Proceedings of the Federated Conference on Computer Science and Information Systems pp. 1461–1466 DOI: 10.15439/2016F424 ACSIS, Vol. 8. ISSN 2300-5963

Token-based Autonomous Task Allocation in Flocking Systems

Andras Kokuti, Vilmos Simon and Bernat Wiandt
Department of Networked Systems and Services
Budapest University of Technology
HU-1117 Magyar tudosok 2, Budapest
Email: {kokuti,svilmos,bwiandt}@hit.bme.hu

***Abstract*—There are serious contributions to the theoretical foundations of flocking systems, but there are only few systems which have the capability of autonomous task allocation, however, many use cases demand this functionality. The implementation of a task allocation algorithm could be a serious challenge even in a simulated environment due to the numerous problems arising from the nature of these systems.**

This paper proposes a novel algorithm to find the optimal allocation of heterogeneous agents to heterogeneous tasks by utilizing distributed auctions based on local peer-to-peer wireless communication and exploiting graph theory with a tree-based multicast protocol. The solution was tested over a number of different scenarios and compared to existing algorithms in order to measure and prove its capability in handling autonomous task allocation in different systems as well.

I. Introduction

THE FLOCKING phenomena (the notion of flocking is used as a synonym of collective motion), observed in various fields in nature (such as insect swarms, fish schools, herds of wildebeests, etc.), can be an appropriate solution in many engineering applications as well. Applications of flocking include massive mobile sensing in an environment [1]; parallel and simultaneous transportation of vehicles or delivery of payloads [2]; performing military missions, such as reconnaissance [3], surveillance [4], and combat using a cooperative group of Unmanned Aerial Vehicles (UAVs). A flocking group of robots can perform tasks like exploration of an area, autonomous navigation for deployment, surveillance, or search and rescue operations [5].

In contrast to the huge number of flocking algorithms already published in the literature, most of them handle the group as a whole, no dynamic and autonomous reconfiguration or partition is possible [6]. Several use cases require utilizing autonomous regrouping of the flock, where a subset of the flock can leave the group, move to a given destination and perform various tasks on the spot. Several important uses cases impose scenarios where the flock has a large set of nodes, but only a few are required to move towards a specific location, for example when monitoring the behavior of the crowd during mass events or when observing the condition of a dam at multiple points with cameras. In cases where there is only one operation center broadcasting commands, the selection of the subset of nodes acting on the command and routing those nodes to the event site needs to happen autonomously based on the implemented control algorithm and the peer-to-peer communication between the nodes. This is an autonomous task allocation problem, which is an NP-hard [7] combinatorial problem.

In this paper we propose an algorithm capable of choosing the optimal task allocation based on the actual requirements without any central supervision, relying only on local interactions of the nodes. The algorithm does not depend on the tasks or the requirements directly, since it can select the optimal allocation with respect to the criteria defined by the use case.

II. Related Works

Multi agent task allocation is a problem that arises where a number of agents are working together in the aim of achieving a common goal or task which is subdivided into a number of subtasks. The problem can be formulated as a Multiple Traveling Salesmen problem which is NP hard [8]. Many solutions try to approximate the optimal solution by relying on auctions.

There are fully centralized solutions, such as [9] that require a central task allocator to determine the optimal allocation of tasks for the whole system. This approach certainly has advantages, such as the optimal global cost can be calculated easily, but it is usually not feasible in a real life scenario, especially in scenarios mentioned in Section I, as a fully centralized approach requires complete and perfect communication between nodes and the central task allocator has to keep track of the state of the whole system including internal knowledge of each agent's state.

Some solutions utilize combinatorial auction [10](where bidders can bid on combinations of items) in a centralized manner in order to find the optimal task allocation such as in [11] and [12]. Combinatorial auction compared to centralized task allocation has many advantages, however, a few disadvantages as well. Due to the distributed cost computation, this method does not require a fully centralized task allocator to keep track of the internal state of each node in the system. However, it has the communication and computational disadvantages of a fully centralized approach. Combinatorial auctioning can provide globally optimal solution based only on local interactions but it is well known to be an exponential algorithm, therefore, it does not scale well with the number of nodes which can result in extreme energy consumption [13].

