Robust Multi-robot Team Formations using Weighted Voting Games

Prithviraj Dasgupta and Ke Cheng

Abstract We consider the problem of distributed multi-robot team formation including the dynamic reconfiguration of robot teams after encountering obstacles. We describe a distributed robot team reconfiguration algorithm, DYN-REFORM, that uses a game-theoretic technique of team formation called weighted voting games(WVGs) along with a flocking-based formation control mechanism. DYN-REFORM works without explicit knowledge of global features such as the presence of obstacles in the environment or the number and location of all other robots in the system. It uses the locally computed metrics of each robot in the team to determine whether a team needs to split or two teams need to merge during reconfiguration. We have tested team reconfiguration using the DYN-REFORM algorithm experimentally within the Webots simulator using teams of e-puck robots of different sizes and with different obstacle geometries. We have also shown that using robots coordinated with the DYN-REFORM algorithm for a distributed area-coverage application improves the coverage performance.[1]

1 Introduction

Distributed formation control of multi-robot teams is an important research direction in robotics that is used in various applications such as convoying or escorting robots or humans for security-related applications, area coverage or clearance for demining, agricultural and other applications, cordoning off regions in hazard control situations, etc. Recently, the problem of robust robotic team formation has gained considerable research interest [6, 9, 10, 13]. Frequent occlusions during the motion of a robot team cause it to repeatedly reconfigure by changing its direction of movement and getting all the team members into a new pose so that it can con-

Prithviraj Dasgupta, Ke Cheng
Computer Science Department, University of Nebraska, Omaha, NE 68182, USA e-mail: pdasgupta, kcheng@mail.unomaha.edu

[1] This research has been sponsored by the U.S. Dept. of Defense, Office of Naval Research as part of the COMRADES project, grant no. N000140911174.

tinue its motion. Reconfiguring the robot team is a costly operation that expends time, consumes robots' energy to communicate with each other and culminates in reduced efficiency of the operation the robots are performing as a team. Therefore, it makes sense to investigate techniques that would reduce this overhead by avoiding inefficient team reconfigurations or by postponing reconfigurations when possible.

In this paper, we posit that robust formation maintenance can be achieved on robot teams if their configuration is dynamically adapted to combine into larger teams or split into smaller teams depending on different environmental and operational conditions. To achieve this, we use a game theoretic technique called a weighted voting game(WVG). We integrate the game-theoretic decision making with a flocking-based formation control technique for the robot teams using a layered approach in an algorithm called DYN-REFORM. We have tested the operation of the DYN-REFORM algorithm experimentally on e-puck robots within the Webots simulator for dynamically reconfiguring teams at obstacles with different geometries. We have also measured an improvement of $5-10\%$ in a distributed area coverage application, obtained by using DYN-REFORM as compared to an algorithm where robot teams do not reconfigure dynamically.

Related Work. Much of the research on formation control with multi-robot teams [1, 8, 10] has been based on Reynolds' model for the mobility of flocks[11]. In [2], the authors describe three reactive behavior-based strategies for robot teams to move in formation, viz., unit center-referenced, neighbor-referenced, or leader-referenced. In contrast to these approaches, Fredslund and Mataric[7] describe techniques for robot team formation without using global knowledge such as robot locations, or the positions/headings of other robots, while using little communication between robots. Complementary to these approaches [6, 13] have used a combination of graph theory and control theory-based techniques to effect multi-robot formations. Our previous work on multi-robot formation [3] uses techniques where a team simply reverses its direction to avoid obstacles or other teams, without dynamically reforming, merging with other teams or splitting into smaller teams, while in [4], we have described preliminary results for team splitting only, using weighted voting games.

2 Dynamic Team Reconfiguration and Weighted Voting Games

We have used a flocking-based, leader-referenced formation control algorithm[11], to maintain a specific formation among the robots in a team while in motion, as shown in Figure 1(a) and described in [3]. Each robot in a team is given a local identifier with '0' as the leader robot's identifier and odd and even numbered identifiers for the follower robots on either side of the leader. The shape of the team can be controlled by varying the angle u to transform it, for example, from a wedge shape to a line shape. Each robot has a specified separation d_{sep} that it must maintain from its neighbors. The leader robot has a predetermined direction α that it wants to move the team in. At the end of each time step, the leader robot determines the position that it should reach at the end of the next time step so that it can continue its desired motion. The leader robot also calculates the positions for each of the follower

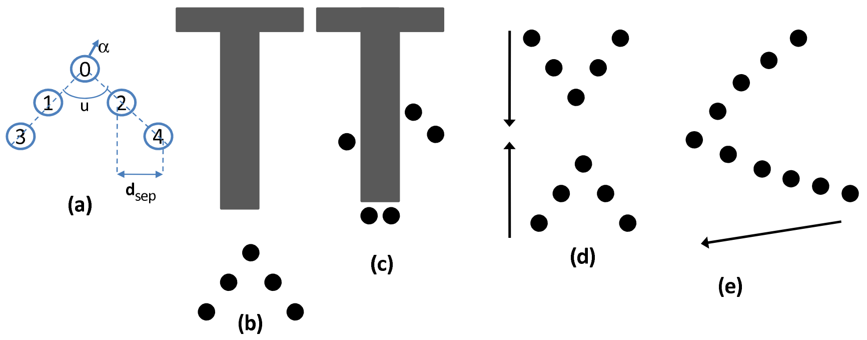


Fig. 1 (a) A robot team showing the position identifiers of each robot. The angular separation in the team is u, the separation between adjacent robots is d_{sep} and α is the heading of the team. (b)-(c) A scenario where a single team in formation encounters a T-shaped obstacle and needs to split. (d)-(e) A scenario where two teams in close proximity of each other encounter each other and need to merge.

robots for the next time step relative to their current positions, so that the formation of the team is maintained. The leader robot then communicates the desired position information to each of the follower robots. Finally, the leader and follower robots start moving towards their respective designated positions.

