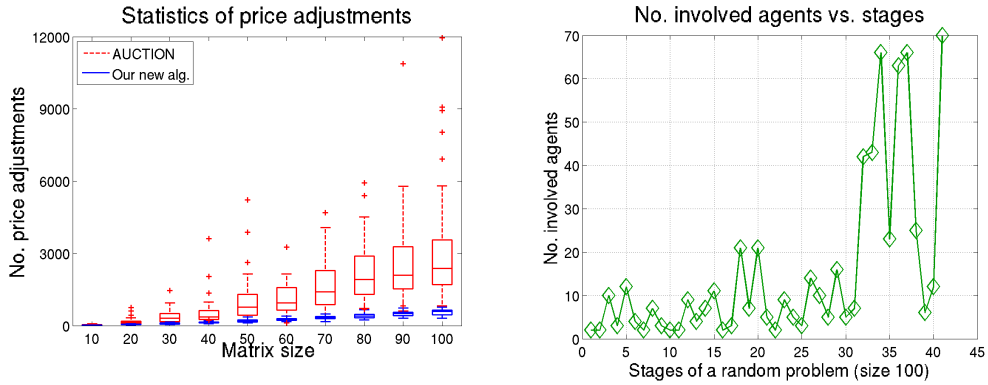
swap-based method (Liu and Shell, 2012a) has a time complexity of $O(n^3 \lg n)$ which is the same as this proposed algorithm, however, this presented algorithm seems to perform much better in practice. This may imply that, although this proposed approach has a worst case of $O(n^3 \lg n)$, the actual amortized time complexity could be $\Theta(n^3)$. (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.)

Since data from particular applications may contain certain structures, we also evaluate our method with data from a simulated multi-robot task allocation scenario, where a swarm of robots are assigned to pursue a set of moving tasks (the scenario is analogous to the pursuit-evasion simulation presented in Liu and Shell (2012b)). The utility matrix is filled by the negated costs (travel distances) among all combinations of robot-task pairs. All values in the matrix are also rounded to be integers. Fig. 4(b) reveals that, comparing with the randomly generated data and for the matrix of the same size, the AUCTION algorithm needs much longer time to compute solutions on data obtained from the robotic scenario. A possible reason is that more ties (similar purchase preferences) are produced in the latter case. In constrast, the performance of our method (blue lines at the bottom) is hardly affected with regards to varying data.

7.2 Performance as a Decentralized Method

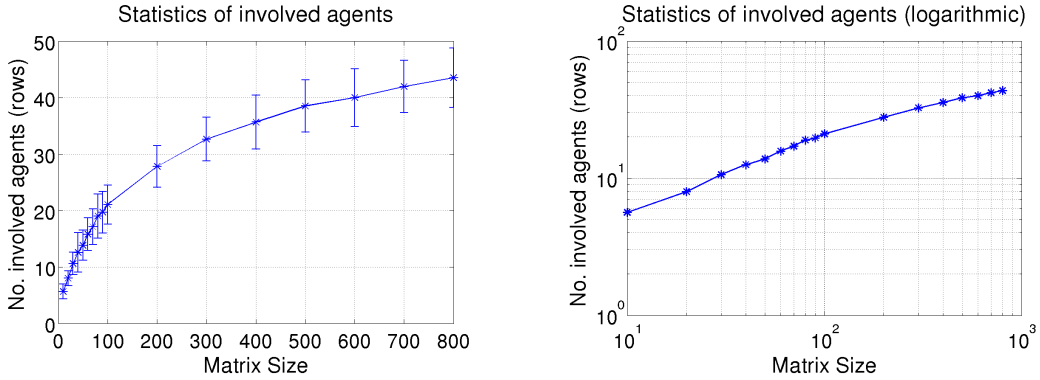
We have shown in Section 6 that our method is as naturally distributable as AUCTION. Here we compare the method in detail with two other distributable optimal algorithms: AUCTION and the task swap-based method.

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



(a) (b)

Figure 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.



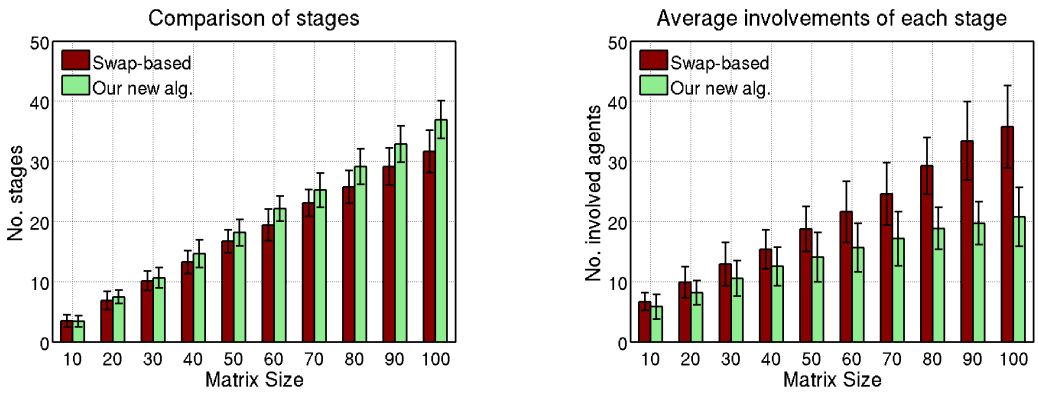
(a) (b)

