

# AUS Repository

## Multi-Objective Task Allocation Via Multi-Agent Coalition Formation

Item Type	Thesis
Authors	Amer, Noha Tarek
Download date	2026-05-20 17:10:07
Link to Item	<a href="http://hdl.handle.net/11073/4071">http://hdl.handle.net/11073/4071</a>

MULTI-OBJECTIVE TASK ALLOCATION VIA MULTI-AGENT COALITION  
FORMATION

by

Noha Tarek Amer

A Thesis Presented to the Faculty of the  
American University of Sharjah  
College of Engineering  
in Partial Fulfillment  
of the Requirements  
for the Degree of

Master of Science in  
Engineering Systems Management

Sharjah, United Arab Emirates

May 2012



## **Acknowledgements**

I would like to express all my gratitude to God for all the favors and support he gave granted me through my life, in particular the completion of this work.

I am also very grateful to my adviser Professor Fouad Ben AbdelAziz, for his continuous support, guidance and patience throughout this work. Without his valuable supervision, encouragement, and advice throughout my MSc. study, I would have not been able to complete this work. I would like to express my deepest appreciation for his valuable efforts with me.

I would also like to extend my gratitude to all ESM faculty members; Dr.Moncer Hariga, Dr. Hazem El Baz, and Dr. Ibrahim El Kattan for their continuous help and support. I am also very grateful for the advice and help provided by my committee members;Dr. Ibrahim El Kattan, Dr. Imran Zualkernan and Dr. Taha Landolsi. I would also like to extend my deepest appreciation to all CEN faculty and staff, specially Ms.Salwa Mohammed for her guidance and encouragement to finish my thesis on time.

I would also like to thank my colleagues in the ESM program and CEN, specially Ms.Salam Taji, Ms.Faten Qutifan, Mr.Mahmoud Hussein, Mr.Nour Nour, Mr.Ibrahim Muhammed, Ms.Sheikha Al Shaeer, Ms.Zenab Khan, Ms.Halah El Tayer, Mr.Mohamed Shaban, Mr.Mohamed Shehata, Mr. Mohamed Amer, Mr.Ahmed Mahdi and Ms.Fatima Ali for their continuous help and support. I would also like to extend my deep appreciation to my office colleagues; Mr.Edi Ali, Mr.Ahmed Ghadban, Mr.Hydar Midhat, Mr.Nasser El Sughayar, Ms. Manal Kakaani, Ms.Assia Lasfer, and Ms.Rana El Haj.

Last but not the least, I would like to deeply thank my family; my father, mother and brothers; Mohammed Tarek Amer and Mustafa Tarek Amer, who provided me with all the emotional support and encouragement to finish my thesis and complete this work.

This thesis is dedicated to all of you...Thank you very much.

## Abstract

Nowadays, tasks are complex and cannot be performed by individual agents. Therefore, there is a need to form coalitions utilizing the resources in order to maximize the efficiency of the system and/or maximize the payoff of each agent. In this thesis, we will formulate an optimization model to form optimal coalitions of agents satisfying both maximizing the efficiency of the system for cooperative settings and/or the payoff of the agent for non-cooperative settings. For small sized problems, we propose an optimization model to get exact solutions. For large sized problems, we propose genetic algorithms to get satisfying solutions. Different experiments were performed for the different settings. For cooperative settings, we observe that the tasks with the maximum payment are being performed, the agents who are most capable are assigned to perform the task provided they are the cheapest, and agents are assigned if and only if they will participate in performing a task to avoid any wasted costs. As for the genetic algorithms, the code is giving exact solution for medium problems. For selfish settings, we observe that the tasks with the maximum payments are being performed because they give maximum payoffs for agents, and agents prefer high payment over a complex task; that is, a capable agent would perform a high paying complex task rather than a low paying simple task. Also, very frequently all agents are assigned to tasks even if they don't contribute to accomplishing the task just to increase their payoff. As for the genetic algorithms, the code is giving exact solution for medium problems with time savings advantages. As for the hybrid setting, we observe that by comparing both cooperative and selfish agents, cooperative agents provide better results for the system. Our results coincide with the theories of game theory that say cooperative games provide a higher utility. However, we also show that combining both behaviors of agents, cooperative and selfish, provides better results. In addition to that, our genetic algorithm is giving exact solution for medium problems with time savings towards problems with a large number of agents. Most applications are realized in e-commerce systems or parallel processing.

**Search Terms:** Multi Objective Programming, Coalition Formation, Task Allocation, Genetic Algorithms, Cooperative Agents, Non-Cooperative Agents

# Table of Contents

<b>Abstract</b>	<b>3</b>
<b>1 Introduction</b>	<b>10</b>
1.1 Overview . . . . .	10
1.2 Agents and Multi Agent Systems (MAS) . . . . .	11
1.2.1 Overview . . . . .	11
1.2.2 Types of Environments in which Agents Operate . . . . .	11
1.2.3 Multi Agent Systems (MAS) . . . . .	12
1.3 Game Theory . . . . .	12
1.3.1 Types of Games . . . . .	14
1.3.2 N-person Games and Coalition Formation . . . . .	14
1.4 Task Allocation and Coalition Formation . . . . .	16
1.4.1 Super Additive Environment . . . . .	16
1.4.2 Non-Super Additive Environment . . . . .	16
1.5 Problem Solving Methodologies . . . . .	17
1.5.1 Meta Heuristics . . . . .	17
1.6 Problem Statement and Complexity . . . . .	19
1.7 Research Significance . . . . .	19
1.8 Research Methodology . . . . .	20
1.9 Thesis Report Organization . . . . .	20
<b>2 Literature Review</b>	<b>21</b>
2.1 Cooperative Agents - Maximizing the efficiency of the system . . . . .	21
2.2 Selfish Agents- Maximizing the payoff of the agent . . . . .	29
2.3 Other Aspects in Coalition Formation . . . . .	33
2.4 Chapter Summary . . . . .	37
<b>4 Overview of Task Allocation via Coalition Formation in Cooperative Agents</b>	<b>38</b>
3.1 Assumptions . . . . .	38
3.2 Notations . . . . .	38
3.2.1 Sets . . . . .	38
3.2.2 Input Parameters . . . . .	39
3.2.3 Binary Decision Variables . . . . .	39
3.3 Constraints . . . . .	40

3.4	Objectives . . . . .	40
3.5	Illustrative Example . . . . .	40
3.5.1	Sensitivity Analysis . . . . .	41
3.6	Computational Results and Conclusion . . . . .	43
3.6.1	Testing Environment . . . . .	43
3.6.2	Genetic Algorithms Application for cooperative agents . . . . .	44
3.7	Selection Strategy . . . . .	46
3.8	Chapter Summary . . . . .	50
<b>5</b>	<b>Overview of Task Allocation via Coalition Formation in Selfish Agents</b>	<b>51</b>
4.1	Assumptions . . . . .	51
4.2	Notations . . . . .	51
4.2.1	Sets . . . . .	51
4.2.2	Input Parameters . . . . .	52
4.2.3	Binary Decision Variables . . . . .	53
4.2.4	Continuous Decision Variables . . . . .	53
4.3	Constraints . . . . .	53
4.4	Objectives . . . . .	53
4.5	Illustrative Example . . . . .	53
4.5.1	Sensitivity Analysis . . . . .	54
4.6	Computational Results and Conclusion . . . . .	57
4.6.1	Testing Environment . . . . .	57
4.6.2	Genetic Algorithms Application for selfish agents . . . . .	58
4.7	Performance Assessment . . . . .	59
4.8	Chapter Summary . . . . .	60
<b>5</b>	<b>Overview of Task Allocation via Coalition Formation in Cooperative AND Selfish Agents</b>	<b>62</b>
5.1	Assumptions . . . . .	62
5.2	Notations . . . . .	62
5.2.1	Sets . . . . .	62
5.2.2	Input Parameters . . . . .	63
5.2.3	Binary Decision Variables . . . . .	63
5.2.4	Continuous Decision Variables . . . . .	64
5.3	Constraints . . . . .	64
5.4	Objectives . . . . .	64

5.5	Illustrative Example . . . . .	64
5.5.1	Sensitivity Analysis . . . . .	64
5.6	Computational Results and Conclusion . . . . .	67
5.6.1	Testing Environment . . . . .	67
5.7	Overview of Genetic Algorithm . . . . .	68
5.7.1	The fitness function . . . . .	68
5.7.2	Genetic Algorithm Pseudo Code . . . . .	70
5.8	Performance Assessment . . . . .	70
5.9	Chapter Summary . . . . .	71
<b>8</b>	<b>Conclusion and Recommendation</b>	<b>73</b>
6.1	Conclusion . . . . .	73
6.2	Future Work . . . . .	74
<b>A</b>	<b>Summary of Exact Solution for Cooperative agents for Medium Problems</b>	<b>79</b>
<b>B</b>	<b>Comparison Between GA Solution versus Exact Solution for Cooperative agents for Medium Problems</b>	<b>81</b>
<b>C</b>	<b>Summary of Exact Solution for Selfish agents for Medium Problems</b>	<b>83</b>
<b>D</b>	<b>Comparison Between GA Solution versus Exact Solution for selfish agents for Medium Problems</b>	<b>85</b>
<b>E</b>	<b>Summary of Exact Solution for Cooperative and Selfish agents for Medium Problems</b>	<b>87</b>
<b>F</b>	<b>Comparison Between GA Solution versus Exact Solution for cooperative and selfish agents for Medium Problems</b>	<b>89</b>
<b>G</b>	<b>Summary of Cooperative versus Selfish versus Hybrid Solutions</b>	<b>91</b>
<b>H</b>	<b>Summary of Solution for Cooperative agents for Big Problems</b>	<b>93</b>
<b>I</b>	<b>Summary of Solution for Selfish agents for Big Problems</b>	<b>94</b>
<b>J</b>	<b>Summary of Solution for Hybrid agents for Big Problems</b>	<b>95</b>
<b>K</b>	<b>Lingo Code for Cooperative Agents</b>	<b>96</b>

<b>L Lingo Code for Selfish Agents</b>	<b>98</b>
<b>M Lingo Code for Hybrid Agents</b>	<b>100</b>
<b>Vita</b>	<b>102</b>