A principal problem with flocking-based formation control is that it can fail if the leader or a follower robot encounters an obstacle that impedes its motion to the desired position for the next time step. The problem of obstacle avoidance while moving in formation is accentuated if robots are not be able to perceive the obstacle boundary before planning their motion for the next time step. For example, in the scenario illustrated in Figure 1(b), it might be inefficient or impossible to continue moving a robot team in formation when the motion of all or some of the team members gets impeded. In such inefficient scenarios, it would be beneficial to reform the members of a team into new teams by splitting the original team (Figure 1(c)). Complimentary to the team splitting scenario, there can be scenarios when two robots teams in close proximity of each other (e.g., Figure 1(d)) could improve some application-specific performance metric such as the sensor diversity, computational capability, or amount of area coverage achieved by each team, if they combined into a single team. In such scenarios, it would be beneficial to merge some or all of the members of two robot teams into a single team, as shown in Figure 1(e). We approach this problem of merging and splitting robot teams as the robot team reconfiguration problem. Reconfiguring a robot team can be viewed as a problem of finding the partition that is the best for all the participating robots. The scenarios shown in Figures 1 (b)-(e) illustrate that a single, hard-coded splitting or merging rule cannot be guaranteed to be the best partition in all scenarios, and, therefore, the partitioning of the robot teams has to be done dynamically. The branch of microeconomics dealing with cooperative or coalitional games uses concepts from utility theory and rational behavior of humans to determine rules for solving the partition problem. Coalition games are particularly attractive for our problem for two reasons. First, they can ensure that the solution is stable, or in other words, it is acceptable to all the participants. Secondly, the players, or robots in our case, do not have to be explicitly informed if a split or a merge is the best thing to do. The rules of the coalition game calculate the set of robots that are incentivized to remain together, based on the performance of each robot in the recent past.

The most important factor in a coalition game is determining the performance that a robot in a team has had because this performance determines which robots will remain together by forming a coalition. The robots participating in a coalition game share their individual performance values with each other before the game. The performance value reported by a robot should reflect its individual efficiency in performing its operation or role while participating in the team. For example, in a team formation setting, it could reflect the distance and time for which the robot has not deviated from its designated position in the formation over some finite time window in the recent past. Metrics from the application domains for the robots could also be incorporated into a robot's performance value. For example, if the robots are performing distributed area coverage, then the performance value could include the area of previously uncovered region that a robot has covered in the recent past. The performance value of a robot represents its 'power' in a coalition and is called the robot's **weight** in the game. For our problem, we have used a suitable and succinct representation of a coalition game called a weighted voting game(WVG). The main parameters of a WVG are the following:

R	Set of players or robots interested in forming a coalition
$\mathscr{R}$	Set of possible partitions among the members of the set R, $\mathscr{R} = 2^R$. Each member of the set $\mathscr{R}$ is called a coalition of robots and denoted by $C_j : j = 1..\mid \mathscr{R} \mid$
w_i	weight of robot $r \in R$
Q	quota or threshold of the WVG. A coalition C_j of robots becomes a winning coalition if the sum of the weights of the players exceeds the quota,
v	value function that denotes whether a a coalition $C \in \mathscr{R}$ is a winning coalition or not, i.e., $v(C) = 1$ if $\sum_{i \in C} w_i >= Q$, and $v(C) = 0$, otherwise.

A few additional concepts in WVGs are useful for the formulation of our problem. A **veto player** is a player such that if the player is excluded from a coalition, that coalition cannot be a winning coalition anymore. As an example. consider a WVG with 4 players A, B, C and D with weights $4, 2, 1,$ and 1 respectively. For this WVG, let quota $Q = 5$, that is any coalition must have a combined weight of at least 5 to be a winning coalition. The set of winning coalitions for this WVG are $\{A,B\}, \{A,C\}, \{A,D\}, \{A,B,C\}, \{A,B,D\}, \{A,C,D\})$ and $\{A,B,C,D\}$. This makes A a veto player because it is present in all the winning coalitions. WVGs can have more than one veto player or no veto players. For example, if we change Q from 5 to 7 both A and B become veto players, while if we change Q from 5 to 2, none of the players is a veto player anymore. We use the notation V to denote the set of veto players. The minimum set of players that can get enough combined weight among themselves to get to the quota is called the **minimum winning coalition(MWC)**.

```
function FindVetoPlayers returns set V
inputs: set R, double Q, $w_i$ ;
variables: double $W$, $W_{-i}$;

$V = \{\emptyset\}$;
$W = \sum_{i \in R} w_i$;
for each $i \in R$
    $W_{-i} = \sum_{j \in R \setminus \{i\}} w_j$;
    if ($W_{-i} < Q$)
        $V = V \cup \{i\}$;
return $V$;
```