Figure 6: Number of agents/rows involved (averaged). (a) Linear axes; (b) Logarithmic axes.

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 then investigated the number of rows involved during each stage, which reflects the number of conflicted robots required for communication in the decentralized variants. Fig. 6 shows results from various matrices with different sizes ranging up to 800. We see that the number of involved agents generally increases, following a logarithmic function, observable in Fig. 6(b). When the matrix is large, *e.g.*, at a size of 800, the average number of robots in a stage is about 40. This sub-linear trend is positive, indicating that the larger the system, the greater the benefit of a decentralized implementation.

We also compared this algorithm with the 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. 7(a) shows that both methods need very similar numbers of stages, and that the swap-based method generally performs slightly better. We then closely compared the average number of agents involved in each stage (reflecting the amount of communication) using up to 100 agents. Fig. 7(b) shows that on average the proposed method 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).



(a) (b)

Figure 7: (a) Total number of stages required; (b) Average number of involved agents in each stage.

8 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 items are chosen in a greedy manner and the 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 item 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 item's price has been escalated (this may change the robot's status in other conflict involvements though).

3. From a purely algorithmic perspective, the proposed method and the Dinic-Kronrod's algorithm possess a certain similarity: each phase of both algorithms must allocate exactly one unallocated task. But it is important to understand that the details are quite different: Dinic-Kronrod's algorithm starts directly from unallocated tasks and reaches what would be analogous to a "conflicting" row as a sink. The process of our algorithm actually proceeds in the opposite manner, starting with conflicted agents, and ending up with an unallocated task as the sink. In order to find the alternate choices, Dinic-Kronrod's algorithm makes comparisons between current records and the assigned entries, enabling a time complexity of $O(n^2)$ per stage; In our method the tasks outside the exploited set have to be ranked for computing the bids, which requires $O(n^2 \lg n)$ for each stage. Although it has slightly inferior time complexity, by overcoming the global communication and computation challenges, this is offset by advantages gained via a decentralized variant (see Section 5.1 and 6).

In future work, we are also interested in investigating the proposed market-based mechanism further. As we have stated, we believe that 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. But resolving assignment ties requires successful communication, which could potentially be problematic in some robotic scenarios. (The review of literature in Section 2 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.

9 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

Agmon, N., Kaminka, G. A., Kraus, S., and Traub, M. (2010). Task Reallocation in Multi-Robot Formations. *J. of Physical Agents 4(2)*, **4**(2).

Bertsekas, D. P. (1979). A distributed algorithm for the assignment problem. *Lab. for Information and Decision Systems Report, MIT*.

Bertsekas, D. P. (1990). The auction algorithm for assignment and other network flow problems: A tutorial. *Interfaces*, **20**(4), 133–149.

Bertsekas, D. P. (1992). Auction algorithms for network flow problems: A tutorial introduction. *Computational Optimization and Applications*, **1**, 7–66.

Bradley, S., Hax, A., and Magnanti, T. (1977). *Applied Mathematical Programming*. Addison-Wesley.

Burkard, R. E., Dell'Amico, M., and Martello, S. (2009). *Assignment problems*. Society for Industrial and Applied Mathematics, New York, NY.

Derigs, U. (1985). The Shortest Augmenting Path Method for Solving Assignment Problems–Motivation and Computational Experience. *Annals of Operations Research*, pages 57–102.

Dias, M. B., , and Stentz, A. (2002). Opportunistic optimization for market-based multirobot control. In *Proc. IROS*, pages 2714–2720.

Dias, M. B., Zlot, R., Kalra, N., and Stentz, A. (2006). Market-Based Multirobot Coordination: A Survey and Analysis. *Proceedings of the IEEE*, **94**(7), 1257–1270.

Dinic, E. A. and Kronrod, M. A. (1969). An algorithm for the solution of the assignment problem. *Sov. Math. Dokl.*, pages 1324–1326.

Gerkey, B. P. and Matarić, M. J. (2000). Murdoch: Publish/subscribe task allocation for heterogeneous agents. In *Fourth International Conference on Autonomous Agents*, pages 203–204.

Gerkey, B. P. and Matarić, M. J. (2002). Sold!: auction methods for multirobot coordination. *IEEE Trans. on Robotics and Autom.*, **18**(5).