## List of Figures

Figure 1	Performance Versus Number of Tasks for Cooperative Settings . . .	49
Figure 2	Performance versus Number of Agents for Cooperative Settings . . .	50
Figure 3	Performance Versus Number of Tasks in Selfish Settings . . . . .	60
Figure 4	Performance versus Number of Agents in Selfish Settings . . . . .	60
Figure 5	Performance Versus Number of Tasks in Hybrid Settings . . . . .	71
Figure 6	Performance versus Number of Agents in Hybrid Settings . . . . .	71

## List of Tables

Table 1	Task's Structure . . . . .	40
Table 2	Agent's Structure . . . . .	41
Table 3	Coalitions Formed with cooperative agents . . . . .	41
Table 4	Sensitivity Analysis for Cooperative Behaviors . . . . .	41
Table 5	Chromosome Encoding . . . . .	45
Table 6	Task's Structure . . . . .	54
Table 7	Agent's Structure . . . . .	54
Table 8	Coalitions Formed with selfish agents . . . . .	54
Table 9	Sensitivity Analysis for Selfish Behaviors . . . . .	54
Table 10	Tasks . . . . .	65
Table 11	Agents . . . . .	65
Table 12	Coalitions Formed for Cooperative and Selfish behaviors combined	65
Table 13	Sensitivity Analysis for Hybrid Behaviors . . . . .	65
Table 14	Summary of Efficiency Results for medium problems . . . . .	79
Table 15	Genetic Algorithms Solution versus Exact Solutions for medium problems . . . . .	81
Table 16	Summary of Payoff Results for Medium Problems . . . . .	83
Table 17	Genetic Algorithms versus Optimal Solutions for Medium Problems	85
Table 18	Summary of Efficiency and Payoff Results for Medium Problems .	87
Table 19	Genetic Algorithms versus Optimal Solutions for Medium Problems	89
Table 20	More Comparisons . . . . .	92
Table 21	Summary of Efficiency Results for big problems . . . . .	93
Table 22	Summary of Payoff Results for big problems . . . . .	94
Table 23	Summary of Payoff and Efficiency Results for big problems . . . .	95

# Chapter 1

## Introduction

### 1.1 Overview

A task is basically an entity that needs to be performed by satisfying a vector of requirements. The requirements of the task can be of different types and demand levels. This means that tasks vary in their level of complexity. Complex tasks can be performed either by dividing it into simpler subtasks or by joining resources from different entities, usually agents, to perform the task. A task is performed because of the incentive of getting its payment at the end of execution.

Agents are anything with sensors to perceive the environment and actuators to act upon it. Agents are rational in the sense that they always try to do the right thing [1]. In addition, each agent has a vector of resources that should be utilized to perform tasks. These resources are of different types and different supply levels. This means that agents vary in their skill. Each agent endures a cost if it participates in performing a task.

Due to the complexities of today's tasks, single agents can not perform tasks individually. Thus, agents need to get into groups to perform the tasks. These groups/coalitions give rise to the problem of Task Allocation via Multi Agent Coalition Formation (TAMACF). In other words, agents should form optimal coalitions to perform tasks. While solving this problem, three main challenges arise:

- Maximize the efficiency of the Multi Agent System as a whole
- Maximize the payoff of each agent
- Maximize both the efficiency of the Multi Agent System and the payoff of each agent

Accurate modeling of the problem while considering all parameters will help in overcoming the challenges. Before proceeding with the literature review and problem modeling, a brief introduction about agents and Multi Agent Systems (MAS) is provided in section 1.2. After that, the origins of the problem from game theory along with its various definitions is illustrated in section 1.3. Next, the two main domains of the TAMACF problem is explained in section 1.4. Also, several applications are explained for illustrative purposes in section 1.5. Then, the problem statement along with

the objectives is highlighted in section 1.6. Finally, the research significance, research methodology and organization of the remaining report are discussed in sections 1.7, 1.8 and 1.9 respectively.

## 1.2 Agents and Multi Agent Systems (MAS)

In this section, different kinds of agents along with their behaviors are explained. After that, an overview of multi agent systems is provided.

### 1.2.1 Overview

Agents are basically anything with sensors to perceive the environment and actuators to act upon it [1]. Agents could be computers, software, robots or even humans [2]. Agents are the building blocks for multi agent system (MAS). This is because they are considered as the players of the game. Each agent has a set of coalitions to choose from, that is, their available strategies. In addition, each agent gets a payoff by performing a coalition or executing a certain strategy. There are mainly two types of how agents behave:

- **Cooperative Agents** These agents work together to increase the total efficiency of the system [3].
- **Non-Cooperative Agents** These agents act selfish by only maximizing their payoff regardless of the performance of the system. In other words, players can not have agreements before making decisions that shall maximize their own payoff [4].

All agents have a generic skeleton that is basically made up of architecture and program [1].

$$Agent = Architecture + Program \quad (1)$$

The architecture depicts the hardware while the program depicts the software.

### 1.2.2 Types of Environments in which Agents Operate

Agents don't only behave according to their architecture and program. In addition to that, their behavior changes according to the environment they operate in [5]. The following are the different types of environments agents can operate in [1].

**Accessible vs In-Accessible** An environment is called accessible if all aspects in an environment that are relevant to the choice of action can be detected through sensors. Therefore, there is no need to maintain an internal state/memory [1].

**Deterministic vs non-deterministic** An environment is called deterministic if the next state is determinable by the current state and the actions performed by the agent [1].

**Episodic vs non-episodic** An environment is episodic if the agent's experience is divided into episodes. The percepts and actions are only 'this' episodes' related. There is no need to care about the consequences [1].

**Static vs Dynamic** An environment is called static when it doesn't change. This means that there is no need to keep track of the world while deciding on what action to take [1].

**Discrete vs continuous** An environment is described discrete if it has defined percepts and actions [1].

The problem in this thesis considers accessible, deterministic, episodic, static, and discrete environments. In simple terms, this work will assume complete information system with a static behavior.

Collaboration and negotiation between agents are widely studied in the field of Distributed Artificial Intelligence (DAI) [6]. DAI is further divided into Multi Agent Systems (MAS) and Distributed Problem Solving Systems (DPS) [6]. This thesis will focus on MAS systems since these systems consider both cooperative and selfish behaviors of agents [6].

### 1.2.3 Multi Agent Systems (MAS)

Multi Agent Systems (MAS) are computational systems consisting of cooperative or non cooperative agents [5], [6]. These agents might need to communicate, cooperate, coordinate and negotiate to solve given problems [5]. This is often done through forming teams. These teams usually hold the generic principles of game theory like having rational self interested agents [2]. Recently, MAS need to interact with humans which results in relaxing the above principles to allow these systems to account for human behavior and social welfare principles [2].

## 1.3 Game Theory

Game Theory is basically the art of the analysis of conflicting interests in an intelligent way [7]. This analysis can happen between corporations, individuals or smart

agents. It was first introduced in the early 1940s by the Mathematician John von Neumann and the economist Oskar Morgenstan in their famous book "Theory of Games and Economic Behavior" [8]. A game is characterized by [7]

- **Players or Agents:** Those are the entities that make decisions throughout the game.
- **Strategy Set:** It is a set of available strategies available to the player.
- **Strategy:** It is the action selected and performed by a player.
- **Outcome:** The outcome of the game is determined by the strategies chosen by the players. Each outcome is accompanied with a set of payoffs, with one payoff to each player.
- **Player's payoff:** This term represents a numerical value of the payoff gained by a player after the game has been sorted out; that is, after all players have picked their strategies.

Game Theory provides a methodology for each player to maximize its' own payoff taking into account the strategies chosen by other players. In practical terms, it can provide companies with tactical strategies to maximize their profits considering the actions taken by their competitors. So, which strategy maximizes the payoff of the player? The answer to this question is by applying the dominant strategy. This drives us to the definition of dominant/strictly dominant strategy.

**Dominant/Strictly Dominant Strategy** It is the strategy that gives the player his maximum payoff with whatever strategies the other players choose [9]. With this definition, a player can choose his optimal strategy by iteratively eliminating all dominated strategies. This method of solving games is called *iterated elimination of strictly dominated strategies* [9]. However, not all games can be solved using this methodology and thus, Nash Equilibrium needs to be introduced.

**Nash Equilibrium** This term provides us with strategies that give players no incentive to deviate from it [9]. Nash Equilibrium is self enforcing as it maximizes all of the players payoffs [9]. This is also called pure strategy equilibrium [7]. Again, this is not applicable to all games.

There are different strategies applicable to game theory like mixed strategies, minimax, maximin, pure value pair...etc [8]. These strategies will not be discussed because they are outside the scope of this thesis.

### 1.3.1 Types of Games

This section provides a gentle introduction to different types of games.

**Cooperative Games** It is a game where players work together to increase the total payoff of the system [3]. This type of games converges to Pareto optimality, which consists of an outcome that gives a higher payoff to all players than any other outcome [3].

**Non-Cooperative Games** It is a game where players act selfish by only maximizing their payoff regardless of the performance of the system. In other words, players can not make agreements before making their decisions [4].

**Zero Sum Games** It is a game when the sum of all payoffs of all players is zero.

**Non-Zero Sum Games** It is a game when the sum of all payoffs of all players is not zero. This type of game is also called *Variable Sum Game*.

### 1.3.2 N-person Games and Coalition Formation

N-Person games are those that involve  $n$  players, where  $n \geq 3$  [4]. In these types of games, groups of players join together their forces because of a mutual advantage [4]. As a result, players tend to get together and form groups called coalitions [9]. This happens because single agents can not perform tasks alone. In addition, when tasks are performed by groups of agents, the efficiency of the whole system increases and the payoff per player might increase. A coalition structure is a way to describe how the  $N$  players divide themselves into groups [4]. A coalition structure should satisfy three properties which are: a coalition should not be empty, a player belongs to one and only one coalition, and the union of all players in all coalitions constitute the grand coalition [4]. Therefore, a system may have several coalition structures and should therefore pick the one that has the maximum payoff for both the system and the players within the coalitions.