(a)

```
function BMWCHeuristics returns Set BMWC
inputs: Set V; ArraySet MWC;
variables:Set S; double[] $\xi$;
$centroid \leftarrow \frac{\sum_{v \in V}(Location(P_v))}{|V|}$;
$avgBearing \leftarrow \frac{\sum_{v \in V}(Angle(P_v))}{|V|}$;
for $i = 1$ to $| MWC |$
    $S = MWC[i] \setminus V$;
    for each robot $r \in MWC[i]$
        $\xi[i] = Distance(r, centroid) +$
            $Angle(r, avgBearing) + ID(r)$;
$j = \arg\min_i \xi[1..| MWC |]$;
return $MWC_j$;
```

(b)

Fig. 2 (a) Algorithm to find veto players in a WVG. (b) Algorithm to find the best minimum winning coalition (BMWC) from a set of MWCs.

In the example above with $Q = 5$, there are three MWCs - $\{A,B\},\{A,C\},\{A,D\}$. MWCs are important because they imply that players in a MWC will not deviate from the coalition they are in because they cannot improve the benefit that they receive by forming a different coalition or a sub-coalition. This makes MWCs stable coalitions which are guaranteed not to break off after the coalition is formed.

The problem solved in a WVG is to identify a set of players that form a minimum winning coalition. This can be achieved in three steps as given below:

1. Identify all the veto players in R, because the veto players, if any, must be there in every winning coalition. The algorithm for identifying the veto players is based on the definition of veto players as players whose exclusion causes any coalition of the remaining players to lose. In other words, the combined weight of players excluding the veto player would fall below the quota. Our *FindVetoPlayers* algorithm shown in Figure 2(a), uses this concept to calculate set of veto players V. It has linear time complexity as it has to inspect each player from the set of players R for checking if it is a veto player or not.

2. Identify all the MWCs, by adding the minimum number of non-veto players to each set of veto players identified in step 1 above. Let w_v denote the combined weight of the veto players found in step 1. Then, $Q' = Q - w_v$, denotes the deficit in combined weight that should come from the non-veto players to reach the quota and form an MWC. Our objective then becomes to determine the set of players from the set $R \setminus V$ that can together reach a combined weight of Q'. This problem is a simplified version of the subset sum problem [5], with the relaxation that we need to find the smallest subset of players that is able to reach a combined weight of at least Q', (instead of exactly Q' of the subset sum problem). We have used a greedy method to solve this problem that has a quadratic time complexity in the worst case. The

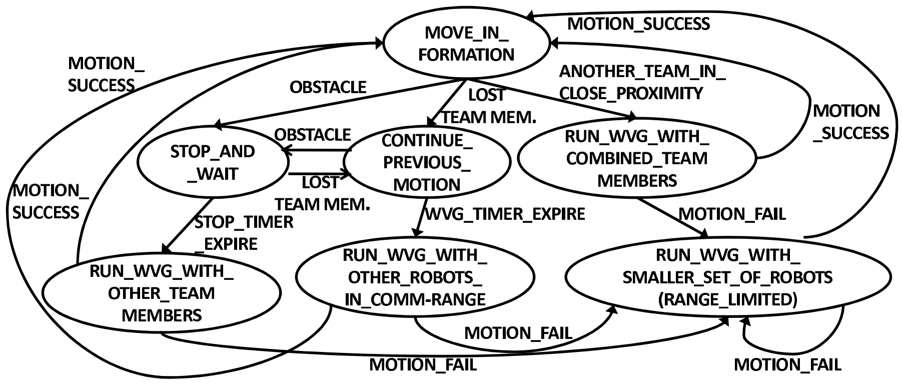

MOVE_IN_ FORMATION
MOTION_SUCCESS
MOTION_ SUCCESS
OBSTACLE
LOST TEAM MEM.
ANOTHER_TEAM_IN_ CLOSE_PROXIMITY
MOTION _SUCCESS
STOP_AND _WAIT
OBSTACLE
CONTINUE_ PREVIOUS_ MOTION
RUN_WVG_WITH_ COMBINED_TEAM MEMBERS
LOST TEAM MEM.
MOTION_ SUCCESS
STOP_TIMER _EXPIRE
WVG_TIMER_EXPIRE
MOTION_FAIL
RUN_WVG_WITH_ OTHER_TEAM MEMBERS
RUN_WVG_WITH_ OTHER_ROBOTS_ IN_COMM-RANGE
RUN_WVG_WITH_ SMALLER_SET_OF_ROBOTS (RANGE_LIMITED)
MOTION_FAIL
MOTION_FAIL
MOTION_FAIL

Fig. 3 State transition diagram for a robot participating in the dynamic reconfiguration algorithm.

output of this algorithm is the set of MWCs.

