The Thirty-Second International Florida
Artificial Intelligence Research Society Conference (FLAIRS-32)

Distributed Coalition Formation with Heterogeneous Agents for Task Allocation

Ayan Dutta,[1] **Emily Czarnecki,**[1] **Asai Asaithambi,**[1] **Vladimir Ufimtsev**[2]
[1]University of North Florida, USA, [2]East Central University, USA
{a.dutta, e.czarnecki, asai.asaithambi}@unf.edu, vufimtsev@ecok.edu

Abstract

In this paper, we study the problem of forming coalitions with heterogeneous agents for allocating them to tasks. Several agents work together to complete a given task. Due to the inherent complexity of real-world tasks and limited capabilities of a particular type of a physical agent such as a robot, it is imperative to form a team consisting of different types of robots to complete the tasks. Our work in this paper proposes a distributed bipartite graph partitioning approach for coalition formation with heterogeneous agents such as humans and/or robots for instantaneous allocation to tasks (ST-MR-IA). We also extend this approach to apply in the scenarios where the tasks might have dependencies among each other (ST-MR-TD). We have implemented the proposed algorithms within the Webots simulator. The proposed strategy allocates near-optimal (up to 98%) agent coalitions to tasks. Results also show that our proposed approach can easily handle as many as 100 agents and 10 tasks while spending an almost negligible amount of time.

Introduction

Real world tasks are usually complex in nature. A team of humans with different skill sets is formed to accomplish them. Although autonomous robots can be used in place of humans to complete computationally intensive or mechanically intrinsic components of the tasks, not all task components can be completed by these robots (yet!). For example, first responders helping injured humans with both mental support and medicine/water in a disaster management situation is still very much a human task. On the other hand, a group of robots jointly deployed with the first responders in the same situation is helpful to rapidly explore difficult-to-navigate regions and locate victims (Kohlbrecher et al. 2015). Alternatively, robots equipped with different sensors can form teams without humans to accomplish a given task.

Forming teams or coalitions in real-world situations including both humans and robots of potentially different types is an important and challenging problem. In this paper, we prove that the heterogeneous coalition formation problem is a NP-hard problem. Most of the previous research in coalition formation deals with homogeneous agents such as robots or software agents (Service and Adams 2011b),(Rahwan and Jennings 2008). The problem with most of the proposed approaches is that the solution becomes computationally intractable with a very small number of agents even as low as in the 20's (Rahwan and Jennings 2008).

This paper presents a *distributed* solution for *heterogeneous* coalition formation using a *bipartite* graph partitioning approach for both the cases where the tasks might or might not have dependencies among each other. In the literature, these problems are known as single task (ST) multi-robot (MR) instantaneous/dependent (IA/TD) task allocation (Zhang and Parker 2013). Unlike previous approaches, the proposed solution can handle as many as 100 agents while incurring an almost-negligible time-cost.

Copyright © 2019, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

Related Work

One of the earliest studies on coalition formation by agents for task allocation is due to Shehory and Kraus (Shehory and Kraus 1998), in which the authors have proposed a greedy algorithm that is guaranteed to find a solution within a factor of $(MAXC + 1)$ of the optimal solution, where $MAXC$ is the maximum size of any coalition formed. Note that optimal solutions are expensive in terms of consumed time and it might be impossible to use them for a large number of agents and on any regular robot's onboard computer (Rahwan and Jennings 2008). Following the taxonomy of coalition formation algorithms for task allocation proposed in (Gerkey and Matarić 2004), this paper solves a single-task agent and multi-agent task problem, i.e., we assume that each agent is assigned to one task at the same time and each task might need a team consisting of more than one agent to finish it. In (Service and Adams 2011a),(Zhang and Parker 2013), similar problems have been studied and polynomial-time algorithms have been proposed. Heterogeneous team formation has been studied by Liemhetcharat and Veloso (Liemhetcharat and Veloso 2012a),(Liemhetcharat and Veloso 2012b).

