2019

Hedonic Coalition Formation for Task Allocation with Heterogeneous Robots

Emily Czarnecki
University of North Florida, n01185574@unf.edu

HEDONIC COALITION FORMATION FOR TASK ALLOCATION WITH HETEROGENEOUS ROBOTS

by

Emily Czarnecki

A thesis submitted to the

School of Computing

in partial fulfillment of the requirements for the degree of

Master of Science in Computer and Information Sciences

UNIVERSITY OF NORTH FLORIDA

SCHOOL OF COMPUTING

November 2019

The thesis "Hedonic Coaliton Formation for Task Allocation with Heterogeneous Robots" submitted by Emily Czarnecki in partial fulfillment of the requirements for the degree of Master of Science in Computer and Information Sciences has been

Approved by the thesis committee: Date: 11/8/19

Redacted

Ayan Dutta, Ph.D.

Thesis Advisor and Committee Chairperson

Redacted

Asai Asaithambi, Ph.D.

Redacted

Xudong Liu, Ph.D.

ACKNOWLEDGEMENTS

First, I would like to thank the University of Florida for supporting the research presented in this thesis with the Professional Scholarship Development grant.

I would also like to offer my sincerest thanks and gratitude to my thesis advisor, Dr. Ayan Dutta for his continued support and mentoring throughout the development of this thesis and for the duration of my time at the University of North Florida.

Additionally, I would like to thank the members of my thesis committee for their continued support: Dr. Asai Asaithambi and Dr. Xudong Liu.

Lastly, I would like to thank my husband for his steady commitment and devotion to my life and career goals, and specifically providing much needed advice and support during the pursuit of my masters degree.

CONTENTS

Figures

Tables

ABSTRACT

Tasks in the real world are complex in nature and often require multiple robots to collaborate in order to be accomplished. However, multiple robots with the same set of sensors working together might not be the optimal solution. In many cases a task might require different sensory inputs and outputs. However, allocating a large variety of sensors on each robot is not a cost-effective solution. As such, robots with different attributes must be considered. In this thesis we study the coalition formation problem for task allocation with multiple heterogeneous (equipped with a different set of sensors) robots. The proposed solution is implemented utilizing a Hedonic Coalition Formation strategy, rooted in game theory, coupled with bipartite graph matching. Our proposed algorithm aims to minimize the total cost of the formed coalitions and to maximize the matching between the required and the allocated types of robots to the tasks. Simulation results show that it produces near-optimal solutions (up to 94%) in a negligible amount of time (0.19 ms. with 100 robots and 10 tasks).

CHAPTER 1

INTRODUCTION

As humans, we often collaborate to complete a given task where participants bring potentially different skill sets to the table. Similar to the humans, in order to complete a task in an autonomous fashion, multiple robots with different abilities need to collaborate. The abilities of the robots depend on their on-board sensors (e.g., laser rangefinder for map building) and their available actuators (e.g., manipulators for object manipulation). Combining the sensory inputs and corresponding actuation, one robot can contribute towards completing a given task. In this paper, we study how a group of heterogeneous robots, i.e., having different sets of sensors and/or actuators, can form teams (or, *coalitions*) to complete a given set of tasks such that the *cost* of forming the coalitions is minimized and the *value* of the formed coalitions is maximized. In literature, this problem is known as **S**ingle **T**ask, **M**ulti-**R**obot **I**nstantaneous task allocation problem (ST-MR-IA) [SA11, ZP13] – a robot can be allocated to a single task at any given time, however, multiple robots can be allocated to a particular task. Finding the optimal solution for this problem is shown to be an NP-Hard problem [SA11, SK95] although many approximation solutions with provable worst-case performance bounds exist [DA19, SA11, SK95].

Multi-robot coalition formation for task allocation has many potential real-world applications ranging from warehouse management to environmental monitoring. The objective is to partition the set of robots into non-overlapping subsets (coalitions), each of which will be allocated to a unique task and the member robots will then cooperate with each other to accomplish the allocated task. Within this thesis, we deal with the

partitioning problem: Given a set of n robots and m tasks, how to form m coalitions such that some given criterion (e.g., utility of the coalitions) is optimized. The required numbers and types of robots to complete a given task is assumed to be known *a priori*.

To solve this stated problem, we use a well-established game theoretic concept, called hedonic coalition game [BJ02]. The word 'hedonic' is originated from the greek word ' hēdonikós' that means 'considered in terms of pleasant'. In a pure hedonic setting, each robot will consider joining a particular coalition if the experience of joining the coalition is 'pleasant', e.g., the robot's sensors are maximally utilized and the cost of moving to the coalition is also low. To form a set of m non-overlapping hedonic coalitions, we have used a weighted bipartite graph matching technique along with an intelligent switch rule that make the formed coalitions provably Nash-stable. To achieve a Nash-Stable environment, each robot must be satisfied in their given coalition. That is, no robot seeks to relocate to another coalition in an effort to improve its utility. To the best of our knowledge, this paper is the first to solve the coalition formation problem for task allocation with a group of heterogeneous robots using a hedonic coalition game formulation. Simulation results validate the theoretically proved stable nature of our solutions. The results show that our proposed algorithm can consistently yield near-optimal solutions (up to 94%).

Furthermore, we have compared our proposed approach with four previous approaches developed to provide an approximation solution to the heterogeneous coalition formation problem. Our approach is compared to two natural greedy heuristics: *MaxUtility* and *AverageUtility*. *MaxUtility* was first presented in [SA11] and further explored along with *AverageUtility* in [ZP13]. To improve upon these two solutions [ZP13] proposed two new solutions to the ST-MR-IA problem: *ResourceCentric* and *ResourceCentricApprox*. We implemented similar solutions to the above mention approaches, altered

slightly to adapt to the nature of the problem and models presented in this thesis. The comparison of the results revealed similar solutions in terms of the ratio to the optimal, however, our solution proved to be significantly faster when compared to all four algorithms. Of the four *MaxUtility* had the fastest run times, but our solution was up to 41 times faster for the given experiment settings. The slowest was *ResourceCentric*. With a setting of $m = 4$ and $n = 12$, our solution was approximately 44,500 times faster.

Lastly, we implemented our proposed approach on a real TurtleBot 3 robot having a Raspberry Pi 3 computing board that is intended to reflect a more realistic environment in which the proposed solution would run on. It could find a solution for 100 robots and 10 task in $.322$ seconds.

CHAPTER 2

LITERATURE REVIEW

To service complex tasks, autonomous robots can work together where each robot can offer a particular set of skills best suited to completing different sub-parts of a task. To best service a task, the robots must form a team in which their collective skills are well-suited to the needs of the task. Coalition formation has received significant attention and a number of studies have been conducted by economists and computer scientists alike. Earlier studies primarily dealt with a set of homogeneous agents, while more recently, studies have expanded to address environments with heterogeneous agents.

2.1 Task Allocation

One of the earliest studies on coalition formation of agents for task allocation is presented in [SK98]. Their work yielded algorithms with polynomial-complexity of a distributive greedy nature providing near-optimal solutions to the homogeneous task allocation program. Their work also included consideration for task with precedence order. The authors' algorithm guaranteed to find a solution within a factor of $(k + 1)$ of the optimal solution, where k is the maximum size of any coalition formed and returned results in less than a minute in a system of up to 60 agents.

Other early studies, providing solutions with homogeneous agents, can be seen in [RJ08]. The authors provide an optimal solution to the coalition generation problem utilizing dynamic programming. The primary motivation of this work is provide a solution that reduces the memory used in previous approaches. The resulting algorithm has an exponential time complexity of $O(3^n)$. Additionally in [RRDJ07], the authors employ a

search-based technique to find a near-optimal solution. The solution generates valid and unique coalition structures and implements a technique to reduce the search space in which they cycle through to find a near-optimal coalition structure. Their results found a solution up to 99% of the optimal for up to 20 agents. More recent studies include the research performed in [DUA19] and [DA19]. In [DUA19] a correlation clustering technique is utilized to provide a near-optimal solution. The solution applies linear programming to correlate robots to clusters of high similarity using cost and value functions. Experiments find near-optimal solutions for up to 100 robots. [DA19] proposes a variant of a classical weighted bipartite matching technique for the allocation of homogeneous robots. The proposed approach can provide similar near-optimality guarantee as [SK98, SA11] while providing a linear time-complexity as opposed to state-of-the-art polynomial solutions. This model is the primary motivation for using the weighted bipartite matching-based hedonic coalition formation solution proposed in this paper albeit for heterogeneous robots.

In real world situations, the complexity of tasks may require more than one type of robot, or robots that offer different capabilities. With this, consideration has been given to the heterogeneous task allocation problem. The work presented in [SA11] provide two solutions for the coalition formation problem. One for a homogeneous environment and the other for a heterogeneous environment. For the former, a dynamic programming based algorithm is introduced which can find an optimal solution in polynomial time. For the latter, the authors present an adjusted solution of the algorithm proposed in [SK98]. Their solution offers a polynomial solution with a complexity of $(O(n^{\frac{3}{2}}m))$, which improves upon Shehory and Kraus's solution with a complexity of $O(n^k m)$; exponential on the size of the largest coalition. However, both [SA11, SK98] report similar sub-optimality guarantees. In [ZP13], the authors examine two natural greedy heuristics for the coalition formation problem within a heterogeneous environment, and then

present an improved heuristic in which inter-task resource constraints are taken into account as well as expected loss of utility. An extension of the new heuristic is also provided to incorporate task dependencies. The studies performed in [LV12a, LV12b] propose a synergy graph to model the compatibility of agents. In [LV12a] a coalition formation algorithm with heterogeneous agents is proposed where agents learn about the capabilities of other agents over time and form better coalitions. Here, a synergy graph is used to model how compatible any two agents are to form a coalition. The solution offered takes a learning approach where the synergy graph developed is built using training observations. A similar synergy-graph model is used in [LV12b] to study role assignments of ad-hoc agents in a coalition. Here, the authors study an environment in which capabilities of the robots and their performance on a team is initially unknown. The research presented in [DCAU19] proposes a distributed bipartite graph partitioning approach with region growing to provide near-optimal solutions for the ST-MR-IA task allocation problem. First, the authors divide the bipartite graph into k sub-problems correlating to the k agent types. The sub-graphs are processed to allocate agents of the same type to tasks. The graph partitioning approach is employed to keep edges of similar end points–an agent task pairing–with edges weights to signify the suitability of an agent/task pairing. Region growing is then performed to either add to a coalition that requires more agents or remove unnecessary agents from a coalition. Furthermore, the work provides an expansion to solve for the ST-MR-TD task problem and considers inter-task dependencies. In [LCS14], the authors consider the heterogeneous task allocation problem for grouped tasks and present an auction-based solution to maximize payoff or minimize cost. The study done in [SPP18] considers the capacity-constrained vehicle routing problem as the base model for the ST-MR-IA and provides a scalable heuristic handling up to 400 robots and 2000 tasks in simulations.

