Article

The International Journal of Robotics Research
1–14
© The Author(s) 2015
Reprints and permissions:
sagepub.co.uk/journalsPermissions.nav
DOI: 10.1177/0278364915595707
ijr.sagepub.com
SAGE

Coalition formation games for dynamic multirobot tasks

Haluk Bayram and H. Işıl Bozma

Abstract

In this paper we study the problem of forming coalitions for dynamic tasks in multirobot systems. As robots, either individually or in groups, encounter new tasks for which individual or group resources do not suffice, robot coalitions that are collectively capable of meeting these requirements need to be formed. We propose a hybrid approach to this problem where coalitions proceed with the task if they have sufficient resources after liberating redundant members while they report it to a task coordinator in cases where their resources do not suffice. In turn, the task coordinator forms capable coalitions based on coalition formation games in which groups of robots are evaluated together in regards to each task's required resources and cost of forming a coalition. The resulting coalitions are such that no group of robots has a viable alternative to staying within their assigned coalition. Thus, as new tasks are confronted, coalitions merge and split so that the resulting coalitions are capable of the newly encountered tasks. Simulations and experiments performed on groups of heterogeneous mobile robots demonstrate the effectiveness of the proposed approach.

Keywords

Multirobot task allocation, cooperative robots, game theory, coalition formation games

1. Introduction

In this paper, we consider the problem of coalition formation for dynamic multirobot tasks that require a multitude of different resources in order to be successfully completed. Such scenarios occur when the resources of an individual robot are not sufficient for task execution. For example, tasks such as data collection or clearing an obstruction in robotic exploration may require different types of equipment or assembly tasks in manufacturing settings may demand a variety of tools, which the encountering robot(s) may not (all) have. In these cases, a group of robots having the required resources need to come together in order to complete the task. While the type of tasks and hence the required resources (sensory information (Chen and Sun, 2011; Tang and Parker, 2005), processing (Hoeing et al., 2007), power or physical labor (Bererton and Khosla, 2002)) will vary depending on the application, in all, assembling robot teams, commonly known as coalitions, that have sufficient resources needs to be addressed effectively. This is a challenging problem. As tasks are dynamic, they are encountered at unpredictable places or times, which implies that the robot teams cannot be assigned a priori. If the coalitions are formed with resources far surpassing the requirements, then it may not be possible to accomplish other encountered tasks. Furthermore, robots' locations will need to be considered since choosing far away robots may lead to delays in the tasks or unnecessary energy consumption. As this is known to be strongly NP-hard, approximate solutions with an emphasis on computational feasibility and practical applicability need to be developed (Korsah et al., 2013).

The contribution of the paper is to propose a novel approach to this problem. It is assumed that each robot can participate in one task at a time and thus can be a member of only one coalition. If a coalition has sufficient resources for a newly encountered task, it proceeds with the task after member robots whose contributions to the coalition's common task are redundant leave the coalition and thus become available to other tasks. In case the coalition does not have sufficient resources, the task is reported to a task coordinator that is responsible for assembling robot coalitions with sufficient resources. The coalition assignments are then conveyed to the robots which then form them as instructed. The novelties of this approach are two-fold.

Intelligent Systems Laboratory, Electrical and Electronics Engineering Department, Bogazici University, Bebek, Istanbul, Turkey

Corresponding author:
Haluk Bayram, Intelligent Systems Laboratory, Electrical and Electronics Engineering Department, Bogazici University, Bebek, Istanbul 34342, Turkey.
Email: hbayram@boun.edu.tr

Downloaded from ijr.sagepub.com at University of New England on January 17, 2016

- First, it is a hybrid approach with both decentralized and centralized decision-making regarding the coalition members.
- Second, the task coordinator forms the coalitions via merge and split operations while considering task related constraints, motivated by work in coalition formation games (CFGs) (Apt and Witzel, 2009).

The advantage of this approach is that, differing from previous related work, coalitions are optimal in the sense that no group of robots has a viable alternative to staying within their assigned coalition as the resulting coalitions are ensured to be $\mathcal{D}_{hp}$-stable. As a result, as new tasks are encountered, coalitions merge and split in a manner where the resulting coalitions are capable of doing these tasks. To the best of the authors' knowledge, while there has been significant progress in multirobot task allocation, we are not aware of such a hybrid approach that utilizes CFGs and thus enables $\mathcal{D}_{hp}$-stability.

The outline of this paper is as follows: First, related work is reviewed in Section 2. Next, coalitions and tasks are formulated in Section 3. The task coordinator is explained in Section 4 including a discussion of coalition stability and computational analysis. Extensive simulations with 50 robots provide insight on performance in a range of multirobot tasks in Section 5. Section 6 presents performance results from the application of this approach on a team of five robots. The paper concludes with a brief summary along with future directions. The details of the experimental setup are discussed briefly in Appendix B.

2. Related literature

The robotics community has addressed coalition formation in multirobot task allocation (MRTA) problems. In the taxonomy[1] of MRTA problems (Gerkey and Mataric, 2004), it is an instance of the ST–MR–IA problems where tasks that require the combined effort of multiple robots are considered. As such, the robots need to be split into task-specific coalitions while taking their individual constraints into consideration. In most settings, all of the tasks are assumed to be known initially, in contrast to dynamic tasks where environmental conditions are fluctuating greatly so that only instantaneous assignment is possible (Fua and Ge, 2005). Optimality is defined either with respect to an overall objective or a set of objective functions that encode demands, resources and gains if possible (Chapman et al., 2011; Lau and Zhang, 2003). Due to the NP-hard nature of the problem, many heuristic methods have been developed (Gerkey and Mataric, 2004).

The decision-making varies from being centralized to being distributed (Parker, 2012; Zhang et al., 2012). With centralized approaches, the constrained optimization is solved after collecting all the relevant data from the robots. For example, in Vig and Adams (2006), a multi-agent task allocation algorithm (Shehory and Kraus, 1998) is modified in order to accommodate the fact that resources of a coalition are not collectively available to member robots and cannot be redistributed among them. The problem is reformulated as a transportation problem by considering suitability and allocating robots to tasks so that their resources match the task's required resources (Fua and Ge, 2005). An anytime algorithm is shown to have bounded solution with a minimal search (Dang and Jennings, 2006). Improved solutions provide solutions in polynomial time in the case of robots of each type being indistinguishable with a fixed coalition population using dynamic programming (Service and Adams, 2011). The problem is generalized by allowing coalitions to be bounded by a fixed number via modifying the greedy iterative algorithm based on the set partitioning problem. Two natural greedy heuristics are extended via a new greedy heuristic that considers the expected loss of utility due to the assigned robots and task as an offset and uses the offset utility for task assignment (Zhang and Parker, 2013). A greedy optimal solution is proposed via a leader follower coalition method where coalition utility is maximized for every assigned task (Chen and Sun, 2011). In general, the efficiency of the resulting solutions is easier to ensure and communication requirements are linear with respect to the number of robots with no negotiation required. However, the computational requirements become unpractical as the number of tasks and robots increases. Distributed approaches solve the constrained optimization problem in a decentralized manner (Cao et al., 1997). For example, in a class of problems known as distributed constraint optimization problems, each robot or group controls one set of variables and together they have the joint goal of maximizing a global objective function (Chapman et al., 2011; Yokoo et al., 1998). One of the most popular approaches that falls into this category is the market-based strategy where auctions can be conducted in a distributed manner such as regional opportunistic centralization (Bernardine Dias and Stentz, 2002; Brunet et al., 2008; Chaimowicz et al., 2002; Gerkey and Mataric, 2002). The auction process may be split into task and robot auctions as is done in the double round auction approach (Guerrero and Oliver, 2012). However, the locality of decisions may block idle, but remote robots coming to assistance. As such, while distributed approaches are advantageous with respect to scalability, the efficiency of the resulting solutions are harder to ensure while communication requirements increase quadratically with the number of robots and special negotiation schemes are required so as to decide when to terminate decision-making (Bayram and Bozma, 2014). In practice, systems may not conform to a strict centralized/decentralized dichotomy and may contain both elements (Wagner et al., 2004). For example, centralized market-based task allocation combines the efficiency of a centralization (the auctioneer decides with overview of the situation) with the advantages of distributed approaches (much of the calculation is done by the individual robots preparing their bids) (Parker, 1998) where a centralized auctioneer is responsible for the optimization of a global objective using either combinatorial (de

Downloaded from ijr.sagepub.com at University of New England on January 17, 2016

Vries and Vohra, 2003) or greedy (Koenig et al., 2006) approaches. However, scalability issues (such as when to stop the auction) arise as the number of robots increases (Ducatelle et al., 2009; Lerman et al., 2006).

