Multi-Agent Coalition Formation for Distributed Area Coverage: Analysis and Evaluation

Ke Cheng
Computer Science Dept.
University of Nebraska
Omaha, NE 68182, USA
kcheng@mail.unomaha.edu

Prithviraj Dasgupta
Computer Science Dept.
University of Nebraska
Omaha, NE 68182, USA
pdasgupta@mail.unomaha.edu

***Abstract*—In the multi-robot area coverage problem, a group of mobile robots have to cover an initially unknown environment using a sensor or coverage tool attached to each robot. Multi-robot area coverage is encountered in many applications of multi-robot systems including unmanned search and rescue, aerial reconnaissance, robotic demining, automatic lawn mowing, and inspection of engineering structures. We envisage that multi-robot coverage can be performed efficiently if robots are coordinated to form small teams while covering the environment. In this paper, we use a technique from coalitional game theory called a *weighted voting game* that allows each robot to dynamically identify other team members and form teams so that the efficiency of the area coverage operation is improved. We propose and evaluate a novel technique of computing the weights of a weighted voting game based on each robot's coverage capability and finding the best minimal winning coalition(BMWC). Also we designed a greedy method and a heuristic method to find BMWC in $O(n\ log\ n)$ time and $O(n^2)$ time respectively. There is a trade-off between computational time and the optimal solution.**

***Keywords*-Area Coverage, Weighted Voting Games, Winning Coalitions**

I. Introduction

In recent years there has been an increasing interest for area coverage using multi-robot systems. Compared with single robot coverage, multiple robots have potentially better performance because increasing the number of robots reduces the time taken to cover the area. Additionally, multi-robot systems offer robustness of the system against failure of individual robots[4], [8]. However, it is challenging to design efficient techniques for coordinating multi-robots to perform area coverage. As Hazon *et al.* [7] mention, optimal single robot coverage algorithms cannot be directly used or produce identical results in multi-robot cases. Area coverage with multi-robot systems involves challenges such as exchanging coverage maps judiciously between robots, allowing robots to move intermittently out of each other communication ranges and dispersing robots in adequate numbers forwards uncovered regions. To address these challenges, we envisage that multi-robot team formations can be used as an appropriate technique the multi-robot coverage problem. Multi-robot collaborations and cooperative decision-making play an important role on multi-robot team formation because the achievement of a robot team is based on the performance of each robot. We propose to use weighted voting games (WVG), which provide a model of decision-making in many political voting procedures, as a model for structured team formation in our multi-robot system. In a WVG, each player has a weight, and a coalition of players wins the game if the sum of the weights of its participants exceeds a certain threshold. As weights can affect the outcome, one of the key issues to use this model in multi-robot area coverage is how to dynamically compute such a weight for each robot. Finding a stable and unique coalition which is a well-known problem in coalitional games [10], is another practical challenge.

In this paper, we illustrate a method of calculating the weight of a robot that can be used in a WVG, based on its coverage history. Also, we extend the weighted voting game with robot domain knowledge to calculate a unique coalition called the best minimal winning coalition(BMWC). We give two acceptable methods to find BMWC. Experiment results show that the greedy method use least of the computational time to find the approximate BMWC solution, and the heuristic method can find the optimal solution.

II. Preliminary Definitions

Weighted Voting Games A weighted voting game G is a simple game that is described by a vector of players' *weights* **w** = $(w_1, ..., w_n)$ and a *quota* **q** [11]. In such games, a coalition is winning if the sum of its total weights meets or exceeds the threshold quota **q**. Formally, weights w_i assigned to each player $i \in N$. Let W be $\sum_{i \in N} w(i)$, and $q \in [0, W]$. For any $C \subseteq N$, $v(C) = 1$ if $\sum_{i \in N} w(i) > q$; $v(C) = 0$ otherwise.

Minimal Winning Coalition S is a winning coalition, if $\forall p_i \in S$, the coalition $T = S - \{p_i\}$ is a losing coalition, S is denoted as minimal winning coalition [12]. For example, let us consider the game $G = (N, q : 6, 3, 2)$, which has three players $N = \{n_1, n_2, n_3\}$ who cast 6, 3, and 2 votes, respectively. If we set the quota q at the ”majority” level,

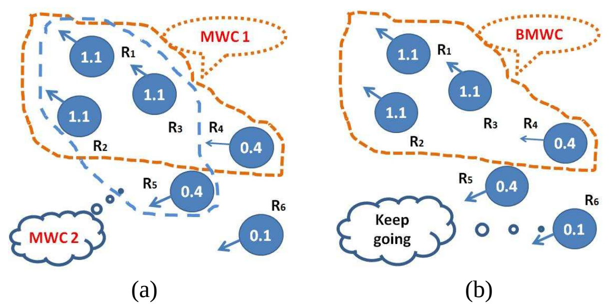

Figure 1. (a) 6 robots calculate two minimal winning coalitions in the communication range , (b) 4 robots form a best minimal winning coalition and head to a new team direction.