Additional complexities to the heterogeneous task allocation problem such as uncertain

costs have also been taken into account. Uncertainty is captured in the edge costs of transport graphs in [Pro19]. The authors use redundant robots to combat the uncertainty of travel times with the goal of reducing the waiting times at the desired goal location. They present a polynomial time algorithm using distributive aggregate functions to assign redundant robots to paths to minimize average waiting time. The research in [NS17] considers uncertain costs and risk levels associated with unknown settings such as surface topology. The authors consider two approaches. One solution allows the input of risk tolerance preference and this risk affects the optimal assignment. In the second, the risk position does not affect the optimal assignment.

2.2 Hedonic Games

The concept of hedonic Games, originally presented by [DG80] found its first applications in economics. Specifically the idea that individuals form teams to accomplish activities, and the motivation for an individual to join a team is based on preferences. Since then a number of studies have been conducted to further define and understand hedonic games and its applications. The work in [BJ02] is a primary example of one of the earlier studies researching the stability of coalition structures in hedonic games. In this work, the authors review the settings for hedonic and non-hedonic situations and the existence of stability in hedonic settings. They focus primarily on core stability and individual stability. Further studies have continued to research stability and complexity which can be found in [GS10] and [AB12]. In the former, the authors research computational complexity when achieving stable results. They consider hedonic games with additively-separable utilities and different stable outcomes such as Nash and individual stability. The latter study further explores the existence of stability in hedonic coalition formation. In this work, we observe the consideration of a player's preference of one coalition over another. The work offers solutions considering preference restrictions and guarantee existence of stable outcomes.

A sub-set of hedonic games known as *Fractional Hedonic Games* has also amassed a decent amount of attention. The studies presented in [ABH14] focus on the applications and stability of fractional hedonic games. In a fractional hedonic game, the utility is an average value a player credits the others in the coalition. Here the authors present algorithms to compute a core stable outcome for a fractional hedonic game. The work presented in [BFF$^+$14, BFF$^+$18] also considers fractional hedonic games and provides various graph topologies in order to model these games and their complexities when aiming to achieve a Nash stable outcome.

Game theory has previously been used for robot planning problems [LH93]. However, specifically proposing hedonic games for multi-robot systems appears to have received less attention. Recent works, done in the last decade by Saad et al. presented in [SHB$^+$09, SHB$^+$10, SHB$^+$11] focuses on using hedonic games to produce self-forming coalitions in different multi-agent systems. These systems include unmanned air vehicles, wireless communication agents, and secondary base station corporation in cognitive radio networks. The primary approach of the thesis was motivated by the work presented in [SHB$^+$09, SHB$^+$10, SHB$^+$11]. However, the problem addressed in the thesis will work with heterogeneous robots. The solution also aims to reduce the number of preference relations built in [SHB$^+$09, SHB$^+$10, SHB$^+$11].

CHAPTER 3

PROBLEM STATEMENT

Over the past few decades, increasing attention has been given to multi-robot systems. It is becoming increasingly common to see cooperative multi-robot systems being utilized in real-world day-to-day tasks[GM04]. Within cooperative multi-robots systems lies the problem of multi-robot task allocation. The core focus of this thesis has studied the Single Task, Multi-Robot, Instantaneous Allocation problem (ST-MR-IA) with heterogeneous robots. The problem studies coalition formation of multiple robots assigned to service a specific task. Each task requires a set of sensors for it to complete successfully. A robot's set of sensors, when combined with sensors from other robots, works cooperatively together to complete a task. The set of sensors a robot has is considered the value it can offer a task. The value of a coalition assigned to a task considers the collective sensors of the robots within a coalition. When forming coalitions, cost also plays an important role. The cost it takes for a robot to service a task is represented by the distance traveled by the robot to get to the task. Traveling to a task not only uses battery power but also contributes to the wear and tear of the robot. This impacts the overall lifetime use of the robot. The value and cost are combined to form a utility function. The utility function indicates how well a robot is suited to a task when considering other robots in the coalition. The utility is then utilized by the Hedonic Coalition Formation Game to create self-forming coalitions.

3.1 Model and Notations

Let $R = \{r_1, r_2, \cdots, r_n\}$ denote a set of n robots. Each robot r_i has the following attributes: position P_i and sensor/actuator set S_{r_i} where $S_{r_i} = \{s_1, s_2, \ldots\ s_k\}$ denotes a set of sensors/actuators for the robot r_i. From here on, we will use the term sensors to indicate the on-board sensors and/or actuators w.l.o.g. We assume that the robot's sensor set size $|S_{r_i}| \geq 1$ and each robot cannot have multiple same sensor. It is also assumed that each robot is localized in the environment using an on-board GPS or an overhead camera. A set of m tasks are available in the environment denoted by $T = \{t_1, t_2, \ldots\ t_m\}$. A task t_j has the following attributes: position P_j and sensor requirements $S_{t_j} = \{s_1, s_2, \ldots\ s_l\}$ – a set of sensors that the task t_j requires to be optimally completed. Each task's sensor set will adhere to $|S_{t_j}| \geq 1$ as well, and a task can require multiple sensors of the same type to be completed. Note that $\bigcup_m S_{t_j} \subseteq S$ and $\bigcup_n S_{r_i} \subseteq S$ where S denotes the set of all possible sensors.

A coalition $c \subseteq R$ is a group of robots assigned to complete a task. A coalition structure *CS* can be viewed as a set of disjoint coalitions, which cover all the robots. Let $CS = \{c_1, c_2, \ldots\ c_m\}$ denote a set of coalitions where each coalition is matched with a single task and as such $|CS| = |T|$.

3.1.1 Value Function

This function determines the non-negative value a robot offers a task, i.e. how effective that robot is in contributing to the needs of any particular task. In our scenario, the value of a robot allocated to a task is defined by how many sensors the robot r_i can contribute to the task t_j's requirements. On the other hand, an unused sensor is a wasted resource and therefore negatively affects the overall value a robot offers a task. The better value

a robot can offer a task, the more likely the robot will choose that task to be assigned to. The value function is a weighted formula of sensor matching and is defined as the following:

$$val(r_i, t_j) = \begin{cases} bI - pO, & \text{if } bI - pO > 0 \\ 0, & \text{otherwise} \end{cases} \tag{3.1}$$

Where b is the benefit ascribed to I, which is the size of the common subset of sensors belonging to r_i and the task t_j's requirement set, i.e., $I = |S_{r_i} \cap S_{t_j}|$. p is the penalty received for each sensor not utilized, and O is a subset of sensors belonging to r_i not needed by the task, i.e., $O = S_{r_i} \setminus (S_{r_i} \cap S_{t_j})$. If the result is negative, the value is set to 0 indicating this robot offers no value to the task being considered. It is important for the value to be non-negative in order to achieve a Nash-stable outcome [BFF+18]. Both b and p are constants, values of which can be changed depending on how much desired benefit or penalty can be ascribed to the matching. For the purpose of this paper we chose a baseline benefit and penalty of $b = 1$ and $p = 0.1$ for most simulations. However, we are interested in how benefit and penalty work together and perform experiments altering both b and p. The benefit and penalty weights work in delicate balance with each other and can offer flexibility in the importance of matched and unused sensors.

The penalty p should be chosen to avoid a negative $value(\cdot, \cdot)$. A penalty may only be added if at least one sensor match is made, otherwise the value defaults to zero as defined in Eq. 3.1. Therefore, the maximum value O can have is the $\max(|S_{r_i}|) - 1, \exists r_i \in R$, which we denote by $max(O)$. As the robots do not have duplicate sensors (e.g., two LIDARs), $max(|S_{r_i}|)$ is equivalent to the number of sensor types. For example, with

maximum of three sensor types, $max(O) = 3-1$. To ensure $pO \leq bI$, the penalty value must be: $p \leq \frac{bI}{max(O)}$ where $I \geq 1$. On the other hand, with $b = 1$ and $max(O) = 2$, the upper bound on p is 0.5. It is important to note that While there is a maximum bound on the penalty p, the value used for p is not required to be the maximum in all the cases.

The used value for p depends on the specifics of the application. The penalty is used to aid in reducing wasted resources. If wasted resources cannot be tolerated, the penalty should be set to its maximum. Depending on the number of sensor types, setting the penalty to the maximum value may affect how close to the optimal the solution may reach. Lowering the penalty will allow for more choice among the robots. In our problem where we consider three types of sensors, the maximum penalty of 0.5 gives it a significant weight. This greatly restricts the robots from potentially relocating to a better coalition. For most of the simulations the base penalty setting of $p = 0.1$ to allow freer movement of robots to tasks.

3.1.2 Cost Function

For a robot to service an assigned task, the robot must travel to the task location to complete it. When a robot considers coalitions to join, it must consider the cost to travel to its chosen task. Travel not only impacts battery life but also overall wear and tear of the robot. Therefore, it is an important factor to consider in the formulation. For our scenario the cost is determined by the distance traveled. With self-forming coalitions, the cost function will aid the robots in deciding if they want to stay in their current coalitions, or if other coalitions are preferred based on how far they have to travel to reach the tasks. The cost function is motivated from [DUA19] and defined as:

$$cost(r_i, t_j) = 1 - \left(\frac{d(r_i, t_j)}{\sqrt{length^2 + width^2 + 1}} \right) \tag{3.2}$$

Where d denotes the euclidean distance between a robot and a task and $length$ and $width$ are the dimensions of the environment. Similar to the value function, we subtract this from 1 to avoid negative cost amount to be used in the utility function described below.

3.1.3 Utility Function

The utility function indicates overall how well a robot fits within its coalition in terms of value and cost. When a robot considers a coalition it will use the utility metric to determine how well it is suited to service the task. The higher the utility the better suited the robot is to the task be considered. It is defined as:

$$u(r_i, t_j) = val(r_i, t_j) + cost(r_i, t_j) \tag{3.3}$$

To follow the hedonic coalition game model, each robot's utility must solely depend on the members of the coalition [BJ02]. To adhere to this property a robot r_i's value can *only* be determined based on the other robots in the coalition under consideration as r_i must take into account the other robots' sensors contributed to the coalition as well as the task as it defines the sensors needed in that coalition.