Coalition formation is a subset of  $n$ -person games. The general setting consists of  $N$  players,  $S$  coalitions where  $S \subseteq N$ . Each coalition  $s$  consists of  $n$  players. According to [9], the characteristic function assigns a value  $v(s)$  to each subset of  $N$ . That is, the value function is assigned to each coalition to evaluate it based on a certain function. [4] proposes several properties for the value  $v$ , the relevant ones to this research are:

- A Player prefers more value over less value
- A coalition forms when it's players agree on how they will be dividing the value  $v$  among themselves

- The amount  $v(s)$  doesn't depend on the actions of other members
- The value  $v$  is known to all players in the game
- The value of the empty coalition  $v(\emptyset)$  is 0

After the coalition picks its alternative and gets its value, it is now time to divide the gain among the members of the coalition. This is done using the methodology of payoff distribution. Von Neumann and Morgenstran [10] propose the idea of n-tuple of numbers  $(x_1, x_2, \dots, x_n)$  where  $x_i$  is the payoff of player  $i$ . The properties of the n-tuple are listed below [7]:

- Individual Rationality: This property indicates that a player should get a strictly higher payoff joining a coalition than staying alone. Otherwise, no coalitions will be formed. Mathematically:  $x_i \geq v(i), \forall i$ , where  $i$  indicates a player.
- Collective Rationality: This property indicated that the sum of all payoffs should be equal to the value of the grand coalition. Mathematically:  $\sum_{i \in N} x_i = v(N)$

A payoff of n-tuple is called an imputation if it satisfies the two above conditions [7]. The payoff vector should be pareto-optimal and stable. This means that a payoff vector should give the maximum possible value to players that is not dominated by any other payoff vector. In addition, the payoff vector should ensure stability; so as not to make the players deviate from the coalitions they are part of and move to other coalitions [3]. These solutions can be found by studying the concepts of the core, the shapely value and the kernel.

### **The core**

The core is the set of all un-dominated imputations. It was first introduced by F.Edgeworth [7]. Mathematically: The core is the set of all imputations  $(x_1, x_2, \dots, x_n)$  such that  $\forall S \subseteq N$ ,

$$\sum_{i \in S} x_i \geq v(s) \quad (2)$$

In other words, the core presents the minimal set of acceptable allocations between the players and the tasks.

### **The shapely value**

The shapely value provides a methodology for a fair distribution of the payoffs of the coalition amongst its members. This idea was first introduced by Lyold Shapely in 1953 [7]. In simple terms, the shapely value of player  $i$  is the marginal value that

player  $i$  contributes when the grand coalition forms given all orders in which coalitions can form [7]. That is, it is the marginal value of a player calculated by averaging up his value out of each and every coalition. This will help players determine how much they will benefit from participating in a coalition.

### **The Kernel**

The kernel provides stability for a coalitional configuration [11]. The kernel doesn't actually provide a methodology to calculate stable payoffs. However, it acts as a test to ensure that a certain coalitional configuration and a payoff vector are stable [11]. For example, coalitional configuration and a payoff vector will be proven stable by the kernel if two agents who are equivalent can not out weight one another from the coalition they belong in [11]. That is, it determines the set of payoff vectors where no player can have powers over another player.

## **1.4 Task Allocation and Coalition Formation**

Task allocation means that tasks should be allocated to agents for execution. Single agents can not perform tasks alone, and if they do, they are not done efficiently. Therefore, agents should join resources in groups/coalitions to perform tasks. This is widely known as coalition formation. Multi Agent coalition formation has two different environments in which they can operate; super additive environments versus non super additive environments.

### **1.4.1 Super Additive Environment**

Super additive environment means that the value of two groups combined is greater than the sum of both groups [4]. For example, let's assume that we have two coalitions  $T$  and  $X$ . The super additivity is illustrated mathematically as follows:

$$V(T \cup X) \geq V(T) + V(X) \quad (3)$$

This indicated that groups with more members gain more than groups with less members. This eventually results in forming the grand coalition [4].

### **1.4.2 Non-Super Additive Environment**

On the other hand, non super additive environments do not imply an increase in value by having more members in the coalition. This is usually important when the

"cost of cooperation is an ascending function in the number of cooperating agents" [11]. Moreover, this environment assumes dropping the property of collective rationality. Non-super additive environments are more practical since the problem emphasizes on forming teams of agents and allocating tasks to them.

## 1.5 Problem Solving Methodologies

Task Allocation via Multi Agent Coalition Formation problem highlight two main challenges:

- To decide which coalitions should form . This factor should be driven by maximizing the efficiency of the whole system.
- To structure a methodology of how to divide the coalition's gain among all players. This factor should be driven by maximizing the payoff of the player [7].
- To model which coalitions should form highlighting both maximizing the efficiency of the system and the payoff of each agent.

This problem can exist in both small/medium and big sized problems. For small problems, exact algorithms through optimization techniques can be modeled to achieve the above mentioned objectives. However, this problem is proven to be of NP hard [12]; that is, they are decision problems that can not be solved in a polynomial time [13]. Therefore, several proposals were made to solve the problem. The meta-heuristics approach is the most widely used one because it gives approximate solutions in an acceptable time [14].

### 1.5.1 Meta Heuristics

**Genetic Algorithms** Genetic algorithms is one of the most popular meta heuristics. They are based on Darwin's theory. This means that they act like biological processes. That is, the population is represented via chromosomes. These chromosomes are usually binary encoded. The chromosomes of each population are evaluated through a fitness function and the good ones are selected for reproduction. Reproduction in Genetic Algorithms is done through two main operators; cross over and mutation. Cross over means recombination of information from two parents and encapsulating them in the child. Mutation means modifying the child to further improve its fitness. The algorithms iterates over the population through reproduction operators and stops when the stopping criteria is met [15].

### **Simulated Annealing**

Simulated Annealing is based on statistical and optimization measures. It simulates the process of gradually cooling down a metal from an initial temperature until it reaches the global minimum. Therefore, the system starts by specifying the initial temperature and population. After that, the fitness of each member is evaluated. Then, selected candidates will move to a neighboring solution and the system is evaluated again. The temperature then keeps decreasing until it reaches global minimum [15].

### **Tabu Search**

Tabu Search is a global optimization meta heuristic that allows a move out from the current solution to a neighboring solution in a hope of finding better solutions. Some moves are restricted to avoid blind search. The moves will be repeated until the termination criteria is satisfied [15].

### **Ant Colony**

Ant Colony meta heuristic follows the behaviors of ants searching for their food. Ants search for their food randomly. Once they find food, they leave a chemical named pheromone to track their path. The next batch of ants going to search for food will follow the pheromone to guide them for shorter paths. This pheromone tends to evaporate quickly. However, the more ants pass over it, the more dense it gets thus eventually guiding to the best solution [15].

It is important to consider several factors when choosing the algorithm:

- Time: time consumption
- Space: Memory consumption
- Optimal: Strategy guaranteed with finding highest quality solution
- Complete: Strategy guaranteed to find a solution

In this thesis, we will consider genetic algorithms as it encodes the problem in a natural way similar to the TAMACF formulation. In addition, it is proven to give acceptable solutions in an acceptable time. Moreover, there are no memory requirements [15]. Most importantly, this algorithm was chosen due to the availability of a rich literature in several implementations via Genetic Algorithms [15].

## 1.6 Problem Statement and Complexity

In this thesis, we will discuss the problem of multi objective task allocation via multi agent coalition formation (MO-TAMACF). The general formulation of the problem considered in this thesis consists of:

- M candidate tasks  $T = t_1, t_2, \dots, t_M$
- A set of K requirements for each task  $t_i$ ,  $D = d_i^1, d_i^2, \dots, d_i^k$
- A payment per task that will be given out to the performing coalition if the task is completed  $P_i$
- N agents  $A = a_1, a_2, \dots, a_N$
- A set of K resources for each agent  $a_j$ ,  $S = s_j^1, s_j^2, \dots, s_j^k$
- A cost  $C_j$  that will imposed if the agent participates in performing a task

These parameters shall guide us to formulate the problem. This problem highlights three main objectives:

- Objective 1: Defining and maximizing the efficiency of the system
- Objective 2: Defining and maximizing the payoff of each agent
- Objective 3: Defining and maximizing the efficiency of the system and the payoff of each agent

These objectives correspond to different agent's behaviors. The first objective assumes cooperative agents, the second objective assumes selfish agents, and the third objective assume both cooperative and selfish agents.

## 1.7 Research Significance

Surprisingly, the (TAMACF) problem was not posed in the literature in this way. In addition, no one has considered agents with negative capabilities which is very much similar to real cases where some agents can contribute negatively to coalitions.

In this thesis, we address the multi-objective task allocation problem. We consider a set of agents each with a vector of limited and/or negative resources that shall be cooperating in order to perform a set of tasks. The cooperation leads to a numerous number of coalitions. In addition, We consider three objectives; maximizing the

efficiency of the system, maximizing the agent's payoff, and maximizing both the efficiency of the system and the payoff of the agents. These three objectives represent the different types of agent's behaviors; cooperative agents, selfish agents and both cooperative and selfish agents. To solve this problem, we design a genetic algorithm approach and we compare the obtained results with those obtained by an exact algorithm for small sized problems. In addition, we will apply our model to an Agile application, a software development methodology, to solve some of its highlighted challenges in the literature.

## **1.8 Research Methodology**

The research methodology to achieve the objectives is to formulate, run and test an optimization model to solve the problem and achieve the objectives mentioned above on a small scale. As for large scaled problems, we shall use Genetic Algorithms that are proven to give good approximated solutions in a reasonable time. We will test the results of the algorithm against exact solutions for small problems. After that, we will conduct a set of experiments to find out whether cooperative, selfish or hybrid agent's behavior is proven to be more beneficial. Finally, we will gather the main problems in Agile Software Development methodologies from the literature and try to customize our model in a hope to solve these problems.

## **1.9 Thesis Report Organization**

This thesis will proceed with literature review in coalition formation in chapter 2. After that, literature review in coalition formation applications in the Information Technology domain will be illustrated in chapter 3. After that, our methodology for both cooperative, selfish, and hybrid (cooperative and selfish) agents will be presented in chapters 4, 5, and 6 respectively. Finally, the conclusions and recommendations will be summarized in chapter 8.