Distributed auctions are used to exploit the peer-to-peer communication between nodes in [14] and [12]. This class of methods is the primary mechanism for providing intentional coordination between agents. Distributed auctions have the advantage of not requiring the auctioneer to have a model of each agent (for example a robot). They do not require full communication between the agents and the auctioneer either (which means the network graph does not have to be complete, only connected) and can be robust with respect to communication failures. However, they are inherently suboptimal when a complex global cost function is used.

Fully distributed approaches [15], [16] do not use auctions in a conventional manner, therefore, they do not require direct communication between nodes. Each robot in this case retains local control and chooses its own actions based on its observations of the environment. Cooperative actions however, can be achieved by estimating models of the other agents' strategies. Distributed approaches are robust to communication failures, but can result in worse performance than distributed auctions due to a lack of intentional coordination via an auctioneer.

After studying the listed solutions thoroughly, we have chosen the method of distributed auctions for our algorithm, as it does not require full communication between the agents and being robust to wireless communication failures. The initiator event can be triggered only on one agent, but it uses graph theory solutions to select the most appropriate local auctioneers that do the auctions in a parallel and distributed way. To convey the tasks and the bids between the agents our solution uses tokens which can be transferred via peer-to-peer communication through the whole network.

III. Proposed Distributed Task Allocation Algorithm

We are investigating the problem of allocating a subset of agents based on local events, such as when a node senses disturbances, hazards, etc. or global events: the control center orders nodes to perform tasks. The event can originate from a higher lever entity in the system but the allocation process has to be performed in a self-organized manner among the nodes. A global or local event can be associated with certain requirements, such as a minimum energy level required or various properties of the nodes: speed, agility, maximum operating altitude, presence/accuracy of sensors/actuators, etc. The goal then is to select a subset of the nodes with the best fit with respect to the requirements associated with the event.

A. *Problem formalization*

An event triggers a request in the system that can be translated into a logical entity called the token. In our terminology a token is a simple data structure which contains the requirements (e.g. in a key-value map) associated with the event and nodes can manipulate it during the allocation process. This is very important, since a node needs to add its own data to the token in order to indicate that it has previously seen the token. A token can be transferred between nodes via wireless communication. It follows that a token is the subject of auctions, since every node is competing for the tokens based on the requirements in the token and the node's properties. Therefore, the selection process can be viewed as a highly distributed auction, where the bidders are the agents in the system and the "goods" are the tokens. The main benefit of the token representation of tasks is that the initial abstract problem can be mapped to a more formal problem which now can be solved by using graph theory and other mathematical tools. Thus the autonomous task allocation problem can be formalized as a token translation step and based on the token a distributed auction between the nodes. In the token translation step the algorithm should translate the requirements of the event to a transferable entity called token. And with distributed auctions (starting from the triggered agent) the nodes in the system can exchange the tokens in a controlled manner. At the end of the auction a task allocation can be done based on the bids contained by the token. In order to guarantee the optimal allocation of tokens to nodes, each node must bid on each token at least once, otherwise there could be a node in the system which did not bid to a token but would be the perfect fit.

In a distributed system limiting the number of local auctions is beneficial, since an auction can consume a huge amount of resources. In this case the auctioning algorithm is coupled with a flocking algorithm that depends on wireless communication to synchronize the state of the nodes such as position, velocity and heading. A naive solution would use no auctions at all but nodes that have the token could bid on it. This solution however involves unnecessarily broadcasted messages, as all tokens have to be transferred to potentially all nodes in the system. This approach is one extremity, where the number of auctions is minimized. Thus the algorithm should find an optimal trade-off between the number of the auctions and the required time to complete the allocation process. To minimize used resources and ensure that an optimal solution is found, our algorithm can be broken down to these general steps:

1) Select the auctioneer nodes based on the network graph
2) Build up a multicast tree in which the source is the node where the event was triggered and the destinations are the auctioneer nodes
3) Use the multicast tree to make communication more efficient in the process of allocating nodes to tokens

In the next subsection we focus on the first sub problem, since the second and third can be solved by enhancing traditional tree-based multicast algorithms (will be described in a later subsection).

