Theory of Computation Final Project

Grace Diehl

June 11, 2020

1 Introduction

Effective cooperation enables individuals to complete tasks that they could not complete alone. Cooperation is particularly beneficial in robotics, due to the difficulty and heterogeneity of real-world tasks [10]. Heterogeneous multi-robot systems have been used in domains such as scientific data collection [7], military [1, 4] and first response [6, 5].

The problem of partitioning robots (or any other type of agent) into the teams that can best enable them to complete a set of tasks is known as coalition formation for task allocation. Coalition formation for task allocation is NP-hard, as demonstrated by reduction from the set packing problem [9]. Due to coalition formation's complexity and the highly dynamic nature of many robotic domains, numerous algorithms have been developed to find reasonably high-valued solutions quickly (e.g., [8, 11, 12]).

This project introduces an algorithm that solves the coalition formation decision problem by reducing it to a satisfiability modulo theory (SMT). A brief performance evaluation was conducted, in order to evaluate the effect of different problem parameters on the scalability of the coaliton formation solver. The results of this evaluation suggest that the number of agents and tasks are the factors that most significantly limit the algorithm's scalability.

2 Problem Description

This project focuses on the decision problem for coalition formation for task allocation. The input to this problem is an integer, $k \geq 0$, a set of agents, $A = \{A_1, \ldots, A_n\}$, and a set of tasks, $T = \{T_1, \ldots, T_m\}$. An agent, A_i, has a resource vector, $\{r_{i1}, \ldots, r_{ic}\}$, where r_{il} is an integer representing the quantity of resource l that agent i possesses. Each agent can only provide resources to a single task. A task, T_j, also has a resource vector $\{r_{j1}, \ldots, r_{jc}\}$, where r_{jl} is an integer representing the minimum quantity of resource l required to complete task j. Each task, T_j, task has an inherent *utility*, u_j, which represents the reward that the system expects to receive if it completes the task.

Define u'_j to be the reward that the system expects to receive from T_j. Formally,

$$u'_j = \begin{cases} u_j & \text{agents assigned to } T_j \text{ have sufficient resources} \\ 0 & \text{otherwise} \end{cases}$$

The decision problem asks whether there is an assignment of agents to tasks such that the following condition holds.

$$\sum_{j \in m} u'_j \geq k$$

This decision problem can be solved via a reduction to SMT.

3 Reduction to SMT

The reduction from the coalition formation decision problem to SMT is composed of three gadgets.

1. Each robot can be assigned to at most one task.

The first gadget enforces the rule that each robot can be assigned to at most one task. The reduction algorithm creates a new variable, A_iT_j, for each robot/task pair, to represent the assignment of robot A_i to task T_j. Then, for each pair, the reduction algorithm creates the following set of clauses:

$$\forall T_l \neq T_j \qquad A_iT_j \implies \overline{A_iT_l} \tag{1}$$

The first gadget is the disjunction of these clauses, which will be denoted as $oneTaskRule$.

The creation of the first gadget introduces $O(|A||T|)$ variables and produces $O(|A||T|^2)$ clauses.

2. A task's resource requirements are satisfied (and, therefore, the task can be completed) if and only if its assigned robots have sufficient resources.

The second gadget enforces the rule that a task's resource requirements are satisfied (and, therefore, the task can be completed) if and only if its assigned robots have sufficient resources. The reduction algorithm creates a new variable, T_jSat, to represent the assignment of agents with sufficient resources to task T_j.Then, for each task T_j, the reduction algorithm creates the following clause.

$$T_jSat \iff (\sum_{i=1}^{n} A_iT_j * r_{i1} \geq r_{j1}) \wedge \cdots \wedge (\sum_{i=1}^{n} A_iT_j * r_{ic} \geq r_{jc}) \tag{2}$$

The second gadget is the disjunction of these clauses, which will be denoted as $taskSatRule$.

The creation of the second gadget introduces $O(|T|)$ clauses of size $O(|A|c)$.