$q = 6$, then we have the following collection of minimal winning coalitions: $\{n_1\}$. So the veto player of the game is n_1. Players n_2 and n_3 are dummies in this game because they are not members of any minimal winning coalition. If we set quota q less than ”majority” level, for example, $q = 5$, the minimal winning coalitions of this weighted voting game are $\{n_1\}$ and $\{n_2, n_3\}$. There are no veto players or dummy players.

Core A payoff vector x is in the core of a coalitional game (N,v) if and only if $\forall S \subseteq N, \sum x_i \geq v(S)$ [11]. It is the significant solution concept of coalitional games. Intuitively, no players plan to leave the coalition C, which is in the core, because they can not get better payoffs by forming any other coalitions. The core provides the stability for coalitional games.

Weighted voting games are widely used representations of coalitional games in practice. With exponentially large number of players, [6] shows that the computational complexity of coalitional games is infeasible.

III. System Modeling and Algorithm Design

A. *Single Robot Coverage Capability*

The goal of multi-robots area coverage is to search or explore an unknown environment efficiently while reducing the redundancy of the area covered. This implies that each robot has to maximize the area it can cover within a certain time period, and simultaneously reduce the overlap between regions covered by itself and other robots. To facilitate our modeling, we posit that the contribution of a robot towards the team it belongs to is measured in teams of the area of a region the robot covers within a certain, fixed number of timesteps T. We call this metric the robot 's *coverage capability*. For physical robots, the coverage capability can vary dynamically depending on the local environment conditions, such as the size and number of obstacles, as well as intrinsic on robot capabilities including motor speed, and sensor detectable range. We propose to use robot coverage capability as the weights in a WVG.

We model the environment as a grid-based graph. Each cell in the graph is the area which a robot can cover in one time step. We assume that each robot has 4-DOF. The coverage map maintained by a robot r is used to store the regions (number of cells) in the vicinity of its current location that it has covered. A robot updates its coverage map every T timesteps. Each location in the coverage map is associated with a numeric value calculated using a node-counting technique [8], which represents the number of times the location has been visited by the robot.

B. *Best Coalition Formation*

Coalition structures represent the different combinations of teams which a robot can form within other robots within its communication range. Every time, there are two or more teams of robots in each other's communication range, we can search the coalition structures in the solution space. Our goal is to find an efficient partition, which increases the payoff of the robot team without reducing the payoff of any single robot. To get a winning coalition, we can set a robot's own weight as a linear function of the robot coverage capability defined by $w_i = \alpha \times C_i + \beta$, where C_i is the robot's coverage capability introduced in former section, and α and β are the adjustment constants to make w_i remain within a certain interval. In each round of the voting game, the weight w_i of each robot and the quota q are both fixed values. A problem with applying the winning condition from general WVGs to our context is that sometimes there could be more than one minimal winning coalition. Suppose there are 6 robots with the weight set $\{1.1, 1.1, 1.1, 0.4, 0.4, 0.1\}$, and quota $q = 3.5$, as shown in Figure 1 (a). We can get two winning coalition $\{r_1, r_2, r_3, r_4\}$ and $\{r_1, r_2, r_3, r_5\}$. As the minimal winning coalition is not unique, we design a function $\xi = f(x_k, d_{i,j}, \varphi_{i,j}) = argmin_{S\in I}(\frac{\sum_{i,j,k\in S} g\times x_k + e\times d_{i,j} + f\times \varphi_{i,j}}{|S|})$. We call the coalition which has the optimized value ξ the best minimal winning coalition(BMWC). The value x_k is a prime number, which can be a unique number such as the robot's ID. $d_{i,j}$ is the distance between two robots i and j. $\varphi_{i,j}$ is the angle between the two robots i and j. e , f ,and g are adjustment constants. I is the set of minimal winning coalitions, and S is a minimal winning coalition in I. ξ has the minimum value of each minimal winning coalition. The definition of ξ considers the angle and distance between robots in the minimal winning coalitions as well as the intrinsic value of, such as an id of each robot to come up with a unique value for each coalition. An example of determining the BMWC is shown in Figure 1 (b). If there is no winning coalition, we just keep the status of each robot unchanged. We can find the unique and stable BMWC of the voting game as proved in [2].

```
function FindVetoPlayers returns set $I${
inputs: set $N$, double $Quota$ ;
variables:
        player $i$;
        double $W$, $W_{-i}$;

$W = \sum_{i \in N} w_i$;
for each $i \in N$
        $W_{-i} = \sum_{j \in N-\{i\}} w_j$;
        if($W_{-i} < Quota$) $I = I \cup \{i\}$;
}
```

Figure 2. The algorithm with linear time used to find veto players in weighted voting games.

