Weighted Voting Game Based Multi-robot Team Formation for Distributed Area Coverage

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). We theoretically prove the feasibility of our model, and give algorithms to find the BMWC as well. We have also evaluated the performance of our algorithms within a robot simulation platform using up to 40 robots.

Categories and Subject Descriptors

I.2.9 [**Artificial Intelligence**]: Robotics

General Terms

Algorithms, Performance, Design, Experimentation

Keywords

Area Coverage, Weighted Voting Games, Winning Coalitions

1. 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[6]. However, it is challenging to design efficient techniques for coordinating multi-robots to perform area coverage. As Hazon *et al.* [8] 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, 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 theoretically prove the feasibility of this model, and give two algorithms to find veto players and the minimal winning coalitions. We have also experimentally verified the performance of our algorithm on a robot simulation platform.

2. PRELIMINARY DEFINITIONS

Coalitional Games A coalitional game (CG) with transferable utility is given by (N, v), where $N = \{1, 2, \ldots, n\}$ is a set of agents or players (the grand coalition); and $v : 2^N \mapsto \Re$ is a utility function that maps each group of agents $S \subseteq N$ to a real-valued payoff. Intuitively, $v(S)$ is the profit that the members of coalition S can attain by working together [11, 12].

A lower case coalitional game is *simple* if $v(S)$ can only take values 0 and 1, e.g., $v : 2^N \mapsto \{0, 1\}$. In such games, we call a coalition C ($C \subseteq N$) *a winning coalition* if $v(C) = 1$, and *a losing coalition* if $v(C) = 0$. An agent i is a *veto* player, if $i \in C$, $v(C) = 1$, $v(C\backslash\{i\}) = 0$. In other words, *veto* players are the players included in all winning coalitions. On the contrary, *dummy* players are those players which can be removed from a winning coalition without affecting the winning condition of the coalition, e.g., for any $C \subseteq N$, and $i \in N$, we have $v(C \cup \{i\}) = v(C) = 1$, i is called a *dummy* player.

Weighted Voting Games A weighted voting game G is a simple game that is described by a vector of players' *weights* $\mathbf{w} = (w_1, ..., w_n)$ and a *quota* $\mathbf{q}$ [1]. In such games, a coalition is winning if the sum of its total weights meets or exceeds the threshold quota $\mathbf{q}$. 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, $q = 6$, 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.

Imputation: Given a coalitional game $G = (N, v)$, the imputation set, denoted P, is defined as $\{x \in \rho | \forall i \in N, x_i \geq v(i)\}$ [12].

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. The core of a coalitional game contains imputations such that no other sub-coalitions can obtain a better imputation for themselves by defecting from the grand coalition. 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. For example, the voting system of the European Union is a combination of weighted voting games [2]. With exponentially large number of players, [7] shows that the computational complexity of general representation of coalitional game is infeasible. In other words, it is a NP-hard problem.

3. SYSTEM MODELING AND ALGORITHM DESIGN

3.1 Single Robot Coverage Capability

The goal of multi-robots area coverage is to search or explore a certain unknown environment efficiently, and reduce 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 y 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

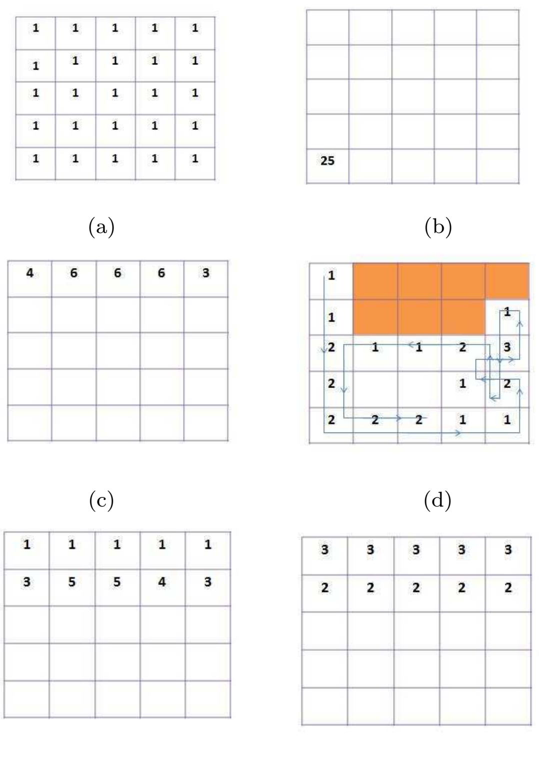

Figure 1: After 25 timesteps, the coverage map of a robot shows:(a) efficiently covering its entire coverage map , (b) stationary at the same place, (c)going back and forth in a row, (d) covering an area with an obstacle, (e) covering in the first row but going back and forth in the second one, (f) covering only inside two rows in a cycle.