3.1.4 Problem Objective

The objective is to have self-forming set of coalitions where the final coalition structure maximizes the utility. The utility of the coalition structure is the sum of utilities of all the robots allocated to coalitions and is defined as follows:

$$u(CS) = \sum_{j=1}^{m} \sum_{i=1}^{|c_j|} u(r_i, t_j)$$

The objective of the coalition formation problem for multi-robot task allocation is to find a coalition structure CS^* that offers the maximum utility and it can be formally represented as follows:

$$CS^* = arg \max_{CS \in \zeta} u(CS)$$

where ζ denotes the set of all possible coalition structures. After an initial allocation, each robot will consider other coalitions in the CS and choose a task, if necessary, in order to increase its utility. This will result in a final Nash-stable coalition structure.

CHAPTER 4

HEDONIC COALITION FORMATION GAME

Hedonic games were first established as a way to model economic situations in which we can observe the natural instinct of individuals to partition themselves into groups they identify as most beneficial given a set of properties [DG80]. The study of hedonic games has evolved beyond economics and has found usefulness in many topics of computer science, specifically coalition formation. In these situations, the agents (robots in our case) have a preference as to which coalition they wish to belong to [BFF+18]. Some examples of situations are social groups, political parties, organizations etc. An agent's motivation to belong to a coalition can be quantified by its utility. The utility is only dependent on the other members within the coalition [BJ02]. One such example is a political party. Here, a member's utility depends on the party's values and identities of other members [BFF+18]. Within a hedonic game, the goal for an agent is to improve its utility by choosing a coalition that maximizes said utility. This approach presents itself to be greedy in nature where an agent chooses a coalition based information known at specific stage within the algorithm. To establish our solution as hedonic, it must follow the properties for hedonic coalition formation. We formalize these by defining the following key properties [BJ02, SHB+11]:

1. Utility of a robot depends solely on the other robots in the coalition the robot currently belongs to.

2. The coalitions are formed based on the preferences the robots have over the set of all possible coalitions.

The concept of Nash stability states that a coalition structure CS is stable if there exists no robot that can improve its utility by moving to another coalition in CS. Simply put, all robots have maximized their utility and are in the coalitions they most prefer. Our objective in this work is to follow the properties of the hedonic model to build self-forming coalitions and form a final coalition structure that is Nash-stable, which is formally defined next [SHB$^+$11].

Definition 1 *A coalition structure CS is Nash-stable if $\forall\, r \in R$, $c'(r_i) \succeq_{r_i} c \cup \{r_i\}$ for all $c \in CS \cup \{\emptyset\}$*

To form such coalitions, we utilize concepts presented in [BFF$^+$18, DA19] focusing on solutions based on bipartite matching. To model the coalitions, we use an undirected, weighted bipartite graph unlike the unweighted graph model used in [BFF$^+$18]. We first perform a matching on a bipartite graph modeled with the robots and the tasks. Thereafter, we give the robots the opportunity to improve their utilities by relocating to different coalitions as discussed next[1].

4.1 Maximum Bipartite Graph Matching

We begin with a maximum weighted bipartite matching formulation to establish initial coalitions where exactly one robot is assigned to each task. Although we have used the classical weighted bipartite matching algorithm (Algorithm 1) presented in [MB07], any other weighted matching algorithm can be used it its place.

We define our undirected, weighted bipartite graph as $G = (\{V, U\}, E, W)$. V and U are two sets of vertices where V represents the robots and U represents the tasks. E is the set of all possible edges between $u - v$ pairs where $u \in U, v \in V$. W is the set of edge weights. The weight of an edge is the utility between a robot-task pair and as

[1]The presented algorithm here has been published in [CD19]

mentioned earlier, it is calculated using Eq. 3.3. Let $M \subseteq E$ denote the set of edges that do not share any end vertices. Algorithm 1 will produce such an edge set, called *matching*, which is also the maximum in size.

Algorithm 1: Weighted Maximum Bipartite Matching

Input: $G(\{V, U\}, E, W)$: A weighted bipartite graph
where V = set of robots and U = set of tasks
Output: M: a weighted maximum matching.

1 **for** *each* $v \in V$ **do**
2 $S_v \leftarrow U$;
3 $M \leftarrow \emptyset$;
4 $T_A \leftarrow \emptyset$;
5 **for** *each* $u \in U$ **do**
6 $v \leftarrow bm(u)$;
7 **if** $bm(u) = v$ **then**
8 /*u and v are mutually best for each other*/
9 $M \leftarrow M \cup \{u, v\}$;
10 $T_A \leftarrow T_A \cup \{u, v\}$;
11 **while** $T_A \neq \emptyset$ **do**
12 $u \leftarrow$ end vertex $\in U$ from an edge in T_A;
13 $T_A = T_A \setminus \{u, v'\}$ where $\{u, v'\} \in M$;
14 **for** *each* $u \in S_v$ *where* u *is unmatched* **do**
15 **if** $bm(bm(u)) = u$ **then**
16 $T_A = T_A \cup \{u, bm(u)\}$;
17 $M = M \cup \{u, bm(u)\}$;
18 return M;

Algorithm 1 will take each task and match it to a robot where the robot-task matching is *mutually best* for each other. A mutually best robot-task pair means the edge $\{u, v\}$ is the maximum weighted edge among all edges incident to u and v. First, for each $v \in V$ we obtain the potential matches and store in S_v (lines $1 - 2$). We initialize S_v to U because initially, every $u \in U$ is a potential match. We initialize M to an empty set. While there are tasks remaining to be matched to a robot, for each such task, we find a mutually best robot pairing (lines $8 - 9$). Once a robot-task pair is found to be mutually best for each other, they are added to M. This matched pair is also added to the set T_A

so that it can be used later for assigning other tasks. After matching the initial set of mutually best robot-task pairs, the remaining unmatched tasks are assigned in a similar fashion (lines $11 - 17$). This process terminates when each task is assigned exactly one robot to it as $m < n$. At this stage, the maximum matching creates the initial set of coalitions. However, all the robots might not have been allocated to some tasks at this stage.

4.2 Hedonic Coalition Formation Algorithm

After the initial maximum weighted matching is found, we use that for the final hedonic coalition formation where a robot may choose to leave its current coalition for another if doing so will increase its utility. We first create the initial coalition structure, i.e., where each robot is assigned to exactly one task. Next, the robots with improving deviations consider moving to other coalitions when there is an opportunity to do so.

4.2.1 Improving Deviation

A robot r_i has an improving deviation if r_i can improve its utility by leaving its current coalition c and moving to another coalition c' [BFF$^+$18]. Any robot with an improving deviation will belong to a set defined as $R_{ID} = \{r_1, r_2, \dots\ r_N\}$ where $R_{ID} \subseteq R$.

To develop self-forming coalitions a robot r_i with an improving deviation will choose a coalition to relocate to based on *preference*. Therefore, for a robot considering relocation, we define a preference relation between itself and all other coalitions. We define the preference relation as follows:

$$c_1 \succeq_{r_i} c_2 \leftrightarrow u_{r_i}(c_1) \geq u_{r_i}(c_2) \tag{4.1}$$

In layman's terms, a robot r_i will strictly or equally prefer coalition c_1 over c_2 *iff* r_i's utility associated to c_1 is greater than or equal to r_i's utility in c_2. The preference relation is specific to some robot r_i and accordingly we denote the preference as $\succeq_{r_i}$. In our approach, only one robot may change coalitions at a time. This robot will have the lowest edge weight i.e. the lowest utility among the robots in R_{ID}. The robot in question will define a preference relation for itself by considering all possible coalitions in the current CS.

4.2.2 Moving to a New Coalition

After the robot with the lowest edge weight in R_{ID} generates a preference relation between itself and the current coalitions in the CS, it will join another coalition that increases its utility. A robot r_i will leave its coalition by using the following a *switch rule*, similar to that of [SHB$^+$11]: Given a coalition structure $CS = \{c_1, c_2, \dots\ c_m\}$, the candidate robot $r_i \in R_{ID}$ will leave its coalition c and join another coalition c', if and only if $c' \cup \{r_i\} \succ_{r_i} c$. We extend the unweighted hedonic coalition formation algorithm in [BFF$^+$18] to a weighted setting and form hedonic coalitions.

Algorithm 2 defines the pseudo-code for coalition formation using principles in fractional hedonic games. Given an undirected weighted bipartite graph the algorithm will return a Nash-stable coalition structure CS. It will begin by first computing a weighted maximum matching defined in Algorithm 1. The matching returned by Algorithm 1 establishes the initial set of m coalitions (line 4). Next, for each robot (v) not yet in a coalition, it will select a task using a greedy methodology. A robot will allocate itself to a task by selecting the edge between itself and some task (u) such that the edge is the maximum value for that robot among all potential tasks. This will continue until all robots have a task allocation (lines $5 - 8$). This provides an initial CS where each robot is allocated to some task. Finally, the algorithm selects the robot in R_{ID}, the set

Algorithm 2: Hedonic Coalition Formation

Input: R: A set of Robots; T: A set of Tasks
Output: CS: A Final Coalition Structure

Create the bipartite graph $G(\{V, U\}, E, W)$;
$M \leftarrow$ Weighted maximum matching from Algorithm 1;
$CS \leftarrow \emptyset$;
$Covered \leftarrow$ Set of all robots part of M;
for *each* $v \notin Covered$ **do**
    choose $u \in U \mid \{u, v\} \in E$ and $w_{u,v} \geq w_{u',v} \; \forall \; u' \neq u$;
    $c_u \leftarrow c_u \cup v$;
    $Covered \leftarrow Covered \cup v$;
Update CS;
Compute the set of robots with Improving Deviation, R_{ID};
while $R_{ID} \neq \emptyset$ **do**
    Select $r_i \in R_{ID}$ with lowest edge weight;
    r_i considers switch operations using the preference relation calculated in Eq. 4.1;
    r_i selects a coalition using the switch rule;
    Update CS;
return CS;

of robots with improving deviations, with the minimum edge weight allows this robot to move to another coalition using the switch rule. The robot will leave its coalition and choose the coalition that maximizes its utility, i.e., the task that has the highest edge weight for that specific robot. The current coalition structure CS is thereby updated. When the set R_{ID} is empty, i.e., no robot can improve its utility by moving coalitions, CS is considered stable and the final coalition structure CS is returned. The final coalition structure can be considered Nash-stable and the proof of stability is described in the research performed in [BFF$^+$18]. After the final CS is formed, the robots will move to their allocated tasks. Inter-robot collisions can be avoided while minimally increasing the initially estimated path lengths by using techniques proposed in [DD17].

4.2.3 Discussion on Complexity

Time complexity of Algorithm 1 is $\mathcal{O}(mn)$ [MB07]. If an optimal algorithm for maximum bipartite matching such as Hungarian [Kuh55] is used instead, the complexity would become $\mathcal{O}(\max(m,n)^3)$.