3. The sum of the utilities of tasks with satisfied resource requirements must be at least k.

The third gadget enforces the requirement that the sum of the utilities of tasks with satisfied resource requirements must be at least k. The reduction algorithm creates the following clause, which will be denoted as $kUtilitySum$.

$$\sum_{T_j \in T} T_jSat * u_i \geq k \tag{3}$$

The creation of the third gadget introduces a single clause of length $O(|T|)$.

Thus, a coalition formation decision problem can be represented by the SMT formula:

$$oneTaskRule \wedge taskSatRule \wedge kSumUtility \tag{4}$$

This formula can be produced in time $O(|A||T|^2 + |T||A|c)$ and is of size $O(|A||T|^2 + |T||A||c|)$.

4 Implementation

The reduction algorithm described above was implemented using the pySMT interface [3], and the resulting formulas were solved using MathSAT [2]. The code from this project (with directions for its use) are available at https://github.com/hmc-cs-gdiehl/ToCFinalProject.

5 Empirical Evaluation

The algorithm's ability to scale in terms of number of agents, number of tasks, and number of resources was empirically evaluated. Problems were generated using a method similar to the one described by Service and Adams [10]. Each agent was assigned a random number of each resource in the interval [0, 5], such that its total number of resources was nonzero. Each task was similarly assigned a random number of each resource in the range [0, 6], such that its total number of resources was nonzero. Each task was additionally assigned a random utility in the interval [1, 20], and k was randomly selected to be between 1 and the sum of the task utilities.

Twenty problems were generated for each fixed number of agents, tasks, and services. The running times below are the average running times for each set of twenty problems, as measured on a ASUS laptop with an Intel i7-8750H processor. Percent satisfied denotes the percentage of the twenty trials for which a satisfying assignment was found in at most five minutes. This time limit reflects the need to find solutions quickly in real-world applications of coalition formation, such as first response.

5.1 Agents

Figure 1 shows the effect of varying the number of agents. The number of agents was varied between 5 and 30 in increments of 5. The number of tasks and resource types were held constant at 10 and 3, respectively.

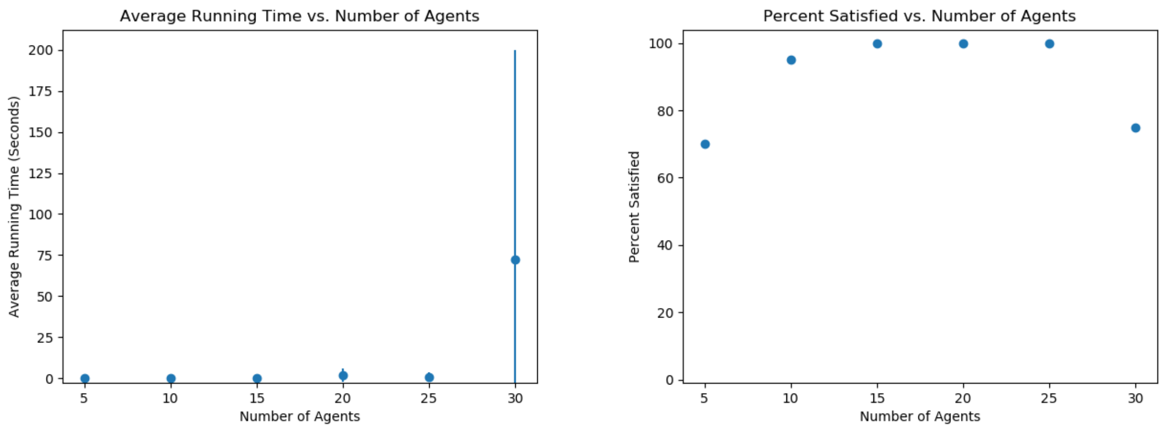

Figure 1: Average running time vs. number of agents (left) and percent satisfied vs. number of agents (right). 20 trials were run with 10 tasks and 3 resource types. The standard deviation of the running time is shown.

The coalition formation solver's running time (Figure 1 left) remains relatively constant with up to 25 agents, but becomes highly variable with 30 agents. Some of the 30 agent trials completed in less than one second, while 5 trials reached the five minute time limit before a satisfying assignment was found. This trend suggests that there are certain coalition formation problems that are more difficult for the solver than others, and that the difference between the time required to solve these problems and simpler ones increases superlinearly as the number of agents increases – in this case, when the number of agents increases to 30. Inspection of the 5 trials that reached the five minute limit did not show a clear correlation between k value or satisfiability and running time. Thus, a potential future research question is what characteristics make certain problems harder than others of the same size.