3. Select one MWC called the best minimum winning coalition (BMWC) from the set of MWCs identified in step 2 above that appears to be most amenable to robot team formation. We measure the eligibility of an MWC towards forming a robot team using a heuristic-based fitness function ξ. For designing this heuristic function, we first consider the pose of the veto players because the veto players must be included in the final winning coalition. We calculate the centroid of the locations of the veto players and the average of the bearing between them. Then, for each of the non-veto players in each *MWC*, we calculate the distance and relative bearing with the centroid and average bearing of the veto players. If there are still any ties remaining, we use a prime number calculated from the robot id to make the value of ξ unique. The minimum of the ξ values for each *MWC* gives the best *MWC*. This algorithm is shown in Figure 2(b) and it has polynomial running time because it takes $O(\mid R\mid^2)$ steps in the worst case to calculate the value of ξ for each non-veto robot in each *MWC*. Integrating all the three steps, the worst case time complexity of running a WVG among R robots is $O(\mid R\mid) + O(\mid R\mid^2) + O(\mid R\mid^2) = O(\mid R\mid^2)$.

DYN-REFORM Algorithm. The DYN-REFORM algorithm realizes dynamic team reconfiguration by integrating the WVG algorithm and the flocking-based formation control algorithm. This integration is important and challenging because the WVG works only with a performance value or weight for each robot while the formation control algorithm relies on operational conditions such as presence of obstacles, proximity of robots, etc. Before running the WVG, the DYN-REFORM algorithm provides methods to determine the set or subset of robots from a team that will participate in the WVG. After a coalition has been computed by the WVG, the DYN-REFORM algorithm provides mechanisms to handle situations where the computed coalition might not be realizable by the formation control algorithm (e.g., because of occlusions that prevent robots in a coalition from reaching their designated positions in the new team). It also specifies the operation of the robots that are excluded from the winning coalition after running the WVG.

The operation of the DYN-REFORM algorithm is summarized by the state transition diagram of a robot shown in Figure 3. First, we consider the case when a team moving in formation encounters an obstacle and could possibly have to split. When any member of the team encounters an obstacle, it enters into a STOP_AND_WAIT state for a certain time period given by the value of a timer called STOP-TIMER. When this timer expires, the robot runs a WVG. Other team members that encountered an obstacle and stopped before the STOP-TIMER expires, possibly in the vicinity of the robot that is running in the WVG, are included as participants in the WVG. In certain scenarios, only some members of a robot team might encounter an obstacle while other robots in the same team do not. The team members that do not encounter the obstacle continue their previous motion (CONTINUE_PREVIOUS_MOTION state) by moving in a straight line. Because the original team has lost some of its members due to an obstacle, it would make sense for the robots continuing their motion to try and reconfigure with other robots in the system. To achieve this, these robots schedule to run a WVG at some time in the future by starting a timer called the WVG-TIMER. When the WVG-TIMER expires on a robot, it runs a WVG including the robots that are within its communication range.

After the WVG has determined the robots comprising the best minimum winning coalition (BMWC), the robot that has the highest weight in the BMWC is selected as their leader robot. The leader robot then selects a new position and heading, usually in the direction opposite to which the obstacle that caused the reconfiguration was sensed. It then starts running the flocking-based formation algorithm to get the follower robots in their desired positions and start moving together as a team in formation. In certain cases, for example, when the vicinity of the robots forming a coalition is occupied by obstacles, the coalition of robots calculated by a WVG might not be amenable to get into formation and move together as a team. When this happens the robots that are not able to get into the desired position reattempt to get into their desired positions for NUM-FORMATION-REATTEMPT iterations. At the end of the reattempts, the robots that managed to get into their desired position for the new team, exclude the unsuccessful robots from the team and continue their motion. The unsuccessful robots attempt another WVG among themselves. If these robots are unsuccessful to form a team after NUM-WVG-REATTEMPT successive WVGs (and included formation reattempts), they continue to move individually using Braitenberg motion. Also, after running a WVG, if there are some robots that are not included as part of the best minimum winning coalition, they continue to move individually using Braitenberg motion until they encounter another team and possibly get assimilated with that team after running a WVG.

When a robot moving individually or a robot team gets within close proximity of another robot team, they run a WVG with the combined team members as the participating robots. Finally, to avoid identical yet repetitive calculation of the BMWC in a WVG by all the participating robots, we have selected the robot with the lowest local identifier to run the WVG. This robot receives the weights from all the participating robots and reports the outcome of the WVG to all the participants.

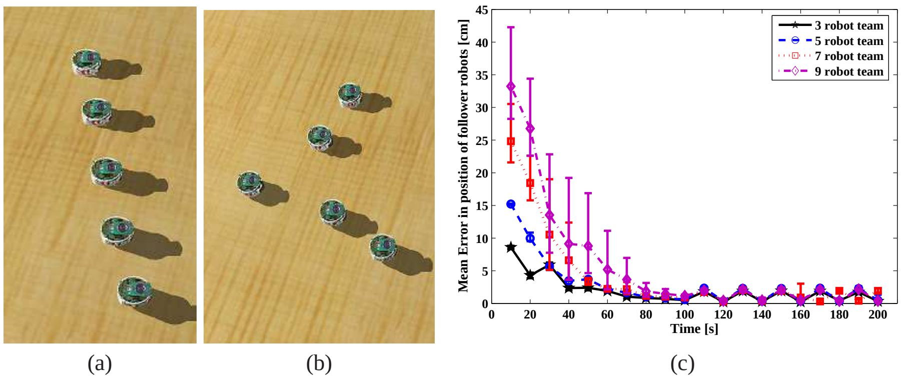


Fig. 4 (a) Initial positions of a set of 5 robots before starting the formation control algorithm. (b) Wedge-shaped formation achieved by 5 robots after starting the formation control algorithm. (c) Average error in the position of follower robots for different team sizes. Each team starts out of formation and after getting into a wedge-shaped formation moves over a distance of 3 m at a speed of 2.8 cm/sec with no obstacles in the path of the moving team.