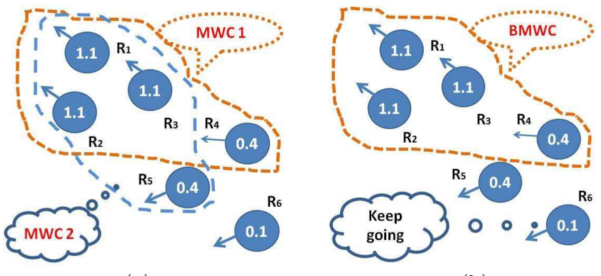

Figure 2: (a) 6 robots calculate two minimal winning coalitions in the communication range , (b) 4 robots form a best minimal winning coalition and head to a new team direction.

as the weights in a WVG.

We model the environment as a grid-based graph (Figure 1(a)). 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 [9], which represents the number of times the location has been visited by the robot. Figure 1(a)-(f)shows the coverage map of a robot for different scenarios after 25 timesteps. In Figure 1(a), the robot has visited each cell once, which corresponds to the most efficient coverage; In contrast, inefficient coverage is shown in Figure 1(b), which the robot stays in only one cell for 25 timesteps. Other cases with different degrees of coverage are shown in Figure 1(c)-(f).

A robot uses the coverage information from its coverage map to calculate its coverage capability. The coverage capability of robot i is denoted as $C_i = a \times \theta_i - b \times \eta_i + C_0$. θ_i is the coverage rate of robot i in recent time period T, and η_i is the redundancy rate. The coverage rate is $\theta_i = \frac{V_{cov}}{V_{map}}$, where V_{cov} is the area robot i covered in the last T timesteps, and V_{map} is the area of its whole coverage map. For example, if robot i can record a $100 \times 100 cm^2$ local area in its coverage map, and in T timesteps it has covered a $50 \times 50 cm^2$ area, we can calculate the value θ_i is 0.25. The redundancy rate is $\eta_i = \frac{V_{red}}{V_{map}}$, where V_{red} is the area of the revisited region within the coverage map, and V_{map} is the area of the whole coverage map. a and b are normalizing constants, and C_0 is the initial value of coverage ability. Thus, we can get C_i in the range $[0, 1.96]$, when $a = 2$, $b = 1$, $C_0 = -0.04$. In Figure 1(a) we get the value of the upper boundary of C_i as 1.96, where $\theta_i = 1$ and $\eta_i = 0$; In Figure 1(b) C_i has a lower bound of 0, with $\theta_i = 0.04$ and $\eta_i = 0.04$. The coverage capability C_i is equal to 0.24 and 0.96, respectively in Figure 1(c) and 1(d). In Figure 1(e) and Figure 1(f) $C_i^e = 0.64 > C_i^f = 0.44$, because in Figure 1(e) a robot has the same amount of coverage, but less redundant coverage than in Figure 1(f).

The coverage capability C_i of robot i is upper and lower bounded. θ_i is tight lower bounded to $\frac{1}{M}$, where M is the number of cells in a robot's local coverage map, because a robot will at least occupy one cell in its coverage map. θ_i is tight upper bounded to 1, if $\frac{T}{t_{step}} \geq M$, where T is the time interval to update the coverage map, and t_{step} is the time step a robot take to cover a single cell in its coverage map. η_i is lower bounded by 0, if no cells in the coverage map are visited. Also, η_i is upper bounded by $\frac{\frac{T}{t_{step}}}{2 \times M}$, with definitions of T, M and t_{step} remaining the same as before. Without loss of generality, we suppose $\frac{T}{t_{step}} = M$, C_i can then be upper bounded by $a + C_0$, which the robot has done less repeated coverage among the cells in its coverage map. C_i can be lower bounded by $\frac{a-b}{M} + C_0$, which means the robot is stuck in a single cell during time T. As the range of C_i can be bigger than 0 by setting the normalizing constants properly such as $a > b$, we can use C_i as the measurement of robot i's weight w_i.