Coalition formation has also been addressed in multiagent systems where the goal is to find a coalition structure that maximizes the sum of the values of the coalitions. It is an instance of a set partition problem which is known to be a NP-complete (Garey and Johnson, 1979). As the exorbitant number of coalition structures does not allow exhaustive search for the optimal one, the focus has been on finding a coalition structure via a partial search with guaranteed proximity to the optimum. It is shown that the number of coalition structures that need to be searched for establishing a bound is required to be greater than a calculated threshold (Sandholm et al., 1999) along with an algorithm that establishes a tight bound within this minimal amount of search. Shehory and Kraus (1998) use a greedy heuristic to yield a coalition structure that is provably within a bound while limiting the coalition sizes. This approach is general as it can be applied in environments that are not necessarily superadditive.[2] Such problems have also been considered within game theory (Hajdukova, 2006; Saad et al., 2009) so that the grand coalition is no longer optimal (Aumann and Dreze, 1974). Since the addition of more robots to a coalition increases interference between the robots and computational cost, the multi-robot systems fall into the non-superadditive category (Vig, 2006). In CFGs, the focus is on obtaining stable partitions of the players, which implies that players have no incentives to change their coalitions. As such, stability is related to the type of membership changes allowed. In hedonistic games where only one individual player is allowed to change its coalition at a time, stability definitions vary from contractual individual stability to individual stability to Nash stability (Bogomolnaia and Jackson, 2002). In more general settings, groups of players are allowed to change their coalitions simultaneously. In this case, stability is related to the set of split and merge rules (Apt and Witzel, 2009). In all, a comparison operator that orders the sets of coalitions is defined. This comparison operator is either based on the coalition value that quantifies the worth of a coalition or the individual players' payoff (Saad et al., 2011). Our proposed approach is motivated by these ideas where the process of finding optimal robot coalitions is modeled as a CFG, considering task-related preferences such as resource satisfaction, resource utilization and site proximity.

3. Multirobot coalitions and tasks

A multirobot system consists of a set of $\mathcal{P} = \{1, \ldots, p\}$ robots. We assume that each robot $i \in \mathcal{P}$ is uniquely identifiable. It is associated with a time-varying position vector $b_i \in \mathbb{R}^2$. The robots are assumed to be heterogeneous, which imply that they vary in their resources. Assuming there are N_r different types of resources, each robot i is also associated with a resource vector $r_i = [r_i(1),\ldots,r_i(N_r)]^{\mathrm{T}}$ with $r_i(j) \geq 0, j = 1,\ldots,N_r$ where $r_i(j) \in \mathbb{R}^{\geq 0}$ denotes the amount of j th resource that robot i has. If robot i does not have any of resource j, then $r_i(j) = 0$.

3.1. Coalitions and resources

A coalition C_c is a non-empty subset of $\mathcal{P}$. A coalition with just one robot is referred to as singleton coalition while the set $\mathcal{P}$ is known as the grand coalition. Each coalition C_c is associated with a set of resources with possible types of the resources known. The resource vector is denoted $R_c = [R_c(1),\ldots,R_c(N_r)]^{\mathrm{T}}$ with $R_c(j) \geq 0$, $j = 1,\ldots,N_r$. It is assumed that resources are additive - namely $R_c(j) = \sum_{k \in C_c} r_k(j)$. Furthermore, each coalition has a leader (head). The leader coordinates the coalition. The leader may change over time as the coalitions evolve. The rules for selecting leader are as follows. First, the leader does not change as long as it does not leave the coalition. Second, if the leader leaves the coalition or the coalition does not have a leader, the robot with the smallest robot ID value becomes the coalition leader.

3.2. Tasks and resources

As a robot or a coalition of robots is moving around the workspace, it will come across a number of tasks. As these tasks are dynamic, there is no a priori information regarding their spatial locations or when they are likely to encounter one. Once a task T is encountered, the coalition leader records the task including the following.

- Required resources: $\tau = [\tau_1, \ldots, \tau_{N_r}]$ where $\tau_i \in \mathbb{R}^{\geq 0}$, $i = 1,\ldots,N_r$. If a resource l is not required for the task T, then $\tau_l = 0$.
- Location of the task: $b \in \mathbb{R}^2$.
- Time of encounter: $t_e \in \mathbb{R}^{\geq 0}$.
- Time-out duration: $\Delta t_o \in \mathbb{R}^{>0}$. This indicates the maximum allowed waiting period for getting the sufficient resources and starting with the task.
- Time when the task starts being handled: $t_s \in \mathbb{R}^{>0}$.
- Task duration: $\Delta t_d \in \mathbb{R}^{>0}$.
- Status of the task: $s \in \{-1, 0^-, 0^+, 1^-, 1^+, 1\}$ -

$$s = \begin{cases} 0^- & \text{if task is waiting in the coalition} \\ 0+ & \text{if task is waiting in the coordinator} \\ 1+ & \text{if task is being handled} \\ 1^- & \text{if a coalition is assigned, but task has not started yet} \\ 1 & \text{if task was completed} \\ -1 & \text{if task could not be completed} \end{cases}$$

When a task is initiated, $s = 0^-$ - which indicates the task has been just encountered. When resources are found to be insufficient, $s = 0^+$. The case $s = 1^-$ indicates that a

Downloaded from ijr.sagepub.com at University of New England on January 17, 2016

coalition has been assigned, but all the coalition members have not reached to the task site.

3.3. Coalition value functions

The coalition value function v relates a given coalition C_c with a given task T via encoding resource sufficiency, resource utilization and members' proximity to task site. It is defined as:

$$v(C_c, T) = \frac{1}{1 + \beta(C_c, T)} \tag{1}$$

where the term $\beta(C_c, T)$ is composed of three terms:

$$\beta(C_c, T) = w_1 \sum_{j=1}^{N_r} \gamma(\tau_j - R_c(j)) + w_2 \sum_{j=1}^{N_r} \left(1 - \frac{R_c(j)}{\tau_j + \epsilon}\right)^2 + w_3 \sum_{i \in C_c} \frac{\delta_{i,T}}{2\rho_o}$$

where $\epsilon > 0$ is a small value that ensures the denominator is never zero. The first term measures whether the coalition has sufficient resources to complete the task where

$$\gamma(x) = \begin{cases} 0 & \text{if } x \leq 0 \\ x^2 & \text{if } x > 0 \end{cases}$$

The second term indicates preferences for robots that utilize their resources to a greater extent. The third term considers the proximities of coalition members to the task site, as nearby free robots will be preferred where $\delta_{i,T}$ is the distance between robot i and the location of task T and the parameter ρ_0 is a normalizing factor for distance, usually taken to be the radius of the workspace. The parameters w_1, w_2 and w_3 are relative weighting parameters of resource satisfaction, resource utilization and site proximity, respectively with values set according to particular preferences.

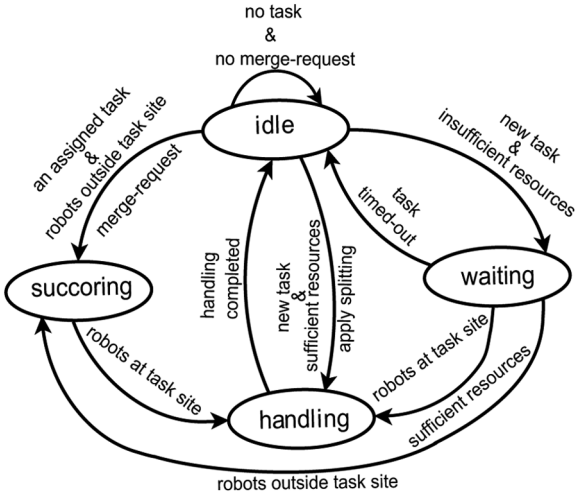


Fig. 1. Decentralized decision-making: task automaton of a coalition.

3.4. Handling tasks

Each coalition has a task automaton for handling tasks that is coordinated by the coalition leader. The task automaton as shown in Figure 1 is designed to have four states: "idle", "handling", "succoring" and "waiting". These states are selected to reflect logical modes of operation. Normally, the coalition is in idle state which indicates that the coalition is not associated with or engaged in a task and thus is ready for new tasks. The coalition goes into the handling state if it has sufficient resources and at the task site. The coalition goes into the waiting state if it has encountered a task, but does not have the sufficient resources. Finally, the succoring state indicates that a coalition has been assigned a task and is moving to the task site.

State transitions among different states are based on five rules. These rules are defined considering the encountered tasks as well as the encountering coalitions. In particular, the sufficiency of the coalition's resources, task site and time-out are paid special attention as follows.

