A Distributed Algorithm for the Multi-Robot Task Allocation Problem

Conference Paper · June 2010

DOI: 10.1007/978-3-642-13022-9_72 · Source: DBLP

3 authors, including:

Stefano Giordani
University of Rome Tor Vergata
71 PUBLICATIONS **993** CITATIONS
SEE PROFILE

Marin Lujak
IMT Lille Douai
49 PUBLICATIONS **413** CITATIONS
SEE PROFILE

A Distributed Algorithm for the Multi-Robot Task Allocation Problem

Stefano Giordani[1], Marin Lujak[1], and Francesco Martinelli[2]

[1]Dip. Ingegneria dell'Impresa - University of Rome "Tor Vergata", Italy
[2]Dip. Informatica Sistemi e Produzione - University of Rome "Tor Vergata", Italy

Abstract. In this work we address the Multi-Robot Task Allocation Problem (MRTA). We assume that the decision making environment is decentralized with as many decision makers (agents) as the robots in the system. To solve this problem, we developed a distributed version of the Hungarian Method for the assignment problem. The robots autonomously perform different substeps of the Hungarian algorithm on the base of the individual and the information received through the messages from the other robots in the system. It is assumed that each robot agent has an information regarding its distance from the targets in the environment. The inter-robot communication is performed over a connected dynamic communication network and the solution to the assignment problem is reached without any common coordinator or a shared memory of the system. The algorithm comes up with a global optimum solution in $O(n^3)$ cumulative time ($O(n^2)$ for each robot), with $O(n^3)$ number of messages exchanged among the n robots.

Key words: Multi robot task allocation, assignment problem, distributed algorithm

1 Introduction

In this work, we consider the Multi-Robot Task Allocation problem (MRTA) which objective is to find the assignment of n robots to a set of n tasks based on the optimization of some global objective function (see, e.g., Gerkey and Mataric, 2003). MRTA corresponds to the (linear sum) assignment problem for which the first developed algorithm was the Hungarian method (see Kuhn, 1955).

The Hungarian method or Kuhn-Munkres algorithm is a well known iterative algorithm which maintains dual feasibility during calculation and searches for a primal solution satisfying complementary slackness conditions. If the primal solution is feasible, the solution is optimal. If the primal solution is not feasible, the method performs a modification of the dual feasible solution after which a new iteration starts. Hungarian method can be implemented using the alternating trees so that its worst case time complexity is limited by $O(n^3)$ (see, e.g., Papadimitriou and Steiglitz 1982).

We assume that the decision making environment is decentralized with as many decision makers (agents) as the robots in the system. To solve this problem, we developed a distributed version of the Hungarian Method. The robots autonomously perform different sub-steps of the Hungarian algorithm on the base of the individual and the information received through the messages from the other robots in the system. It is assumed that each robot agent has the information regarding its distance from the targets in the environment or, more generally, the assignment costs to the same. The inter-robot communication is performed over a connected dynamic communication network. The algorithm comes up with a global optimum solution in $O(n^3)$ cumulative time ($O(n^2)$ for each robot), with $O(n^3)$ number of messages exchanged among the n robots. There is no shared memory among the processors and the solution to the assignment problem is reached without any common coordinator of the system. The tasks are memoryless and are indeed considered as objects without any calculation or memory capacities.

The outline of the paper is as follows. In Section 2, we introduce related work. In Section 3, some basic definitions are stated together with the formulation of the problem. In Section 4, the distributed allocation algorithm is presented. In Section 5 we present the simulation results. We close the paper with the possible directions of the future work and the conclusion in Section 6.

2 Related Work

There are several main approaches to the Assignment problem (see, e.g., Burkard and Çela, 1999).

The classical centralized assignment methods find a matching solution through the iterative improvement of some cost function: in primal simplex methods it is a primal cost, and in Hungarian, dual simplex and relaxation methods it is a dual cost (see Bertsekas, 1992).

The Auction algorithms can improve as well as worsen both the primal and the dual cost through the intermediate iterations, although at the end the optimal assignment is found (see Bertsekas, 1991). Bertsekas in this work introduces the auction algorithm in which the agents bid for the tasks in iterative manner, and in each iteration, the bidding increment is always at least equal to ϵ (ϵ-complementary slackness). If $\epsilon < \frac{1}{n}$ the algorithm finds the optimal solution, and it runs in $O(n^3 \cdot \max\{c_{ij}\})$ time. Zavlanos et al. (2008) provide a distributed version of the auction algorithm proposed by Bertsekas for the networked systems with the lack of global information due to the limited communication capabilities of the agents. Updated prices, necessary for accurate bidding can be obtained in a multi-hop fashion only by local exchange of information between adjacent agents. No shared memory is available and the agents are required to store locally all the pricing information. This approach calculates the optimal solution in $O(\Delta \cdot n^3 \cdot \max\{c_{ij}\})$ time, with $\Delta \leq n-1$ being the maximum network diameter of the communication network.

