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. However, it is challenging to design efficient techniques for coordinating multi-robots to perform area coverage. As Hazon et al. 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[11], 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 the BMWC. Experiment results show that the greedy method uses lower computational time to find the approximate BMWC solution, while 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 $w = (w_1, \ldots, w_n)$ and a quota $q$ [12]. 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 [13]. 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 \(\forall S \subseteq N, \sum x_i \geq v(S)\) [12]. 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. In [7], Elkind gives a dynamic programming method to solve weighted threshold game in polynomial time.

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 minimal winning coalitions as well as the intrinsic value. The definition of \(\xi\) considers the angle and distance between robots in the minimal winning coalitions as well as the intrinsic value. An example of determining the BMWC is 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 \(\xi = f(x_k, d_{i,j}, \varphi_{i,j}) = \arg\min_{S \in I} x_k + e \times d_{i,j} + f \times \varphi_{i,j}\). We call the coalition which has the optimized value \(\xi\) 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 \(j\). \(e, f, q\) are adjustment constants. \(I\) is the set of minimal winning coalitions, and \(S\) is a minimal winning coalition in \(I\). \(\xi\) has the minimum value of each minimal winning coalition. The definition of \(\xi\) considers the angle and distance between robots in the minimal winning coalitions as well as the intrinsic value. 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].
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 with 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 $W_i$, the sum weights of all the players in linear time. Then, for each player $i$, we calculate $W_{\sim i}$ as the sum of weights excluding that player. We add the player $i$ to the set of veto players $I$ if the $W_{\sim i}$ 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 $\xi$ 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 $P_v$, 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 using a weighted voting game. After they build their group, how to keep robots moving in a particular formation is another important issue. In other words, at a higher level, robots select a team based on maximizing their single utility or payoff to fulfill a certain task. At the lower level, robots need to keep their team in formation by maintaining a certain shape while moving. We use our former ‘V’ shape flocking formation technique, which uses leader referenced flocking with 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. This means that 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 if it is the leader of the team or a follower in the team, it will not be in the new BMWC. When two or more than two teams meet within each other’s 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 on a laptop computer with 2 GHz processor run time and 1 GB memory. This hardware is similar to the hardware of a CoroBot robot [3]. We ran the three algorithms respectively over 10 times using 10 to 100 robots, as shown in Figure 3 in the logarithm scale. Not surprisingly, the greedy algorithm ran fastest within 20 millisecond for all the tests. The lowest one is the baseline algorithm, which ran for over 1 second for 100 robots. The middle one is the heuristic based algorithm which has 10 to 100 millisecond run time while increasing the number.
of robots. All these algorithms ran at around 1 millisecond when there are less than 10 robots. However, if the numbers of robots or agents are very large, the greedy algorithm becomes a better choice.

As the greedy algorithm is an approximate solution. We checked the $\xi$ value between the baseline algorithm and the greedy algorithm, as shown in Figure 4. The average $\xi$ values of the two different algorithms are nearly the same. The range of $\xi$ 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 the 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.

REFERENCES