The percentage of problems for which the solver finds a satisfying assignment (Figure 1 right) is approximately 75% with 5 agents, 100% for 10-25 agents, and 80% for 30 agents. The initial increase in the percent of trials during which satisfying assignments found can be explained by the fact that adding more agents generally increases the number of tasks that can be completed. The subsequent decrease in satisfying assignments found is due to the five minute time limit. Out of the 20 trials conducted with 30 agents, the solver found none that were unsatisfiable and 5 that were interrupted by the time limit. It is unlikely that many of the 5 interrupted trials were unsatisfiable, based on the percentage of satisfiable inputs with 20 and 25 agents, so some other factor likely makes these inputs more challenging than others of the same size.

Overall, this experiment demonstrates that the average running time scales superlinearly to the number of agents, despite the problem's $O(|A||T|^2 + |T||A||c|)$ size. For the input range tested, the solver begins to perform poorly in terms of running time when there are 30 agents.

5.2 Tasks

Figure 2 shows the effect of varying the number of tasks. The number of tasks was varied between 5 and 15 in increments of 5. The number of agents was held constant at 15, and there were 3 resource types.

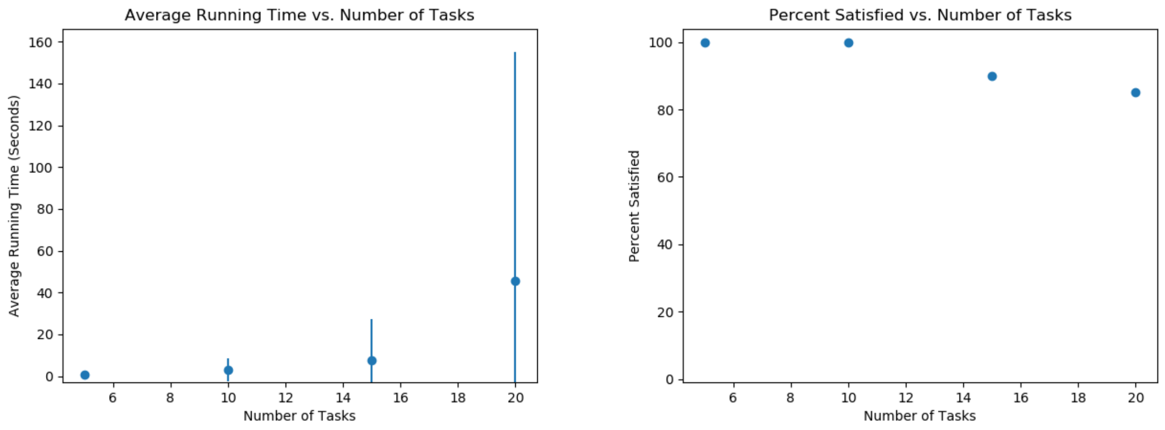

Figure 2: Average running time vs. number of tasks (left) and percent satisfied vs. number of tasks (right). 20 trials were run with 20 agents and 3 resource types. The standard deviation of the running time is shown.

The coalition formation solver's average running time is shown (Figure 2 left) to increase superlinearly as the number of task increases, with a sharp increase between 15 and 20 tasks. This trend is expected, based on the $O(|A||T|^2 + |T||A||c|)$ problem size. Additionally, the variability in running time increases with the number of tasks, as it was previously shown to do with the number of agents.

The percentage of problems for which the solver finds a satisfying assignment is expected to decrease as the ratio of tasks to agents increases. This trend is not directly present in Figure 2 (right), likely because the tast to agent ratio was insignificantly high. Instead, the main cause of the decreasing percent satisfied is the five minute time limit. Three trials were interrupted by the this time limit before a satisfying assignment was found.

These results suggest that number of tasks is a significant limiting factor in the solver's scalability. For the problem sizes tested, the solver scales well up to 15 tasks.