C. Algorithm analysis of finding the BMWC

There are three steps to find the BMWC [2]. First, we need to find the veto players in the communication range that have more coverage capability than the non-veto players. In other words, veto players can get benefits in the winning coalition. Next, we can use the set of veto player to find the minimal winning coalition set. Finally, we can find the unique BMWC while different strategies. In [2], we described an $O(n^3)$ time complexity method as the baseline algorithm to fulfill these steps. Here we describe an algorithm that can reduce the worst case time complexity for finding the BMWC from cubic time to log-linear time without guaranteeing optimality. To guarantee that the BMWC is optimal, we also describe a quadratic time algorithm.

Veto players are the players who appear in all winning coalitions. Therefore, if we remove all the veto players from a certain coalition, the resulting coalition should not remain a winning coalition anymore. We use this intuition to determine the set of veto players in linear time as shown in Figure 2. First, we calculate W, the sum weights of all the players in linear time. Then, for each player i, we calculate W_{-i} as the sum of weights excluding that player. We add the player i to the set of veto players I if the W_{-i} falls below the quota without player i.

To get the unique BMWC, we propose two methods: a greedy method and a heuristic method. The heuristic method can find the optimal solution in $O(n^2)$ time. In contrast, the greedy method can only guarantee to find a suboptimal solution. However, it can find this solution in $O(n\ log\ n)$ time. Thus, there is a tradeoff between the computational time and the optimal solution. First, we sort the non-veto player set in descending order by a single robot's weight. By selecting the sorting algorithm judiciously and assuming that the veto player weights are in randomized order, the worst case time complexity of the sorting algorithm becomes $O(n\ log\ n)$. Although there can be more than one MWC, we only choose the first MWC we get as the BMWC.

The heuristic method gives a way to find the optimal coalition. By checking every member of all MWC sets with all members of the veto player set, we can find the optimal coalition with the minimal ξ value. Instead of calculating the distance and the angle between non-veto players to veto players in each MWC, we compute these values with the virtual player P_c, which is the centroid of all the veto players. Finally, we remove veto players inside each of the MWCs. In the worse case and the average case, the computational time is $O(n^2)$. However, in the best case, the computational time becomes linear.

IV. Adaptive Flocking Teams with WVGs

A. Team shape maintenance

In the former section, we described how robots find the best team based on a weighted voting game module. After they built their group, how to keep robots moving in a particular formation is another important issue. In other words, in the higher level, robots select a team based on maximizing their single utility or payoff to fulfill a certain task. In the lower level, robots need to keep their team in an optimal physical shape to get the most team utility and reduce extra moving costs. Our goal is to keep the robot team moving efficiently while maintaining a certain shape. We use our former 'V' shape flocking module, which has a team leader and several followers [1], [5] to achieve this. Particularly, we select the robot with the highest weight as the team leader in the team. In other words, the team leader has the highest coverage capability.

B. WVGs for team (re)formations

Based on calculating the best minimal winning coalition, a robot can make the decision of joining a team or leaving a team. If a team member's coverage capability is low, no matter it is the leader of the team or the follower of the team, it will not be in the new BMWC. When two or more than two teams meet in the communication range, based on recalculation of the BMWC, team leaders and team followers may exchange their roles inside the team. In this process of reformation, the number of robots of each team can also be increased or decreased to gain the maximum coverage capability. In short, the stability and uniqueness of BMWC guarantee the distributed decision-making of each robot to achieve optimal team coverage.

V. Experiment and Evaluation

To evaluate the run time of three algorithms to find BMWC, we implemented them in a laptop computer with 2 GHz processor run time and 1 GB memory. These hardware are similar to the hardware of CoroBot [3]. We ran the three algorithms respectively over 10 times using 10 to 100 robots, as shown in Figure 3 in the logarithm scale. In fact, these three line should show no-linear relationship in the normal scale. Not surprisingly, the greedy algorithm ran fastest within 20 millisecond for all the tests. The lowest

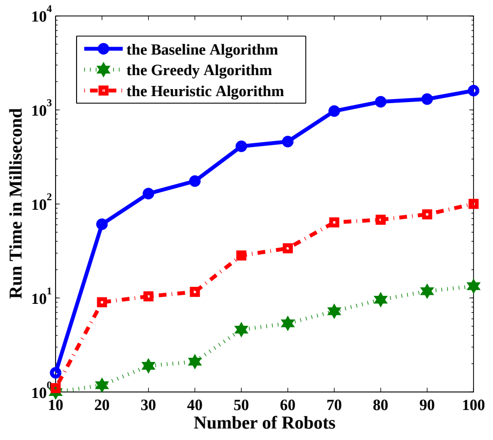

Figure 3. The running time (in the logarithm scale) of three algorithms for finding the BMWC with different numbers of robots.