The time complexity of Algorithm 2 would primarily depend on lines $2, 5-8,$ and $10-15$. As discussed above, line 2 will incur a complexity of $\mathcal{O}(mn)$. Lines $5-8$ will incur a time complexity of $\mathcal{O}(n)$ as all the robots are unassigned at this point in the worst case scenario. Complexity of line 10 in Algorithm 2 is $\mathcal{O}(mn)$ as each robot will check whether it can improve its utility by moving to any of the existing coalitions, the maximum count of which is m. As the R_{ID} set can maximally have n robots in it and each robot $r_i \in R_{ID}$ will be checking for the *switch*, lines $11-15$ will incur a complexity of $\mathcal{O}(mn)$. Thus, the worst-case time complexity of Algorithm 2 is $\mathcal{O}(mn)$.

Now, we compare time complexity of our approach with that of the four existing state-of-the-art algorithms in 4.1. $\mathcal{C}$ denotes the total number of coalitions ($= 2^n$) possible with n robots. Therefore, it should be noted that all these algorithms will incur a high, exponential time complexity whereas our proposed hedonic game-based coalition formation approach incurs only a polynomial time complexity. This helps our algorithm to be highly scalable whereas the compared state-of-the-art algorithms can not handle more than only tens of robots. One should also note that *ResourceCentricApprox* algorithm [ZP13] also incurs high space complexities due to the fact that it creates numerous hash tables to store and track coalitions with resource constraints. As the number of coalitions ($\mathcal{C}$) grows exponentially with n, they will incur exponential space complexities. On the other hand, our proposed method does not take this approach and requires only polynomial space ($\mathcal{O}(mn)$). These practical implications are also demonstrated in the next section.

Table 4.1: Summary of time complexity comparison

Algorithm	Reference	Complexity
Average Utility	[SA11, SK98, ZP13]	$\mathcal{O}(\min(m,n) \cdot m\mathcal{C})$
Max Utility	[SA11, SK98, ZP13]	$\mathcal{O}(\min(m,n) \cdot m\mathcal{C})$
ResourceCentric	[ZP13]	$\mathcal{O}(\min(m,n) \cdot m^2\mathcal{C}^2)$
ResourceCentricApprox	[ZP13]	$\mathcal{O}(\min(m,n) \cdot m^2 n\mathcal{C})$
Our approach	this paper	$\mathcal{O}(mn)$

CHAPTER 5

EXPERIMENTS

5.1 Settings

We have implemented our proposed hedonic coalition formation algorithm using the Java programming language. The tests were run on a laptop with an Intel $i7 - 3615$QM processor and 16GB RAM. The number of robots (n) has been varied between $[4, 100]$, and the number of tasks (m) has been varied between $[2, 10]$. We have made sure that in no test case the number of tasks exceeds 50% of the number of robots used. The distinct 2D locations of the robots and the tasks are randomly generated from a bounded square area with sides of length 100m. The total number of possible sensors ($|S|$) was set to 3 and the sensor distribution was randomly generated for both robots and tasks. To begin, robots received a random combination between one and three unique sensors. To determine how many sensors each task would receive, the sum of sensors among the robots was randomly partitioned based on the number of tasks in the environment. The sum of each type of sensor given to the robots was used to randomly generate the sensor requirements of the tasks. This results in an environment where the tasks have sensor requirements that can be filled by the robots in the environment. Each setting was run 20 times with an average result calculated and illustrated in the graphs in the results section. The bars in the plots indicate the maximum and the minimum value obtained for any particular metric.

We have compared the performance of our algorithm against four previous approaches. The first two approaches – *MaxUtility* and *AverageUtility* – implemented for compari-

son are greedy algorithms developed in [SK98] as well as modifications of this approach in [SA11] and [ZP13]. In both algorithms – beginning with the first task – all coalitions of a size equal to that of the robot requirement for the given task are generated. Then each candidate coalition is evaluated. With *MaxUtility* the total utility of the coalition is considered. The alternate approach – *AverageUtility* – considers the average utility of the candidate coalition. Given that the goal is to maximize utility, the candidate coalition with the highest utility (Max Utility), or highest average (Average Utility) is assigned to the task being considered in greedy fashion. The task and robots are then removed from consideration and the process repeats with remaining tasks and robots.

The other two solutions implemented for comparison are developed in [ZP13]. The *ResourceCentric* and *ResourceCentricApprox* are greedy heuristics that consider inter-task resource constraints and an approximation of that heuristic to improve performance respectively. For both, the first step is to generate all possible coalitions given a maximum size. In *ResourceCentric*, an undirected graph is generated where each node is a coalition and edges are added where a conflict exists between two coalitions. A conflict occurs when two coalitions contain the same task or robots as a task cannot be assigned more than one set of robots and robots cannot be in multiple coalitions. After the graph is generated, while the graph is not empty, for each coalition (node) and for each of its neighbors a value ρ is calculated which is the utility of the coalition minus the sum of the conflicting assignments multiplied by the utility of the neighbor coalition. The coalition that maximizes ρ is selected and this coalition, its neighbors and connecting neighbors are removed from the graph. The key differences in *ResourceCentricApprox* as compared to *ResourceCentric* is an approximation of the calculation ρ and the use of hash tables to track conflicts instead of an undirected graph.

5.2 Results

A number of experiments were run to determine the overall quality and effectiveness of the solutions. These experiments and their results are presented in the following subsections.

5.2.1 Utility Comparisons

First, we are interested in the investigation of the approximation ratio (higher is better, 1 being the optimal solution) achieved by our proposed method to an optimal solution. To this end, we have implemented a brute-force method [Orl02] to obtain the optimal solution and we compute the ratio of our solution's utility to that of the optimal solution. The brute-force method employs the same cost, value, and utility calculations presented in Chapter 3, Section 3.1. The result is presented in Fig. 5.1. This plot demonstrates the near-optimal nature of our solutions. For example, with $m = 2$, the highest and the

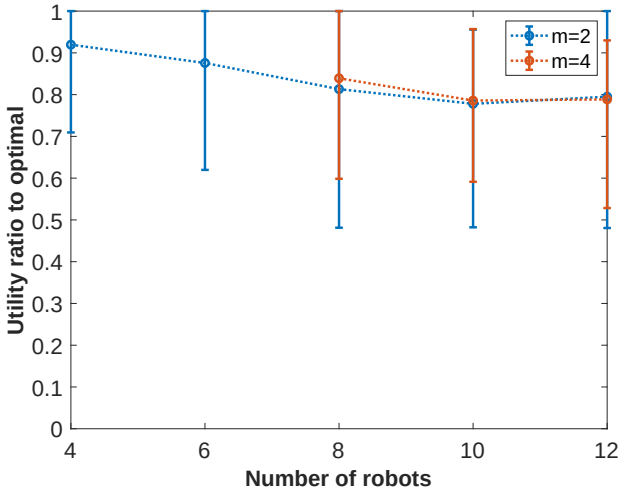

Figure 5.1: Approximation ratio to optimal

lowest approximation ratios obtained are 0.91 and 0.78 for $n = 4$ and 10 respectively. The approximation ratios were quite similar with $m = 4$ as compared with $m = 2$, and

we obtained an approximation ratio of 0.84 with $n = 8$ and 0.79 with $n = 12$. This experiment was run with a baseline benefit of 1.0 and penalty of 0.1. Improved utility results were obtained with a different benefit and penalty and is explored further in subsection 5.2.2. Next we look at how well our solution performs in terms of ratio to the optimal in comparison to the four additional greedy solutions implemented.

The result in comparing our approach to the *MaxUtility* solution is shown in Fig. 5.2. Here we can observe that *MaxUtility* provided better solutions with $m = 2$, however the solution degraded with $m = 4$ and our solution provided better results. The highest approximation ratio for *MaxUtility* was 0.96 with $m = 2$ and $n = 12$, and the lowest was 0.70 with $m = 4$ and $n = 10$.

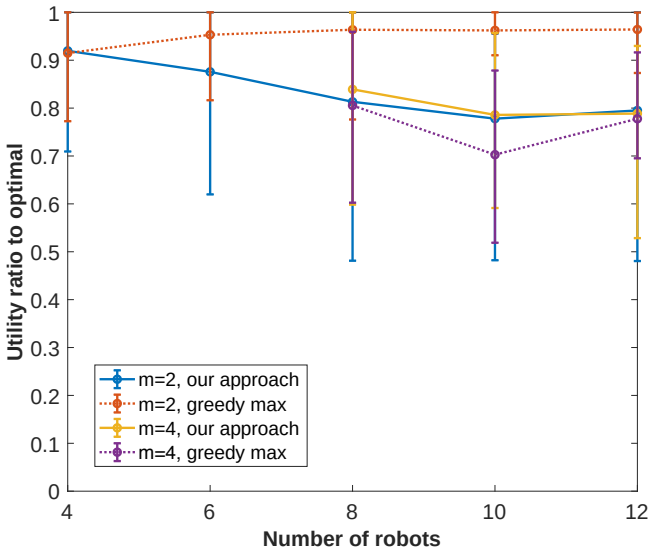

Figure 5.2: Approximation ratio to optimal (MaxUtility)

In Fig. 5.3, we view the results of our solution compared with *AverageUtility*. The results show *AverageUtility* provided a lower quality solution in all settings than our approach. The highest approximation ratio for *AverageUtility* was 0.86 with $m = 2$ and $n = 4$, and the lowest was 0.53 with $m = 4$ and $n = 12$.

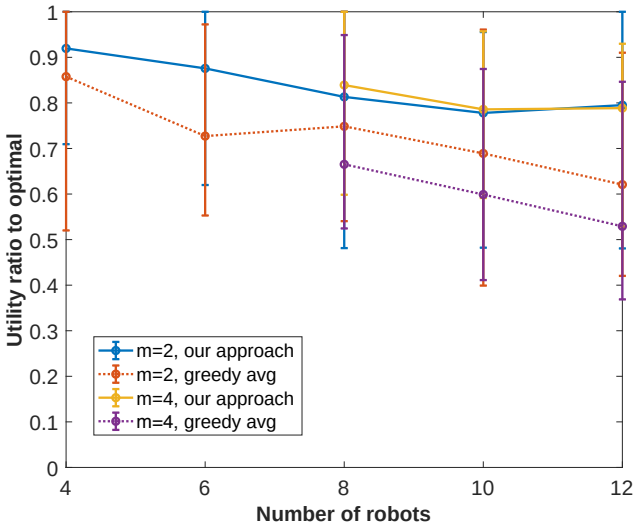

Figure 5.3: Approximation ratio to optimal (AverageUtility)

Next we investigate the results when comparing our solution with the *ResourceCentric* and *ResourceCentricApprox* heuristics. In Fig. 5.4, we observe the comparison of *ResourceCentric* with our approach. This result illustrates that *ResourceCentric* provides

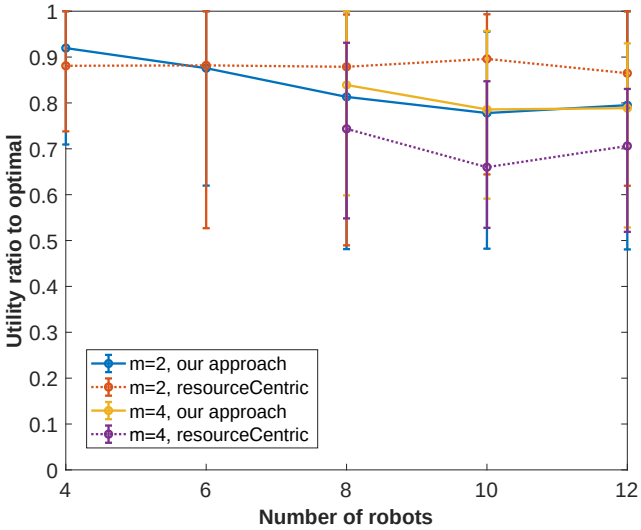

Figure 5.4: Approximation ratio to optimal (Resource Centric)

a better solution in some cases with $m = 2$, but with $m = 4$, our approach provides

a better solution. The highest approximation ratio for *ResourceCentric* was 0.89 with $m = 2$ and $n = 10$, and the lowest was 0.66 with $m = 4$ and $n = 10$.