3 Experimental Results

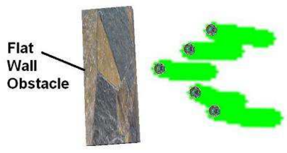
We have evaluated our multi-robot team reformation using e-puck robots on the Webots simulator. The e-puck robot has a diameter of 7 cm. Each wheel is 4.1 cm in diameter and is capable of a maximum speed of about 12 cm/sec. We have used the following sensors that are available on the e-puck robot: (1) Eight infra-red distance distance sensors to detect of obstacles in a range of 7 cm, and, (2) Bluetooth capability for wireless communication. Each e-puck robot is also provided with a local positioning system using a GPS and a compass node within Webots to enable it to determine its position in the environment within a 2-D coordinate system. For all team formations in our simulations, the robots forming the team had to get into a wedge-shaped formation with an angular separation $u = 60$ degrees. The inter-robot separation between a pair of follower robots in a team is set to 15 cm. For multi-team merging scenarios, the distance between two team leaders to run the WVG-based team merging algorithm is set to 70 cm. For our experiments, one time step is defined as the time required by a robot to cover an area equal to its own footprint. For a robot speed of 2.8 cm/sec, the value of the time step is calculated as 2.5 sec. The performance function used to compute the weight value for each robot for participating in the WVG uses the mean error in the robot's desired position and the area of previously uncovered region covered by the robot, over the last 25 time steps. Based on this function, for the DYN-REFORM algorithm, the duration of the WVG-TIMER and STOP-TIMER is set to 25 time steps and 15 time steps respectively. The number of reattempts by a newly formed team to get into formation after running a WVG (NUM-FORMATION-REATTEMPT) was set to 5. To prevent the formation of excessively large teams that have a high communication and computation overhead in the DYN-REFORM algorithm, we took two steps. First, we set the quota for a WVG at $0.9\times$ (sum of the weights of participating robots). This guarantees that robots with very poor performance, and, consequently low weights get excluded from the new team after reconfiguration. Secondly, we limited the maximum team size in our experiments to 7 robots. If the winning coalition calculated by the WVG has more than 7 robots, the excess robots that have the lowest weights in the coalition are removed from the team and move individually without forming a team, until they merge with another team at a later stage. All results were averaged over 10 simulation runs.

3.1 Single Team Moving in Formation Experiments

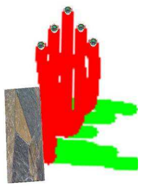
In our first set of experiments, we quantified the efficacy of forming a single team using the leader-referenced formation control technique described in Section 2. We considered different sets of robots that were initially out of formation (arranged in a straight line), but within communication range of each other, as shown in Figure 4(a). These robots use the flocking-based formation control algorithm to get into a wedge-shaped formation (Figure 4(b)) and move in formation within an obstacle free environment over a distance of 3 m. Because of sensor-related positioning errors, the leader robot treats a follower robot that is within a radius of ± 5 cm a from its exact desired position as 'in formation'. In this experiment, our objective is to measure the deviation in the team as a whole from a formation. To measure this, we calculated the mean error in the positions of the follower robots at intervals of 10 sec. over a period of 200 seconds. At $t = 0$, a leader robot was selected for each set of robots and the leader robot started running the flocking-based formation maintenance algorithm. The results of our experiments for team sizes of $3, 5, 7$ and 9 robots are shown in Figure 4(a). Initially, as the robots are not in formation, larger teams need longer times to get into formation. However, after the robots get into formation they are able to successfully use the flocking-based algorithm to remain in formation as shown by the low value of the mean error in the position of the follower robots after 100 sec.

3.2 Team Reformation Experiments

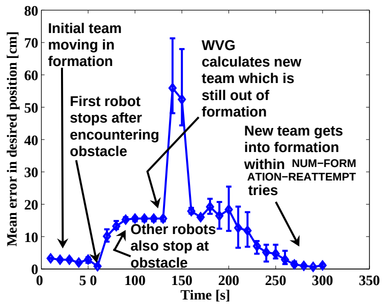
For our next set of experiments, we verified the performance of the DYN-REFORM algorithm. We considered three types of obstacles - a flat wall obstacle, a non-uniform wall obstacle and a perpendicular, narrow wall obstacle, as shown in Figures 5(a), 6(a) and 7(a). For this set of experiments, we have traced the trail of the robots within Webots to show their motion before, during and after reconfiguration. For the flat wall obstacle, the leader robot encounters the wall first, enters into the STOP_AND_WAIT state and starts the STOP-TIMER, according to the DYN-REFORM algorithm. The follower robots successively encounter the wall and also enter the STOP_AND_WAIT state and start their individual STOP-TIMERs. The leader robot's STOP-TIMER expires first and it runs the WVG including the other robots that are stopped in its vicinity as the WVG's participants. As an example of the weight and quota values used in this WVG, one of the reported runs for this experiment had weights of the five robots as $0.72, 1.0, 0.96, 1.16$ and 1.24 respectively and quota $Q = 0.9 \times \sum_i w_i = 4.57$. The leader robot of the team had the worst performance recently because it encountered the obstacle first and stopped, giving