Rule 1: If the coalition is idle, has enough resources for this task - namely $\forall \tau_j,\ j = 1, \ldots, N_r, \quad R_c(j) \geq \tau_j$ and is at the task site, it goes into the handling state.

Rule 2: In the transition from the idle to the handling state, if the coalition has surplus of resources: namely $\exists j$ such that $R_c(j) - \tau_j > 0$, then it splits as much as possible in order to maximize coalition value $v(C_c, T)$. If the leader is taken out, the coalition then chooses the next leader. The result is reported to the task coordinator, where the robots that leave the coalition go into the idle state. When the task is completed, it reports task completion to the task coordinator and goes back to the idle state.

Rule 3: If the coalition is idle, but does not have enough resources, then it reports the task to the task coordinator and goes into the waiting state where it remains until it hears back or time-out occurs. In case it is given sufficient resources (additional members), then it goes either into the succoring state or the handling state depending on whether it is at the task site or not.

Rule 4: If the coalition is idle and is assigned a task, but is not at the task site, it then goes into the succoring state where it starts moving to the task site. Upon all reaching the task site, the state changes to handling.

Rule 5: If the coalition is not idle when it encounters a task, the coalition leader informs the coordinator of this task.

In the handling state, before starting with the task, the coalition checks for excessive resources and attempts to be as lean as possible, as described by Rule 2. In case of having excessive resources, without contacting the coordinator, the leader splits its coalition so that some robots may be free to participate in other coalitions. If the leader is taken out, the coalition then chooses the next leader. The information is relayed to the task coordinator and then the coalition starts handling the task. At the end of task completion,

Downloaded from ijr.sagepub.com at University of New England on January 17, 2016

the task information is transmitted to the task coordinator and the coalition goes back to the idle state. The coordinator is informed of this because only idle robots or coalitions can be considered for task assignment. Rule 3 determines the transition from the idle to the waiting state. This occurs if the coalition does not have enough resources. In this case, it reports this task to the task coordinator and switches to waiting state. The coalition remains at the location where it encounters the task, waiting for a response from the task coordinator. If the task coordinator does not convey an assignment within a certain time, the coalition goes into idle state again. If the waiting robots are assigned particular tasks, they move to the task site as defined by Rule 4. During this movement, the coalition will be in succoring state. Once all of the robots reach the task location, the coalition goes into handling state. The coalition may encounter new tasks while it is not in the idle state. In this case, Rule 5 designates that the task is reported to the task coordinator. Finally, whenever the coalition goes into the idle or waiting state, this is reported to the task coordinator. Hence, communication requirements will vary depending on the encountered tasks and coalition's resources.

4. Task coordinator

The task coordinator[3] is responsible for forming the robot coalitions capable of performing encountered tasks. The coordinator maintains a list of reported tasks which are waiting (either in the coalition or in the coordinator). Let this list be denoted by $\mathcal{T} = [T^1, \ldots, T^{n_T}]$ where n_T is their total number. This list expands as new tasks are reported and shrinks as tasks get assigned or timed-out.

The set of coalitions is defined by a time-varying set $C(t) = C^+(t) \cup C'(t)$ where $C^+(t)$ denotes the coalitions (consisting of either a single robot or multiple robots) that are currently engaged in a task while $C'(t)$ denotes the coalitions that are idle or waiting composed of member robots $\mathcal{P}' \subseteq \mathcal{P}$. Periodically, the task coordinator considers the current list of pending tasks $\mathcal{T}$, updates the set of coalitions $C'(t)$ in order to assign coalitions to these tasks and informs the robots accordingly. Any family of $\mathcal{P}' = \{C_1, \ldots, C_{n_C}\}$ of mutually disjoint coalitions is referred to as a collection in $\mathcal{P}'$. If $\bigcup_{c=1}^{n_C} C_c = \mathcal{P}'$ where n_C is the number of coalitions, then C' is called a partition on $\mathcal{P}'$ (Apt and Witzel, 2009). As such, in each update, the task coordinator needs to find a partition C' of $\mathcal{P}'$ and the assignment of tasks to member coalitions so that the pending tasks can be completed in a maximal manner. Note that depending on the tasks encountered and robots' resources, the newly formed partition C' may vary from being identical to being very different as compared with $C'(t)$. Of course, it may not be possible to find a coalition for each task or any of the pending tasks given the currently available robots. The simplest approach to this problem is to conduct the exhaustive search of all of the possible partitions while considering all possible assignment of tasks. As the number of possible partitions is given by the Bell number $\mathbf{B}_p$ which is very large even with a modest robot population size $p \approx 10$ (Sandholm et al., 1999), the computational complexity of the required search is exorbitant.

4.1. CFG

In the proposed approach, the process of finding such a partition is modeled as a CFG. The coordinator periodically starts a CFG, considering all of the pending tasks $\mathcal{T}$. A CFG starts with the current coalition structure as defined by the partition $C'(t)$ on $\mathcal{P}'$. There are two aspects in defining each game. First, two partitions are compared using a predefined $\rhd$ comparison relation.

Definition 1. *A comparison relation $\rhd$ is defined for comparing two collections A and B that are partitions of the same set $\mathcal{P}'$. If $A \rhd B$, then the partition A is preferred to partition B.*

Note that each comparison relation is used only to compare partitions of the same set of players. Partitions of different sets of players are incomparable. As such, different coalitions are allowed to interact, taking the decision to merge or split based on the comparison relation $\rhd$. Various criteria can be used as comparison relation between partitions (Apt and Witzel, 2009). An adequate individual value order that can be used is the Pareto order. The Pareto order is defined as

$$A \rhd B \Leftrightarrow \varphi_i(A, T) \geq \varphi_i(B, T) \forall i \in \mathcal{P}'$$

with at least one strict inequality ($>$) for one robot $k \in \mathcal{P}'$ and where the robot payoff $\varphi_i(C, T)$ describes the overall utility a robot $i \in \mathcal{P}'$ receives for being in coalition $C_c \in C(t)$ that is associated with task T. The robot payoff function is defined by the coalition value function as

$$\varphi_i(C, T) = \varphi_i(C_c, T) = v(C_c, T) \tag{2}$$

Second, the game evolution is based on two operations, called "merge" and "split", that allow to modify a partition C as follows (Apt and Radzik, 2006).

- **Merge**: If $\bigcup_{j=1}^{k} C_j \rhd \{C_1, \ldots, C_k\}$, then merge $\{C_1,\ldots,C_k\}$ as $\bigcup_{j=1}^{k} C_j$ - namely $\{C_1, \ldots, C_k\} \cup C \rightarrow \bigcup_{j=1}^{k} C_j \cup C$.
- **Split**: If $\{C_1,\ldots,C_k\}$ is a collection such that $\{C_1, \ldots, C_k\} \rhd \bigcup_{j=1}^{k} C_j$, then split $\bigcup_{j=1}^{k} C_j$ as $\{C_1,\ldots,C_k\}$, namely $\bigcup_{j=1}^{k} C_j \cup C \rightarrow \{C_1, \ldots, C_k\} \cup C$.

The coordinator uses merge and split operations on the existing coalitions. With the Pareto order, the task coordinator decides to merge or split coalitions only if at least one coalition is able to strictly improve its individual value through this process without decreasing the other coalitions' values. Therefore, the merge operation by Pareto

Downloaded from ijr.sagepub.com at University of New England on January 17, 2016

Algorithm 1. Centralized decision making: Task coordinator automaton.

```
1:  while change in partition C', repeat do
2:    while change in partition C', merge do
3:      for all T^k ∈ T | s^k = 0^- or s^k = 0^+ do
4:        if s^k = 0^+ then
5:          Assign C_c with the highest v(C_c, T^k)
6:        end if
7:        Find a coalition C_d such that C_c ∪ C_d ▷ C_c
8:        if C_d ∉ ∅ then
9:          C' = (C' − C_c) − C_d, C_c = C_c ∪ C_d, C' = C' ∪ C_c
10:       end if
11:     end for
12:   end while
13:   while change in partition C', split do
14:     for all T^k ∈ T | s^k = 0^- or s^k = 0^+ do
15:       if a coalition C_c is assigned to T^k then
16:         Find i∈C_c such that C_c − {i} ▷ C_c
17:         C_c = C_c − {i}, C' = C' ∪ {i}
18:       end if
19:     end for
20:   end while
21: end while
```

order is a binding agreement among the robots to operate together if it is beneficial for the tasks.