3.2 Best Coalition Formation

Coalition structures are different combinations of teams which robots can form in a certain 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. For example, there are 4 that are within each other's communication range. Suppose robots can form $2^4 = 16$ possible coalitions. To get a winning coalition, we set robot's own weight $w_i = \alpha \times C_i + \beta$ as a linear function of the robot coverage capability, 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. Because the coverage capability is guaranteed to stay within an interval, the weight of the robot is also contained in an interval. For example, $W_i \in$ [0,1] with $\alpha = \frac{1}{2}$ and $\beta = 0.02$. The weight W_i associated with a robot gets updated with its coverage map and coverage capability after every T timesteps. We set the winning threshold of WVG to $q = c \times n$, where n is the number of robots and c is an adjustment constant. In each round of the voting game, the weight w_i of each robot and the quota q are both fixed values. For example, we consider a weighted voting game with n = 4 robots and individual robot weights at $w_1 = 1.3, w_2 = 1.1, w_3 = 1.2, w_4 = 0.2$, in a concise form $G = \{3 : 1.3, 1.1, 1.2, 0.2\}$. We can get the minimal winning coalition $\{r_1, r_2, r_3\}$, when we set $q = 3$, with $c = 0.75$. However, 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 2 (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 $\xi = argmin_{S \in I}(g \times \prod_{k \in S} x_k + \frac{\sum_{i,j \in S}(e \times d_{i,j} + f \times \varphi_{i,j})}{|S|})$. 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, which make $g \times \prod_{k \in S} x_k \ll \frac{\sum_{i,j \in S}(e \times d_{i,j} + f \times \varphi_{i,j})}{|S|}$. 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 2 (b). If there is no winning coalition, we just keep the status of each robot unchanged.

Theorem 1. *The core of a weighted voting game $G = (I, \{w_1, \cdots, w_n\}, T)$ is non-empty if and only if there is an agent i that is present in all winning coalitions. In other words, the core of WVG is non-empty, if and only if it has a veto player.*

PROOF. As weighted voting game is the subclass of simple game, we can derive this property straightforward from simple game as proved in [13].

□

Theorem 2. *The best minimal winning coalition is in the core.*

PROOF. For $T \supseteq N$, let MW_T denote the set of all best minimal winning coalitions $T \in N$. We establish $A \in T$.

```
function FindVetoPlayers returns Veto player set $I${
inputs: all player set $N$;
variables:
        boolean $flag$; temp set S;
        player $P_0$, $P_i$, $P_j$;

for each $P_i$ in $N$
        $S = N\backslash\{P_i\}$; $flag$ = FALSE;
        while flag == FALSE and $S \neq \emptyset$ do
              flag = TRUE; $P_0 = 0$;
              for each $P_j$ in $S$
                  if flag == TRUE and $V_j(S) == 0$ then
                  $P_0 = P_j$; flag = FALSE;
                  endif
              endfor
              if $P_0 \neq 0$ then S = $S\backslash\{P_0\}$ ;
        endwhile
        if S == $\emptyset$ then
           put $P_i$ in $I$;
        endif
endfor

return $I$;
}
```

Figure 3: The algorithm used to find veto players in weighted voting games.

Suppose B belongs to the core, and $B = \{j\} \bigcup A$, which means B is not in set T. x_j is the payoff of j. Based on the definition of the **core** in the former section , $\sum_{i\in N} x_i = v(N)$ and $\sum_{i\in B} x_i \geq v(B), \forall B \subseteq N$. $\sum_{i\in B} x_i = \sum_{i\in A} x_i + x_j \geq v(B) > v(A)$. However, A is in the winning coalition set T, so the payoff $v(A) \geq v(N) \geq v(B)$. So, there does not exist any B which belongs to the core, but does not belong to the minimal winning coalition set T. Thus if we can find a best minimal winning coalition A, it must be the core. □