(a)



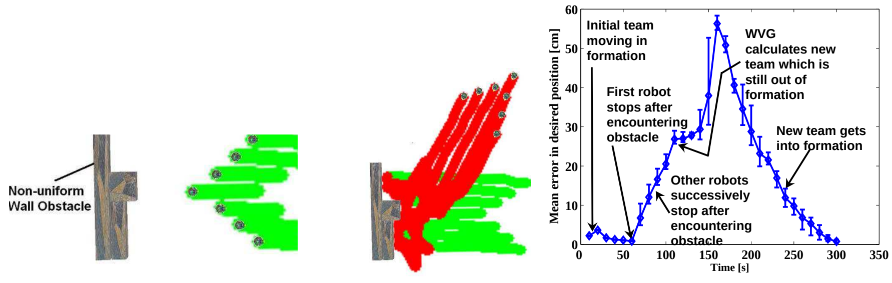
(b)



(c)

Fig. 5 (a) Initial configuration of a team of 5 robots moving in a wedge-shaped formation. (b) Reformed team moving in new direction after encountering a flat wall obstacle. (c) Mean error in the desired position of the robot team during the reconfiguration.

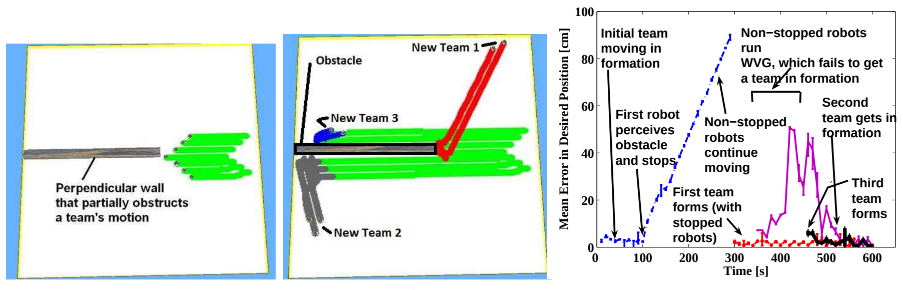
it a weight 0.72 in the WVG. The fringe robots that stopped last have the highest weights of 1.16 and 1.24 respectively. The BMWC contains all the five robots in this scenario. which means that all the robots should stay together in the new team. The new leader robot then selects a position in a direction opposite to the direction in which it encountered the wall, a pose or heading for the new team, and communicates the desired position of the follower robots to get in formation in the new team. After the formation succeeds, the new team starts moving as shown in Figure 5(b). Figure 5(c) shows the mean error in the position of the robots during the entire reconfiguration process. Initially, between $0-50$ seconds, the team is moving in formation before encountering the obstacle, and this is shown by a low mean error of about 3 cm in the position of the robots. When the robots successively start encountering the obstacle, between $50-100$ seconds, the error between their actual positions and their desired positions to remain in formation increases. This happens because when the robots successively stop at the obstacle they form straight line along the boundary of the wall, while their current formation, in which they had been before encountering the obstacle, requires them to form a wedge shape. While the WVG runs, the robots are stopped and their mean error in position remains unchanged, as seen between $100-125$ seconds. After determining the best minimum winning coalition, the robots get a new formation and the mean error in the position of the robots decreases back to the low value of around $3-4$ cm over $125-250$ seconds. We notice that the mean error suddenly spikes to about 60 cm around 150 seconds when the new team calculated by the WVG attempts reformation. This happened because all the robots stopped at the flat wall forming a horizontal line and they are in close proximity of each other. The robots themselves occlude each others paths when they try to get into a new formation. However, within 20 seconds, which was within NUM-FORMATION-REATTEMPT$=5$ iterations, the formation control algorithm is able to resolve this problem and the robots are able to regain formation.



(a) (b) (c)

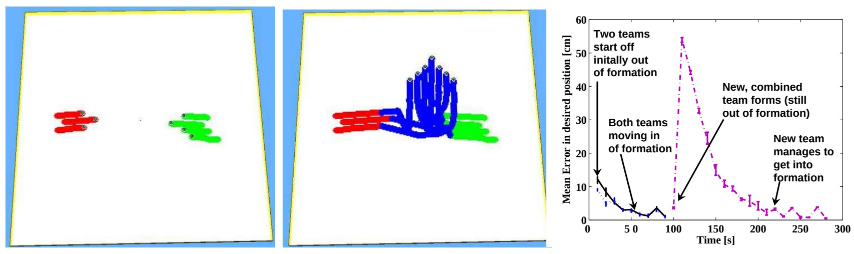
Fig. 6 (a) Initial configuration of a team of 7 robots moving in a wedge-shaped formation. (b) Reformed team moving in new direction after encountering a non-uniform wall obstacle. (c) Mean error in the desired position of a robot team during the reconfiguration.