There are also many parallel algorithms based on the Hungarian method. For a good survey see, e.g., Burkard and Çela, 1999, Bertsekas et al., 1995.

Another way of seeing the assignment problem is through the Game theoretical formulation (see, e.g., Arslan et al., 2007).

In the robotic community, there are many heuristic methods developed for the MRTA problem. Smith and Bullo (2007) describe two algorithms, namely ETSP, and Grid algorithm, for the task assignment in the sparse and dense environments. The agents have a full knowledge of the tasks in the environment and, in the ETSP algorithm approach the closest ones. If the closest task is occupied, they pre-compute an optimal tour through the n remaining tasks, and search for the first non-occupied task based on certain criteria. Gerkey and Mataric (2003, 2004) describe certain relevant heuristic algorithms for the MRTA problem through the prism of three important factors: computation and communication requirements, and the manner in which tasks are considered for (re)assignment, i.e. whether all or just a part of the tasks are considered for (re)assignment in each iteration. The problem with these approaches, as the authors state, is that there is no characterization of the solution quality that can be expected of the algorithms. Kwok et al. (2002) apply the classic combinatorial methods, such as Hungarian and Gabow algorithms, to the assignment problem in the context of a set of mobile robots which need to be moved to a desired matrix of grid points to have a complete surveillance of the desired area.

Top-down centralized methods and, in general, centralized control, are highly complex and costly in the cases with increased number of controlled components, in our case robots, and are influenced by resource limitations, performance bottlenecks, and critical failures (see, e.g., Sugar and Kumar, 2000). On the contrary, the coordination system in a bottom-up multi-agent modular approach distributes computational resources and capabilities across a network of interconnected agents, and therefore reduces the complexity and increases the robustness of the same. MAS does not suffer from the "single point of failure" problem associated with centralized systems (see e.g. Wooldridge, 2002). Moreover, in the case of an unknown dynamic environment or the unknown moving obstacles situation as is the case in the factories with mobile production robots and stochastic demand situation (see, e.g., Giordani, Lujak, and Martinelli, 2009), local planning is preferred over global planning. The latter is expensive if there are many agents in the system or if there exists varying demand with multiple combinations of production machines capable of satisfying it. Some of the further advantages of the decentralized implementation of this algorithm in respect to the centralized one are: (see, e.g., Lueth and Laengle 1994) modularity: the robots are by nature independent; decentralized knowledge bases: each robot uses a local knowledge base that stores relevant information for the robot and is capable of exchanging this information with other robots; fault-tolerance and redundancy: in case of an error on a singular robot, the multi-robot system continues to function, and extendibility: new robots can be added to the original system without any change in the system architecture.

3 Definitions and Problem Formulation

We consider a system consisting of a set R of n collaborative mobile robot agents positioned in an environment with a set T of n target locations (tasks) and a given $n \times n$ matrix of the distances (or costs) c_{ij} between the robots that are represented by the index j, and the tasks represented by i. Each robot can be assigned to maximally one task, and each task can be performed by not more than one robot. The problem consists of finding a feasible assignment of minimum total cost (distance) between the robots and the tasks, i.e., determining the minimum weight perfect matching of the weighted bipartite graph $G = (T \bigcup R, E)$ with weight c_{ij} on edge $(i, j) \in E$. The mathematical formulation of the problem is:

$$\min \sum_{i,j} c_{ij} x_{ij} \tag{1}$$

subject to

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

and

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

$$x_{ij} \geq 0\ . \tag{4}$$

Although fractional solutions exist, there always exists an optimal integer solution which corresponds to a real matching (see, e.g., Papadimitriou and Steiglitz, 1982). In particular, $x_{ij} = 1$ if the robot j is matched to the task i; otherwise $x_{ij} = 0$.