Theorem 3. *The best minimal winning coalition is unique.*

PROOF. $d_{i,j}$ is less than the length L of the communication range and bigger than 0. $\varphi_{i,j}$ is in the interval $[0, 2\pi]$. $\frac{\sum_{i,j\in S}(e\times d_{i,j}+f\times\varphi_{i,j})}{|S|}$ is in the interval $(0, e \times L + 2\pi \times f]$. Suppose, two winning coalitions S and $S^{'}$ have the same value $\xi = \xi^{'}$, where exists one robot $n \in S^{'}$ and n is not contained by S. As both coalitions are minimal winning coalitions, they have the same number of elements $|S| = |S^{'}|$. From the definition of ξ, we have $\frac{\sum_{i,j\in S}(e\times d_{i,j}+f\times\varphi_{i,j})}{|S|}) = \frac{\sum_{i,j\in S'}(e\times d_{i,j}+f\times\varphi_{i,j})}{|S'|})$, which means the geometric shape of the two coalitions are same. In other words, they have the same perimeter and the sum of all angles. Thus, we know $\prod_{k\in S} x_k \neq \prod_{k\in S'} x_k$. However, as $|S| = |S^{'}|$, if we get $\xi = \xi^{'}$, x_k can not be prime numbers. It is contradictive for the define of the x_k. Therefore, S and $S^{'}$ are the same coalition, which infers that the best minimal winning coalition is unique .

□

3.3 Algorithms of BMWC

To find a best minimal winning coalition, we need to find whether there exist veto players or not based on Theorem 1.

```
function FindMinimalWinningCoalitions
returns Minimal winning coalition set $MWC${
inputs: veto player set $I$;
        all player set $N$;
variables:
        temp set S, M, C;
        player $P_i$;
C = I; $S = N\backslash I$;
sort $S$ in descendent order;
// The veto players can form the
// minimal winning coalition by themselves.
if $V(C) == 1$ then put $C$ in $MWC$;
else
        while $S \neq \emptyset$ do
            for each $P_i$ in $S$
                $C = C \cup \{P_i\}$;
                if $V(C) == 1$ then
                    put $C$ in $MWC$;
                    $S = S\backslash\{P_i\}$;
                    break;
                else
                    put $P_i$ in $M$;
                endif
            endfor
            // The remain member in S with the veto players
            // can not form a minimal winning coalition.
            if $M == S$ then $S = \emptyset$;
        end while
endif
return $MWC$;
}
```

Figure 4: The algorithm used to find minimal winning coalitions.

Figure 3 shows the algorithm to find veto players. The input is the set of all players in the range, and the output is the set of the veto players. The idea of the algorithm is to check the payoffs of all the coalitions without P_i. If there exist a coalition S, which does not include P_i, and has a payoff value $V(S)$ equal to 1, P_i is not a veto player; otherwise, P_i is a veto player. We put all the veto players in set I. Suppose the worse time to compute $V(S)$ is A. The worse computation time of the algorithm in Figure 3 is $O(An^3)$. We know veto players must be inside the minimal winning coalitions, but the minimal winning coalitions are not formed only by veto players. We can find the minimal coalitions by the algorithm shown in Figure 4. The input is the set of veto players I, and the output is the set of minimal winning coalitions. Based on the definition of the minimal winning coalition, we need to check out if the worth of the coalition $V(C)$ is equal to 0, and if the worth of the coalition $V(C \cup \{P_i\})$ is equal to 1. Thus the coalition $MWC = C \cup \{P_i\}$ is the minimal winning coalition. The computation complexity of the algorithm in Figure 4 is $O(An^2)$. However, the minimal winning coalitions are not unique. In the last step, we can use a sorting algorithm to get the value ξ, so that we can find the best minimal winning coalition in the set of minimal winning coalitions. The computational time to find value ξ is $O(n^2)$, as there could be n coalitions in the minimal winning coalition set, and n players in each minimal winning coalition.

4. ADAPTIVE FLOCKING TEAMS WITH WVGS

Table 1: Different parameter values used in the experimental settings for our simulations.

Symbol	Parameter	Value
$\mid N \mid$	Number of robots	$\{5, 10, 20, 40\}$
ξ_c	Communication range (m)	{0.7, 1.1, 1.5 }
	Percentage of environment occupied by obstacles	{0, 10, 20 }
q	Quota for WVGs	$\{0.3W_{max}, 0.5W_{max}, 0.7W_{max}\}$

4.1 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 [3, 4, 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.