The task coordinator algorithm is as given in Algorithm 1. The main loop of the algorithm consists of two consecutive loops for merge and split operations in which only the coalitions in idle or waiting state are considered. The merge phase is described in lines 2–12 of Algorithm 1. Here, the coordinator checks if a task T^k is associated with a coalition or not by checking task status s^k. If $s = 0^-$, the coalition that reported it has insufficient resources. The coordinator starts merge operations - using this coalition. If there is no coalition associated with this task ($s^k = 0^-$), it determines the coalition C_c with the highest coalition value $v(C_c,T^k)$ - namely $C_c = \arg\max_{C_c \in C'} v(C_c, T^k)$. Then, it starts a merging phase so as to increase the coalition value based on the Pareto order. The merge loop continues as long as there is a change in the partition C'. After passing the merge loop, in the split loop (lines 13–20 of Algorithm 1), the assigned coalitions with excessive resource for their tasks are split. In the split operation, one of the robots in the coalition is selected based on the Pareto order such that the coalition without this selected robot has higher coalition value. This loop continues as long as there is a change in the partition C'. Merge–split operations are iteratively applied until all the coalitions associated with all of the tasks stabilize. As a result, each task $T \in \mathcal{T}$ is associated either with a coalition or empty set, which implies that the coordinator cannot find a coalition capable of performing that particular task.

4.2. Convergence and stability

There are two issues regarding the behavior of each CFG: namely whether the game terminates and in the case of convergence, the properties of the resulting partition. The convergence of the CFG is ensured by the following theorem (Apt and Witzel, 2009).

Theorem 1. *Suppose that $\rhd$ is a comparison relation. Every iteration of the merge and split operations terminates.*

As each merge–split operation increases the values of the coalitions with an assigned task, the process terminates.

The resulting partition $C' = \{C_1, \ldots, C_{n_C}\}$ is evaluated with respect to the stability of the coalition structure. Stability captures the idea that no robot or group of robots (as defined) has an incentive to change the existing coalition structure (Apt and Radzik, 2006). Thus, it depends on the type of coalition membership changes allowed. Allowable membership changes are defined by a defection function $\mathcal{D}$ that associates with each partition C' of $\mathcal{P}'$ a group of collections in $\mathcal{P}'$ such that robots can leave the partition C' by forming new and separate group of robots $\bigcup_{j=1}^{l} P_l$ divided according to one $P = \{P_1,\ldots,P_l\}$ of these collections. As such, different stability notions are obtained by considering different defection functions.

A partition $C' = \{C_1,\ldots,C_l\}$ of $\mathcal{P}'$ is $\mathcal{D}$-stable if no group of robots is interested in leaving C' when the robots who leave can only form the collections allowed by $\mathcal{D}(C')$ (Apt and Witzel, 2009). Mathematically, it is defined using the partition comparison relation $\rhd$ as follows.

Definition 2. (**$\mathcal{D}$-stability**) *A partition C' is called $\mathcal{D}$-stable if for all $P \in \mathcal{D}(C')$ such that $P[C'] \neq P$, $P[C'] \rhd P$ where $P[C']$ denotes the collection P in the frame of C' defined as $P[C'] = \{C_1 \cap \bigcup P, \ldots, C_{n_C} \cap \bigcup P\} \setminus \{\emptyset\}$.*

The most general case is with a defection function $\mathcal{D}_c$ that maps each partition C' to the family $\mathcal{D}_c(C')$ of all collections in $\mathcal{P}'$. As such, any group of robots can leave C' and create an arbitrary collection in $\mathcal{P}'$. However, $\mathcal{D}_c-$ stability is hard to attain as it requires the value functions to be superadditive: a condition that will not hold in many applications. An alternative definition is based on C'-homogeneity. A partition $Q = \{Q_1,\ldots,Q_l\}$ is C'-homogeneous if for each $j \in \{1,\ldots,l\}$, there exists $i \in \{1,\ldots,n_C\}$ such that either $Q_j \subseteq C_i$ or $C_i \subseteq Q_j$. Any C'-homogeneous partition arises from C' by allowing each coalition either to split into smaller coalitions or to merge with other coalitions. With this definition, the defection function $\mathcal{D}_{hp}$ is defined such that for each partition C', $\mathcal{D}_{hp}(C')$ is the family of all C'-homogeneous partitions in $\mathcal{P}'$. Theorem 2 admits the following characterization of $\mathcal{D}_{hp}$-stability (Apt and Radzik, 2006).

Theorem 2. *A partition $C' = \{C_1, \ldots, C_{n_C}\}$ of $\mathcal{P}'$ is $\mathcal{D}_{hp}$-stable if and only if the following two conditions are satisfied.*

1. *No coalition has an incentive to split; namely, for all $i \in \{1, \ldots, n_C\}$ and for each partition $\{P_1,\ldots,P_l\}$ of coalition C_i, $C' \triangleright \tilde{C}$ where $\tilde{C} = (C' - C_i) \cup \{P_j\}_{j=1}^{l}$*
2. *No set of coalitions has an incentive to merge; namely, for all $L \subseteq \{1,\ldots,n_C\}$ $C' \triangleright \tilde{C}$ where $\tilde{C} = \left(C' - \{C_i\}_{i\in L}\right) \cup \left\{\bigcup_{i\in L} C_i\right\}$*

Downloaded from ijr.sagepub.com at University of New England on January 17, 2016

This result implies that robots are allowed to leave the partition C' only by means of merges or splittings, albeit with multiple applications. With the value functions as defined by (1) and (2), the two conditions can equivalently be expressed as:

1. $\forall C_c \in C'$ with C_c assigned to task $\alpha(C_c) \in \mathcal{T}$ and for each partition $\{P_1,\ldots,P_l\}$ of coalition C_c, $v(C_c, \alpha(C_c)) \geq \sum_{i=1}^{l} v(P_i, \alpha(C_c))$;
2. $\forall T \in \mathcal{T}$, $\forall L \subseteq \{1, \ldots, n_C\} \sum_{i=1}^{L} v(C_i, T) \geq v\left(\bigcup_{i \in L} C_i, T\right)$.

An immediate consequence of Theorem 2 is that a partition C' is $\mathcal{D}_{hp}-$ stable if and only if it is the outcome of the merge and split rules. As such, C' will be $\mathcal{D}_{hp}$-stable.

4.3. Computational complexity

The complexity of each CFG is due to the complexity of the merge and split operations, since $\mathcal{D}_{hp}$-stability is reached via merge and split operations (Saad et al., 2011). During one merge process, the task coordinator attempts to merge a coalition with the other available coalitions in a pairwise manner. In the worst-case scenario, it needs to make a merge attempt of every coalition with all of the other coalitions, before finding a suitable merge partner. This implies $n_C - 1$ attempts for merge for the first coalition, $n_C - 2$ attempts for the second and so on. The total worst case number of merge attempts will be $\sum_{i=1}^{n_C} i = \frac{n_C(n_C-1)}{2}$. This is repeated for each task with the number of coalitions getting smaller. In practice, the merge process requires a significantly lower number of attempts since finding a suitable coalition does not always require going through all of the merge attempts, since the merge will occur immediately once a suitable coalition is identified. Moreover, once the p robots are assembled into coalitions after an initial run of the CFG, in the subsequent CFGs, the task coordinator will deal with a set of coalitions n_C that is much smaller than the number of robots p. As such, the merge attempts per coalition will be further reduced. Finally, as coalitions become populated themselves, the number of available n_C gets smaller, which implies that the number of merging possibilities will decrease.

For the split process, in a worst-case scenario, splitting can involve finding all of the possible partitions of the set formed by the robots in a single coalition. As this number is given by the Bell number $\mathbf{B}_{n_P}$ where $n_P = \max_{C \in \mathcal{P}} |C|$ denotes the maximum coalition population size, it grows exponentially with the number of robots in the coalition. However, the cost function ensures that the size of each coalition is limited as much as possible which implies that the coalitions have relatively small sizes. Thus, the split complexity is limited to finding possible partitions for small sets which is reasonable in terms of complexity. Furthermore, in most cases, the task coordinator does not need to search for all possible split forms. Once it identifies a split structure that satisfies the Pareto order, the robots in this coalition will be split, and the coordinator will not further search for splitting. In summary, the practical aspects as dictated by the sequential split search as well as the cost function will significantly diminish the split complexity. This analysis suggests that while ensuring $\mathcal{D}_{hp}$-stability, the computational complexity of the proposed approach will be relatively much lower as compared to finding the optimal partition. As such, in case of the failure of the task coordinator, another robot can easily assume this role - as the required computational resources will not be high.

Table 1. Simulation settings.