Model and Notations

Let $A = \{a_1, a_2, .., a_n\}$ denote a set of n agents (e.g., humans, robots), and let $H = \{h_1, h_2, .., h_k\}$ denote the set of k types of agents. For example, a human, a flying robot, and a ground robot may be three different types of agents

available. Each agent is characterized by the tuple $\langle P_i, h_{s_i} \rangle$ where P_i denotes the position, and h_{s_i} the type of agent a_i, with $1 \leq s_i \leq k$. Let $T = \{t_1, t_2, .., t_m\}$ denote a set of m tasks ($n > m$). Any task t_j is characterized by a tuple $\langle P_j, O_j \rangle$ where P_j and O_j respectively denote the task location and optimal distribution of heterogeneous agents needed to finish that task, i.e., $O_j = \{o_{j1}, o_{j2}, .., o_{jk}\}$. In layman's terms, O_j represents how many agents of each type should be assigned to task t_j to finish it optimally. O_j for each task t_j is fixed and this information is assumed to be available with the agents for the purpose of task allocation.

A coalition ($c \subseteq A$) is a group of agents working together for completing a given task. A partitioning of the agents into non-overlapping coalitions is called a coalition structure (CS), i.e., $CS = \{c_1, c_2, \cdots, c_m\}$ where c_i is assigned to task t_i. We define a partial coalition structure pCS^i to be a set of non-overlapping m coalitions with only agents of type h_i constituting the coalitions. Note that a pCS^i does not cover all the agents.

With teams of heterogeneous agents, the effectiveness of any agent coalition depends on the difference between the allocated distribution of agent types to task t_i, which we denote by D_i, t_i's required optimal distribution of agent types, which we denote by O_i. Note that $D_i = \{d_{i1}, d_{i2}, \cdots, d_{ik}\}$ represents the allocated distribution of the heterogeneous agents. We define a *value function* Val that assigns a virtual reward to a coalition and is defined as $Val(c_i) = \sum_{j=1}^{k} o_{ij}^2 - (o_{ij} - d_{ij})^2$. Val ensures that if the coalition c_i assigned to the task t_i has the optimal distribution of agents, i.e., if $D_i = O_i$, then that coalition earns the maximum possible value. On the other hand, if D_i is different from the associated O_i, then the value of the coalition is not the highest. This formulation checks the effectiveness of the allocated coalition to the task in terms of the distribution of agent types. We define the value of a coalition structure as the summation of values of all the coalitions in it, i.e., $Val(CS) = \sum_{\forall c_i \in CS} Val(c_i)$. Note that the maximum value of any coalition structure can be computed as follows: $MAX\_VAL = \sum_{1 \leq j \leq m} \sum_{1 \leq j \leq k} o_{ij}^2$.

Cost of a coalition depends on the distance travelled by all the member robots to reach the task location. The same cost function proposed in (Dutta, Ufimtsev, and Asaithambi 2019) has been used in this paper. Finally, the problem objective is to find a set of coalitions for all the tasks such that the generated coalition structure has the minimum cost, while its value is the maximum.

Algorithm for ST-MR-IA

In this section, we discuss how we subdivide the problem into k sub-problems. Let $G = ((V, U), E, w)$ be an undirected, weighted, bipartite graph. V is a set of vertices which corresponds to the agents in A, and U is a set of vertices corresponding to the tasks in T. E is the edge set which consists of all possible pairs with exactly one agent from V and one task from U (thus $|E| = |T| \times |U|$). The edge weight function $w : E \rightarrow \mathbb{R}$ is as defined in (Dutta, Ufimtsev, and Asaithambi 2019). In the complete graph model used in (Dutta, Ufimtsev, and Asaithambi 2019), $|E| = \binom{|R|+|T|}{2}$. In this paper, by using a bipartite graph, we are able to drastically reduce the number of edges to be processed which in turn significantly reduces the time complexity. For example, in a complete graph with 100 agents and 10 tasks (one of the test cases in (Dutta, Ufimtsev, and Asaithambi 2019)), $|E| = 5,995$ whereas the bipartite graph formulation in this paper reduces this number to $1,000$, a reduction by a factor of 5.995.

After the bipartite graph is created, one leader for each of the k types of agents is elected. The algorithm described in (Dutta, Dasgupta, and Nelson 2018) can be used to elect the leaders. Each of these leaders will process a subgraph $G'_i = ((V_i, U), E_i, w_i)$ of G to allocate agents of only its own type to tasks. $V_i = A'_i$ where A'_i represents all the agents of type i. The corresponding edges and edge weights constitute E_i and w_i.