B. *The auctioneer selection algorithm*

As we are considering mobile agents, the graph representing the network connections is not stable and not known a priori, thus some assumptions have to be made. We assume that each node has a local network table, which contains all other nodes in an adjacency matrix. This table is built up using the periodically received topology information (containing the whole network from the sender's perspective) from the neighbors (similar to the distance-vector routing method [17]). Thus, every node transmits its local network table periodically (like the heartbeat packet in distributed systems) in order to ascertain the node is still operating (since a device can be broken down at any time because of many reasons) and to create a local copy form the whole network topology in all nodes. The next assumption deals with the structure of the network, since in our case the graph is undirected and connected before the task allocation phase. However after it can change, due to for example a task which requires that few nodes from the connected flock move away for a time and do dedicated tasks such as exploration and then join back to the flock. The undirected graph can be guaranteed by allowing only pairwise connections. Thus, if only one node can communicate with the other but the other can not (because of different sizes of transmission ranges) then this asymmetric connection will not be placed in the network matrix and hence the undirected communication graph is guaranteed.

Based on these assumptions auctioneer selection can be formalized and solved by using tools from graph theory. It is a selection problem and the goal is to select the minimum number of auctioneer nodes in the system. Let G denote the graph representation of the whole network, and $G = (V, E)$, where V are the vertices (the nodes) of the graph and E is the set of edges, representing the communication links between the nodes. Thus, the problem is to select a subset (V') from V, which has the minimum number of elements (minimum cardinality set) and can cover all the nodes in the network with the edges covered by the vertexes in the subset. More formally, we are seeking a subset V' of V, such that for all $u \in G$ there exists $(u, v) \in E$, where $v \in V'$ and V' is the minimum cardinality set. This set is very similar to the minimum vertex cover, however, in this case the goal is to cover all nodes instead of all edges.

Assume that every vertex has an associated cost of $c(v) \geq 0$. Then the problem discussed earlier can be formulated as the following integer linear program (x_v is a label which means whether the node is chosen as the element of the minimal set):

$$\text{minimize} \sum_{v \in V} c(v) * x_v \tag{1a}$$

$$\text{subject to} \sum x_u + x_v \geq 1 \ \forall v \in V(G), \ \forall u : (u, v) \in E(G) \tag{1b}$$

$$x_v \in \{0, 1\} \ \forall v \in V(G) \tag{1c}$$

The formulation as an integer linear program can be done in polynomial-time (since from the adjacency matrix the equations can be formulated, processing the whole matrix in O(n^2). As the graph should be processed n times, the reduction can be done in O(n^3)). The 0-1 Integer Linear Programming (a special case of the ILP) problem is one of Karp's 21 NP-complete problems. Thus, we were able to reduce our problem in polynomial-time (i.e. Karp reduction, where $c(v) = 1 \ \forall v$) to an NP-complete problem, therefore, our problem is NP-hard. It follows straight-away from the fact that ILPs are NP hard.

Algorithm 1 Calculating the covering nodes

```
   $V' \leftarrow \emptyset$ (the set of the selected nodes), and $V'' \leftarrow \emptyset$ (the set of covered nodes);
2: while $V \neq \emptyset$ do
       $v \leftarrow$ the node with the maximum degree;
4:     $V' \leftarrow V' \cup v$;
       $V'' \leftarrow V'' \cup v \cup \forall u$ where $(u, v) \in E(G)$;
6:     $E(G) \leftarrow E(G) \setminus (E(G(V'')) \cup \forall E(u, v), where\ u \in V\ and\ v \in V'')$;
       $V \leftarrow V \setminus \forall u$, where $d(u) = 0$;
8: end while
   return $V'$
```

This problem also contains the well-known minimum vertex cover problem as a special case, when in the subject we do not summarize all the neighbors instead one (thus subject to $x_u + x_v \geq 1 \ \forall (u, v) \in E(G)$).