## **Chapter 2**

### **Literature Review**

Several papers tackle different aspects of coalition formation. This section is divided as follows: Related literature for maximizing the efficiency of the system and related literature for maximizing the payoff of the agent. In addition, some papers discussed coalition formation taking into account different aspects like multi objective applications, learning capabilities by agents, and skill uncertainty of agents. Within each of these sections, the literature is further divided chronologically; that is, from the oldest to the most recent or divided by topic relevance.

#### **2.1 Cooperative Agents - Maximizing the efficiency of the system**

In this section, coalition formation with cooperative agents is discussed where the main objective is to perform tasks while maximizing the efficiency of the system. Several papers discuss this issue where they vary the definition of efficiency then provide and apply algorithms to solve the problem.

According to [16], It is usually important to group agents together when a single agent cannot perform a task or when it performs the task inefficiently. The challenge here is to assign the right group of agents to the right task in respect to the capabilities and costs of the agents. The authors propose a solution concerning coalition formation for non-supper additive environments. They approach their solution by combining combinatorial algorithms along with graph theory. More importantly, their system is not centralized; thus, agents try to achieve an efficient solution by themselves. This is important to minimize the complexity of the system.[16] make the following assumptions:

- Agents can communicate, negotiate and transfer resources.
- Coalitions work on a single task at a time
- Each agent is allowed to join only one coalition
- Agents are coalition-ally rational. This means that they don't join a coalition unless they benefit from it as much as they would benefit if they were left alone.
- Agents join coalitions that will increase the overall value of the system. This doesn't mean that it is a supper additive environment.

The authors formulate their problem as  $M$  candidate tasks  $\mathcal{T} \{T_1, \dots, T_M\}$ . Tasks are expressed as  $T_j$ . Each task has a vector of capabilities  $B_j = (b_{1j}, \dots, b_{rj})$ . There are  $N$  agents  $A \{A_1, \dots, A_N\}$ . Agents are expressed as  $A_i$ . Each agent has a vector of abilities  $B_i = (b_{1i}, \dots, b_{ri})$ . There is an evaluation function that is attached to each type of capability. Such a function shall transform the capabilities into monetary terms. A group of agents may form a coalition  $C$ . The coalition  $C$  sums up all ability vectors of its agents  $B_c = \sum_{A_i \in C} (B_i)$ . This is also considered and named the Value  $V$  of the coalition. A coalition can only perform a task when its ability vector is more than or equal to the requirements vector of the task  $\forall 0 \leq i \leq r, b_i^j \leq b_i^C$ . The cost of the coalition is simply the reciprocal of the value of the coalition. This means that increasing the value automatically decreases the cost.

The authors attempt to solve their problem using set covering and set partitioning. Lets assume that there are  $N$  agents  $A \{A_1, \dots, A_N\}$  and a set  $S$  which is a subset of  $N$  where  $S = \{C_1, \dots, C_m\}$ . A set cover  $S'$  is basically the union of all subsets in  $N$ . This is mathematically expressed as follows:  $\bigcup_{C_j \in S'} C_j = N$ . The cost of a set cover is assumed positive:  $\sum_{C_j \in S'} c_j$ . On the other hand, a set partitioning problem is mainly pairwise disjoint. That is, there are no common members between two different coalitions. The set covering problem is NP-Complete. Many heuristic based algorithms have been proposed to solve this problem. The authors pick Chavatal's algorithm, a logarithmic ratio bound algorithm. Some of the properties of the algorithm include:

- greedy distributed set partitioning algorithm with a low ratio bound
- It is an any-time algorithm. This means that if the algorithm is stopped before it is complete, it is guaranteed to give a solution that is better than the initial setting.

To further minimize the complexity of the problem, the authors introduced the following heuristics

- Coalitions shall avoid communication and computational cost and complexity. Therefore, only small sized coalitions shall be formed. This will be implemented by introducing an integer 'k' which denotes the maximum size of a coalition.
- Iterative greedy process where agents decide on which coalitions they want to join and form them
- The benefits of the cooperation may be distributed.

The algorithm has three main stages:

- Stage One: Coalitional values shall be calculated.
  - Calculate all of possible coalitions up to size  $k$  in which you are a member. Form a personal list of coalitions.
  - For each coalition, contact each member and record their capabilities. Construct a personal list of agents and avoid redundancy.
  - publicize that you are in charge of value calculation for coalitions that you and the contacted agents are members of.
  - If you were contacted by an agent, erase all common calculations from your personal list

To calculate the values

- check which capabilities are necessary to perform the task
  - calculate the expected outcome of the task; Monetary outcome = Monetary sum of coalitions capabilities - Monetary sum of task capabilities
- Step Two: Iterative greedy process where agents decide on which coalitions they want to join and then form them. Agents that join coalitions quit the coalition formation process and only single agents shall be left.
    - The weight of the coalition is calculated  $w_i = c_i/|C_i|$  where  $c_i$  is the cost of the coalition and  $C_i$  is the size of the coalition.
    - Each agent will choose the coalition with the lowest weight from its list
    - Each agent will announce the weight of its chosen coalition and the lowest among these will be chosen by all candidates
    - Committed agents will be deleted from lists

This is done until all agents are deleted from the lists. It is worth noting that the coalition value is always re-calculated as the configuration is changed due to different assignment of the tasks. The only value that changes is the coalition value. All other values remain unchanged.

- Step Three: the benefits of the cooperation may be distributed if necessary. This was outside the scope of their paper.

This paper present a low logarithmic ratio bound to solve the problem. The average computational complexity will be of order  $O(n^{k-1})$  while communication complexity will be of order of  $O(n.m)$ .

In [17], the authors start by differentiating between cooperative settings and non cooperative settings. Cooperative agents are utilized using a central controller who is in charge of the assignment. Moreover, their costs and capabilities are revealed. However, for non-cooperative settings, despite the fact that the agent's capabilities are still known, the controller can not assign agents unless they express their willingness. This means that non-cooperative agents can create a strategy to pick up their preferred tasks to maximize their payment. In their paper, the authors assume cooperative behavior of agents. First, they assume that each task is composed of several subtasks. Moreover, every task has a predetermined payment set by the owner. When all the subtasks of the task are performed, the task's payment shall be fully paid out. These subtasks are assigned to agents who are capable of performing it. Moreover, the sum of their costs shall be less than the tasks's payment. An example in real life could be in the form of an organization that receives a set of projects to be assigned to departments. The organization knows the capabilities and costs of each department. Thus, this problem does exist in the practical life. The algorithm proposed in this paper, for cooperative setting, is exponential in number of tasks and polynomial in number of agents. As number of tasks is usually small, the algorithm is assumed polynomial.

A typical problem involve M candidate tasks  $\mathfrak{T}\{T_1, \dots, T_M\}$  with a predefined payment  $P(T_i)$ . Subtasks are expressed as  $ST_{i_l}$  where i indicates the task's index and l indicates the subtask's index. There are two important conditions for a task to be considered complete:

- All subtasks of that task should be performed
- The payment of the task should be fully paid out

These subtasks are assigned to a set of N agents  $A\{A_1, \dots, A_N\}$ . Agents are expressed as  $A_j$ . An agent can perform a subtask if the knowledge capability is one. Knowledge capability  $\varphi(A_j, ST_{i_l})$  define if a certain agent can perform a certain subtask. It is one if true and zero otherwise. Using this, a summary of agents and subtasks can be formalized. Agent's capabilities, or subtasks that they can perform, are expressed as follows  $\psi_j = \{ST_{i_l} | \varphi(A_j, ST_{i_l}) = 1\}$ . Consequently, Subtask's performers or agents are expressed as follow:  $\Omega_{i_l} = \{A_j | \varphi(A_j, ST_{i_l}) = 1\}$  In addition, each agent has a cost to perform a subtask  $ST_{i_l}$ . This cost is expressed as follows:  $C_{i_l}^j$ . An allocation is

defined as a match between agents and subtasks. This is expressed as follows:  $\Phi = \{(A_{i_1}, ST_{i_{1l}}), \dots, (A_{i_n}, ST_{i_{nl}})\}$ . Here,  $\Phi_T$  indicates the set of tasks in the allocation  $\Phi$ , where  $\Phi_T \subset \Phi$ . An allocation is feasible if:

- Each agent in  $\Phi$  is allocated to at most one subtask.
- Any task is either fully allocated or not allocated.
- The total cost of the agents performing the task should not exceed its payment

To have a quantitative analysis, the authors assigned a value to the allocation function,  $V(\phi)$ .  $V(\phi)$  is the difference between the total payments of the tasks and the total costs of the agents. In mathematical terms  $V(\Phi) = \sum_{T \in \Phi_T} P(T_i) - \sum_{(A_j, ST_{i1}) \in \Phi} C_{i1}^j$ . The feasible allocation with the highest  $V$  is called  $\Phi_{eff}$ . Consequently, the efficiency of any solution could be defined as  $efficiency(\Phi) = \frac{V(\Phi)}{V(\Phi_{eff})}$ .

The complexity for the above formalization is NP hard. Solution for big problems will be based on the algorithm of finding the minimum weighted perfect matching in a bipartite graph. Each of the agents and the subtasks are defined using vertices in  $V$  such that the capability of an agent to perform some subtask is indicated by an Edge  $E$  between the corresponding vertices. The agent's cost to perform some subtask is presented as a weight on the edge  $E$ . Obviously, the sum of the edge's weight should be minimum  $G = (V, E)$ . Accordingly, given a set of tasks  $\mathfrak{S} = \{T_1, \dots, T_M\}$ , they define the power set of  $\mathfrak{S}$  to be  $\Lambda = \{\lambda | \lambda \subseteq \mathfrak{S}\}$ . For each  $\lambda$  in  $\Lambda$ , they construct a bipartite graph  $G_\lambda = (V_1, V_2, E)$  where  $V_1$  represent all the subtasks, ST, associated with one task in  $\lambda$  and  $V_2$  corresponds to all agents in  $A$ . If the number of vertices in  $V_2$  is greater than the number of vertices in  $V_1$ , the algorithm extends the size of  $V_2$  by adding  $|V_2| - |V_1|$  that are connected to all vertices in  $V_2$  with a weight of 0. Their algorithms, therefore, is stated as follows: For each  $\lambda$  in  $\Lambda$  do:

- Build a bipartite graph,  $G_\lambda = (V_1, V_2, E)$
- Find the minimum weighted perfect matching  $M_\lambda$  in  $G_\lambda$ .
- check whether  $M_\lambda$  is feasible
- if  $M_\lambda$  is feasible, compute its allocation value  $V(M_\lambda)$

Finally, they return the most efficient allocation,  $M_\lambda$ , which is a feasible solution with the highest value,  $V(M_\lambda)$ .