Let TA_i denote the allocation of i-th type of agents to tasks. TA_i is represented as a 2D array, where the first column contains the agent IDs and the second column contains the task IDs. It indicates which agent is allocated to which task. As agents are forming coalitions, more than one agent can be allocated to the same task. Once agents of the i-th type are allocated to the tasks, i.e., once TA_i's are finalized using the algorithms proposed in (Dutta, Ufimtsev, and Asaithambi 2019), the leaders will broadcast and share these partial allocation information with the other leaders. Then the complete coalition structure will be formed and consequently the task allocation process will be complete. This information will be stored in $TA \leftarrow TA_1 \cup \cdots \cup TA_k$. In this way, we are able to break down the allocation problem into k sub-problems and solve it in a distributed manner. Additionally, noting that each TA_i actually corresponds to the pCS^i introduced earlier, we can say that the heterogeneous agent coalition formation problem is the union of k homogeneous agent coalition formation sub-problems. Also, the desired heterogeneous coalition structure CS is a member of the Cartesian product of the k partial coalition structures pCS^i.

Algorithm 1: Allocation of heterogeneous agents to tasks

Input: A: A set of agents; T: A set of tasks.
Output: TA: An allocation of agents to tasks;

```
Create the bipartite graph G.
Divide G into k subgraphs, each of which contains a
  unique type agents and all the tasks.
a^i_lead ← the leader of the i-th type of agents.
Each a^i_lead executes the following:
        TA_i ← allocateTaskPerType(G_i) [following
  (Dutta, Ufimtsev, and Asaithambi 2019)].
TA ← TA_1 ∪ ··· ∪ TA_k.
return TA.
```

Allocation in ST-MR-TD

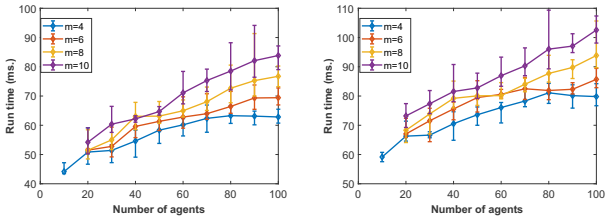
When considering real-world scenarios of task allocation, task dependencies often come into play (Zhang and Parker

2013),(Shehory and Kraus 1998). That is, some task t_i must be completed before the next task t_j can begin. To appropriately consider task dependencies it is important to note that tasks, based on their dependencies, may or may not be able to be assigned simultaneously. In this section, we discuss the extension of the above-mentioned algorithms developed for the ST-MR-IA task allocation problem to incorporate the task dependencies. In order to do this, we define a precedence order based on the given task dependencies. Tasks within the same order of precedence are assigned together and can only be assigned if their predecessors have been assigned (Shehory and Kraus 1998). More simply, we consider the tasks which can be assigned together based on their dependencies and group them together into bins. For example, with a set of tasks $T = \{t_1, t_2, t_3, t_4\}$, suppose that the task dependencies are as follows: t_2 and t_4 depend on t_1; and t_3 depends on t_2. Then, the resulting set of bins would be $B = \{\{t_1\}, \{t_2, t_4\}, \{t_3\}\}$. The tasks in the bins are assigned together in the order the bins are formed allowing assignments for the tasks only after their dependencies are met. Each bin in B is then given to the ST-MR-IA solution method (Algorithm 1) to make assignments. The tasks that are already assigned are removed from T. We can either *remove* the agents that are already assigned to a particular task or we can update their current locations to the assigned task locations in order to allocate them to remaining tasks. *Keeping* the agents in A for future allocations will be important if $\sum_m \sum_k o_{mk} < n$.