Since the problem is NP-hard, we provide a constructive heuristic algorithm which finds only a suboptimal solution but does it in polynomial time. The main idea behind the algorithm is to select nodes with higher degree as they contain more previously uncovered nodes of the graph. Obviously the degrees of the nodes have to be updated after each selection since the selected node and all of its neighbors are covered. Thus, our algorithm can be described by the following pseudo-code:

Theorem: The Algorithm 1 does find a (sub optimal) solution in polynomial time.

Proof: It is easy to see that the above algorithm does find a solution, since it can stop only after Step 2, and after this step it always stands that $V'' \cup V$ equals to all the vertices in the graph, and it returns only if $V = \emptyset$, thus V'' (which is the set of the covered nodes) contains all the devices in the network. And it can be proved as well, that it stops in polynomial time as the algorithm steps in the loop (Step 1-2) at most n times, where n is the number of vertices in the graph ($n = |V(G)|$). To find the node with the highest degree from the adjacency matrix is O(n) and the update process of the matrix (after removing the vertices and the edges) is O(n^2). Thus, the running time is $O(n*(n+n^2))$ = $O(n^3)$. The $O(n^3)$ time relates to only the auctioneer candidate selection which runs locally on the triggered node and therefore, can be really fast when the adjacency matrices are stored in-memory.

It would be beneficial as well if an upper bound could be determined for the number of selected nodes, since it is a good indicator for the speed of the whole process (includes the distributed auction as well).

Theorem: An upper bound for the number of the covering nodes is $\lfloor |V(G)/2| \rfloor + 1$, where $V(G)$ is the set of the vertices in G.

Proof: It is easy to see, that if we can guarantee this upper bound on a spanning tree of the graph, then the bound is valid for the original graph as well. Since the original graph has more (or at least equal number of) edges than the spanning tree, the nodes we selected based on the spanning tree still cover all vertices in the original graph. In a tree the set we are looking for is identical to the well known minimum vertex cover from graph theory (since in this case to cover all vertices we have to cover all the edges as well). From the Gallai theorem we know that in a graph the sum of the minimum vertex cover and the maximum independent set is equal to the number of vertices. In a tree the number of nodes in a maximum independent set is equal to or greater than $\lceil |V(G)|/2 \rceil$, since trees are bipartite graphs, thus, can be divided into two distinct sets and the vertices of either set are independent from each other. Therefore the number of nodes in the minimum vertex cover is equal to or less then $\lfloor |V(G)/2| \rfloor$. The additional 1 vertex is added to the upper bound because by using a heuristic algorithm in trees it may happen that it selects the wrong one from the two distinct sets (this happens only when the difference between the two sets is only 1).

In this section we have proposed an algorithm able to select a subset of vertices from a connected G graph covering all other vertices in the graph through their edges. We have proved that the algorithm does the selection in polynomial time and we could define an upper bound for the number of selected vertices. Thus, with the help of this algorithm we are able to determine the agents in the system where the local auctions have to be performed in order to find the optimal task allocation.

C. The tree-based multicast algorithm

We have seen previously that the main problem was divided into 3 sub problems and in Section III-B we have provided a heuristic algorithm to approximate the solution for the first subproblem. In this subsection our goal is to identify the appropriate tree-based multicast algorithm to solve the remaining two problems based on the aforementioned selection process result. Thus, the role of the multicast algorithm is to deliver the tokens from the source node to the selected local auctioneer nodes relying on a multicast tree.