Parameter	Value
Mission duration	60 minutes
Number of robot types	10
Number of resource types N_r	5
Number of task types	10
Time-out duration Δt_o	{1, 2} minutes
Task rate λ	{5, 10, 20} tasks per hour
Task resource τ_i levels	Low - [20, 25] and High - [45, 50] units
Robot resource $r_i(j)$ levels	Low - [1, 5] and High - [6, 10] units
Task duration Δt_d	{2, 4} minutes

5. Simulations

Extensive simulations have been conducted with $p = 50$ robots placed in a workspace of roughly 31,400 m^2. The robots are assumed to be patrolling in this workspace. They are cylinder shaped with radii 15 cm and can move with maximum speed of 0.3 m/s. The simulation settings are as given in Table 1. There are $N_r = 5$ different resources. We assume that each robot is one of 10 types with either all low resources $r_i(j) \in [1, 5]$ or all high $r_i(j) \in [6, 10]$. The task locations are generated dynamically via a Poisson process with parameter λ tasks per hour. The average number of tasks increases with increased λ value, thus making the overall mission more challenging. There are 10 different type of tasks, where all τ_i are either in the low [20, 25] range or high [45, 50]. A sample scenario of robot types and task types is shown in Tables 2a and 2b where robots have low resources while tasks encountered also require low resources. In this case, the coalition populations capable of accomplishing the encountered tasks are expected to vary between 4 and 25 robots. The parameters of the coalition value function are set as $w_1 = 1/3$, $w_2 = 1/3$, and $w_3 = 1/3$ in order to give equal importance to resource satisfaction, resource utilization and site proximity. The coordinator starts a CFG every 10 seconds. As an example, 86 and 113 tasks are encountered during two sample missions (having $\lambda = 5$ and $\lambda = 20$ respectively) with locations as shown in Figure 2. Note that all these tasks are dynamically encountered and are not known a priori. The

Downloaded from ijr.sagepub.com at University of New England on January 17, 2016

Table 2. Sample mission scenario: Robots have low level of resources while tasks also require low level resources.

(a) Robots' resources.

Robot	Resources				
Type	$r(1)$	$r(2)$	$r(3)$	$r(4)$	$r(5)$
1	1	1	3	5	3
2	4	5	4	4	2
3	3	3	2	3	3
4	5	4	3	2	5
5	2	4	4	1	1
6	2	3	4	4	4
7	4	1	4	3	5
8	3	4	3	2	4
9	5	1	1	3	1
10	3	4	2	1	4

(b) Tasks' resources.

Task	Required Resources				
Type	τ_1	τ_2	τ_3	τ_4	τ_5
1	20	21	21	21	25
2	20	24	21	22	23
3	22	20	21	21	21
4	25	21	20	21	25
5	24	23	24	23	20
6	24	23	24	25	25
7	24	23	20	23	23
8	23	25	23	20	24
9	22	25	20	23	21
10	24	22	21	25	23

performance results with these sample missions are also shown in the same figure where tasks that are successfully completed are shown in green plus signs while those that were timed-out are in red minus signs. When the number of tasks is fewer ($\lambda = 5$), 87.8% of the 86 tasks are completed. As the number of tasks is increased ($\lambda = 20$), the percentage of the completed tasks goes down to 78% with 88 out of the 113 tasks being completed.

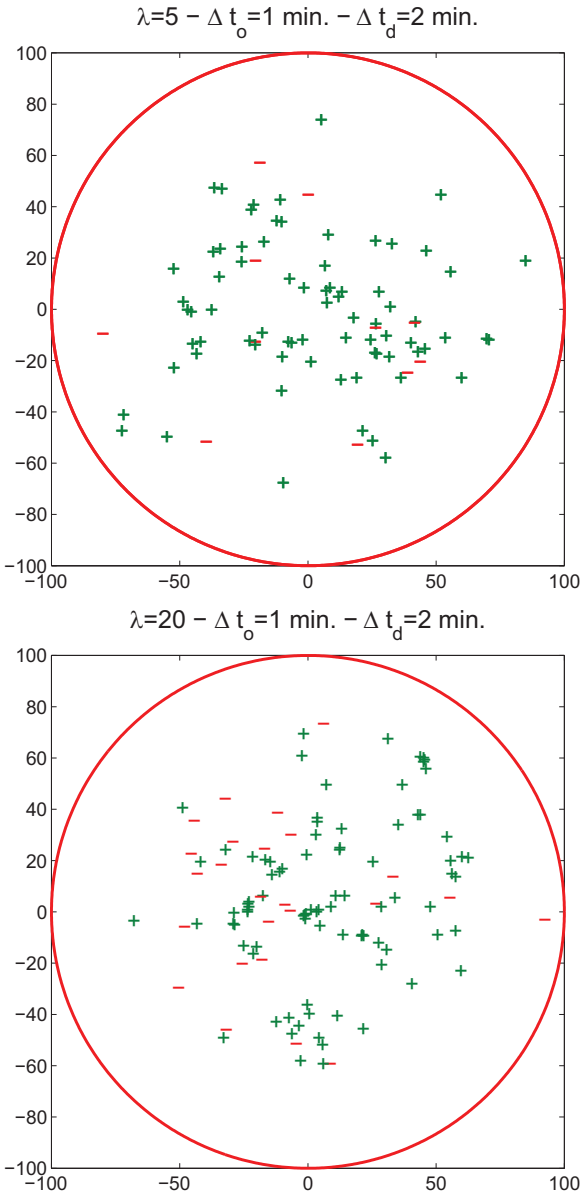


Fig. 2. Two sample missions with dynamically encountered 86 tasks ($\lambda = 5$) and 113 tasks ($\lambda = 20$). With the tasks' required resources and robots' resources both high, $\Delta t_o = 1$ minute and $\Delta t_d = 2$ minutes the completed tasks are as shown by green plus signs while timed out tasks are shown by red minus signs.

5.1. Statistical results

The missions are repeated 20 times, all starting at random initial locations for each $\lambda \in \{5, 10, 20\}$, time-out duration $\Delta t_o \in \{1,2\}$ and handling duration $\Delta t_d \in \{2, 4\}$ minutes. Hence, there are all together 240 missions that are conducted. Note that time-out periods with 1 or 2 minutes imply relatively fast response times for successful task completion. The mission performances are analyzed with respect to six measures:

- M_1, the number of completed tasks;
- M_2, percentage of timed-out tasks;
- M_3, number of coalitions;
- M_4, number of times the coordinator makes coalition assignments;
- M_5, average coalition populations at each instant;
- M_6, average percentage of required resources with respect to the coalitions' available resources for completed tasks over a mission.

The resulting performance statistics are given in Table 3. First, it is observed that the number of completed tasks M_1 varies roughly between 47 and 268 depending on Δt_o, Δt_d, λ and the level of resources as the percentage of timed-out tasks M_2 varies between 0.5% to 78.6%. With high Δt_o, low Δt_d and low λ values, the coalitions are able to complete many tasks they encounter. In contrast, with low Δt_o,

Downloaded from ijr.sagepub.com at University of New England on January 17, 2016

Table 3. Simulation results. TR, task resource level; RR, robot resource level; L, low; H, high.