Gerkey, B. P. and Matarić, M. J. (2004). A formal analysis and taxonomy of task allocation in multi-robot systems. *International Journal of Robotics Research*, **23**(9), 939–954.

Giordani, S., Lujak, M., and Martinelli, F. (2010). A Distributed Algorithm for the Multi-Robot Task Allocation Problem. *LNCS: Trends in Applied Intelligent Systems*, **6096**, 721–730.

Goldberg, D., Cicirello, V. A., Dias, M. B., Simmons, R. G., Smith, S. F., and Stentz, A. (2003). Task Allocation Using a Distributed Market-based Planning Mechanism. In *Proceedings of the International Joint Conference on Autonomous Agents & Multiagent Systems, (AAMAS)*, pages 996–997, Melbourne, Australia.

Koenig, S., Keskinocak, P., and Tovey, C. A. (2010). Progress on Agent Coordination with Cooperative Auctions. In *Proc. AAAI*.

Kuhn, H. W. (1955). The Hungarian Method for the Assignment Problem. *Naval Research Logistic Quarterly 2:83–97*, **2**, 83–97.

Lagoudakis, M. G., Markakis, E., Kempe, D., Keskinocak, P., Kleywegt, A., Koenig, S., Tovey, C., Meyerson, A., and Jain, S. (2005). Auction-based multi-robot routing. In *Robotics: Science and Systems*.

Liu, L. and Shell, D. A. (2012a). A distributable and computation-flexible assignment algorithm: From local task swapping to global optimality. In *Proceedings of Robotics: Science and Systems*.

Liu, L. and Shell, D. A. (2012b). Large-scale multi-robot task allocation via dynamic partitioning and distribution. *Auton. Robots*, **33**(3), 291–307.

Liu, L. and Shell, D. A. (2013). Optimal market-based multi-robot task allocation via strategic pricing. In *Proceedings of Robotics: Science and Systems*, Berlin, Germany.

Luo, L., N.Chakraborty, and Sycara, K. (2013). Distributed algorithm design for multi-robot task assignment with deadlines for tasks. In *ICRA*.

Mankiw, N. (2011). *Principles of Economics*. Economics Series. Cengage Learning.

McLurkin, J. and Yamins, D. (2005). Dynamic task assignment in robot swarms. In *Proceedings of Robotics: Science and Systems*.

Michael, N., Zavlanos, M. M., Kumar, V., and Pappas, G. J. (2008). Distributed Multi-Robot Task Assignment and Formation Control. In *IEEE International Conference on Robotics and Automation*, Pasadena, CA.

Nanjanath, M. and Gini, M. (2006). Dynamic task allocation for robots via auctions. In *Proc. ICRA*, pages 2781–2786.

Parker, L. E. (1998). Alliance: An Architecture for Fault-tolerant Multi-robot Cooperation. *IEEE Transactions on Robotics and Automation*, **14**(2), 220–240.

Pentico, D. W. (2007). Assignment problems: A golden anniversary survey. *European Journal of Operational Research*, pages 774–793.

Sandholm, T. (2002). Algorithm for optimal winner determination in combinatorial auctions. *Artif. Intell.*, **135**(1-2), 1–54.

Smith, S. L. and Bullo, F. (2007a). 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*, volume 364, pages 271–284.

Smith, S. L. and Bullo, F. (2007b). Target assignment for robotic networks: Asymptotic performance under limited communication. In *American Control Conference*, pages 1155–1160.

Turpin, M., Michael, N., and Kumar, V. (2012). Trajectory planning and assignment in multirobot systems. In *International Workshop on the Algorithmic Foundations of Robotics (WAFR)*.

Vail, D. and Veloso, M. (2003). Multi-robot dynamic role assignment and coordination through shared potential fields. In A. Schultz, L. Parker, and F. Schneider, editors, *Multi-Robot Systems*. Kluwer.

Vincent, R., Fox, D., Ko, J., Konolige, K., Limketkai, B., Morisset, B., Ortiz, C., Schulz, D., and Stewart, B. (2008). Distributed multirobot exploration, mapping, and task allocation. *Annals of Mathematics and Artificial Intelligence*, **52**(2-4), 229–255.

Wolfstetter, E. (1994). Auctions – an introduction. *J. Economic Surveys*, **10**(4), 367–420.

Zavlanos, M. M., Spesivtsev, L., and Pappas, G. J. (2008). A Distributed Auction Algorithm for the Assignment Problem. In *Proceedings of the IEEE Conference on Decision and Control*, pages 1212–1217, Cancun, Mexico.

Zhang, Y. and Parker, L. E. (2013). Considering inter-task resource constraints in task allocation. *Autonomous Agents and Multi-Agent Systems*, **26**(3), 389–419.