The problem of finding a minimum cost multicast tree is well-known as the minimum Steiner tree problem. According to [18] this problem is also NP-complete, even when every edge has the same cost, by reduction from the exact cover by 3-set. Although it is widely assumed that a Steiner tree is the minimal cost multicast tree, it is not generally true in multihop wireless networks [19]. The problem of minimizing the cost of a multicast tree in an ad hoc network needs to be re-formulated in terms of minimizing the number of the data transmissions. Since in a broadcast medium, the transmission of a data packet from a given node to any number of its neighbors can be done with a single transmission, thus, the minimum cost tree is the one which connects sources and receivers by issuing a minimum number of transmissions, rather than having a minimal edge cost. However, finding this optimal tree in a wireless network is also NP-hard [19], therefore, a heuristic algorithm will be used in this case as well.

There are many heuristic algorithms to compute minimal Steiner trees: for instance, the MST algorithm in [20] provides a 2-approximation, and Zelikovsky [21] proposed an algorithm which obtains a 11/6-approximation. However, these algorithms try to solve the minimum cost tree in general instead in a broadcast manner. Authors in [19] proposed two heuristics (a centralized and a distributed one), both resulting in a tree with lower or equal data-overhead then the MST Steiner tree. Since in our case each node knows the whole network (because of the locally maintained adjacency matrices), the centralized solution seems to be an appropriate choice.

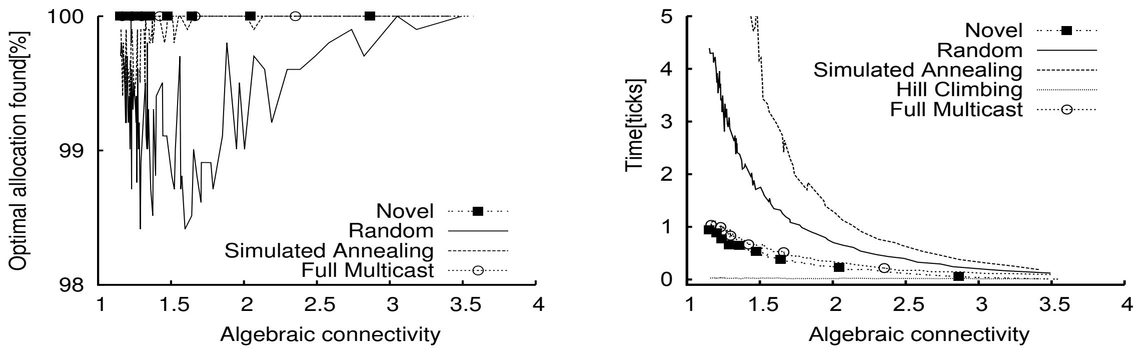
Thus, in the first step of the algorithm it selects the most appropriate auctioneer nodes, in the second step it creates a multicast tree where the multicast receivers are the nodes from the first step. This tree is not only usable for spreading the tokens from the source to the receivers (local auctioneers) but for the opposite direction as well. Therefore, if an auctioneer finishes the local auction, it transfers its token back on the multicast tree to the source. If a node on the path from auctioneer to source possesses two or more tokens at a time, it merges them in order to decrease the number of broadcast transmissions. Finally the first node (in worst case the source) which obtains all the tokens, immediately calculates the winner nodes and instructs them to carry out the tasks formulated in the token. Thus basically, only the auctioneer node selection step runs locally on a single node (where the task allocation event has been triggered) and then the algorithm uses distributed auction backed by multicast trees in order to increase the efficiency of the local auctions.

IV. Performance through Simulation

In the previous section we have proven that our algorithm can find the optimal allocation and does it in polynomial time, but our objective is to evaluate the effect of the proposed auction based task allocation algorithm in the context of a flock environment. Thus we have chosen to conduct a simulation study using the SimPy process-based discrete-event simulation framework [22] in a realistic scenario.

The simulations are based on a static context, thus the agents in our simulation do not move, since primarily our goal is to prove the algorithm can operate optimally in cases where the dynamics of the agents in the system are negligible compared to the communication speed. The nodes in the simulation communicate with each other via a radio interface, such as Wi-Fi, Bluetooth or anything else in a predefined communication range. It is important to note that in the simulation a perfect communication model is assumed as well, therefore, interference and packet loss are not considered.