This paper present a solution to allocate complex tasks. They come up with an algorithm that is exponential for the number of tasks and polynomial for the number of agents. Moreover, they mathematically proved that the algorithm always produces an efficient solution. In [18], the authors aim at partitioning a group of agents to perform activities and gain more. For example, in e-commerce systems, buyers get together to guarantee a group discount. According to the authors, coalition formation impose three main steps:

- Dividing the agents in the system into exhaustive disjoint groups
- Optimize the value of each coalition by pooling the resources of each member into the coalition
- Dividing the value of the coalition among the agents according to some method

The authors claim that coalition generations discussed in the second step is NP-hard due to the exponential number of groups that need to be examined. Therefore, they start by formulating their problem as a set of Agents  $A$  where  $n$  is the number of agents. Next, they define coalitions as subsets of agents. Each subset is associated with a value  $V$ . Moreover, they define the value of the coalition structure as the sum of all values of coalitions in the structure.

To solve the problem, the authors present an anytime algorithm to solve big sized problems. They start by defining  $L$  as the set of all coalitional structures with size  $k$  in the system. They only examine a subset of  $L$  based on the cardinality. Their results show that their solution is within a finite bound from the optimal.

[19] discusses coalition formation for cooperative agents. They claim that in order to make multi agent systems optimal, they shall consider the possibility of the majority of coalitions combinations. This means that it is a complex combinatorial problem. Optimization problems usually involve maximizing or minimizing a fitness function. OPT: Minimize/Maximize  $f(x)$  with respect to  $x$  in  $S$  Where  $x$  is a vector of parameters and  $S$  is the solution's space. The fitness function is usually given, but it is too complex to solve. Therefore, the authors shall apply genetic algorithms. Genetic algorithms shall involve three main things:

- Stochastic Choice: Enable the best individual in the community to have more opportunities and to give it to younger generations.
- Cross Over: It is the exchange of information.

- Mutation: It introduces new characteristics to the community.

A typical problem would involve  $M$  candidate tasks  $\mathfrak{S}\{T_1, \dots, T_M\}$ . Tasks are expressed as  $T_j$ . Each task has a vector of requirements  $S_j = (s_{1j}, \dots, s_{rj})$ . There are  $N$  agents  $A\{A_1, \dots, A_N\}$ . Agents are expressed as  $A_i$ . Each agent has a vector of abilities  $B_i = (b_{1i}, \dots, b_{ri})$ . A group of agents may form a coalition  $C$ . The coalition  $C$  may sum up all ability vectors of its agents  $B_c = \sum_{A_i \in C} (B_i)$ . A coalition can only perform a task when its ability vector is more than or equal to the requirements vector of the task  $\forall 0 \leq i \leq r, r_i^j \leq b_i^C$ . Therefore, the authors introduced a value function which measures the efficiency of the total (MAS) system. The value function is positive when the equation above holds. The total efficiency of the MAS system is presented as follows:  $V = \sum_i (V_i)$ . The rules for agents to form coalitions are:

- An agent is allowed to join a coalition only if it increases the efficiency of the MAS system.
- Once an agent joins a coalition, it is not allowed to join any other coalition.

Genetic algorithms are basically based on Darwin's theory. This means that they act like biological processes. Below are the step of Genetic Algorithm:

- Generate an initial solution
- Evaluate the current setting
- Crossover the best individual together to get new children
- Make random mutation over the children
- Go back to second step

The authors start by creating a novel 2D chromosome encoding. Basically, tasks are represented in columns while agents are represented in rows.  $a_{ij} = 1$  means that Agent  $i$  in Coalition  $j$  is involved in task  $j$ . After that, the authors apply the cross over operator. It is basically a bit wise OR operation between two parents. At the end of this process, the children shall be a valid one. After that, the mutation operator is applied. Basically, it is based upon the idea of two coalitions exchanging one respective member. At the end, some final touches are applied to make sure of the efficiency of the MAS system. Basically, agents are divided into legitimate and non legitimate. Legitimate

individuals are those who can perform all tasks while non-legitimate agents are those who can not perform all tasks. As a result, the algorithm attempts to exchange agents within coalitions to make sure that every coalition can perform every task.

The authors test their algorithm using 3 groups with 10, 20, and 30 agents respectively. Number of tasks were between 4 and 5. After testing, they concluded that the convergence rate of the algorithm in this paper is fast and that the robustness is strong.

[20],[21], and [22] attempt to solve the problem in a similar behavior by applying meta-heuristics, specifically genetic algorithms.

[20] modifies the previous approach of [19]. First, they start by listing their main assumptions:

- Each agent has a vector of resources
- Each task required a vector of requirements to be satisfied
- A single agent may not have enough resources to perform tasks individually
- Agents are cooperative and can solve a finite number of tasks
- Each agent can join only one coalition

The authors solve the problem via evolutionary algorithms by only applying a simple one dimensional array to represent their chromosomes. To evolve their population, they utilized two point cross over operator and a mutation operator with a probability of 0.01. In addition, the fitness of each member is evaluated via the number of tasks performed. That is, an individual who solves more tasks is preferred as they aim at maximizing the number of tasks performed. They prove that their algorithm gave no worse than 80 percent of optimal solution.

Another attempt to solve the problem via genetic algorithms was done by [21]. Their algorithm hold three main assumptions; agent's rationality in trying to increase the welfare of the system, central decision making mechanism, and non super additive environments. They encode the chromosomes using a one dimensional array where the index represent the agents and the values represent the tasks they are assigned to. They evolve the population where each chromosome is evaluated by summing up the coalitional values in it. The coalitional value is calculated by how important the task performed is, and the cost of the coalition. In addition, the population evolves via the

uniform cross over operator and the swap mutation operator. Their approach is 100 percent correct for small problems while 56 percent correct for big problems.

[22] illustrates a different way other than meta heuristics to solve the problem of coalition formation. They distribute the coalitional values calculations among all agents thus utilizing the resources of the system, saving more memory and requiring less communication. Their main idea is to divide all possible coalitions equally among the agents. They test their algorithm with 25 agents and the results were found in less than 0.02 percent of the time with less than 0.000006 percent of memory use.

## 2.2 Selfish Agents- Maximizing the payoff of the agent

On the other hand, some papers focused their attention on selfish agents. These agents aim mainly at increasing their own payoff regardless of the performance of the system. Different authors tackled the question of "solving optimization problems among selfish agents". In this section, the literature discussing maximizing the payoff of the agents is discussed. Several approaches are presented below.

Shehory et al [11] propose a feasible coalition formation among autonomous agents in non-super environments. They make several assumptions which are:

- The environment is a complete information system;i.e Information about agents abilities and payoff functions are accessible to all other agents
- Communication between agents is allowed. However, it is expensive.
- Resources and payoffs are transferable between agents.

The main objective of their solution is to find a pareto optimal solution of payoff vectors that are stable.

They provide two approaches to solve the problem; a computational approach and a negotiation approach. The computational approach searches an exponential space and leads to forming stable coalition with maximal individual payoffs. Their second approach leads to stability, and efficiency of coalition formation process. These approaches are discussed below:

**The computational approach** Using this approach, the author suggest a kernel-based coalition formation framework to minimize complexity of computation and communication overheads. A summary of the steps are shown below:

- Each agent should calculate the coalitional values for the coalitions in which it is a member. After that, it should transmit these values to other agents.
- Each agent is assigned a random variable  $z_i$  using a distributed random method
- Each agent should calculate  $z_i$  coalition configurations
- Each agent should find a stable payoff vector for each coalitional configuration
- Each agent should make a list of personally Pareto optimal k-stable payoff vectors. Here, k indicates the size of the coalition.
- All agents should merge their personal lists
- The agents should choose one of the coalitional configurations using a decision making technique

This algorithm is of exponential complexity

#### **The negotiation approach**

Another methodology to solve the coalition formation problem is by using negotiation among agents. This is illustrated in the summary below:

- Each agent should calculate coalitional values
- Each coalition will coordinate its actions through a leader
- For each coalition, do:
  - Transmit a proposal to another coalition to join forces
  - The receiving coalition either accepts or rejects the proposal
- Steps above are performed until a steady state is reached.

This algorithm is of a polynomial complexity.

Philip, Samir et al [23] talk about how to form coalitions which find a pareto optimal solution without aggregating the preferences of the agents. Moreover, they continue by discussing how to dynamically restructure the coalitions to meet the changes in the environment. Philip and Samir focus mainly on complex tasks where coalitions are needed to accomplish them. Also, they focus on dynamic environments where tasks may be added, canceled or modified constantly. Moreover, they mainly consider that the

choice of an agent depends mainly on the members of the coalition it is in. However, externalities should be stable during the negotiation process. They go on with their paper and give the following definitions:

- **COALITION:** A coalition is formed for each task. Each coalition shall achieve a task.
- **COALITION SET:** It represents a solution to the problem of coalition formation.
- **GROUP OF COALITION SETS:** It is a group of coalitions brought together in order to be computed and transmitted collectively.
- **CONTEXT:** A set of unspecified parameters that should be stable during the negotiation process.
- **UTILITY FUNCTION:** If cardinal, it associates a utility with a set of coalitions within a given context. If it is ordinal, it compares two sets in a given context, maybe comparing against the reference situation.
- **REFERENCE SITUATION:** For the agents to know if they have to accept a set of coalitions as solution, they need to be able to compare it with what they are sure to obtain during the negotiation.
- **ACCEPTABLE SET:** a set is acceptable to an agent if it prefers it over another situation
- **PARETO OPTIMAL:** It is a situation where it is impossible to improve the situation of the agent.

Philip and Samir propose a negotiation algorithm to come up with a pareto optimal solution.

**Phase 1: initialization** When a new task appears, or when an agent changes its preference, the initiator agent notifies the others that it will start a negotiation process. Other agents that want to start negotiation should then wait. The initiator asks each other agent to send it their tasks. The agent then deduces the set of tasks that need to be performed and associate it to coalitions. At the end, the initiator send the allocations to other agents that shall start negotiations.