Simulations

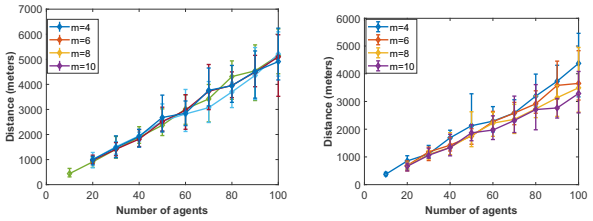
Settings We have implemented our distributed graph partitioning algorithm using the Java programming language within the Webots simulator. The tests are run on a laptop with an Intel i7-3615QM processor and 16GB RAM. The number of agents (n) has been varied between 4 and 100, and the number of tasks (m) has been varied between 4 and 10. We have made sure that in no test case the number of tasks exceeds 50% of the number of agents used. The distinct 2D locations of the agents and the tasks are randomly generated from a bounded square area with sides of length 100m. Task dependencies are built as a data structure similar to a tree. The root represents a task with no dependencies. This data structure can have multiple roots. Each child node is dependent on its parent nodes, i.e. the parent task must be assigned before a child task can be assigned. Type counts ($|H|$) of agents have been varied between 2 and 3. In each run, a random set of O_i's has been generated using an integer partitioning program. We have run each test case 10 times and the average result is presented here. The bars in the plots indicate the maximum and minimum y-axis values in that plot.

ST-MR-IA results: First we are interested to see how well our proposed algorithm scales with n and m. The results are shown in Fig. 1 for two different values of H. The result shows that the maximum run time of the proposed solution scales almost linearly with n. As there are more partitions to search for with higher values of m, it takes more time to get the final solution for $m = 10$ than $m = 2$. The run



(a) (b)

Figure 1: Run time of the ST-MR-IA solution (a) $|H| = 2$; (b) $|H| = 3$



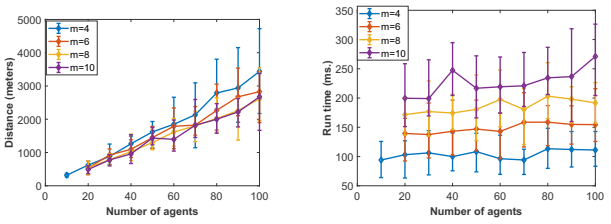
(a) (b)

Figure 2: Total distance traveled by the agents to reach the allocated tasks ($H = 2$): (a) IA, (b) TD (remove).

time is slightly higher with $H = 3$ as there are more types of agents and each type gets allocated in an order. However, the maximum run time is almost-negligible – 110 ms. with $n = 100, m = 2, |H| = 3$. Next we see how the cost (i.e., the total distance that the agents need to travel to reach the allocated tasks) of the solution varies with n. Fig. 2(a) shows that the total cost increases in a linear fashion with increasing n and it varies negligibly for different values of m.

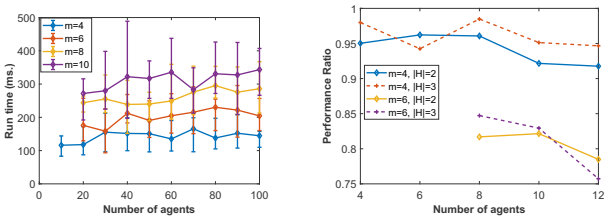
ST-MR-TD results: We now present the results of the solution for the ST-MR-TD problem. First we take the case where the agents are *removed* from A if they are allocated to one task once. Similar to IA allocation, we can see that the total traveled distances are increasing linearly (Fig. 2(b)). But as the agents allocated to one task are removed for further allocation and also with less number of tasks, the inter-task distances are higher, we notice that the agents travel more distances with $m = 4$ than $m = 10$. Although the same trend is noticed when we *keep* the agents in A, we see that the total traveled distance by the agents reduces compared to the case when we remove them (Fig. 3(a)). For example, with $n = 100, m = 10$, the total distance is around 4374 m. while removing the agents, but this value reduces to around 3438 m. while keeping the agents. If the inter-task distances are lower than the initial inter-agent distances, this phenomena can be observed. Similar result has been observed for $|H| = 3$.

Next we are interested in observing the run times of the proposed algorithm for the ST-MR-TD problem. The results are shown in Fig. 3(b) and 4(a). Two noticeable changes can be observed in these plots from the run times of ST-MR-IA solution (Fig. 1). The overall run time has increased by almost three time because of the fact that IA solution is run for each of the task bins separately. Secondly, run times do



(a) (b)

Figure 3: ST-MR-TD (keep), $|H| = 2$: (a) Total distance traveled by the agents to reach the allocated tasks; (b) Run time.