In Fig. 5.5, we compare *ResourceCentricApprox* with our solution. *ResourceCentricApprox*, similarly to some of the earlier comparisons, resulted in better solutions at $m = 2$. However, did not perform as well as approach for $m = 4$. The highest approximation ratio for *ResourceCentricApprox* was 0.94 with $m = 2$ and $n = 6$, and the lowest was 0.70 with $m = 4$ and $n = 10$.

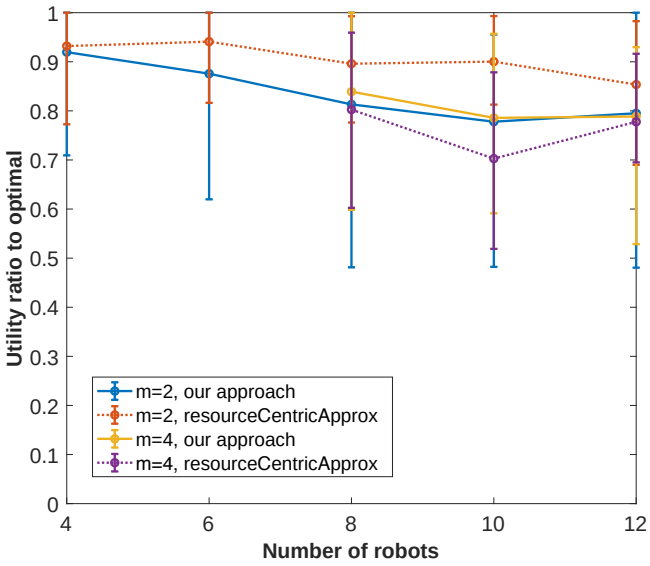

Figure 5.5: Approximation ratio to optimal (Resource Centric Approx.)

Following [DA19], we are also interested to investigate how many times out of the 20 simulation runs, we get a *good* solution, i.e., at least 80% of the optimal. The result is presented in Fig. 5.6. We observe that for $m = 2$, we can always achieve a solution 50% of the time that is within 80% of the optimal. Similarly for $m = 4$, we are also able to obtain a *good* solution in at least half of the simulation runs.

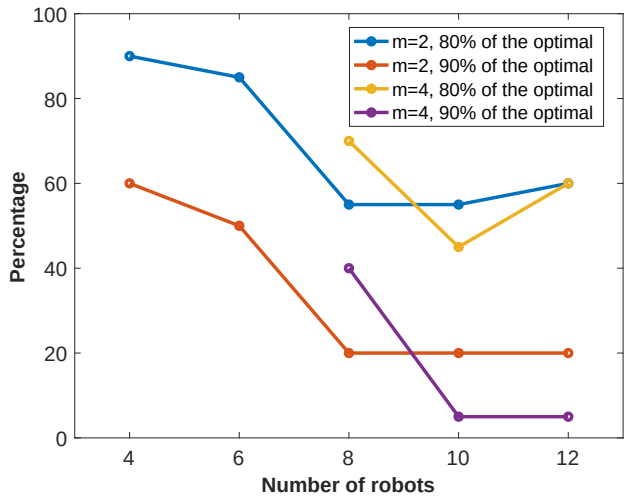

Figure 5.6: Percentage of tests in which our proposed approach generated a coalition structure with utilities 80% and 90% of the optimal.

In comparing our approach with the results achieved with *MaxUtility* as shown in Fig. 5.7, we see that *MaxUtility* outperforms our approach with $m = 2$, however does not perform as well with $m = 4$. *MaxUtility* can achieve a solution that is within 80% of the optimal 90% of the time, and at minimum 50% of the time within 90% of the optimal

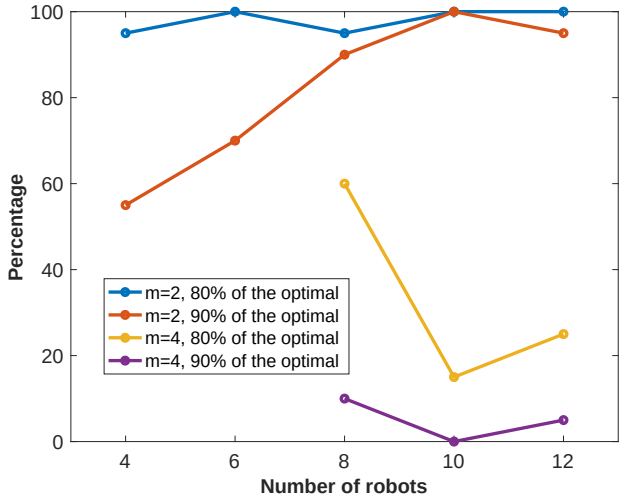

Figure 5.7: Percentage of tests in which the MaxUtility algorithm generated a coalition structure with utilities 80% and 90% of the optimal.

for $m = 2$. For $m = 4$, in most cases *MaxUtility* achieves solutions within 80% or 90% of the optimal less than 30% of the time.

AverageUtility overall was completely outperformed by our approach. The result is presented in Fig. 5.8. In all scenarios save one, *AverageUtility* could not achieve solutions within 80% or 90% of the optimal in half of the simulation runs.

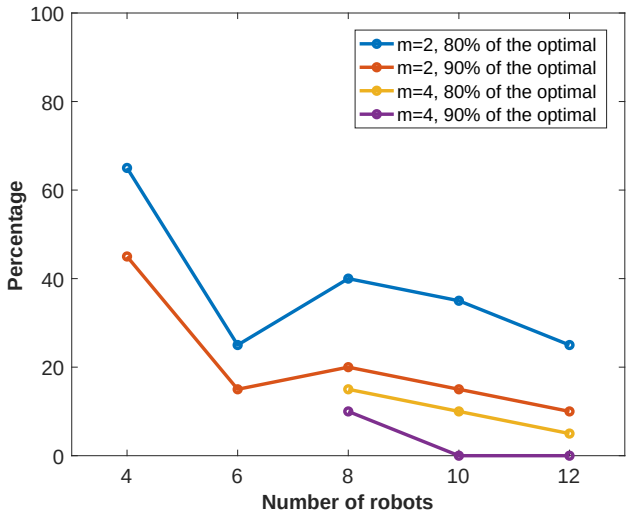

Figure 5.8: Percentage of tests in which the AverageUtility algorithm generated a coalition structure with utilities 80% and 90% of the optimal.

The *ResourceCentric* heuristic performed well with $m = 2$ as can be seen in Fig. 5.9. It provided a good solution–within 80% of the optimal–typically above 80% of the time. In at least half of the runs, it was able to produce a solution within 90% of the optimal. As we have seen with the other comparison algorithms though, it does not perform as well as our approach with $m = 4$ and does not achieve a solution within either 80% or 90% of the optimal more than 30% of the time.

A similar assessment can be made for the *ResourceCentricApprox* approach presented in Fig. 5.10. It provides a good solution in most cases for $m = 2$ at least 80% of the time.

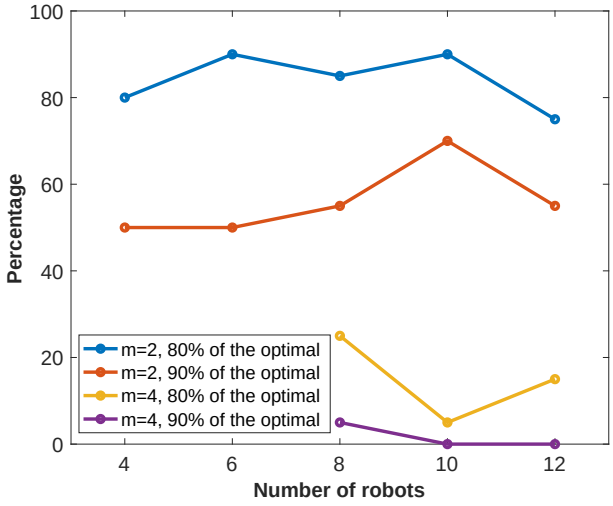

Figure 5.9: Percentage of tests in which the Resource Centric algorithm generated a coalition structure with utilities 80% and 90% of the optimal.

It performs more comparably with our approach with $m = 2$ when seeking a solution within 90% of the optimal. Again, similar to the other algorithms used for comparison, it does not perform as well when $m = 4$ and in all cases except where $n = 8$ does not achieve a solution within either 80% or 90% of the optimal more than 30% of the time.

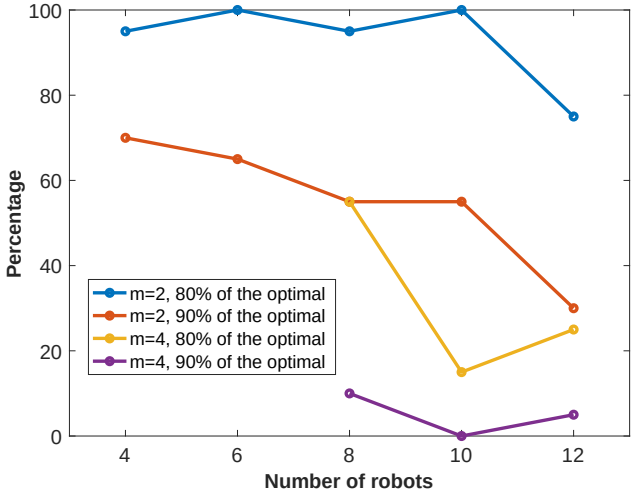

Figure 5.10: Percentage of tests in which the Resource Centric Approx. algorithm generated a coalition structure with utilities 80% and 90% of the optimal.