4.2 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. In a certain time period, such as 25 time steps, the robot with highest weight of the team will calculate the BMWC. If a team member's coverage capability is low, it is the leader of the team or a follower of the team, it will not be in the new BMWC and leave the team. When two or more than two two teams meet in the communication range, based on recalculation of the BMWC, a team leader may become a team follower, and a team follower may become a team leader. 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 can obtain team behaviors.

5. EXPERIMENT AND EVALUATION

We have tested our algorithms on the Webots robot simulation platform. Webots allows realistic modeling of robots and environments including the parameters of different sensors on robots and the physics of the environment. Each robot in our simulated system is modeled as a mobile robot with a GPS and an onboard compass. The speed of each wheel was set to 2.8 cm/sec, and a robot can cover $0.07 \times 0.07 meter^2$ area during each time step. Unless otherwise stated, the experiments are repeated over 10 times with simulated physical robots for 2 hours, and we collected the average results. We used internal and external parameters to evaluate the effect of the coalition formation, and the system performance as well. The parameters included in each experiment set are presented in Table 1. W_{max} is the maximum weight of robots in the communication range. As the weight of each robot is upper bounded, W_{max} is a constant in each round of a weighted voting game.

Table 2: Percentage of the environment that is covered by 5, 10, 20, or 40 robot with different percentages of the environment occupied by obstacles.

		Number of robots			
		5	10	20	40
Obstacles 0%	Max.	88.29	97.99	99.39	100
	Avg.	85.78	97.17	99.27	99.79
	Min.	82.38	96.21	99.20	99.63
Obstacles 10%	Max.	88.61	97.75	99.90	100
	Avg.	86.42	97.13	99.81	99.90
	Min.	84.88	96.19	99.20	99.59
Obstacles 20%	Max.	90.17	97.95	100	100
	Avg.	86.76	97.18	99.81	99.94
	Min.	81.9	96.75	99.64	99.80

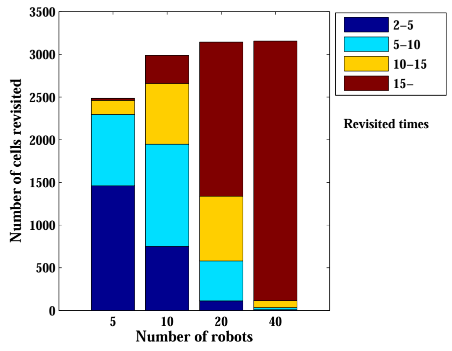

Figure 5: Redundancy in coverage with different number of robots with $q = 0.7W_{max}$ and 10% percentage of the environment occupied by obstacles.

In our first set of experiments, we verify the percentage of environment covered by different numbers of robots in the environment. The environment is occupied by 0%, 10%, or 20% obstacles respectively. By adding more obstacles in the environment, the percentage of the coverage increases by 1-2 % as shown in Table 2. Although with more obstacles there is less room to be covered, the robots are hard to form a team, or frequently rebuild a new team. Thus, in the environment occupied by obstacles 20% the amount of coverage measured as a percentage does not improve strikingly. Quite expectedly, as the number of robots increases the amount of coverage of the environment measured as a percentage improves.

In our second set of experiments shown in Figure 5, we observe the redundancy in coverage by different number of robots. We categorizing the redundancy into four different intervals. The cells that are more than 15 revisited times increase sharply with 40 robots, because the environment

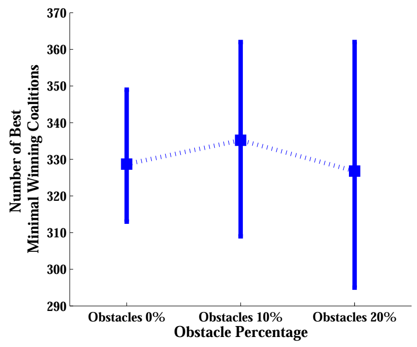

Figure 6: Number of best minimal winning coalitions for coverage with 20 robots with different percentages of the environment occupied by obstacles. Quota q is set to $0.7W_{max}$.