5.3 Resource Types

Figure 3 shows the effect of varying the number of resource types. The number of resource types was varied between 2 and 10 in increments of 2. The number of resource types was not increased beyond 10, because agents, particularly robots operating in multi-robot systems, tend to have relatively limited capabilities. The number of agents and tasks were held constant at 20 and 10, respectively.

The running time (shown in Figure 3 left) remains relatively constant across the different numbers of resource types. Additionally, the solver found satisfying assignments for all problems (Figure 3 right). These results suggest that the number of resource types is unlikely to be a limiting factor in scalability or satisfiability, compared to the number of agents or the number of tasks. This result is likely due to the number of resource type's limited contribution to the $O(|A||T|^2 + |T||A||c|)$ problem size and the fact that multi-agent systems often have significantly more agents and tasks than resource types.

5.4 k Value

Figure 4 shows the effect of varying the k value. For this experiment, the same set of twenty input problems was run with k values varying between 10 and 70 in increments of 10. The number of agents, tasks, and resource types were held constant at 20, 10, and 3, respectively.

The running time (shown in Figure 4) remains relatively constant as the k value increases. Meanwhile, the number of problems for which the coalition formation solver finds a satisfying assignment decreases. This

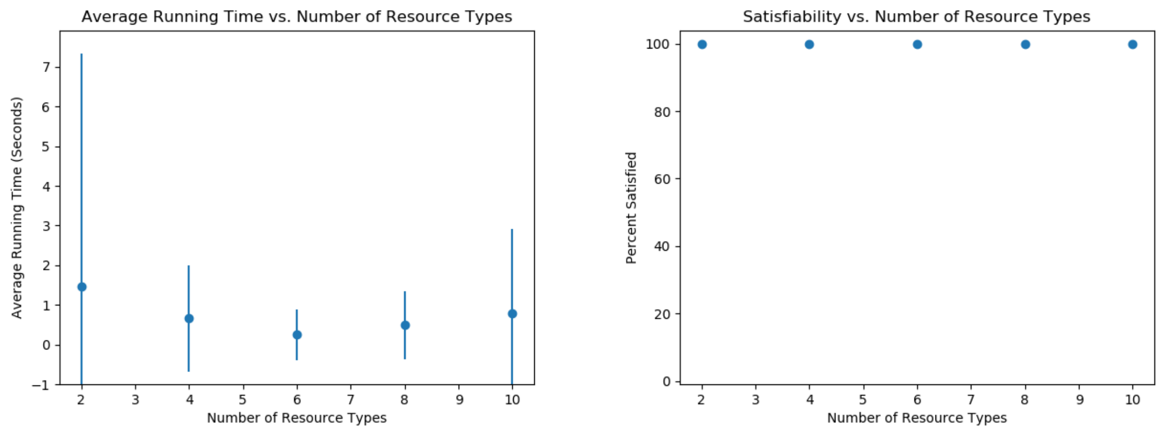

Figure 3: Average running time vs. number of resource types (left) and percent satisfied vs. number of resource types (right). 20 trials were run with 20 agents and 10 tasks. The standard deviation of the running time is shown.