	λ	5				10				20			
	Δt_o	1		2		1		2		1		2	
TR-RR	Δt_d	2	4	2	4	2	4	2	4	2	4	2	4
L-L	$\bar{M}_1$	82.3	67.5	75.6	57.5	105.4	83.9	89.7	69.6	126.6	105.9	102.3	82.7
	$\bar{M}_2$	14.1	35.1	12.3	15.7	22.2	38.8	21.1	25.5	30.4	47.8	25.6	33.6
	$\bar{M}_3$	10.5	9.5	10.7	10.2	9.2	9.4	9.3	8.4	8.2	8.3	8.0	8.2
	$\bar{M}_4$	103.8	142.8	140.0	151.4	166.6	190.7	227.4	212.8	230.1	239.2	277.9	272.4
	$\bar{M}_5$	9.4	9.8	9.7	9.3	9.3	9.4	9.7	9.4	9.4	9.5	10.0	9.4
	$\bar{M}_6$	83.1	82.2	83.0	83.7	83.9	83.5	82.5	83.2	82.4	84.1	81.6	83.8
L-H	$\bar{M}_1$	141.5	110.3	139.5	110.0	205.0	144.3	203.4	143.7	268.4	175.8	252.8	165.5
	$\bar{M}_2$	6.2	5.6	4.2	6.2	5.7	7.1	5.9	9.0	9.9	10.7	6.1	7.8
	$\bar{M}_3$	23.6	19.2	22.2	19.7	17.7	16.7	18.1	16.8	16.3	16.0	15.6	15.7
	$\bar{M}_4$	105.5	75.5	117.0	110.0	115.3	98.5	166.2	161.8	176.2	142.2	191.1	175.0
	$\bar{M}_5$	3.4	3.5	3.6	3.5	3.6	3.6	3.5	3.5	3.6	3.5	3.7	3.6
	$\bar{M}_6$	82.9	82.1	80.6	81.9	80.1	81.8	81.4	81.2	81.1	81.6	79.1	80.6
H-L	$\bar{M}_1$	68.4	58.9	56.5	47.4	90.7	75.6	72.3	64.3	117.2	123.8	93.0	89.0
	$\bar{M}_2$	45.7	59.6	34.8	51.4	58.0	64.5	44.3	56.7	62.3	78.6	50.3	68.5
	$\bar{M}_3$	11.6	11.4	10.3	9.6	9.0	8.3	8.2	8.3	6.5	8.0	6.5	7.2
	$\bar{M}_4$	178.8	181.3	230.7	230.0	244.3	218.6	274.4	277.3	273.6	300.6	307.5	325.9
	$\bar{M}_5$	19.1	19.2	18.6	19.4	19.3	20.1	19.2	18.8	19.8	19.7	19.0	19.5
	$\bar{M}_6$	87.5	87.2	87.5	85.4	87.2	84.1	86.8	85.8	83.9	84.8	85.2	84.1
H-H	$\bar{M}_1$	105.8	71.8	106.5	69.8	132.8	85.1	129.7	81.9	152.5	96.0	144.2	91.7
	$\bar{M}_2$	2.9	3.9	0.5	1.9	3.7	9.1	1.7	6.2	5.2	13.7	1.8	9.7
	$\bar{M}_3$	11.0	10.2	11.0	10.3	9.7	9.7	10.2	9.7	9.4	9.0	9.4	9.3
	$\bar{M}_4$	50.0	41.1	40.1	45.0	70.9	81.7	83.5	111.0	113.2	119.4	120.7	158.9
	$\bar{M}_5$	6.9	6.9	6.9	6.9	6.9	7.0	6.9	7.0	6.9	7.0	6.9	6.9
	$\bar{M}_6$	86.9	86.6	86.5	86.6	86.5	86.8	87.0	86.7	86.7	86.3	86.6	86.7

high Δt_d and high λ values, the tasks are likely to time out. Hence, with the increase in time-out durations, the number of handled tasks increases. Average number of coalitions decreases with the increase in λ. This is because with the low λ values, there are less encountered tasks, so the need for coalition merging is lower. It is interesting to observe that while the maximum of M_4 is 360 with the coordinator checking for waiting coalitions every 10 seconds, in practice, M_4 turns out to be much lower. Coalition populations M_5 vary between 3 and 20 robots. They are observed to related to tasks' required resource levels as well as robots' resources. The populations are the lowest in cases where tasks require low level of resources while robots are endowed with high level of resources. In the other extreme case of L–H scenarios, M_5 increases approximately five times. This is expected since the tasks require too many resources. As a result, the population of each coalition needs to be greater. This leads to fewer tasks to be handled and the percentage of timed-out tasks M_2 increases considerably. It is observed that regardless of the variations among different scenarios, population sizes remain reasonable, considering the worst-case split complexity as discussed previously. Of course, for the H–L scenarios, this number is quite large, however it still computable within the limits. Of course, larger coalition populations will not be recommended, as worst-case split complexity will lead to increased time-outs and hence the deterioration of task completion performance. The percentage of required resources to handling coalition's resources for completed tasks M_6 varies between 79.1% and 86.9%. It is observed that as tasks require higher level resources, M_6 becomes higher which implies that the resource utilization becomes lower. With lower λ, the coalitions are able to handle their tasks without informing the coordinator. The computational complexity of the coordinator's decision making is studied by computing the average iteration numbers of the CFGs in Table 4. It is observed that the games converge after a small number of iterations, which implies that game durations are negligible, as computational analysis indicated.

We also investigate the effects of the weighting parameters. Four alternative sets $\mathcal{W}^1$, $\mathcal{W}^2$, $\mathcal{W}^3$ and $\mathcal{W}^4$ are considered with parameter values depending on the relative preference of resource satisfaction, resource utilization and site proximity. The set $\mathcal{W}^1$ weighs each equally with $w_1 = w_2 = w_3 = 1/3$. In the set $\mathcal{W}^2$, resource satisfaction is relatively more important with $w_1 = 2/3$, $w_2 = w_3 = 1/3$. Resource utilization has a higher weight in the set $\mathcal{W}^3$ with $w_2 = 2/3$ and $w_1 = w_3 = 1/3$. Finally, in the set $\mathcal{W}^4$, site proximity has a higher priority with $w_3 = 2/3$ and $w_1 = w_2 = 1/3$. In these simulations, λ, Δt_o, and Δt_d are set to $\lambda = 10$, $\Delta t_o = 1$ minute and $\Delta t_d = 2$ minutes respectively. Statistical results are as shown in Table 5. Time-outs are lowest when resource satisfaction has the highest priority, except for the H–L scenarios where both having resource utilization and site proximity at higher priority leads to slightly better performance. Changing the weighting

Downloaded from ijr.sagepub.com at University of New England on January 17, 2016

Table 4. Average number of iterations (sum of merge and split iterations). TR, task resource level; RR, robot resource level; L, low; H, high.

	λ	5				10				20			
	Δt_o	1		2		1		2		1		2	
	Δt_d	2	4	2	4	2	4	2	4	2	4	2	4
TR-RR	L-L	7	4	6	4	5	4	5	4	5	4	5	3
	L-H	8	7	7	5	7	6	5	4	5	5	5	4
	H-L	4	3	4	3	4	3	4	3	3	3	3	3
	H-H	9	8	11	9	8	5	7	5	6	4	6	4

Table 5. Effects of weighting parameters. TR, task resource level; RR, robot resource level; L, low; H, high.

		Weighting parameters			
TR-RR		$\mathcal{W}^1$	$\mathcal{W}^2$	$\mathcal{W}^3$	$\mathcal{W}^4$
L–L	$\bar{M}_1$	103.3	100.9	117.4	110.2
	$\bar{M}_2$	22.8	14.0	35.2	26.4
	$\bar{M}_3$	9.6	9.6	8.9	9.4
	$\bar{M}_4$	170.4	137.1	219.4	190.6
	$\bar{M}_5$	9.5	9.2	9.2	9.3
	$\bar{M}_6$	83.1	83.5	84.5	84.4
L–H	$\bar{M}_1$	208.8	205.0	219.9	212.8
	$\bar{M}_2$	7.3	0.4	15.3	9.2
	$\bar{M}_3$	17.8	16.9	17.9	17.7
	$\bar{M}_4$	130.5	66.7	187.2	144.5
	$\bar{M}_5$	3.5	3.6	3.5	3.5
	$\bar{M}_6$	81.0	80.0	82.4	81.4
H–L	$\bar{M}_1$	91.8	97.3	92.2	90.7
	$\bar{M}_2$	58.7	59.9	57.0	57.4
	$\bar{M}_3$	9.3	9.9	8.7	8.5
	$\bar{M}_4$	248.5	259.5	240.3	233.1
	$\bar{M}_5$	19.8	18.5	18.9	20.1
	$\bar{M}_6$	85.9	86.9	84.2	84.5
H–H	$\bar{M}_1$	130.3	131.0	135.6	132.2
	$\bar{M}_2$	3.0	3.3	5.9	3.5
	$\bar{M}_3$	9.7	9.6	10.0	9.9
	$\bar{M}_4$	64.9	69.1	84.1	70.1
	$\bar{M}_5$	6.9	6.9	6.8	6.9
	$\bar{M}_6$	86.8	87.2	87.5	87.0

parameters does not have much effect on M_3 and M_5 as both seem to be mainly dependent tasks' resource requirements and robots' resources. Resource utilization is observed to vary between 80% and 87.5%. As expected, using the set $\mathcal{W}^3$ results in better resource utilization with higher M_6 values while the coordinator has the highest decision-making load, except for the H–L scenario. Interestingly, in the H–L scenario, as the time-out percentage M_2 is considerably high around 57–59.9% regardless of the parameter set, the particular choice of the weight parameters does not seem to have much effect on the resulting performance. In summary, task performance measures can be programmed according to the relative weighting preferences of these three considerations.



Fig. 3. The experimental workspace.