Overall we can conclude in some cases the comparison algorithms provide better solutions when looking at ratio to the optimal solution, but in all cases, our solution shows better results as the number of tasks increase demonstrating better scalability.

5.2.2 Impact of Benefit and Penalty

Next we investigate how the benefit(b) and penalty(p) weights affect the solution in terms of utility and proximity to the optimal solution. In these experiments, we chose baseline values of $b = 1.0$ and $p = 0.1$. To understand the impact of these two values, we ran an experiment with b values varying between 1.0 and 1.75 with increments of 0.25, and an additional experiment with p values between $[0.1, 0.4]$ with increments of 0.1. First, we review the impact of the b values on the approximation ratios (Fig. 5.11). Note that, in this case, p was static at 0.1. Overall, we can observe, that the change in b does indeed impact the approximation ratio. Furthermore, we can observe, that the baseline benefit of 1.0 produces the lowest approximation ratios. Overall, when looking at all scenarios, a benefit of $b = 1.50$ offers, in general, higher approximation ratios. At $m = 4$ and $n = 12$ the solution produces a mean approximation ratio of 0.85. The

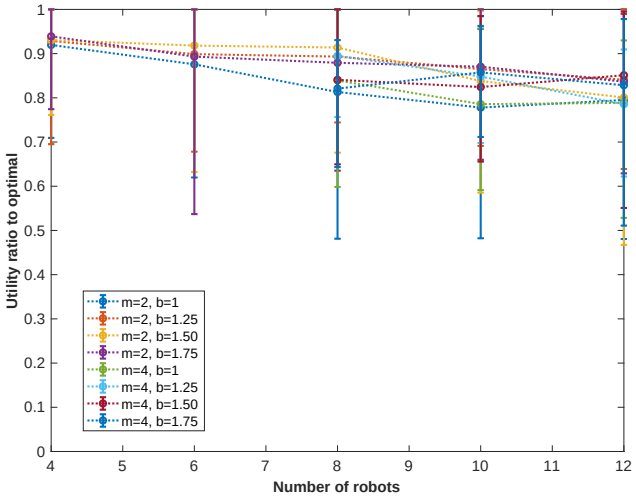

Figure 5.11: Approximation Ratio to Optimal with varying benefit(b) values

highest mean approximation ratio of 0.94 was achieved with $b = 1.75$ at $m = 2$ and $m = 4$.

We now analyze the results with varying penalty values. Our results presented in Fig. 5.12 show the approximations ratios with varying p values and static b value of 1.0. As with benefit, we observe that the penalty value does impact ratio to the optimal. However, $p = 0.1$ did produce the highest singular mean approximation ratio of 0.92 at $m = 2$ and $n = 4$. When looking at the quality of solutions in all scenarios though, a penalty of $p = 0.2$ results in a higher approximation ratios producing a ratio of 0.83 with $m = 4$ and $n = 12$.

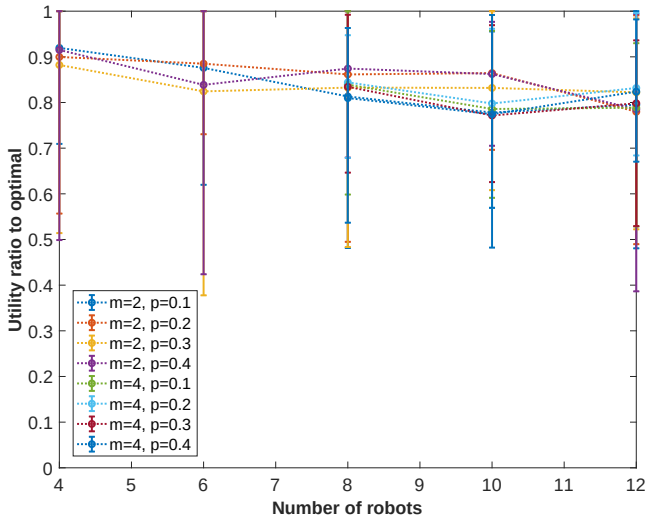

Figure 5.12: Approximation Ratio to Optimal with varying penalty(p) values

Finally, after identifying benefit and penalty values that perform well, we were interested in reviewing the results in terms of approximation ratio when employing those values. We can observe these results in Fig. 5.13. In this result, we compare the baseline benefit and penalty (1.0, 0.1 respectively), to our observed better performing benefit and penalty (1.50, 0.2 respectively). As can be seen, overall higher approximation ratios were achieved with the latter. This is particularly notable with $m = 4$ and n values of

$8, 10, 12$. The highest approximation ratio of 0.94 was achieved at $m = 2$ and $n = 4$. Approximation ratios of $0.86, 0.84, 0.82$ were achieved with $m = 4$ and $n = 8, 10, 12$ respectively.

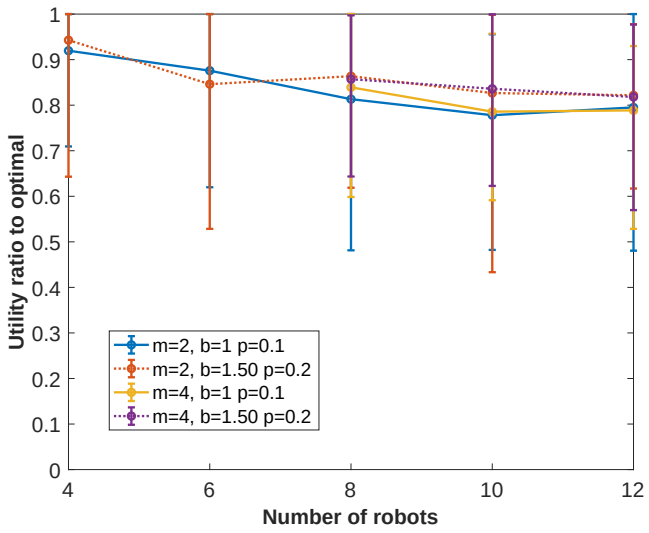

Figure 5.13: Approximation Ratio to Optimal with benefit = 1.50 and penalty = 0.2

5.2.3 Performance Comparisons

Next, we are interested in understanding how well our algorithm performs in terms of time in comparison with the brute-force method and the four comparison algorithms. Additionally, we are interested is how our solution scales with a large set of robots. For our first result, as expected, when compared to the brute-force method, our algorithm outperforms it in terms of run time. The result is shown in Fig. 5.14. With $m = 2$ and $n = 12$, our approach takes a negligible 0.028ms. While the brute-force takes 4.36ms. The difference becomes much more drastic with $m = 4$ where our approach takes 0.031ms. to run while the brute force algorithm takes $59,183.71$ms.

Run time comparison result against the *MaxUtility* solution is shown in Fig. 5.15. As can be observed, only in the first scenario of $m = 2$ and $n = 4$, *MaxUtilty* outperforms

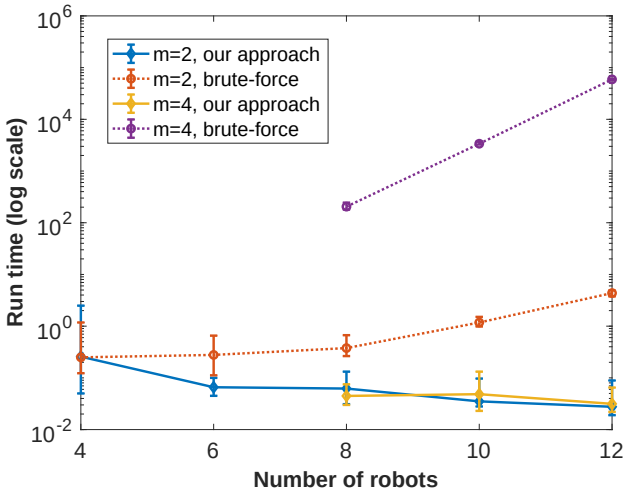

Figure 5.14: Run time comparison to a brute-force method [Orl02]

our solution with respect to run time. At very low numbers of tasks and robots, *MaxUtility* has very few possible coalitions to consider, and given the simplicity of the greedy

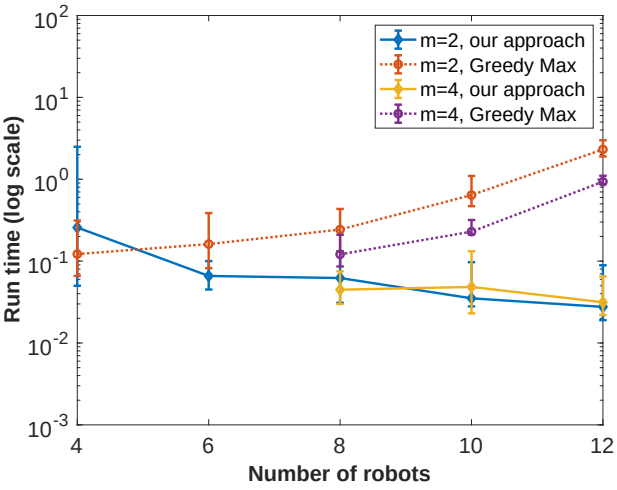

Figure 5.15: Run time comparison to Max Utility approach

choices in this algorithm, it makes sense that this algorithm would be quick initially. However, it quickly grows in run time as the number of robots and tasks increases due to the exponential nature of possible coalitions to consider. With $m = 4$ and $n = 12$,

MaxUtility takes 0.933ms, as compared to our 0.031ms, resulting in our approach being approximately 30 times faster.

Similar results can be observed with *AverageUtility* as illustrated in Fig. 5.16. Again, the first scenario of $m = 2$ and $n = 4$ shows a faster run time than our solution, but its run time increases significantly as the number of robots and tasks increase. With $m = 4$ and $n = 12$, *AverageUtility* takes 1.27ms, as compared to our 0.031ms, resulting in our approach being approximately 41 times faster.

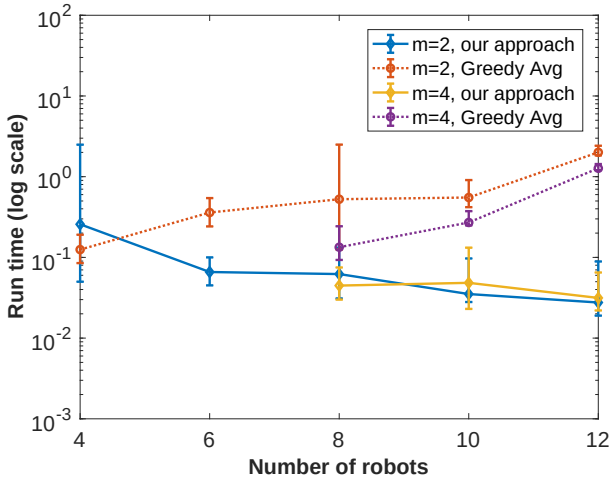

Figure 5.16: Run time comparison to Average Utility approach

Comparison result against the *ResourceCentric* algorithm is shown in Fig. 5.17. Here we can see our solution significantly outperforms *ResourceCentric*. As with *MaxUtility* and *AverageUtility*, the algorithm considers all possible coalitions of a maximum size, but performs more complex calculations taking resource constraints into account, resulting in a slower performance. At $m = 4$ and $n = 12$, *ResourceCentric* takes 1336.38ms with our solution taking 0.031ms, making our solution approximately 44,500 times faster.