To model the communication graph we used instances of the connected Watts and Strogatz graph [23], which is a random graph and can be parametrized to resemble the communication graph of flock. The communication network of a flocking system is a type of graph which is similar to a regular grid, therefore the valency of the nodes are nearly the same (since the agents in the flock are placed almost the same distance from each other). We generate a random connected Watts and Strogatz graph by setting the algebraic connectivity parameter as well. The algebraic connectivity of a graph is the second-smallest eigenvalue of the Laplacian matrix of the graph and



(a) For a clearer picture only the optimal or near to optimal solutions are depicted.

(b) Speed of the selected algorithms.

Fig. 1. Simulation results of the autonomous task allocation from all the aspects.

the magnitude of this value reflects how well connected the overall graph is. It depends on the number of vertices as well as the way in which they are connected. In random graphs, the algebraic connectivity decreases with the number of vertices, and increases with the average degree (and is greater than 0 if and only if the graph is connected). The simulations are performed with different algebraic connectivities in order to simulate different kind of flocks. Every graph with a unique algebraic connectivity value was tested 100 times and the results were aggregated in order to minimize the impact of outlier cases.

As it has been stated before, the requirements generated by an input event are translated into a token. However, our framework is capable of handling multiple types of requirements (and cost functions as well) since the algorithm does not restrict it. Thus it is really important to note that the proposed system does not depend on the type of the requirements or the cost function, instead these can be defined differently from case to case and the algorithms will adopt dynamically. In this paper we have considered a simple scenario where a token only contains a minimal energy level as requirement. An energy level is generated for each agent before the simulation, following a uniform distribution between 70% and 100%. In the generated event, the energy requirement is 65%, therefore every node can perform the task to ensure that a solution exists. In this configuration there are optimal and suboptimal allocations based on the generated energy level distribution.

Five different algorithms have been compared against each other in the simulator. The random walk algorithm, which transfers the token (the translated request) on randomly selected edges until some conditions (such as all nodes placed a bid or the token has been transferred at a given threshold) are fulfilled. The well known hill climbing algorithm, that transfers the token only to a node that has a higher bid than the current one, therefore, in our case it means that the tokens move towards nodes with higher energy. The simulated annealing algorithm: it tries to avoid suboptimal selection by allowing temporary worse states (so the selected node's bid is lower than the current highest). The algorithm uses acceptance probabilities of making the transition from a current node to a candidate one, which depends on the energy of the nodes and on a globally-varying time parameter, called the temperature.

A. Simulation Results

The results will be shown in two different aspects. The first is the main focus of the paper, that relates to the success of the task allocation. Measuring the success ratio of finding the optimal allocation among the nodes, 0% to 100% indicates the effectiveness of the used solution. And the second aspect focuses on the simulation time, which indicates the speed of the algorithm. Therefore, the goal is to minimize the time and select the most appropriate agents as quickly as possible.

Figure 1a presents the results from the first aspect. It has to be noticed that the hill climbing algorithm is not depicted since it performs really bad (around 40% at lower density and improves to around 80% in more connected networks) because its performance heavily depends on the location of the initial request event. And obviously the hill climbing algorithm produces the worst selections as it chooses nodes greedily, therefore it terminates at suboptimal nodes (local optimums). Therefore, for a clearer picture only solutions with close to the optimal performance are shown. When observing the percentage of the optimal coverage it could be seen that there are only two algorithms able to achieve the optimal selection regardless of the structure of the network graph: our proposed algorithm and the full multicast algorithm. The full multicast algorithm finds the optimal allocation in any case by definition, since it transfers the token to every node. The simulated annealing algorithm results in an almost optimal selection in all cases, but in really sparse environments (graphs with lower algebraic connectivity) it has some errors. The relatively good results of this algorithm is because it's parameters (such as the initial and the terminating temperature and the transfer rate) have been trained on many networks. The random solution (which passes the token to randomly selected nodes) also achieves a nearly optimal selection in all cases, since its input parameter (the number of token transfers) has also been trained on the same dataset. It is important to notice as well that the results are improved when increasing the algebraic connectivity of the graphs, as higher algebraic connectivity values mean a denser and more connected network graph. Therefore local information becomes more global, in the extreme case of a fully connected graph each node has global knowledge about the whole graph.