Denoting the above problem formulation as the primal problem, it is possible to define the dual problem as follows. Let α_i be a dual variable related to task i, and β_j the same related to robot j. Both variables are real and unrestricted in sign. The objective of the dual problem is

$$\max\{\sum_{j} \beta_j - \sum_{i} \alpha_i\} \tag{5}$$

subject to

$$\beta_j - \alpha_i \leq c_{ij} \quad ,\forall\, i\ ,\ \forall\, j\ . \tag{6}$$

It is possible to give an economic interpretation to the dual problem and the dual variables: α_i is the price that each robot j will pay if it gets matched with the task i, while β_j is the robot j's utility for the matching with a certain task. The constraints (6) of the dual problem state that the utility β_j of robot j cannot be greater than the total cost $(c_{ij} + \alpha_i)$ of allocation (matching) to task i. The dual objective function (5) to be maximized is the difference between the sum of the utilities of the robots and the sum of the prices of the tasks, i.e., the total net profit of the robots. On the base of the duality in linear programming,

$\sum_j \beta_j - \sum_i \alpha_i \leq \sum_{ij} c_{ij} \cdot x_{ij}$; therefore, the total net profit of the robots cannot be greater than the total assignment cost that the robots have to pay for moving to the tasks, and only at the optimum, those two are equal. Obviously, at the optimum, each robot j will be matched to task i for which the utility of the robot is $\beta_j = c_{ij} + \alpha_i$, i.e., exactly equal to the total cost that the robot j would have for matching with task i.

4 Distributed allocation algorithm

It is assumed that each robot agent has an information regarding its position and can receive the information regarding the position of all the tasks in the environment through the coordinates on a map of the environment. The inter-robot communication is performed over a connected dynamic communication network and the solution to the assignment problem is reached without any common coordinator or a shared memory of the system.

To solve the problem in this context, we developed a distributed version of a Hungarian Method with the augmenting paths from the graph theory that we recall briefly in the following.

We refer to the centralized (Hungarian) method implemented by means of so-called alternating trees (see, e.g., Burkard and Çela, 1999). The algorithm is iterative and, maintaining a feasible dual solution, considers the admissible bipartite graph $\bar{G} = (T \bigcup R, \bar{E})$ where $\bar{E} = \{(i,j) : c_{ij} + \alpha_i - \beta_j = 0\}$. The algorithm searches for a matching of the maximal cardinality in the graph $\bar{G}$. If the matching is perfect, i.e., every robot is matched with a task, then the matching represents the optimal solution and the algorithm stops. If the matching is not perfect, the algorithm updates the dual variables so as to increase the dual objective function such that at least one new admissible edge is added to $\bar{G}$, and continues with a new iteration.

Given $\bar{G}$, let $M \subseteq \bar{E}$ be the current maximal matching in $\bar{G}$. The edges belonging to M are called matched edges, and the ones in $\bar{E} \backslash M$ are free edges. Given $\bar{G}$ and the current maximal matching $M \subseteq \bar{E}$, the algorithm iteratively improves the matching along augmenting paths over alternating trees in $\bar{G}$. An alternating tree is a subgraph of $\bar{G}$ rooted at some free task vertex t connected to all its adjacent robot vertices through free edges of $\bar{E}$, and with each robot vertex connected to a task vertex through a matched edge. When the above mentioned tree reaches a robot leaf vertex which is adjacent to some free robot vertex r, an augmenting path is found, i.e., a path that lets the augmentation of the cardinality of matching exchanging the free and matched edges. When all the possible augmentation steps are done, the dual variables are updated and the new admissible edges are added. Then a new iteration begins searching for the maximum matching on the new admissible graph (see, e.g., Burkard and Çela, 1999).

In our implementation we maintain the forest F^1 of all the alternating trees rooted in free task vertices. Moreover, we maintain a forest F^2 of the alternating

trees in $\bar{G}$ rooted in robot vertices containing all the robot-task vertices not contained in F^1. Clearly, the latter are also not connected with vertices in F^1.