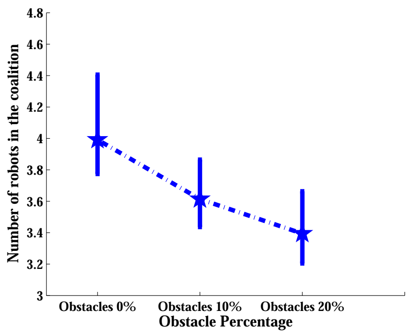

Figure 7: Number of robots in the best minimal winning coalition for coverage with 20 robots with different percentages of the environment occupied by obstacles. Quota q is set to $0.7W_{max}$.

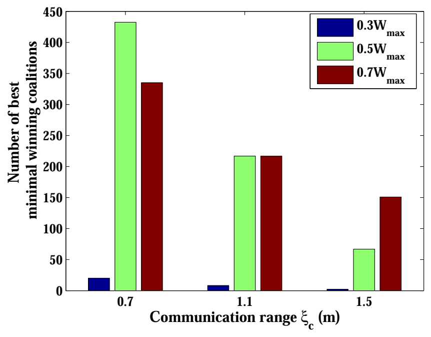

Figure 8: Number of best minimal winning coalitions for coverage with 20 robots with 10% percentage of the environment occupied by obstacles, for different communication range value ξ_c with $q = 0.3W_{max}$, $q = 0.5W_{max}$, and $q = 0.7W_{max}$.

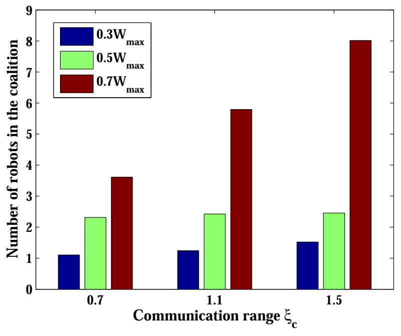

Figure 9: Number of robots in the best minimal winning coalition for coverage with 20 robots with 10% percentage of the environment occupied by obstacles in different communication range value ξ_c with $q = 0.3W_{max}$, $q = 0.5W_{max}$, and $q = 0.7W_{max}$.

is crowded, and robots or robot teams find it difficult to disperse away from each other.

To analyze the effect of changing the environment occupied different percentage of obstacles while keeping the number of robot and communication range of the robots fixed at $n = 20$ and $\xi_c = 0.7$m respectively. Not surprisingly, as the percentage of obstacles increases in the environment the number of robots in the coalitions decrease as shown in Figure 7. In other words, the size of team become smaller in the environment full of obstacles.In the environment occupied by 10% obstacles there are around 10 more best minimal winning coalitions than the other two as shown in Figure 6. The number of best minimal winning coalitions are nonlinear with increasing the percentage of obstacles. We also observed that more robots have high weights to join the team in open space than the one full of thorns.

In the last set of experiments, we observe the effect of changing the communication range ξ_c of each robot and the quote q of WVGs with fixed number of robots and percentages of the environment occupied by obstacles. By increasing the communication range, the number of minimal winning coalitions decreases as shown in Figure 8, and the number of robots in the coalitions increases as shown in Figure 9. Obviously, more robots can communicate to each other and form the bigger team with more length of the communication range. However, increasing the value of the quota does not mean more chances to build the team. In our experiment with $0.9W_{max}$, there is even no BMWCs no matter how long the communication range is. Because it is hard for each robot to get the maximum weight. Meanwhile, the value of quota is too large for robots in the communication range to achieve the winning coalition.

6. 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 the feasibility of our model, and give theoretical proof and solutions to find the BMWC. The result of our experiments show that this model can be applied for multi-robot coverage. 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.

7. REFERENCES

[1] Y. Bachrach, and E. Elkind. "Divide and Conquer: False-Name Manipulations in Weighted Voting Games", Autonomous Agents and Multiagent Systems (AAMAS), pp. 975-982 , 2008.

[2] J. M. Bilbao, J. R. Fernandez, N. Jiminez, and J. J. Lopez. "Voting power in the European Union enlargement", European Journal of Operational Research, vol. 143, pp. 181-196, 2002.

[3] K. Cheng and P. Dasgupta, "Coalition game based distributed coverage of unknown environments using robot swarms," International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS), pp. 1191-1194 2008.

[4] 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.

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

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[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.