(a) (b)

Figure 4: (a) Run time of ST-MR-TD (keep), $|H| = 3$; (b) Performance ratio of the cost of ST-MR-IA to the optimal cost (higher is better).

not change significantly with increasing n. All the agents might not be assigned to the tasks at the same time. Because of the uniform distribution in the agent requirement model, the number of bins increases in proportion to the number of agents. Thus, the number of agents required for the tasks in each bin stay almost the same.

Comparison with the optimal: Finally, to check the solution quality against the optimal solution, we implement a brute-force method of generating all possible allocations with n agents and m tasks (Orlov 2002). We could not test against this approach beyond 12 agents and 4 tasks, the test became prohibitive on our machine. After getting the final optimal allocation from this implementation, we compare the distance cost of it against the cost of the ST-MR-IA solution. Ratio of the optimal cost to our solution's cost is named *Performance Ratio* (PR) and the result is shown in Fig. 4(b). As can be understood, if our solution's cost is close to the optimal, PR will be close to 1. In Fig. 4(b), we observe that with $m = 4$, PR is always greater than 0.9, i.e, our solution is at least 90% close to the optimal solution for both two and three types of agents while the highest ratio reaching up to 98% with $n = 8, m = 4, |H| = 3$. With $m = 6$, the value of PR is around 0.8 for $|H| = 2$ and it drops to 0.75 for 12 agents and $|H| = 3$.

Conclusion and Future Work

We have proposed a *distributed* solution for coalition formation with *heterogeneous* agents. Assuming there are k types of agents, our solution strategy subdivides the multi-agent heterogeneous coalition formation problem into k multi-agent homogeneous coalition formation sub-problems. This strategy has helped us obtain near-optimal solutions requiring an almost-negligible computational power. We have also gracefully handled the inter-task dependencies while extending the instantaneous task allocation model for the task dependent model. We also plan to better distribute the graph so that the leaders have similar load in terms of processed number of edges.

References

Dutta, A.; Dasgupta, P.; and Nelson, C. 2018. Distributed adaptive locomotion learning in modred modular self-reconfigurable robot. In *Distributed Autonomous Robotic Systems*. Springer. 345–357.

Dutta, A.; Ufimtsev, V.; and Asaithambi, A. 2019. Correlation clustering based coalition formation for multi-robot task allocation. In *ACM/SIGAPP Symposium on Applied Computing*. ACM.

Gerkey, B. P., and Matarić, M. J. 2004. A formal analysis and taxonomy of task allocation in multi-robot systems. *The International Journal of Robotics Research* 23(9):939–954.

Kohlbrecher, S.; Romay, A.; Stumpf, A.; Gupta, A.; Von Stryk, O.; Bacim, F.; Bowman, D. A.; Goins, A.; Balasubramanian, R.; and Conner, D. C. 2015. Human-robot teaming for rescue missions: Team vigir's approach to the 2013 darpa robotics challenge trials. *Journal of Field Robotics* 32(3):352–377.

Liemhetcharat, S., and Veloso, M. 2012a. Modeling and learning synergy for team formation with heterogeneous agents. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems-Volume 1*, 365–374. International Foundation for Autonomous Agents and Multiagent Systems.

Liemhetcharat, S., and Veloso, M. 2012b. Weighted synergy graphs for role assignment in ad hoc heterogeneous robot teams. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, 5247–5254. IEEE.

Orlov, M. 2002. Efficient generation of set partitions. *Engineering and Computer Sciences, University of Ulm, Tech. Rep*.

Rahwan, T., and Jennings, N. 2008. An improved dynamic programming algorithm for coalition structure generation. In *Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems*, 1417–1420.

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

Service, T. C., and Adams, J. A. 2011b. Constant factor approximation algorithms for coalition structure generation. *Autonomous Agents and Multi-Agent Systems* 23(1):1–17.

Shehory, O., and Kraus, S. 1998. Methods for task allocation via agent coalition formation. *Artificial Intelligence* 101(1):165–200.

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