Initially, the forests are set up in the following way. Forest F^1 is composed of n isolated task vertices. Forest F^2, in the similar way, is made of n isolated robot vertices. Dual variables are assumed to be $\alpha_i = 0$ for all the tasks i, and $\beta_j = \min_i c_{ij}$, for all the robots j, and are updated as follows. Let $minSlack_j = \min_{i \in F^1} (c_{ij} + \alpha_i - \beta_j)$ and n_j be the task corresponding to the $\arg\min$, for each robot vertex $j \in F^2$, and $\delta = \min_{j \in F^2} minSlack_j$. New values of the dual variables are:

$$\alpha_i := \alpha_i - \delta \ \ ,\forall \text{ task vertex } \ \ i \in F^1$$
$$\beta_j := \beta_j - \delta \ \ ,\forall \text{ robot vertex } \ \ j \in F^1$$

After updating the dual variables, even though there might be more than one new admissible edge, w.l.o.g., we can add the new edges one at a time and after adding only one edge, search for a new matching on the augmented admissible graph. The new edge connects the two forests. In more detail, the edge starts at the task vertex t^* of F^1 and ends on the robot vertex r^* of F^2. If r^* is free (unmatched), then there is an augmenting path $(t, \ldots, t^*, r^*)$ starting from the root task vertex of F^1, connected with t^* through the alternating path, and ending in r^*. By exchanging the free and matched edges in this augmenting path we get the augmented matching. Since task vertex t is no more free, all the tree of F^1 rooted in t moves to F^2 and is connected to root r^* over t^*. If r^* is not free, i.e., it is matched, there is no augmenting path and the forest of alternating tree is updated by disconnecting robot r^* with all its subtree from forest F^2 and connecting it with all its subtree to the task vertex t^* in forest F^1.

4.1 Algorithm details

The robots interactively execute the steps of the algorithm based on the local information and the one received through the messages from their neighbors in a communication graph. The structure of the communication graph is made of the forests F^1 and F^2 restricted to robots. The interconnected robots recalculate the structure of the forests each time the forests change. Therefore, the connection graph is updated each time the update of the forests F^1 and F^2 occurs.

Each robot keeps in its memory 4 pointers relating to the robots on its neighboring positions in the tree, i.e., the father robot, the oldest son, next older brother, and next younger brother, as seen from the Fig. 1. The first two pointers are used to move upwards and downwards in the tree while the two latter ones are used to explore a branch. Each robot in its memory also keeps the data about its position (whether it is in F^1 or F^2), its matched task M_j (if there is none, $M_j = 0$), and the father task vertex. Each robot j memorizes also the column vector j of the matrix of the distances, that is, its vector of distances from its position to the positions of all the tasks, its dual variable β_j, and the $minSlack_j$ along with the task n_j which obtains the $minSlack_j$.

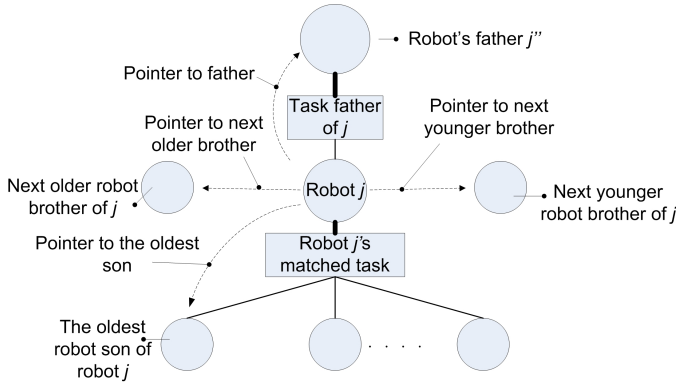

Fig. 1. Four pointers of each robot in the tree

Through autonomous calculations and the communication with the neighbors, robots get and share the information about α_i of all the tasks i, δ value for the dual variables' update, the tasks which are in F^1, the new edge between the forests F^1 and F^2, and the root robots $r(F^1)$ and $r(F^2)$ of forests F^1 and F^2 respectively (the first robot vertex in a depth-first search of the forest). In this way, there is no common coordinator or a shared memory of the robots' system. In detail, the robots, depending on the position in the forests, (e.g., whether they are in the augmenting path, in forest F^1 or F^2, etc.) change their roles, and accordingly do some of the steps of the distributed algorithm. In the following we describe the robots actions according to their roles.

Initialization (done by each robot j):

- Let $M_j := 0$ (robot j is initially unmatched); let $\alpha_i := 0$ for all the tasks i, and $\beta_j := \min_i\{c_{ij}\}$.
- Initialize the pointers in such a way that robot j is connected only with its younger $(j-1)$ and older brother robot $(j+1)$ and has no father and no sons. This means that the forests are initialized as described above.
- Each robot positioned in F^2 calculates $[minSlack_j, n_j]$ without exchanging any message with the others.

The root robots $r(F^1)$ and $r(F^2)$ start the message exchange based on depth-first search (DFS) asking if there is an unmatched robot. In the contact with the first unmatched robot, the same informs $r(F^2)$ to start a new iteration. In the following we describe the actions of the robots in respect to their roles during one iteration.