**Phase 2: Negotiation** When an agent receives the set, it sorts the groups of sets according to preferences. Each agent keeps on sending to the next agent until they all finish.

phase 3: **Transmission of Solution** The last agent sends the pareto optimal solution to other agents as the solution for the negotiation.

Finally, when there are dynamic changes in the environment, the agents should start from the current optimal solution. According to [23], Coalition formation is one type of coordination when it comes to bargaining. The authors claim that peer coordination provides better results than a random model. Moreover, they claim that "it is one way to economize on transaction costs"

Another approach for non-cooperative agents in coalition formation is discussed by Arib et al [24]. They introduce a plan for each agent. A plan is basically a set of ordered actions to achieve a certain goal. The plans of the agents are presented using acyclic graphs. Acyclic graphs consist of nodes that are connected via and/or edges. Here, the nodes will present actions. Because the outcome of the agent does not only depend on the actions of this agent, but on other agents as well, there is a global inter-agent dependence in the system.

They formulate their problem as a set of agents  $N = a_1, a_2, \dots, a_n$ , a set of plans  $P = p_1, p_2, \dots, p_K$  where each plan contains a set of actions  $A = b_1, b_2, b_3, \dots, b_m$ . In addition, each action is associated with a utility  $u_i$ . Also, they assume that agents prefer higher utility coalitions. Moreover, they define a coalition structure as a subset of  $N$ .

Their proposal is composed of two main steps: coalition search and negotiation process algorithm.

- Coalition search: In this step, each agent will sort its actions in a decreasing order of preference to reach a certain goal. To do this, the authors first introduce the concept of desirability. This means that actions are sorted based on their relevant importance to other actions in the system. Each agent will form coalitions based on its preferences. It then takes the coalition structure proposal to the negotiation process.
- Negotiation Process: The initiator broadcasts a coalition structure based on the actions sent by other agents. The receiving agents will either accept, if the utility of the coalition structure is at least as good as its reference point, otherwise it rejects. At the end, when the agents agree, the last agent in the process will send to the initiator who will then broadcast the results.

The authors here try to maximize the efficiency of the system which is defined by maximizing the number of performed tasks. The main constraint in this algorithm

is that agents should join forces with those that will help them perform their preferred tasks.

### 2.3 Other Aspects in Coalition Formation

In addition to the discussion above, some papers tackled different issues in coalition formation.

For example, [25] presents an application of multi-objective coalition formation of ATMs in banks. Banks operate mainly using the concept of ATMs network; this network includes their ATMs and other bank's ATMs. This network helps their customers gain access to more machines. Moreover, it is used as a sort of back up in case all ATMs of this bank fails. Due to the above mentioned benefits, banks decide to create a network of ATMs. In addition, banks introduce interchange fees whenever a customer uses the ATM of another bank to count for imbalance in the accounts. This means that, bank  $i$  will pay a fee  $f_{i,j}$  to bank  $j$  whenever a customer of bank  $i$  uses the ATM of bank  $j$ . The authors propose a model for ATM networks in the form of multi-objective game involving two main objectives:

- The cost whenever a cash is withdrawn by the customers of the bank
- The quality of the transaction which is a measure of how many ATM machines are available

Therefore, the main objective of this paper is to solve the problem of multi objective transferable utility game to model the problem of ATM network efficiently. A multi-objective game is a pair  $(N, V)$  where  $N$  represents the number of players and  $V$  is the value function.  $V_S$  is the characteristic set assigned to coalition  $S$  where  $S \subseteq N$  and  $V_S \subseteq R^k$  such that  $V_\emptyset = 0$ . In words, vector  $V_S$  is the vector valued utilities that the members in coalition  $S$  can guarantee by themselves.  $K$  denotes the number of objectives. Non-dominated points in  $V_S$  are defined by:  $E(V_S) = \{a \in V_S, \text{such that } \nexists b \in V_S, a \leq b, a \neq b\}$ . Vector valued games can be seen as multi-objective games with  $|V_S| = 1 \forall S \subseteq N$ . Therefore, the maxima of the characteristic set of the multi objective function can be defined as  $|E(V_S)| = 1 \forall S \subseteq N$ . The payoff  $z \in E(V_N)$  is divided by players  $N$  who decided to operate in the form of a 2D payoff matrix. The allocation based on the criteria is presented by  $k$  rows and the players are presented by  $n$  columns. Obviously,  $X_S$  is the total payoff obtained by coalition  $S \sum_{i \in S} X^i \in R^k$ . In this paper, the authors will define

an allocation of all  $k \times N$  such that  $X^N \in E(V_N)$  and this will be denoted by  $I^*(N, V)$ . To get to the result of non-dominated allocation, two different solution concepts exist:

- "We do not admit less worth, componentwise, than all the utility vectors that we already can guarantee by ourselves"
- "We accept compromise payoffs which are not in the set of vector that we can guarantee by ourselves"

Their model aim to solve mainly two contradicting objectives: Firstly; The costs involved in cash withdrawals by the customers of the organizations when they have a common network. Secondly; The quality of the service offered by the ATMS which is basically the number of ATMs. The notation for the example is as follows:

- Number of organizations  $1, \dots, m$
- $S$ : any subset of  $N = 1, 2, 3, \dots, n$
- $d_j$ : is the fixed annual cost of operating one ATM by bank  $j$
- $t_j$ : is the total number of ATMs owned by bank  $j$
- $C_{ij}$  is the variable cost of a transaction by a customer of bank  $i$  using the ATM of bank  $j$
- $n_{ij}$  is the number of transaction by customer  $i$  on ATM of bank  $j$
- $K$ : it is the average cost of cash withdrawals by means other than ATM
- $m_j$ : it is the number of cash withdrawals by non ATM means of customer of bank  $j$

Therefore, the total cost, the first objective, is:  $C = \sum_{j \in N} R_j = \sum_{j \in N} (d_j * t_j + m_j * k + \sum_{i \in N} c_{ij} * n_{ij})$ . The second objective, quality of ATM, is:  $A = \sum_{i \in N} t_i$ . If bank  $j$  decides to leave the network, the customer of bank  $j$  still in the network would either like to use other ATMs in the network or use non-ATM means. Therefore, coalition  $S$  can increase their ATMs by a fraction  $\beta(s)$ . This is a fraction of transactions on non- $S$  ATMs which are performed by customers of  $S$ . However, as this might not always be possible neither desirable, the authors will try to achieve a fraction of it defined by  $0 \leq \lambda \leq 1$ . The idea is to increase the number of ATMs as not to make the customers

feel any difference in the quality of ATMs offered by the coalition. Therefore, the number of ATMs in a coalition should be defined as:  $t_\lambda(s) = \sum_{i \in S} t_i + \lambda * \beta(s) \sum_{j \ni S} t_j$ . The characteristic function for the coalition  $S$ ,  $v(S)$  should consider both objectives. The equation shall be written as follows:  $V_s = (C_\lambda(S), t_\lambda(S)) \in R^2$ . Because both objectives are contradicting, the cost will be denoted as a negative vector. The authors conclude their paper by applying their model in a four bank network in UK and they prove minimizing the average interchange fee paid by each bank.

Another approach by [26] discusses agents with multi skills where each skill is valued differently. In addition, the skills are uncertain and an agent do not know about other agent's skills. However, each agent has a belief space about the skills of other agents. Therefore, the coalition value is calculated differently for each agent as the value function is dependent on the belief space. In specific, the value function is the belief space in terms of it's probability. The authors proceed by explaining their algorithms which is a repeated formation of coalitions by agents. At each period, the proposer agent can propose staying in the current coalition, leaving the current coalition to join another coalition, or leave the current coalition to stay alone. At the end, all the affected agents must agree for the change to happen. Their experimental results show that environments with perfect information produces best coalitions in terms of values.

Alternatively, [27] proposes reinforcement learning and tasks performing in parallel by making groups of agents in coalitions. Coalition formation in this paper is based on Markov decision processes. Their model evolves mainly on the reward and probability from moving from state  $s$  to state  $t/s$  via performing action  $a$ . A state is defined as a set of resources by agents and a set of tasks selected to be performed. Also, an action  $A$  is defined as utilizing a specific resource of an agent. In addition, the authors apply reinforcement learning which is a methodology that relates actions to states via trial and error in an attempt to reach optimal state. Their model assumes that the current state highly depends on the previous state. At the end, they testify their markov decision making capabilities in coalition formation theoretically.

There are several applications for coalition formation in real world. For example, it is used in the fields of economics, game theory, mathematics, politics [4]. This topic is specially popular in the field of computer science as it has applications in parallel processing, network modeling, disaster response applications, and learning agents. Moreover, this area of research aims at having better interactions between software agents. For example, In e-commerce systems, buyer and seller agents can form coalitions based on trust and witness reputation that can be used in virtual organizations

[28]. Similarly in e-commerce systems, by joining coalitions, buyers can buy at a lower price and sellers can sell at a higher price [14]. This section will highlight some of the main applications in the IT field and in Software Engineering Methodologies for Software development.

Task Allocation via Coalition Formation can be realized in e-commerce systems. [29] proposes coalition formation in electronic commerce systems via genetic algorithms. They state that the buying power of a single buyer is limited, and if they coordinate, they can realize the same value for a cheaper cost. The main challenge here is which buyers should get together to find the global minimum cost. In their paper, they assume different buyers with different shopping lists shopping from different sellers with different retail prices. Forming groups between buyers will help them realize discount rates for high cost transactions. One might think that all buyers will all join together to get the maximum discount rate. However, this is not applicable as different buyers have different shopping lists. From experimentation results, they prove that their algorithms is feasible and effective to find optimal solutions.

Nematbaksh et al [30] address a greedy model for large complex cooperative multi agent systems. They adopt an organizational model because they aim at making (MAS) systems act like intelligent human systems. In this model, each task is performed as a whole and not decomposed. In addition, each agent has capabilities and roles to achieve some goals in a structured way. For example, in their model, they assumed two roles for agents; Supervision role and Operation role. The main objective of the model is to try to tackle as many tasks with minimal use of resources. Their solution adopts Schwaninger's model which is an organization model adopted for cooperative agents. It consists of three main parts; Structural model, activity model, and behavioral model.