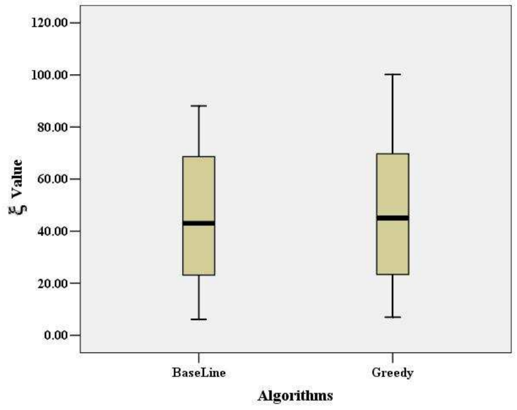

Figure 4. The ξ values of the baseline algorithm and the greedy algorithm using different numbers of robots.

one is the base line algorithm, which ran over 1 second for 100 robots. The middle one is the heuristic algorithm which has 10 to 100 millisecond run time by increasing the number of robots. All these algorithms ran very fast around 1 millisecond when there are less than 10 robots. However, if the numbers of robots or agents are really large, the greedy algorithm becomes a better choice.

As the greedy algorithm is an approximate solution. We checked the ξ value between the baseline algorithm and the greedy algorithm, as shown in Figure 4. The average ξ values of the two different algorithms are nearly the same. The range of ξ value is larger for the greedy algorithm than the base line algorithm.

VI. Future Work

In this paper, we proposed and verified a technique of computing the weights of a weighted voting game based on robot's coverage capability and finding the stable and unique best minimal winning coalition. We studied two methods to find BMWC. The result of our experiments show that there is a trade-off between computational time and the optimal solution. We plan to evaluate the robustness of our approach to sensor and actuator noises. We are working towards evaluating our technique with physical robots, such as the CoroBot. We envisage that this technique can support heterogeneous robots with different memory and communication limitations as well.

References

[1] K. Cheng, P. Dasgupta, Yi Wang "Distributed Area Coverage Using Robot Flocks", World Congress on Nature and Biologically Inspired Computing (NaBIC'09), Coimbatore, India, pp. 678-683, 2009.

[2] K. Cheng, P. Dasgupta, "Weighted voting game based Multi-robot team formation for distributed area coverage," 3rd Practical and Cognitive Agents and Robots (PCAR) Workshop, (co-located with AAMAS 2010), Toronto, Canada, pp. 9-15, 2010.

[3] http://robotics.coroware.com/corobot, Accessed by March 25th, 2010.

[4] N. Correll, "Coordination Schemes for Distributed Boundary Coverage with A Swarm of Miniature Robots: Synthesis, Analysis and Experimental Validation", Ph.D. Dissertation, Ecole Polytechic Federale Laurence, 2007.

[5] P. Dasgupta, K. Cheng, and L. Fan, "Flocking-based Distributed Terrain Coverage with Mobile Mini-robots," IEEE Swarm Intelligence Symposium, pp. 96-103, 2009.

[6] X. Deng and C. H. Papadimitriou, "On the complexity of cooperative solution concepts", Math. of Oper. Res., vol. 19, no. 2, pp. 257-266, 1994.

[7] N. Hazon, G. Kaminka, "On Redundancy, Efficiency, and Robustness in Coverage for Multiple Robots", Robotics and Autonomous Systems, Volume 56 , Issue 12, pp. 1102-1114, 2008.

[8] S. Koening, B. Szymanski, Y. Liu, "Efficient and Inefficient Ant Coverage Methods", Annals of Mathematics and Artificial Intelligence, Vol 31, no. 1-4., pp. 41-76, 2001.

[9] O. Michel, "WebotsTM: Professional mobile robot simulation," International Journal of Advanced Robotics Systems, vol. 1, no. 1, 2004.

[10] R. B. Myerson, "Game Theory", Harvard University Press, 1997.

[11] Y. Shoham and K. Leyton-Brown, "Multiagent Systems: Algorithmic, Game Theoretic and Logical Foundations", Cambridge University Press, 2009.

[12] A. Taylor, W. Zwicker, "Simple Game: Desirablility Relations, Trading, Pseudoweightings", Princeton Univerity Press, Princeton, 1999.