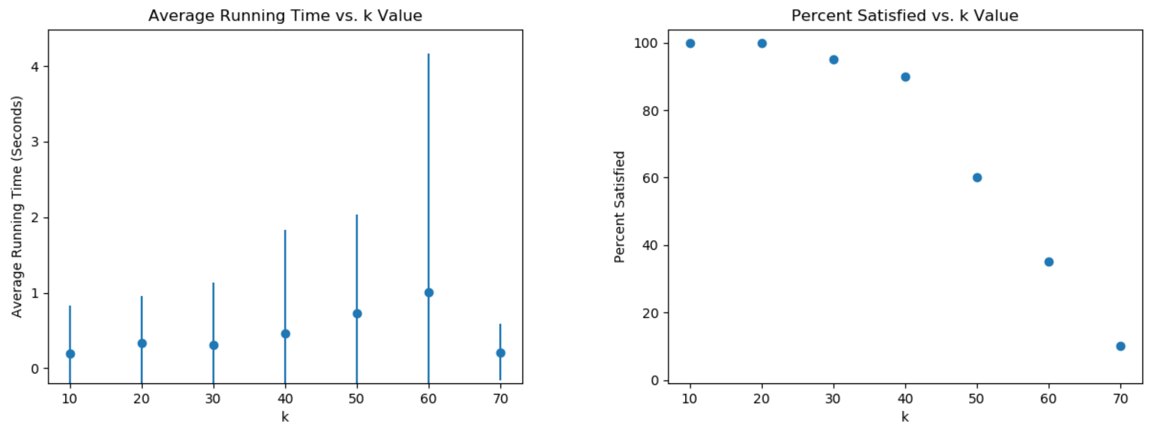

Figure 4: Average running time vs. k value (left) and percent satisfied vs. k value (right). 20 trials were run with 20 agents, 10 tasks, and 3 resource types. The standard deviation of the running time is shown.

decrease in satisfiability is expected, because a higher k value means that more tasks or higher utility tasks must be completed. This experiment does not demonstrate any connection between satisfiability or k value and runtime. The lack of correlation may be due to the fact that k does not impact the size of the input problem (for any meaningful value of k) or the fact that the problem sizes tested were such that the solver was able to determine the satisfiablility of each input without difficulty.

6 Conclusion

This project introduced a reduction from coalition formation to task allocation to SMT and evaluated a coalition formation solver that uses this reduction. The evaluation demonstrated that the solver scales well in terms of number of resource types and k values, but is unlikely to scale to the numbers of robots or tasks that are required for large-scale multi-robot systems.

References

[1] Defense Advanced Research Projects Agency. OFFensive Swarm-Enabled Tactics (OFFSET), 2019. URL `https://www.darpa.mil/work-with-us/offensive-swarm-enabled-tactics`.

[2] Alessandro Cimatti, Alberto Griggio, Bastiaan Schaafsma, and Roberto Sebastiani. The MathSAT5 SMT Solver. In Nir Piterman and Scott Smolka, editors, *Proceedings of TACAS*, volume 7795 of *LNCS*. Springer, 2013.

[3] Marco Gario and Andrea Micheli. Pysmt: a solver-agnostic library for fast prototyping of smt-based algorithms. In *SMT Workshop 2015*, 2015.

[4] M. Haque, M. Egerstedt, and A. Rahmani. Multilevel coalition formation strategy for suppression of enemy air defenses missions. *Journal of Aerospace Information Systems*, 10(6):287–296, 2013.

[5] Maryam Khani, Ali Ahmadi, and Hajar Hajary. Distributed task allocation in multi-agent environments using cellular learning automata. *Soft Computing*, 23(4):1199–1218, 2019.

[6] Somchaya Liemhetcharat and Manuela Veloso. Weighted synergy graphs for effective team formation with heterogeneous ad hoc agents. *Artificial Intelligence*, 208(1):41–65, 2014.

[7] NASA. Distributed spacecraft autonomy project conducts successful joint preliminary design and technical assessment periodic reviews. URL `https://ti.arc.nasa.gov/news/DSA-PDR-TAPR/`.

[8] Gyeongtaek Oh, Youdan Kim, Jaemyung Ahn, and Han Lim Choi. Market-Based Task Assignment for Cooperative Timing Missions in Dynamic Environments. *Journal of Intelligent and Robotic Systems: Theory and Applications*, 87(1):97–123, 2017.

[9] Tuomas Sandholm, Kate Larson, Martin Andersson, Onn Shehory, and Fernando Tohmé. Coalition structure generation with worst case guarantees. *Artificial Intelligence*, 111(1):209–238, 1999.

[10] Travis C. Service and Julie A. Adams. Coalition formation for task allocation: Theory and algorithms. *Autonomous Agents and Multi-Agent Systems*, 22(2):225–248, 3 2011.

[11] Lovekesh Vig and Julie A. Adams. Multi-robot coalition formation. *IEEE Transactions on Robotics*, 22(4):637–649, 8 2006.

[12] Chiawei Yeh and Toshiharu Sugawara. Solving coalition structure generation problem with double-layered ant colony optimization. In *Proceedings of the 5th IIAI International Congress on Advanced Applied Informatics,*, pages 65–70, 2016.