The scenario with robots encountering the non-uniform wall obstacle, shown in Figures 6(a) and (b), is very similar to the flat wall case. The main difference is that because of the non-uniform surface of the wall, the robots along the fringes of the former team persist longer in the CONTINUE_PREVIOUS_MOTION state than in the flat wall case. This behavior is also seen in the mean error of the robots during reconfiguration, shown in Figure 6(c). The mean error in the position of the robots w.r.t. their positions in the previous formation becomes larger than the flat wall case because the robots move farther from their erstwhile desired positions in formation into the clefts of the wall. However, in this case, we do not see any significant spikes during after the WVG when the new team is getting into its formation because the robots have dispersed further from each other because of the non-uniform surface of the wall. Therefore, they do not occlude each other's path while getting into formation.



(a) (b) (c)

Fig. 7 (a) Initial configuration of a team of 7 robots moving in wedge-shaped formation. (b) Reformed team moving in new direction after encountering a narrow obstacle that causes the team to split. (c) Mean error in the desired position of a robot team during the reconfiguration.

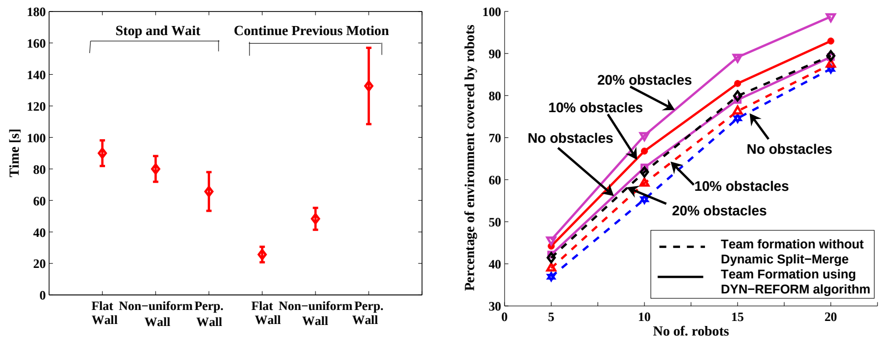


(a) (b) (c)

Fig. 8 (a) Initial configuration of two teams of 3 and 4 robots moving in wedge-shaped formations. (b) Reformed team moving in new direction after encountering each other and merging into a new team. (c) Mean error in the desired position of a robot team during the reconfiguration.

Figures 7(a) and (b) show a scenario where a robot-team encounters a wall perpendicular to its heading that obstructs the team partially. In this scenario, only two robots at the center of the 7 robot team encounter the obstacle, enter into the STOP_AND_WAIT state and start their STOP-TIMERs. The rest of the team members do not encounter the obstacle and enter into the CONTINUE_PREVIOUS_MOTION state while starting the WVG-TIMER. We notice that the obstacle forces two sets of team members to continue their motion along the two sides - above and below the obstacle. When the STOP-TIMER expires for the two stopped robots, they run a WVG. Since there are no other stopped robots in their vicinity, the stopped robots form a new team among themselves and continue moving in a new direction. When the WVG-TIMER expires for the robots that continued their motion, they run a WVG. These robots are in the vicinity of each other and the BMWC contains all these robots. However, the BMWC contains two sets of robots that are located on opposite sides of the obstacle and can never get into a single formation. Therefore, although the WVG succeeds, the new team fails to form within NUM-FORMATION-REATTEMPT($=5$) reattempts. The DYN-REFORM algorithm then causes the robots that were unable to get in formation to to run another WVG and form a new team. Each team gets into its desired formation and continues its movement. The graphs in Figure 7(c) show that the mean error in the desired position of the robots from their erstwhile team keeps increasing because some of the robots do not encounter the obstacle and therefore, do not stop. When the WVGs run, three teams are formed at different times based on the STOP-TIMER expiry (first team) and WVG-TIMER expiry followed by WVG reattempt (second and third team), as shown in Figure 7(c).

Figure 8(a)-(c) show the scenario of two robot teams moving towards each other and merging using a WVG, and the mean error in robot positions during the reconfiguration process for this scenario. Before encountering each other, the two teams are moving in formation and consequently, the mean error in the desired position of the robots is low. When the teams encounter each other (team leaders separated by a distance of 70 cm or less) they stop and run a WVG. Examples of quota and



(a) (b)

Fig. 9 (a) Time spent by the robots in the STOP_AND_WAIT state and CONTINUE_PREVIOUS_MOTION state of the DYN-REFORM algorithm for the three different types of obstacles. (b) Percentage of the environment covered by a set of 5 robots initially configured as a team without and with DYN-REFORM algorithm. The environment is 2×2 m^2 with $0\%, 10\%$ or 20% of the area of the environment occupied by obstacles. Each experiment was run for a period of 30 mins. Error bars were omitted for legibility

weight values from one experiment run show that the weights of the robots in the two teams are $\{1.28, 1.2, 1.2, 1.2, \}$ and $\{1.32, 1.12, 1.16\}$ respectively, while the quota $Q = 0.9 \times \sum_i w_i = 7.6$. The WVG outputs the combined set of all robots in both teams as the BMWC, implying that the two teams have to merge into a new team. As in the case of reformation with the flat wall obstacle, we notice a large spike in the mean error of the positions of the robots around 120 seconds in Figure 8(c) because the robots from the combined teams get in each others' way while forming the new team. However, finally, they manage to get into their desired position before NUM-FORMATION-REATTEMPTS and move together as one team.