6. Experiments

The proposed approach has also been implemented on a multirobot team of five Turtlebot-II robots[4] that are operating in a workspace of 5 m $\times$ 5 m as shown in Figure 3. The experiment scenario is designed to imitate a real scenario in which the robots are assigned to search potential areas of interest and to collect various data from these areas. In particular, the data collection tasks require 5 different resources including cameras (resource 1), laser (resource 2), arms (resource 3), chemical test unit (resource 4) and biological test unit (resource 5). Each robot has resources as shown in Table 6a while the type of encountered tasks require resources shown in Table 6b. Accordingly, the coalition populations capable of accomplishing the encountered tasks are expected to vary between 2 and 5 robots depending on the task requirements. The parameters of the coalition value function are set as $w_1 = 1/3$, $w_2 = 1/3$, and $w_3 = 1/3$ in order to give equal importance to resource satisfaction, resource utilization and site proximity. The coordinator starts a CFG every 5 seconds. The missions are repeated 10 times, all starting at random initial locations for each $\lambda = 3$ per minute, time-out duration $\Delta t_o = 20$ seconds and handling duration $\Delta t_d =$ 10 seconds. Robot 1 is assigned to be the task coordinator. the coordinator can be changed during the mission. It is observed that the number of completed tasks $\bar{M}_1$ is about 10 while the percentage of timed-out tasks $\bar{M}_2$ is 33.3%. Since the succoring takes much time and the time-out

Downloaded from ijr.sagepub.com at University of New England on January 17, 2016

Table 6. Experiment scenario.

(a) Robots' resources.

	Resources				
Robot ID	$r(1)$	$r(2)$	$r(3)$	$r(4)$	$r(5)$
1	1	1	2	2	2
2	1	1	1	2	2
3	1	2	1	1	1
4	1	1	1	2	1
5	1	2	2	1	2

(b) Tasks' resources.

Task	Resources				
Type	τ_1	τ_2	τ_3	τ_4	τ_5
1	1	2	3	4	4
2	2	1	1	3	2
3	4	3	2	4	5

duration Δt_o is considerably low, the encountered tasks time out, hence $\bar{M}_2$ value increases. Average number of coalitions $\bar{M}_3$ turns out to be 2.5. Each CFG converges on average after five iterations of merge and split operations. This supports our computational analysis.

7. Conclusion

This paper introduced a novel hybrid approach to dynamic multirobot tasks requiring a set of resources. If the resources possessed by individual robots are insufficient, the formation of robot coalitions endowed with sufficient resources is essential. Coalitions proceed with the tasks if they have sufficient resources after liberating redundant members while they report it to a task coordinator, in case their resources do not suffice. The task coordinator forms coalitions capable of these tasks based on CFGs in which it evaluates groups of robots while considering resource satisfaction, resource utilization and site proximity with weighting parameters that encode relative preferences. The resulting coalitions are $\mathcal{D}_{hp}$-stable, which implies that no group of robots has a viable alternative to staying within their assigned coalition. Thus, as new tasks are encountered, coalitions merge and split so that the resulting coalitions are capable of doing these tasks. As the computational complexity of each CFG is due to merge and split operations, the proposed approach can be easily utilized in multirobot task applications, as also demonstrated in our simulation and experimental results. Simulation results indicate that the proposed approach enables the assembly of coalitions capable of completing the dynamically encountered tasks. Furthermore, the performance in regards to different tasks measures can be programmed depending on the user's or robots' priority via setting the weighting factors in the cost functions appropriately. The applicability of the proposed approach is verified experimentally after designing and implementing custom ROS software on a team of heterogeneous mobile robots. For future work, we plan to consider scenarios where both tasks' required resources and robots' resources have additional constraints that need to be taken into account. We also plan to study whether the role of the task coordinator can be distributed to the coalitions while considering inexact solutions to $\mathcal{D}_{hp}$-stability.

Acknowledgments

We are grateful for the contributions of H. Samed Cildir and Kadir Türksoy to the experiments.

Funding

This work has been supported in part by TUBITAK (grant number EEEAG-111E285) and Bogazici University (projects BAP 5169 and BAP 7222).

Notes

1. This taxonomy considers three orthogonal dimensions, namely single-task (ST) versus multi-task robots (MT) depending on whether each robot is capable of executing single or multiple tasks at the same time; single-robot tasks (SR) versus multi-robot tasks (MR) depending on whether a task can be completed by a single robot or several robots (a coalition) and instantaneous assignment (IA) versus time-extended assignment (TE) depending of whether only current tasks or future tasks are considered.
2. Superadditivity implies that any two disjoint coalitions, when acting together, can get at least as much as they can when acting separately. This may arise with tasks that concern the entire population such as environmental monitoring that requires all of the resources of all of the robots.
3. This may be one of the robots with the additional task of being a coordinator. As its processing is relatively simple, in case of failure, another robot may easily assume this role.
4. The details of the software that has been designed and developed for this purpose are described in Appendix B.

Downloaded from ijr.sagepub.com at University of New England on January 17, 2016

References

Apt KR and Radzik T (2006) Stable partitions in coalitional games. Arxiv Preprint cs/0605132.

Apt KR and Witzel A (2009) A generic approach to coalition formation. *International Game Theory Review* 11(3): 347–367.

Aumann R and Dreze J (1974) Cooperative games with coalition structures. *International Journal of Game Theory* 3(4): 217–237.

Bayram H and Bozma HI (2014) Decentralized network topologies in multirobot systems. *Advanced Robotics* 28(14): 967–982.

Bererton C and Khosla P (2002) An analysis of cooperative repair capabilities in a team of robots. In: *IEEE international conference on robotics and automation*, vol. 1, pp. 476–482.

Bernardine Dias M and Stentz A (2002) Opportunistic optimization for market-based multirobot control. In: *IEEE/RSJ international conference on intelligent robots and systems*, vol. 3, pp. 2714–2720.

Bogomolnaia A and Jackson MO (2002) The stability of hedonic coalition structures. *Games and Economic Behavior* 38(2): 201–230.

Brunet L, Choi HL and How JP (2008) Consensus-based auction approaches for decentralized task assignment. In: *AIAA guidance, navigation, and control conference*

Cao Y, Fukunaga AS and Kahng A (1997) Cooperative mobile robotics: antecedents and directions. *Autonomous Robots* 4(1): 7–27.

Chaimowicz L, Campos M and Kumar V (2002) Dynamic role assignment for cooperative robots. In: *IEEE international conference on robotics and automation*, vol. 1, pp. 293–298.

Chapman AC, Rogers A, Jennings NR and Leslie DS (2011) A unifying framework for iterative approximate best-response algorithms for distributed constraint optimization problems. *Knowledge Engineering Review* 26: 411–444.

Chen J and Sun D (2011) Resource constrained multirobot task allocation based on leader-follower coalition methodology. *The International Journal of Robotics Research* 30(12): 1423–1434.

Dang VD and Jennings NR (2006) Coalition structure generation in task-based settings. In: *17th European conference on artificial intelligence*, pp. 210–214.

de Vries S and Vohra RV (2003) Combinatorial auctions: a survey. *INFORMS Journal on Computing* 15(3): 284–309.

Ducatelle F, Forster A, DiCaro GA and Gambardella LM (2009) Task allocation in robotic swarms: new methods and comparisons. Technical Report IDSIA-01-09, Dalle Molle Institute for Artificial Intelligence, Lugano, Switzerland.

Fousse L, Hanrot G, Lefèvre V, Pélissier P and Zimmermann P (2007) MPFR: A multiple-precision binary floating-point library with correct rounding. *ACM Transactions on Mathematical Software* 33(2): 13.

Fua CH and Ge SS (2005) COBOS: Cooperative backoff adaptive scheme for multirobot task allocation. *IEEE Transactions on Robotics* 21(6): 1168–1178.

Garey MR and Johnson DS (1979) *Computers and Intractability: A Guide to the Theory of NP-Completeness*. New York: Freeman.

Gerkey B and Mataric M (2002) Sold!: auction methods for multirobot coordination. *IEEE Transactions on Robotics and Automation* 18(5): 758–768.

Gerkey BP and Mataric MJ (2004) A formal analysis and taxonomy of task allocation in multi-robot systems. *The International Journal of Robotics Research* 23(9): 939–954.

Guerrero J and Oliver G (2012) Multi-robot coalition formation in real-time scenarios. *Robotics and Autonomous Systems* 60(10): 1295–1307.

Hajdukova J (2006) Coalition formation games: a survey. *International Game Theory Review* 08(04): 613–641.

Hoeing M, Dasgupta P, Petrov P and O'Hara S (2007) Auction-based multi-robot task allocation in comstar. In: *International joint conference on autonomous agents and multiagent systems*, pp. 280:1–280:8.