Our final results are depicted on Figure 1b, and it shows the speed (the simulation time measured in ticks by the discrete simulator) of the different selection solutions. As it can be observed, all algorithms except the hill climbing are highly dependent on the algebraic connectivity. The relation between the algebraic connectivity and the speed is not linear but exponential, therefore, a less connected graph results in much slower selection. Hill climbing performs the best with respect to speed due to the greedy optimization: blindly passing tokens to agents with higher utility. On one hand this algorithm can terminate in local optimum instead of a global one (see Figure 1) but on the other hand it provides the fastest coverage. The other four examined solutions perform better if the underlying network graph is more connected, since in that case the network can be covered within less time. The second most efficient solution is our proposed algorithm, which can outperform the other three optimization techniques even by 30-40$\%$ in less connected environments. Based on the results from all the aspects, we can conclude that our solution finds the optimal selection in any network and does it in the fastest way among the algorithms which find the optimal or nearly optimal allocations. Only the hill climbing search technique does faster allocation but it can reach an optimal selection only in strongly connected flocking systems.

V. Conclusions

We have proposed and described a novel distributed auction based task allocation algorithm to enhance flocking systems. This task allocation algorithm exploits the network graph maintained locally by nodes in order to build a (sub) optimal multicast path from the source node to the chosen local auctioneers. Using this multicast tree our solution can find an optimal allocation based on the generated requirements.

Real-life flocking examples were used as a case study to evaluate how our novel algorithm can solve the distributed task allocation problem in flocking systems. We evaluated the performance based on multiple environments by varying the algebraic connectivity of the generated graphs. Our experimental results indicate that the proposed algorithm can find the optimal allocation regardless of the algebraic connectivity of the graph and does it relatively fast compared to the other examined solutions such as random walk, hill climbing, simulated annealing and the full multicast algorithm.

References

[1] H. M. La, W. Sheng, and J. Chen, "Cooperative and active sensing in mobile sensor networks for scalar field mapping," *Systems, Man, and Cybernetics: Systems, IEEE Transactions on*, vol. 45, no. 1, pp. 1–12, 2015. http://dx.doi.org/10.1109/tsmc.2014.2318282

[2] X. Wang, J. Qin, and C. Yu, "Iss method for coordination control of nonlinear dynamical agents under directed topology," *Cybernetics, IEEE Transactions on*, vol. 44, no. 10, pp. 1832–1845, 2014. http://dx.doi.org/10.1109/tcyb.2013.2296311

[3] X. Zhu, C. Wei, H. Duan, and Q. Li, "Some new results on bees-mechanism-based flock control with neighbors chosen by topological distance," in *Guidance, Navigation and Control Conference (CGNCC), 2014 IEEE Chinese*, pp. 2681–2686, IEEE, 2014. http://dx.doi.org/10.1109/cgncc.2014.7007591

[4] S. H. Semnani and O. A. Basir, "Semi-flocking algorithm for motion control of mobile sensors in large-scale surveillance systems," *Cybernetics, IEEE Transactions on*, vol. 45, no. 1, pp. 129–137, 2015. http://dx.doi.org/10.1109/tcyb.2014.2328659

[5] S. K. Lee, "Distributed space coverage for exploration, localization, and navigation in unknown environments," 2015.