- All the robots $j \in F^2$ participate in calculating $[\delta, (t^*, r^*)]$. The calculation of δ starts from the root robot $r(F^2)$ and follows via the exchange of messages through the depth-first search of F^2. When the values of $[\delta, (t^*, r^*)]$ are

found, the information is transmitted to all the robots in F^2 through message exchange based on depth-first search (DFS) starting from the root robot of F^2. The same DFS message passing applies to inform the robots in F^1.

- All the robots j do the following: for all $t_i \in F^1$ set $\alpha_i := \alpha_i - \delta$; moreover, if j belongs to F^1, set $\beta_j := \beta_j - \delta$. No messages are exchanged in this phase.
- The father robot of t^* (if any), returns the information about himself and about his oldest son to all the robots through the DFS message passing.
- Root robot $r(F^2)$ informs r^* in F^2 to start the next phase of augmenting or non augmenting the path.
- If r^* is not matched, then it initiates augmenting. Over the message passing from r^* and the robots in the augmenting path, robot r^* orders the swapping of the free and the matched edges along the augmenting path. It sends the messages to the robots in F^2 with the information about the new connection with the subtree of task t^* in F^1. The information contains t^*, and all the objects in the tree going backwards in F^1 up to the first task without father along the augmenting path $(t, \ldots, t^*, r^*)$. The robots simultaneously update the matching in respect to the messages received from r^* and update the pointers to the adjacent robots so that the tree of t^* in F^1 rooted in t moves to F^2 and gets connected to root r^* over t^*. The robots through the DFS message passing get the updated information regarding the position of the tasks in respect to the forests. Each robot positioned in F^2 calculates the new $[minSlack_j, n_j]$ without exchanging any messages with the others.
- If r^* is already matched, then it does not augment the matching. It sends the message to $r(F^2)$ to inform, through the DFS message passing, all the robots in F^2 to decrease minimum slack by δ; r^* disconnects from tree F^2 attaching itself together with its subtree to F^1 over t^*. Each robot in F^2 updates the new minimum slack for each task in the subtree routed at r^* in F^1, which results in $O(n^2)$ messages exchanged.

Regarding the robustness of the method, if the robot during the execution of the algorithm stops responding, it is considered erroneous and is eliminated from the further calculations. In the case where the robot was unmatched in the forest F^2, the calculation continues without any modifications, ignoring the robot in question. Otherwise, the algorithm starts from the beginning excluding the same.

5 Simulation results

The simulation was performed in the C language and executed on a PC with an Intel Core Duo 1.6 GHz CPU and 1 GB of RAM. For each possible n varying from 10 to 250 we considered the average of 10 different instances consisting of sparse random cost matrices with the sparsity varying from 2 to 10% and the cost $c_{ij} \in (0, 100)$.

From the experiments it turns out that the cumulative computational time of the decentralized method, due to the exchange of messages among the robots,

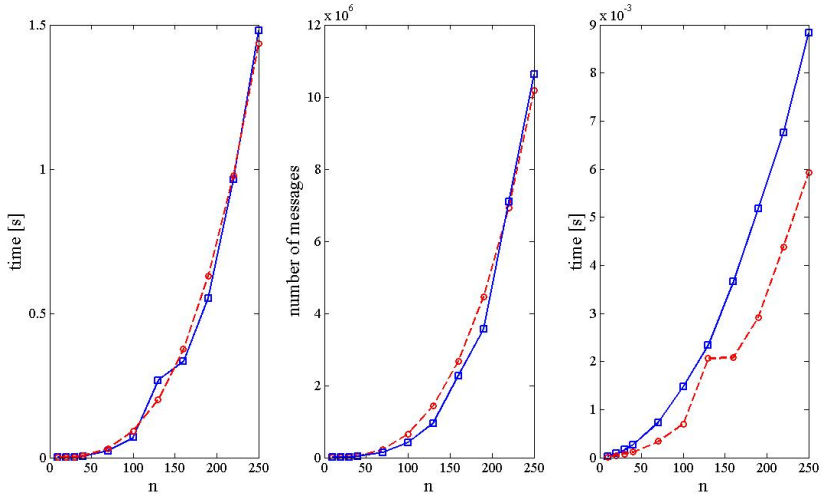

Fig. 2. Left: cumulative time of the decentralized algorithm (solid) vs. n^3 best fitting curve (dashed); center: total number of messages (solid) vs. n^3 best fitting curve (dashed); right: computational time of the (centralized) Hungarian algorithm (solid) vs. computational time of each robot in the decentralized method (dashed).