- Structural Model: It shows the overall design of the organization. They adopted a team-based model organization. First, the organization starts with an initial structuring. The initial structuring is based on the initial location of the agents. After that, they reconfigure according to changes in the environment.
- Behavioral Model: It shows the main functionality of the organization and how it handles the dynamics. The system moves from one state to another if it tackles one of the following situations:
  - A task occurs
  - A new agent enters

- An agent goes out of the system
- Activity Model: It shows the main processes of different departments in the organization. Here, components should work focusing on the goal. The goal in this case is to maximize the overall utility in the system. The utility is calculated as follows:

$$Utility = TaskCompletionRate / MeanTaskCompletionTime \quad (4)$$

Another interesting application in allocating tasks to Multihop Wireless Networks by [31]. They state that currently, required processing powers might be beyond the capabilities of individual nodes. They say that the obvious solution is through parallel processing by dividing tasks into simpler subtasks and then assign those tasks to individual nodes. However, they show that this might not be very applicable in such networks as distributed processing might have a negative effect. Therefore, they propose genetic algorithms to allow for parallel processing along with nodes collaboration to have a cost effective strategy. Their testing results prove significant performance improvement.

## 2.4 Chapter Summary

In this chapter, a comprehensive literature review is presented. Some papers discussed coalition formation for cooperative agents. The authors of these papers apply several techniques to solve the problem like integer programming, greedy algorithms, graph theory, and genetic and evolutionary algorithms. On the other hands, some authors discussed coalition formation for selfish agents who want to maximize their pay-offs. The authors of these papers apply mainly negotiation and communication techniques for agents. Also, some papers tackled different aspects of coalition formation like learning mechanisms by agents, multi objective applications, skills uncertainty for agents and IT applications in task allocation via coalition formation.

## **Chapter 4**

### **Our Contribution to Task Allocation via Coalition Formation in Cooperative Settings**

The main goal of this model is to assign agents to tasks and thus form a coalition for each task while maximizing the efficiency of the whole system. In other words, each task should be assigned the optimally capable agent with the minimum possible cost. The assignments of agents to tasks should be done taking into account two constraints; Each agent should be assigned to one and only one task, and the resources of the agents performing the task should be greater than or equal to the tasks's requirement.

#### **3.1 Assumptions**

- Each task has a vector of requirements that needs to be satisfied before a task could be called done
- There is a finite number of tasks in the system
- Each task has a payment that is given at the end of it's execution
- Each agent has a vector of resources that shall be utilized to perform tasks
- There is a finite number of agents in the system
- Each agent has a cost that is endured once it is assigned to a task
- Agents are cooperative and want to increase the efficiency of the system more than increasing their own good
- A single agent may not have enough resources to satisfy the requirements of a task

#### **3.2 Notations**

##### **3.2.1 Sets**

- Tasks: Set of candidate tasks to be performed in the system  
Tasks= (tasks:1,2,3...J), where J is the number of tasks
- Requirements: set of requirements in the system  
Requirements= (Requirements:1,2,3...K), where K is the number of requirements

- Task-Requirements: set of requirements required by the task  
Task-Requirements= (Task-Requirements:1,2,3...K), where K is the number of requirements required by the task
- Agents: Set of agents in the system  
Agents= (Agents:1,2,3...I), where I is the number of agents
- Resources: Set of resources in the system  
Resources= (Resources:1,2,3...K), where K is the number of resources
- Agent-Resources: set of resources belonging to the agent  
Agent-Resources= (Agent-Resources:1,2,3...K), where K is the number of resources for the agent
- Agents-Tasks: Set of assignments between the agents and the tasks  
Agents-Tasks= (Agents-Tasks:1,2,3...I\*J), where I\*J is the size of the two dimensional matrix linking agents and tasks

### 3.2.2 Input Parameters

- $T_j$ : Task j in the system
- $P_j$ : The payment per task j
- $TR_{jk}$ : The requirement k of task j
- $A_i$ : Agent i in the system
- $C_i$ : The cost per agent i
- $AR_{ik}$ : The resource k of agent i

### 3.2.3 Binary Decision Variables

$$AT_{ij} = \begin{cases} 1 & \text{if agent } i \text{ is assigned to task } j \\ 0 & \text{otherwise} \end{cases} \quad (5)$$

$$Done_j = \begin{cases} 1 & \text{if task } j \text{ is performed} \\ 0 & \text{otherwise} \end{cases} \quad (6)$$

### 3.3 Constraints

There are two main constraints in the systems.

- Each agent is assigned to one and only one coalition. That is, each agent can only perform one task

$$\sum_j AT_{ij} \leq 1, \forall i \in 1, \dots, I \quad (7)$$

- The resources of the agents in the coalitions should be greater than or equal to the requirements of the performed task

$$\sum_i AT_{ij} * AR_{ik} \geq TR_{jk} * Done_j, \forall j \in 1, \dots, J, \text{ and } \forall k \in 1, \dots, K \quad (8)$$

### 3.4 Objectives

The overall objective of this model is to maximize the efficiency of the system; that is, maximize the profits by assigning the cheapest capable agents to tasks.

$$\text{Maximize} \left( \sum_j DONE_j * P_j \right) - \left( \sum_{ij} AT_{ij} * C_i \right) \quad (9)$$

### 3.5 Illustrative Example

In this section, an illustrative example of 3 tasks and 6 agents is illustrated. The figures below shows the system's structure for this example. Table 1 below show the tasks structure while table 2 show the agents structure. The assignment of agents to tasks is shown in table 3

Table 1: Task's Structure

Tasks	Requirement 1	Requirement 2	Requirement 3	Payment
Task #1	173	469	33	18780
Task #2	204	294	20	5446
Task #3	436	489	104	15760

Table 2: Agent's Structure

Agents	Resource 1	Resource 2	Resource 3	Cost
Agent #1	237	204	225	9637
Agent #2	344	343	425	9869
Agent #3	-48	61	141	1921
Agent #4	301	-32	209	5909
Agent #5	435	425	449	5462
Agent #6	375	466	211	6874

Table 3: Coalitions Formed with cooperative agents

Task 1	Task 2	Task 3
Agent 3		
Agent 5		

### 3.5.1 Sensitivity Analysis

In this section, we will apply sensitivity analysis on the example illustrated above. Specifically, we are interested on how the value of the efficiency changes when one of the following changes at a time: A task's payment, a task's requirement, an agent's cost, and an agent's resource. Table 4 below illustrates the simulations.

Table 4: Sensitivity Analysis for Cooperative Behaviors

Parameter Changed	Parameter Value	Coalition Value	Coalition Structure	Change?
Task 2 Payment	5446	11397	Task 1 is Performed	Initial Case
Task 2 Payment	6000	11397	Task 1 is Performed	No
Task 2 Payment	6500	11397	Task 1 is Performed	No

Parameter Changed	Parameter Value	Coalition Value	Coalition Structure	Change?
Task 2 Payment	7000	11523	Task 1 and 2 are Performed	Yes
Task 1 Requirement	173	11397	Task 1 is Performed	Initial Case
Task 1 Requirement	200	11397	Task 1 is Performed	No
Task 1 Requirement	250	11397	Task 1 is Performed	No
Task 1 Requirement	300	11397	Task 1 is Performed	No
Task 1 Requirement	350	11397	Task 1 is Performed	No
Task 1 Requirement	400	6444	Task 1 is Performed	Yes
Agent 4 cost	5909	11397	Task 1 is Performed	Initial Case
Agent 4 cost	4000	11397	Task 1 is Performed	No
Agent 4 cost	3000	11397	Task 1 is Performed	No
Agent 4 cost	2000	11397	Task 1 is Performed	No
Agent 4 cost	1000	11397	Task 1 is Performed	No
Agent 5 resource	435	11397	Task 1 is Performed	Initial Case
Agent 5 resource	400	11397	Task 1 is Performed	No
Agent 5 resource	350	11397	Task 1 is Performed	No
Agent 5 resource	300	11397	Task 1 is Performed	No

Parameter Changed	Parameter Value	Coalition Value	Coalition Structure	Change?
Agent 5 resource	250	11397	Task 1 is Performed	No
Agent 5 resource	200	10646	Task 1 and 3 are Performed	Yes

In this simulation, we start with the initial case of the example illustrated above where only Task 1 is performed and the efficiency of the system is 11397. After that, we gradually change the payment of task 2 to understand if increasing its payment would give incentives for the agents to perform it. When the payment of task 2 is 7000, the coalition structure changes with a new value of 11523 and now task 1 and 3 are performed. This methodology is also applied to Task 1 requirement, Agent 4 cost, and Agent 5 resource. As shown from the simulation run above, the payment seems to be the most sensitive parameter for coalitions to be formed in this example.

### 3.6 Computational Results and Conclusion

The model was tested for medium and big sized problems. However, due to the NP complexity of the problem, the model can give optimal solutions up to 12 tasks, 14 agents and 8 resources/requirements. Big sized problems will be solved via genetic algorithms to solve for good solutions in an acceptable time.

#### 3.6.1 Testing Environment

The optimization model was coded, run and tested in LINGO 12.0 in an Intel(R) Core(TM)2 Duo CPU processor, 2.10 GHz speed and 3.00 GB RAM memory. The problem sets were generated randomly in the excel sheet and input to the LINGO program via a database connection. The following parameters were considered for testing:

- Medium problems: 1-11 tasks, 3-14 agents, 1-8 resources
- Range of task's payments: 1:20,000 units
- Range of task's requirements: 1:500 units
- Range of agent's costs: 1:10,000 units

- Range of agent's resources: -100:500 units

Table 14 in Appendix A show detailed test results for medium problems. The following conclusions could be made about the runs:

- The tasks with the maximum payment are being performed
- The agents who are most capable are assigned to perform the task provided they are the cheapest
- Agents are assigned if and only if they will be participate in performing a task to avoid any wasted costs
- Agents prefer high payment over a complex task; that is, a capable agent would perform a high paying complex task rather than a low paying simple tasks
- A capable agent will not perform a task if it's cost is higher than it's payment
- The payment of the task is the most sensitive parameter that highly affects the coalitional structure and value

### 3.6.2 Genetic Algorithms Application for cooperative agents

The structure of the problem implies a boolean satisfiability problem (SAT). Specifically, agents needs to be assigned to tasks while satisfying all the clauses and constraints specified in the previous section. The problem attempts at finding the optimal assignments while maximizing the efficiency of the system. Therefore it implies a MAX-SAT problem which is a variant of the SAT problem [32]. This problem, as illustrated before and due to its structure, is of NP complexity. Therefore, in this section we propose a meta heuristics, genetic algorithm, to help solve big sized problems in a reasonable time. The details of the algorithm are illustrated in the following section.