[6] B. Wiandt, A. Kokuti, and V. Simon, "Application of collective movement in real life scenarios: Overview of current flocking solutions," *Scalable Computing: Practice and Experience*, vol. 16, no. 3, pp. 233–248, 2015. http://dx.doi.org/10.12694/scpe.v16i3.1099

[7] B. P. Gerkey and M. J. Matari, "Sold!: Auction methods for multirobot coordination," *Robotics and Automation, IEEE Transactions on*, vol. 18, no. 5, pp. 758–768, 2002. http://dx.doi.org/10.1109/tra.2002.803462

[8] M. Badreldin, A. Hussein, and A. Khamis, "A comparative study between optimization and market-based approaches to multi-robot task allocation," *Advances in Artificial Intelligence*, vol. 2013, p. 12, 2013. http://dx.doi.org/10.1155/2013/256524

[9] M. Koes, K. Sycara, and I. Nourbakhsh, "A constraint optimization framework for fractured robot teams," in *Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*, pp. 491–493, ACM, 2006. http://dx.doi.org/10.1145/1160633.1160724

[10] P. Cramton, Y. Shoham, and R. Steinberg, "Combinatorial auctions," 2006.

[11] M. Mito and S. Fujita, "On heuristics for solving winner determination problem in combinatorial auctions," *Journal of Heuristics*, vol. 10, no. 5, pp. 507–523, 2004. http://dx.doi.org/10.1023/b:heur.0000045322.51784.2a

[12] K. Zhang, E. G. Collins Jr, and D. Shi, "Centralized and distributed task allocation in multi-robot teams via a stochastic clustering auction," *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, vol. 7, no. 2, p. 21, 2012. http://dx.doi.org/10.1145/2240166.2240171

[13] T. Sandholm, "Algorithm for optimal winner determination in combinatorial auctions," *Artificial intelligence*, vol. 135, no. 1, pp. 1–54, 2002. http://dx.doi.org/10.1016/s0004-3702(01)00159-x

[14] C. M. Clark, R. Morton, and G. A. Bekey, "Altruistic relationships for optimizing task fulfillment in robot communities," in *Distributed Autonomous Robotic Systems 8*, pp. 261–270, Springer, 2009. http://dx.doi.org/10.1007/978-3-642-00644-9_23

[15] A. Wagner and R. Arkin, "Multi-robot communication-sensitive reconnaissance," in *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on*, vol. 5, pp. 4674–4681, IEEE, 2004. http://dx.doi.org/10.1109/robot.2004.1302455

[16] R. Powers and Y. Shoham, "New criteria and a new algorithm for learning in multi-agent systems," in *Advances in neural information processing systems*, pp. 1089–1096, 2004.

[17] C. Perkins, E. Belding-Royer, and S. Das, "Ad hoc on-demand distance vector (aodv) routing," tech. rep., 2003. http://dx.doi.org/10.17487/rfc3561

[18] R. M. Karp, *Reducibility among combinatorial problems.* Springer, 1972. http://dx.doi.org/10.1007/978-1-4684-2001-2_9

[19] P. M. Ruiz and A. F. Gomez-Skarmeta, "Approximating optimal multicast trees in wireless multihop networks," in *Computers and Communications, 2005. ISCC 2005. Proceedings. 10th IEEE Symposium on*, pp. 686–691, IEEE, 2005. http://dx.doi.org/10.1109/iscc.2005.34

[20] L. Kou, G. Markowsky, and L. Berman, "A fast algorithm for steiner trees," *Acta informatica*, vol. 15, no. 2, pp. 141–145, 1981. http://dx.doi.org/10.1007/bf00288961

[21] A. Z. Zelikovsky, "An 11/6-approximation algorithm for the network steiner problem," *Algorithmica*, vol. 9, no. 5, pp. 463–470, 1993. http://dx.doi.org/10.1007/bf01187035

[22] N. Matloff, "Introduction to discrete-event simulation and the simpy language," *Davis, CA. Dept of Computer Science. University of California at Davis. Retrieved on August*, vol. 2, p. 2009, 2008.

[23] D. J. Watts and S. H. Strogatz, "Collective dynamics of small-world networks," *nature*, vol. 393, no. 6684, pp. 440–442, 1998.