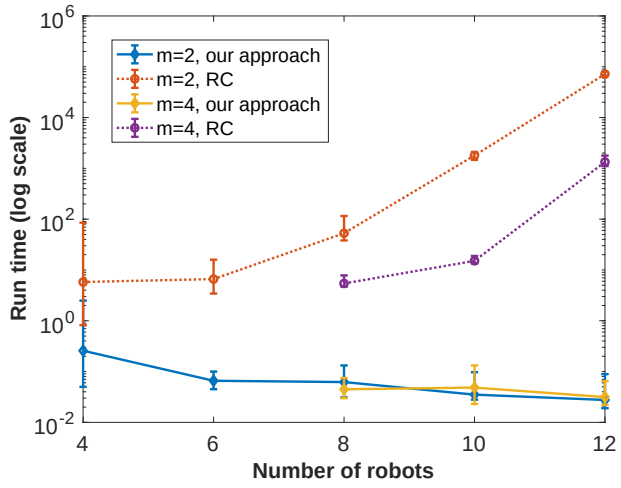

Figure 5.17: Run time comparison to Resource Centric approach

The *ResourceCentricApprox* algorithm is modification on *ResourceCentric* to improve its performance with run time. This can be observed in Fig. 5.18. We can see that it does indeed perform much better in terms of time as compared with *ResourceCentric*, however, it still does not perform as well as our solution.

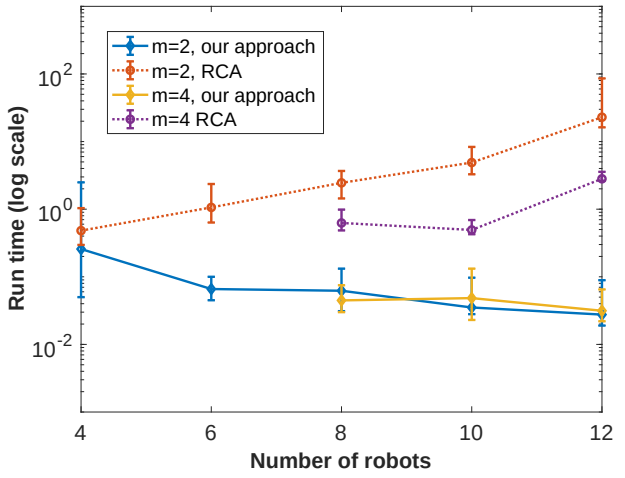

Figure 5.18: Run time comparison to Resource Centric Approx. approach

At $m = 4$ and $n = 12$, *ResourceCentricApprox* takes 2.81ms. As compared with our solution with a run time 0.031ms – making our solution approximately 90 times faster.

As stated earlier, our goal in this study is to develop a scalable algorithm that produces near-optimal solutions. Fig. 5.19 demonstrates the scalability of our proposed solution– the maximum run time is found to be 0.4ms. and the the mean run time was 0.192 milliseconds–with $n = 100$ and $m = 10$. A negligible number especially considering the fact that for this particular setting, the astronomical number of possible coalition structures is 2.75×10^{93}

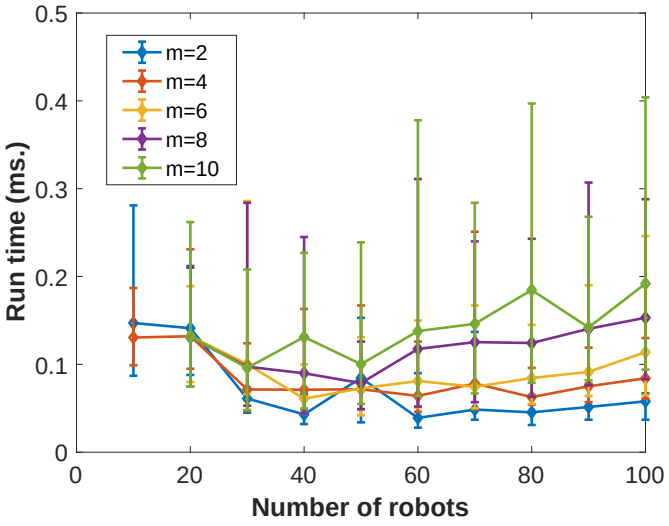

Figure 5.19: Run time of the proposed approach

Lastly, in this section, we consider distances the robots travel to get to their assigned tasks. The robots want to minimize the traveled distance amount to move to a specific task. Therefore, distance is an important performance metric. In Fig. 5.20, we see that the total distance traveled by the robots increases linearly with more robots irrespective of the task counts. As with less number of tasks in the environment, each robot needs to travel more distance to reach a particular task because of the uniform distribution of

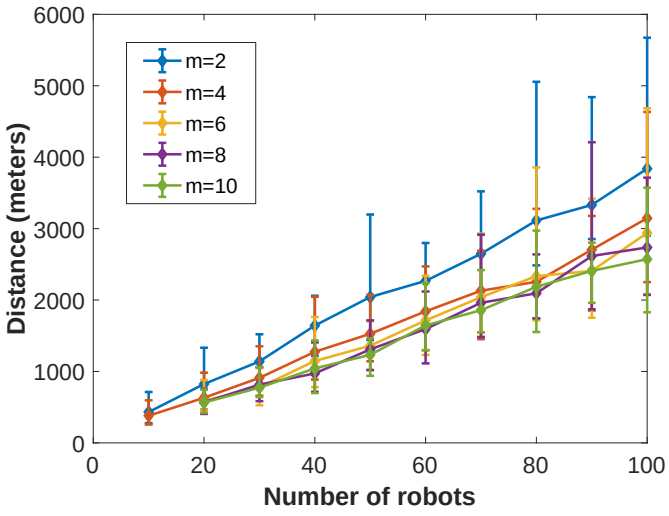

Figure 5.20: Total distance traveled by the robots to reach the allocated tasks.

tasks and robots, it is worth noting, that with less tasks, the robots travel more. This is consistent with the results found in [DA19]. If we observe the total distance traveled by the robots in Fig. 5.20, we can see the maximum distance traveled was with 2 tasks and 100 robots. The mean distance traveled was 3839.647 meters and the maximum was 5673.29 meters. With 10 tasks and 100 robots, the mean distance traveled was 2572.364 meters and a maximum of 3573.26 meters.

5.2.4 Switch Rule Analysis

The number of coalition switches does not show a clear trend (Fig. 5.21). However, we notice that the average number of switches occurred across the tested m and n values is almost negligible and generally higher number of tasks correspond to higher number of switches. As the robots have more coalitions to relocate to, the number of switches are more probable with more tasks. The small, finite number of switches also demonstrates the stability of the solution. For 2 tasks, 100 robots the mean number of switches was 0, for 4 tasks, 100 robots the mean switches was 0.2, for 6 tasks and 100 robots mean switches was 0.55, for 8 tasks 100 robots mean switches was 1 and for 10 tasks 100 robots mean switches was 1 as well. Looking at the maximum number of switches that

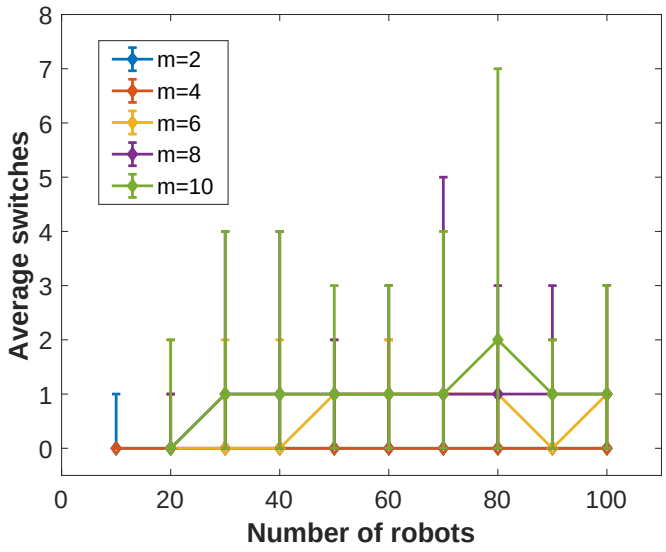

Figure 5.21: Average number of switches performed

took place for the $m = 2, 4, 6, 8, 10$, the maximum switch counts were 1, 2, 3, 5, and 6 respectively showing again an increase in switches as the number of tasks increase. Finally, we want to demonstrate the usefulness of the switch rule used in our model. We see in Fig. 5.22 that generally with a higher number of coalition switches the robots were able to increase the total utility, the maximum being 24% with six switches.

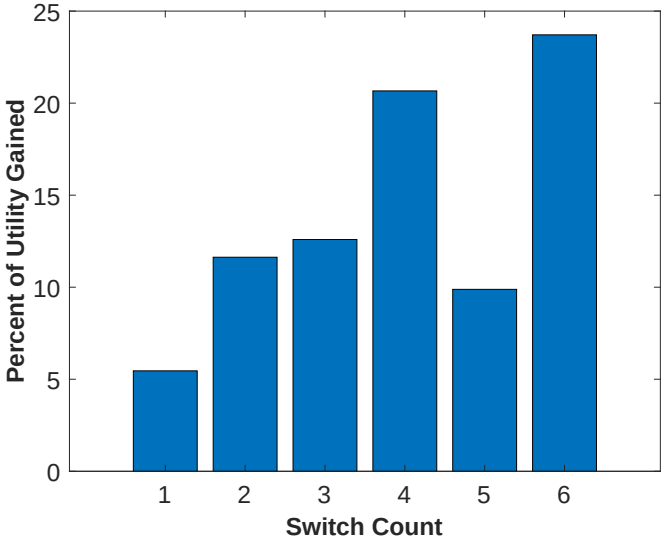

Figure 5.22: Utility gained (%) for different number of switches.

5.2.5 Implementation on a TurtleBot 3

Finally, we implemented the algorithm on a TurtleBot 3 robot equipped with a Raspberry Pi 3. We used the same experiment settings when testing the scalability varying tasks from $[2, 10]$ and robots from $[10, 100]$ taking the average of 20 runs. With $m = 10$ and $n = 100$ we achieved a mean run time of .322 seconds and a maximum run time of .867 seconds. This demonstrates the scalability of the proposed algorithm on a real hardware platform.