Figure 9(a) shows the mean times spent by the robots in the STOP_AND_WAIT state and the CONTINUE_PREVIOUS_MOTION state for the three different obstacle types. The time in the STOP_AND_WAIT state is the highest for the flat wall because all robots stop at the flat wall, while it is the lowest for the perpendicular wall where only two robots stop and the rest of the team continues its motion. A complementary trend happens for the time spent in the CONTINUE_PREVIOUS_MOTION state with a very low value when all team members stop at the flat wall, and a higher value in the perpendicular wall case when some team members never encounter the wall and continue moving until their WVG-TIMER expires and they run a WVG.

Figure 9(b) shows the improvement in coverage achieved using the DYN-REFORM algorithm by a set of 5 robots, intially configured as a team. The robots are placed within a 2×2 m^2 walled environment with 0%, 10% or 20% of the total area of the environment occupied by obstacles. We observe that the robots usin the DYN-REFORM, because of their capability to dynamically reconfigure at obstacles and avoid inefficient configurations, are able to improve coverage by about 5% when there are no obstacles, and about $7-10\%$ when there are obstacles in the environment.

4 Conclusions and Future Work

In this paper, we have described a novel dynamic reconfiguration technique based on weighted voting games called DYN-REFORM, that allows robust and distributed multi-robot team formations. Simulation results on Webots with e-puck robots show that the DYN-REFORM algorithm allows robots to gracefully split and merge into new teams on encountering other teams or obstacles. The robots in our system use absolute positioning information using a GPS and compass for the area coverage application. For a scenario that requires team reformation only without requiring to record coordinates of covered regions, the DYN-REFORM algorithm can work with only relative position information of the robots participating in a WVG, enabled, for example, with IR-based range and bearing sensors. For our experiments, the quota for forming a winning coalition in a WVG was kept constant at a fraction of the team's total weight to lean towards forming teams around $5-7$ robots. A future problem we are investigating is to dynamically adapt the value of this quota, so that the team size can be automatically changed in cluttered or open environments perceived by the robots. Yet another direction is to integrate more perceptual data from the environment, e.g., from laser scans, camera, etc., into the DYN-REFORM algorithm to improve its performance. Finally, a future direction we are investigating is to use teams of heterogeneous robots with diverse sensor capabilities and how to integrate the robot heterogeneity into the WVG framework.

References

1. E. Bahceci, O. Soysal and E. Sahin, "Review: Pattern formation and adaptation in multi-robot systems," CMU Tech. Report no. CMU-RI-TR-03-43, 2003.
2. T. Balch and R. Arkin, "Behavior-based formation control of multi-robot teams," IEEE Transactions on Robotics and Automation, vol. 14, no. 6, 1998, pp. 926-939.
3. K. Cheng, P. Dasgupta, and Yi Wang, "Distributed Area Coverage Using Robot Flocks," World Congress on Nature and Biologically Inspired Computing (NaBIC'09), 2009, pp.678-683.
4. K. Cheng and 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, 2010, pp. 9-15.
5. T. Cormen, C. Leiseron, R. Rivest and C. Stein, "Intro. to Algorithms," McGraw Hill, 2001.
6. R. Falconi, S. Gowal, A. Martinoli, "Graph Based Distributed Control of Non-Holonomic Vehicles Endowed with Local Positioning Information Engaged in Escorting Missions," ICRA 2010, Anchorage, AK, pp. 3207-3214.
7. J. Fredslund and M. Mataric, "A general algorithm for robot formations using local sensing and minimal comm," IEEE Trans. on Rob. and Auton., vol. 18, no. 5, 2002, pp. 837-846.
8. F. Gokce and E. Sahin, "To flock or not to flock: the pros and cons of flocking in long-range migration of mobile robot swarms," AAMAS 2009, pp. 65-72.
9. M. Ji and M. Egerstedt, "A Graph-Theoretic Characterization of Controllability for Multi-Agent Systems," Proc. American Control Conference, New York, NY, 2007, pp. 4588-4593.
10. G. Kaminka, R. Schechter, and V. Sadov, "Using Sensor Morphology for Multirobot Formations," IEEE Transactions on Robotics, 2008, Vol. 24, No. 2, pp. 271-282.
11. C. Reynolds, "Flocks, herds and schools: A distributed behavioral model," Computer Graphics, vol. 21, no. 4, 1987, pp. 25-34.
12. S. Rutishauser, N. Correll, and A. Martinoli, "Collaborative Coverage using a Swarm of Networked Miniature Robots," Robotics and Auton. Systems, vol. 57, no. 5, 2009, pp. 517-525.
13. B. Smith, M. Egerstedt, and A. Howard, "Automatic Generation of Persistent Formations for Multi-Agent Networks Under Range Constraints," Mobile Networks and Applications Journal, Vol. 14, pp. 322-335, 2009.