is from one to two orders higher than the computational time of the centralized algorithm, but is still limited by the $O(n^3)$ time (see Fig. 2, left). The number of messages exchanged among the robots is limited as well by $O(n^3)$ (see Fig. 2, center). Even if the decentralized method is cumulatively heavier, the computational time performed by each robot (in the order of $O(n^2)$) results minor than the time required by the standard (centralized) Hungarian algorithm (see Fig. 2, right).

6 Conclusion

A decentralized implementation of the Hungarian algorithm has been presented to solve an allocation problem for a set of n robots which must execute a given number of tasks in the framework of a decentralized architecture where a centralized controller and a shared memory are not available. The robots must operate autonomously and the decentralized implementation is possible on the basis of an exchange of messages. We have assumed an equal number of robots and tasks but the more general case can be easily handled by introducing fictitious robots/tasks with fictitious costs; also the case where robots can execute only a subset of tasks can be handled with a proper choice of the matrix costs. The cumulative execution time of the decentralized algorithm and the total number of messages exchanged are both in the order of $O(n^3)$, resulting in an average

computational payload $O(n^2)$ for each robot. A decentralized implementation is intrinsically robust: how robot failures can be handled is introduced and will be extended in the future research.

References

Arslan, G. and Marden, J.R. and Shamma, J.S.: Autonomous vehicle-target assignment: A game-theoretical formulation. Journal of Dynamic Systems, Measurement, and Control, 129, 584–596, (2007).

Bertsekas, D.P.: Linear Network Optimization: Algorithms and Codes. MIT Press, Cambridge, MA, (1991).

Bertsekas, D. P.: Auction algorithms for network flow problems: A tutorial introduction. Computational Optimization and Applications, Springer Netherlands 1, (1), (1992).

Bertsekas,D. P., Castanon, D. A., Eckstein,J., and Zenios,S.: Parallel computing in network optimization. Network Models - Handbooks in Operations Research and Management Science, Elsevier, Amsterdam, The Netherlands, 7, 330-399, (1995).

Burkard, R.E. and Çela, E.: Linear assignment problems and extensions. Handbook of Combinatorial Optimization, 4 (1), 221–300, (1999).

Gerkey, B.P., Mataric, M.J.: A framework for studying multi-robot task allocation. In Multi-Robot Systems: From Swarms to Intelligent Automata, A.C. Shultz et al, the Netherlands, Kluwer Academic Publishers, 2 15–26,(2003).

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

Giordani, S., Lujak, M., Martinelli, F.: A Decentralized Scheduling Policy for a Dynamically Reconfigurable Production System. Lecture Notes in Artificial Intelligence 5696, 102–113 (2009) .

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

Kwok, K.S. and Driessen, B.J. and Phillips, C.A. and Tovey, C.A.: Analyzing the multiple-target-multiple-agent scenario using optimal assignment algorithms. Journal of Intelligent and Robotic Systems, Springer, 35 (1), 111–122, (2002).

Lawler, E. L.: Combinatorial Optimization: Networks and Matroids. Holt, Rinehart, and Winston, New York, (1976).

Lueth, T.C. and Laengle, T.: Task description, decomposition and allocation in a distributed autonomous multi-agent robot system. IEEE/RSJ IROS, 1516–1523, (1994).

Papadimitriou, C.H., Steiglitz, K.: Combinatorial optimization: algorithms and complexity. Prentice-Hall, Inc., Englewood Cliffs, N.J., (1982).

Smith, S.L. and Bullo, F.: Target assignment for robotic networks: Asymptotic performance under limited communication. American Control Conference, 2007. ACC'07, 1155–1160, (2007).

Sugar, T., and Kumar, V.: Control and Coordination of Multiple Mobile Robots in Manipulation and Material Handling Tasks. The Sixth International Symposium on Experimental Robotics VI, 15–24, Springer-Verlag (2000).

Wooldridge, M.: Introduction to Multi-Agent Systems. John Wiley and Sons (2002).

Zavlanos, M.M., Spesivtsev, L., and Pappas, G.J.: A distributed auction algorithm for the assignment problem. In Proc. of 47th IEEE Conf. on Decision and Control, Cancun, Mexico, 1212–1217, (2008).