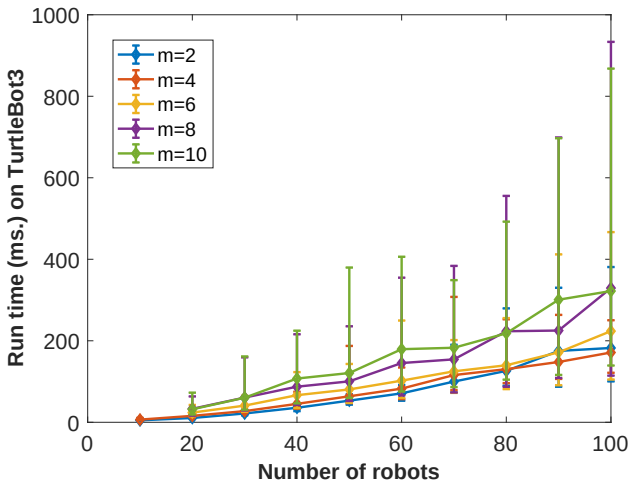

Figure 5.23: Run time of proposed approach on a TurtleBot 3

CHAPTER 6

CONCLUSIONS AND FUTURE WORK

The work explored in this thesis studies the problem of coalition formation with heterogeneous robots for task allocation. This problem has numerous real-world applications, however, finding the optimal solution for the studied problem is shown to be NP-Hard in the literature. The solution presented offers a novel approach to the heterogeneous multi-robot task allocation problem. To this end, we have proposed a hedonic coalition formation solution using concepts from graph matching. To the best of our knowledge, this approach is the first to solve the coalition formation problem for task allocation with a group of heterogeneous robots using a hedonic coalition game formulation. Results show that our proposed solution is fast, does produce near-optimal solutions, and can be applied for a large multi-robot system as well as offering the ability to calibrate the outcome with the benefit and penalty weights. Additionally, simulation runs on a TurtleBot 3 show that our solution can be utilized in practical application settings where computing capabilities may be limited. Comparisons against four state-of-the-art approaches resulted in comparable outcomes in terms of utility ratio, however, our solution provided a significant improvement in run time. There are many interesting extensions to the coalition formation problem. The increasing use of robots and robot teams across industries lends itself to revealing new ways in which robots can coordinate and fulfill task requirements. In the future, we can expand this work to consider inter-task dependencies, as well as incorporate uncertainty into the model. Furthermore, we plan to distribute the proposed method in order to avoid one point of failure.

BIBLIOGRAPHY

[AB12] Haris Aziz and Florian Brandl. Existence of stability in hedonic coalition formation games. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems-Volume 2*, pages 763–770. International Foundation for Autonomous Agents and Multiagent Systems, 2012.

[ABH14] Haris Aziz, Felix Brandt, and Paul Harrenstein. Fractional hedonic games. In *Proceedings of the 2014 international conference on Autonomous agents and multi-agent systems*, pages 5–12. International Foundation for Autonomous Agents and Multiagent Systems, 2014.

[BFF+14] Vittorio Bilò, Angelo Fanelli, Michele Flammini, Gianpiero Monaco, and Luca Moscardelli. Nash stability in fractional hedonic games. In *International Conference on Web and Internet Economics*, pages 486–491. Springer, 2014.

[BFF+18] Vittorio Bilo, Angelo Fanelli, Michele Flammini, Gianpiero Monaco, and Luca Moscardelli. Nash stable outcomes in fractional hedonic games: existence, efficiency and computation. *Journal of Artificial Intelligence Research*, 62:315–371, 2018.

[BJ02] Anna Bogomolnaia and Matthew O Jackson. The stability of hedonic coalition structures. *Games and Economic Behavior*, 38(2):201–230, 2002.

[CD19] Emily Czarnecki and Ayan Dutta. Hedonic coalition formation for task allocation with heterogeneous robots. In *IEEE International Conference on*

Systems, Man, and Cybernetics, SMC 2019, Bari, Italy, October 6-9, 2019, pages 1–6, 2019.

[DA19] Ayan Dutta and Asai Asaithambi. One-to-many bipartite matching based coalition formation for multi-robot task allocation. In *2019 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2019.

[DCAU19] Ayan Dutta, Emily Czarnecki, Asai Asaithambi, and Vladimir Ufimtsev. Distributed coalition formation with heterogeneous agents for task allocation. In *The 32nd International Flairs Conference*, 2019.

[DD17] Ayan Dutta and Prithviraj Dasgupta. Bipartite graph matching-based coordination mechanism for multi-robot path planning under communication constraints. In *Robotics and Automation (ICRA), 2017 IEEE International Conference on*, pages 857–862. IEEE, 2017.

[DG80] Jacques H Dreze and Joseph Greenberg. Hedonic coalitions: Optimality and stability. *Econometrica (pre-1986)*, 48(4):987, 1980.

[DUA19] Ayan Dutta, Vladimir Ufimtsev, and Asai Asaithambi. Correlation clustering based coalition formation for multi-robot task allocation. In *ACM/SIGAPP Symposium on Applied Computing*. ACM, 2019.

[GM04] Brian P Gerkey and Maja J Matarić. A formal analysis and taxonomy of task allocation in multi-robot systems. *The International Journal of Robotics Research*, 23(9):939–954, 2004.

[GS10] Martin Gairing and Rahul Savani. Computing stable outcomes in hedonic games. In *International Symposium on Algorithmic Game Theory*, pages 174–185. Springer, 2010.

[Kuh55] Harold W Kuhn. The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97, 1955.

[LCS14] Lingzhi Luo, Nilanjan Chakraborty, and Katia Sycara. Provably-good distributed algorithm for constrained multi-robot task assignment for grouped tasks. *IEEE Transactions on Robotics*, 31(1):19–30, 2014.

[LH93] Steven M LaValle and S Hutchinson. Game theory as a unifying structure for a variety of robot tasks. In *Proceedings of 8th IEEE International Symposium on Intelligent Control*, pages 429–434. IEEE, 1993.

[LV12a] Somchaya Liemhetcharat and Manuela Veloso. 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*, pages 365–374. International Foundation for Autonomous Agents and Multiagent Systems, 2012.

[LV12b] Somchaya Liemhetcharat and Manuela Veloso. Weighted synergy graphs for role assignment in ad hoc heterogeneous robot teams. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5247–5254. IEEE, 2012.

[MB07] Fredrik Manne and Rob H Bisseling. A parallel approximation algorithm for the weighted maximum matching problem. In *International Conference on Parallel Processing and Applied Mathematics*, pages 708–717. Springer, 2007.

[NS17] Changjoo Nam and Dylan A Shell. Analyzing the sensitivity of the optimal assignment in probabilistic multi-robot task allocation. *IEEE Robotics and Automation Letters*, 2(1):193–200, 2017.

[Orl02] Michael Orlov. Efficient generation of set partitions. *Engineering and Computer Sciences, University of Ulm, Tech. Rep*, 2002.

[Pro19] Amanda Prorok. Redundant robot assignment on graphs with uncertain edge costs. In *Distributed Autonomous Robotic Systems*, pages 313–327. Springer, 2019.

[RJ08] Talal Rahwan and Nicholas R Jennings. An improved dynamic programming algorithm for coalition structure generation. In *Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems-Volume 3*, pages 1417–1420. International Foundation for Autonomous Agents and Multiagent Systems, 2008.

[RRDJ07] Talal Rahwan, Sarvapali D Ramchurn, Viet Dung Dang, and Nicholas R Jennings. Near-optimal anytime coalition structure generation. In *IJCAI*, volume 7, pages 2365–2371, 2007.

[SA11] 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, 2011.

[SHB+09] Walid Saad, Zhu Han, Tamer Basar, Mérouane Debbah, and Are Hjorungnes. A selfish approach to coalition formation among unmanned air vehicles in wireless networks. In *2009 International Conference on Game Theory for Networks*, pages 259–267. IEEE, 2009.

[SHB+10] Walid Saad, Zhu Han, Tamer Basar, Are Hjorungnes, and Ju Bin Song. Hedonic coalition formation games for secondary base station cooperation in cognitive radio networks. In *2010 IEEE Wireless Communication and Networking Conference*, pages 1–6. IEEE, 2010.

[SHB+11] Walid Saad, Zhu Han, Tamer Basar, Merouane Debbah, and Are Hjorungnes. Hedonic coalition formation for distributed task allocation among wireless agents. *IEEE Transactions on Mobile Computing*, 10(9):1327–1344, 2011.

[SK95] Onn Shehory and Sarit Kraus. Task allocation via coalition formation among autonomous agents. In *IJCAI (1)*, pages 655–661, 1995.

[SK98] Onn Shehory and Sarit Kraus. Methods for task allocation via agent coalition formation. *Artificial intelligence*, 101(1-2):165–200, 1998.

[SPP18] Chayan Sarkar, Himadri Sekhar Paul, and Arindam Pal. A scalable multi-robot task allocation algorithm. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1–9. IEEE, 2018.

[ZP13] Yu Zhang and Lynne E. Parker. Considering inter-task resource constraints in task allocation. *Autonomous Agents and Multi-Agent Systems*, 26(3):389–419, May 2013.

VITA

Emily Czarnecki has Bachelor of Fine Arts degree from Northeastern University, Boston, in Graphic Design. Emily expects to receive a Master of Science in Computer and Information Sciences with a concentration in Computer Science from the University of North Florida, December 2019. Dr. Ayan Dutta is serving as Emily's thesis advisor. Emily is currently employed as a software developer at FloridaBlue and has been there for one year. Previous to that, Emily spent five years within the Architecture industry working as a Marketing Manager for the global firm Gensler in their Boston, Massachusetts location.

Her research work focuses on multi-robot task allocation and coalition formation. Her paper *Hedonic Coalition Formation for Task Allocation with Heterogeneous Robots*, co-authored with Dr. Ayan Dutta, was recently presented at the IEEE International Conference on Systems, Man, and Cybernetics (IEEE SMC), 2019. Additionally, she co-authored the paper *Distributed Coalition Formation with Heterogeneous Agents for Task Allocation* which won Best Poster Award at the 32nd International FLAIRS Conference, 2019. She also has co-authored the following journal articles: *Coalition Formation for Multi-Robot Task Allocation Via Correlation Clustering* which has been accepted to appear in the Cybernetics and Systems Journal, 2019 and *Hourly Weather Data Projection due to Climate Change for Impact Assessment on Building and Infrastructure* published by Elsevier in Sustainable Cities and Society in 2019.