Karaoğuz H, Bayram H and Bozma HI (2013) Communication integrated control architecture in multirobot systems. In: *IEEE international conference on robotics and automation (ICRA 2013)*, Karlsruhe, Germany, 6–10 May.

Karagöz CS, Bozma HI and Koditschek DE (2014) Coordinated Navigation of Multiple Independent Disk-Shaped Robots. *IEEE Transactions on Robotics* 30(6): 1289–1304.

Koenig S, Tovey C, Lagoudakis M, Markakis V, Kempe D, Keskinocak P, Kleywegt A, Meyerson A and Jain S (2006) The power of sequential single-item auctions for agent coordination. In: *National conference on artificial intelligence*, vol. 2. AAAI, pp. 1625–1629.

Korsah GA, Stentz A and Dias MB (2013) A comprehensive taxonomy for multi-robot task allocation. *The International Journal of Robotics Research* 32(12): 1495–1512.

Lau HC and Zhang L (2003) Task allocation via multi-agent coalition formation: taxonomy, algorithms and complexity. In: *15th IEEE international conference on tools with artificial intelligence*, pp. 346–350.

Lerman K, Jones C, Galstyan A and Mataric MJ (2006) Analysis of dynamic task allocation in multi-robot systems. *The International Journal of Robotics Research* 25(3): 225–241.

Parker L (1998) ALLIANCE: an architecture for fault tolerant multirobot cooperation. *IEEE Transactions on Robotics and Automation* 14(2): 220–240.

Parker LE (2012) Decision making as optimization in multi-robot teams. In: Ramanujam R and Ramaswamy S (eds.) *Distributed Computing and Internet Technology* (*Lecture Notes in Computer Science*, vol. 7154). New York: Springer, pp. 35–49.

Saad W, Han Z, Basar T, Debbah M and Hjorungnes A (2011) Coalition formation games for collaborative spectrum sensing. *IEEE Transactions on Vehicular Technology* 60(1): 276–297.

Saad W, Han Z, Debbah M, Hjorungnes A and Basar T (2009) Coalitional game theory for communication networks. *IEEE Signal Processing Magazine* 26(5): 77–97.

Sandholm T, Larson K, Andersson M, Shehory O and Tohm F (1999) Coalition structure generation with worst case guarantees. *Artificial Intelligence* 111(12): 209–238.

Service T and Adams J (2011) Coalition formation for task allocation: theory and algorithms. *Autonomous Agents and Multi-Agent Systems* 22(2): 225–248.

Shehory O and Kraus S (1998) Methods for task allocation via agent coalition formation. *Artificial Intelligence* 101: 165–200.

Tang F and Parker L (2005) ASyMTRe: Automated synthesis of multi-robot task solutions through software reconfiguration. In: *IEEE international conference on robotics and automation*, pp. 1501–1508.

Vig L (2006) *Multi-robot Coalition Formation*. PhD Thesis, Vanderbilt University.

Vig L and Adams JA (2006) Multi-robot coalition formation. *IEEE Transactions on Robotics* 22(4): 637–649.

Wagner T, Phelps J and Guralnik V (2004) Centralized vs. decentralized coordination: Two application case studies. In: Wagner

Downloaded from ijr.sagepub.com at University of New England on January 17, 2016

TA (ed.) *An Application Science for Multi-Agent Systems*, volume 10. New York: Springer, pp. 41–75.

Yokoo M, Durfee E, Ishida T and Kuwabara K (1998) The distributed constraint satisfaction problem: formalization and algorithms. *IEEE Transactions on Knowledge and Data Engineering* 10(5): 673–685.

Zhang K, Collins EGJr and Shi D (2012) Centralized and distributed task allocation in multi-robot teams via a stochastic clustering auction. *ACM Transactions on Autonomous and Adaptive Systems* 7(2): 21:1–21:22.

Zhang Y and Parker L (2013) Considering inter-task resource constraints in task allocation. *Autonomous Agents and Multi-Agent Systems* 26(3): 389–419.

Appendix A: Index to Multimedia Extension

Archives of IJRR multimedia extensions published prior to 2014 can be found at http://www.ijrr.org, after 2014 all videos are available on the IJRR YouTube channel at http://www.youtube.com/user/ijrrmultimedia

Table of Multimedia Extension

Extension	Media type	Description
1	Video	5-robot experiment

Appendix B: Experimental software

We have designed and developed custom ROS software implementing the proposed approach. This software is used in each robot, regardless whether the robot is a task coordinator, coalition leader or simply a coalition member. As such, as the robot's role changes, it can accommodate its new responsibilities without requiring programming changes. The control architecture is based on the sense-communicate-act paradigm as presented in Karaoğuz et al. (2013). It consists of the following modules as also shown in Figure 4.

- *Communication manager*: This module handles all the required communication of each robot. While the messages coming from the message decoder module are sent to appropriate, incoming messages from other robots are also published to the message decoder.
- *Message decoder*: The messages coming from task handler, coalition leader and task coordinator modules are converted into data packages that will be sent to other robots by the communication manager. On the other hand, the incoming messages are published to the related modules.
- *Task observer*: This module is responsible for detecting the tasks while moving around the workspace. Once a task is detected, the task related information is sent to the task handler module.

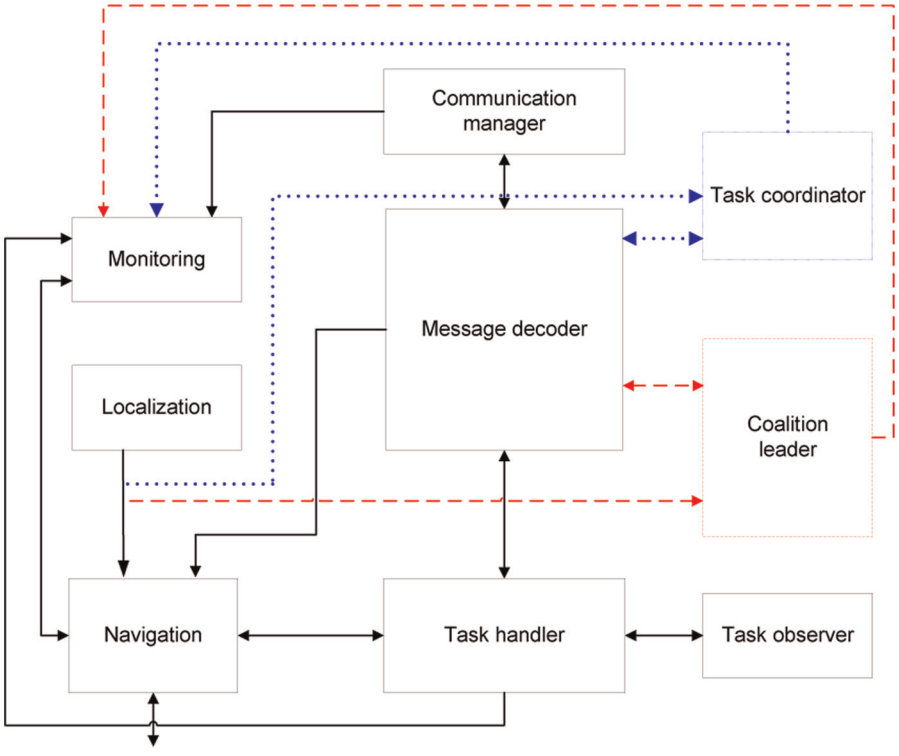


Fig. 4. The ROS software architecture.

Downloaded from ijr.sagepub.com at University of New England on January 17, 2016

- *Task handler*: This module informs the coalition leader of the newly encountered task. It also deals with the task handling issues.
- *Coalition leader*: The leader module coordinates the task handling by using the task automaton in Figure 1. This module is only activated when the robot is a coalition leader.
- *Task coordinator*: This module is responsible for forming the robot coalitions capable of performing encountered tasks. The coordinator maintains a list of reported tasks which are waiting (either in the coalition or in the coordinator). This module is only activated if the robot is the task coordinator.
- *Localization*: This module is responsible for sending the position information to the related modules (navigation, task coordinator, and coalition leader).
- *Navigation*: This module is responsible for collision-free navigation to target position. The controller is based on a decentralized version of feedback-based navigation as presented in Karagöz et al. (2014) and uses the MPFR library in order to manipulate big numbers (Fousse et al., 2007).
- *Monitoring*: It sends the task and robot related state information to the monitoring computer on which a monitoring software is running in order to make the analysis of the collective behavior easier.

Downloaded from ijr.sagepub.com at University of New England on January 17, 2016