#### Chromosome Encoding

The first step in using genetic algorithms is to find an appropriate chromosome encoding methodology. In our algorithm, we are going to utilize the chromosome encoding designed in [19] which is a two dimensional binary system chromosome. Basically, it is a two dimensional matrix of agents versus tasks where  $a_{ij}$  of value 1 means that agent  $i$  is assigned to task  $j$  and 0 otherwise. The encoding is illustrated in table 5.

This encoding structure suits our problem the most due to the following reasons:

Table 5: Chromosome Encoding

Titles	Task 1	Task 2	Task 3	...	Task m
Agent 1	1	0	0	...	0
Agent 2	0	0	1	...	0
Agent 3	0	1	0	...	0
⋮	⋮	⋮	⋮	⋮	⋮
Agent n	0	0	0	...	1

- It retains the binary encoding
- It has a similar representation to the Optimization Model
- It is easy to apply the cross over and mutation operator due to the binary mechanism

#### **Initial Population Generation**

During experimentation and from literature, it is realized that the quality of the initial population highly affects the quality of the solution generated. The initial population is created in different ways which are:

- Completely Random Chromosomes
- Random chromosomes where each row has only one assignment. The purpose of this chromosome is to satisfy the first constraint where each agent is allowed to work on one and only one task
- Chromosomes where all agents are assigned to each task respectively
- A Chromosome with zero assignments
- A Chromosome with full assignments; that is all fields in the two dimensional array are assigned the value of one
- A Chromosome where agents are assigned to the first task until the requirements of that task are satisfied. After that, the agents get assigned to the second task until it's requirements are satisfied and so on...

- Chromosomes where agents are assigned to a random task until the requirements of that task are satisfied. After that, the agents get assigned to another random task until it's requirements are satisfied and so on...

These attempts try to maximize the possible combination of chromosomes in the population. In our approach, we started with an initial population of 300 members. This number was obtained after several experimentation and it proved to give the best results.

### 3.7 Selection Strategy

The selection strategy we implemented was highly inspired by the work of [32]. It is based on diversification and fitness as genetic algorithms recommend. The selected candidates are then used for the reproduction operators; cross over and mutation.

The selection strategy was based on several methodologies illustrated below:

- One good parent with one good parent based on fitness
- One good parent based on fitness with a randomly selected parent
- One good parent with one bad parent based on fitness
- Two randomly selected parents

#### **Cross over operator pseudo code**

The population is mainly generated with the cross over operator. In this algorithm, we have applied the uniform cross over where a new child is constructed by combining two parents randomly. The uniform cross over proves beneficial because it improves diversification. In addition, it outperformed the 'OR' cross over operator during our experimentation. The pseudo code of the cross over operator is illustrated in algorithm 1.

#### **Mutation Operator pseudo code**

The mutation operator was highly inspired by the work of [32]. It is used to improve the children who are produced from the cross over operator. It works by selecting a random variable in the chromosome and flipping it. We have implemented the mutation operator with a probability of 0.2. The pseudo code of the mutation operator is shown in algorithm 2.

#### **Elitism Operator**

---

**Algorithm 1** The Uniform Cross Over Operator

---

**Require:** Two Parents; parent 1 and parent 2 and parent copying percent=0.5

**Ensure:** Two Children

```
while agent  $i$  in Agents do
  while task  $j$  in Tasks do
     $b \leftarrow \text{randomnumberbetween0and1}$ 
    if  $r \leq 0.5$  then
       $child1[agent][task] \leftarrow parent1[agent][task]$ 
       $child2[agent][task] \leftarrow parent2[agent][task]$ 
    else
       $child1[agent][task] \leftarrow parent2[agent][task]$ 
       $child2[agent][task] \leftarrow parent1[agent][task]$ 
    end if
  end while
end while
```

---

---

**Algorithm 2** The Flipping Mutation Operator

---

**Require:** One parent

**Ensure:** One child

```
 $i \leftarrow \text{randomagentindex}$ 
 $j \leftarrow \text{randomtaskindex}$ 
 $r \leftarrow \text{randomnumberbetween0and1}$ 
 $child[i][j] \neq child[i][j]$ 
```

---

Elitism is one of the main operators in Genetic Algorithms that are used to maintain some individuals in the population in case they get lost during the operations discussed above. These individuals need to be maintained to keep a diversified population and to stop the algorithm from being stuck in a local optimum. Elitism is applied in the following way:

- 20% of the population maintains good individuals based on their fitness
- 5% of the population maintains average individuals based on their fitness
- 5% of the population maintains bad individuals based on their fitness

### **The fitness function**

The fitness function is used to rank and rate individuals depending on the efficiency that results in the system from assigning agents to tasks. The outline of the fitness function is shown in algorithm 3.

### **Stopping Criteria**

---

**Algorithm 3** The Fitness Function

---

**Require:** A chromosome

**Ensure:** The fitness of the chromosome

```
if The Chromosome satisfies constraint one and constraint two then
  while There are tasks do
    while There are agents do
      if the agent is assigned to the task then
        Calculate the cost of the agent participating in the task
      end if
    end while
  if the task is performed then
    Calculate  $paymentof\ task - cost\ of\ all\ participating\ agents$ 
  end if
end while
else
   $fitness = -\infty$ 
end if
```

---

Two stopping criteria were considered during the runs. The first one is iterative steps between 1000-10000 iterations. The other rule is the convergence rule. The algorithm keeps on looping until it converges. That is, if the difference between the best individuals of two consecutive populations is 0.0001 for 1000 times. This criteria proves a faster termination of the algorithm than the iterations criteria while still providing same results.

#### **Genetic Algorithm pseudo code**

After explaining the structure of the different operators in genetic algorithm, the structure of the overall algorithm is stated in algorithm 4 below.

#### **Performance Assessment**

As there are no benchmark problems available in the literature to assess the quality of the solution, the genetic algorithm will be bench marked against the exact solutions obtained above. The code was implemented using Java. No Libraries or genetic algorithms engines where implemented. Everything was coded from scratch. Keeping the same testing environment as the optimization model, the algorithm was tested against the problems specified above. The genetic algorithm was performing very well giving 0% deviation for medium problems. In addition, the genetic algorithm is able to solve bigger problems than the exact solution. Figures 1 and 2 below show the performance of both the GA and the Lingo in terms of number of tasks and number of agents respectively. The data points are randomly selected. The blue line indicates

---

**Algorithm 4** Overall Structure of the Genetic Algorithm

---

**Require:** An instance of satisfiability

**Ensure:** An assignment of agents to tasks maximizing the overall efficiency of the system

**while** The algorithm didn't converge **do**

    Generate the initial population

    Select two parents

    Apply the cross over operator

$b \leftarrow \text{randomnumberbetween}0\text{and}1$

**if**  $r \leq 0.2$  **then**

        Apply the mutation operator

**end if**

    Add the children to the new population

**end while**

**return** the best individual A found

---

where the Lingo stopped running. The Lingo and GA points overlap due to the 0% deviation. Therefore, all points after the blue line are Genetic Algorithm runs as the graphs show. The details of the runs are shown below in table 15 in Appendix B.

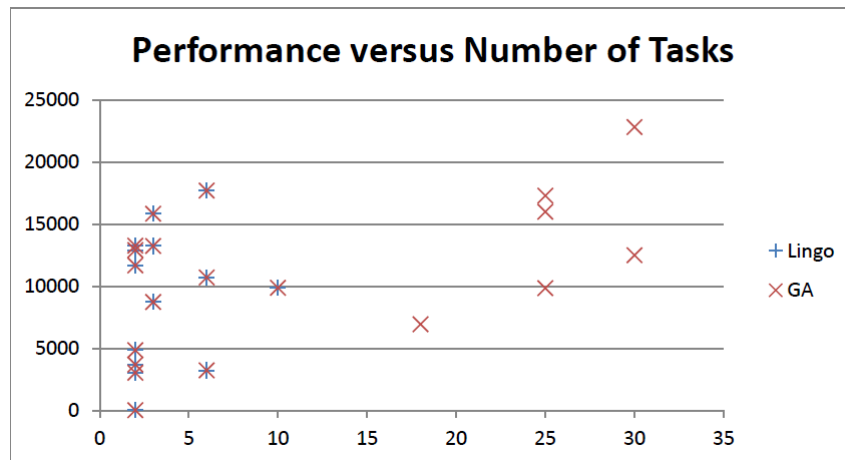


Figure 1: Performance Versus Number of Tasks for Cooperative Settings

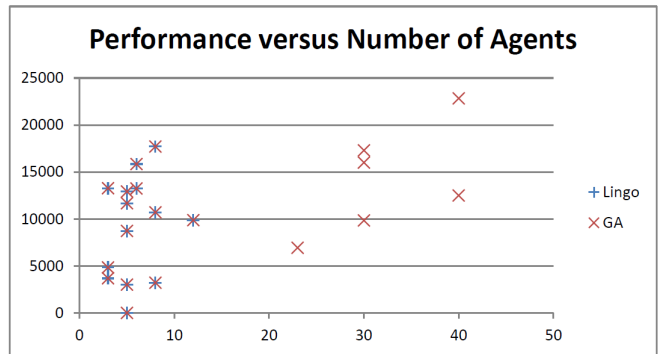


Figure 2: Performance versus Number of Agents for Cooperative Settings

### 3.8 Chapter Summary

In this section, an optimization model was generated to solve for coalition formation in cooperative environments. The coalitions were formed in order to increase the efficiency of the system which is defined as the payments of all the performed tasks minus the costs of all the participating agents. Due to the complexity of the problem, genetic algorithms was developed to solve for big problems in order to get acceptable solutions in an acceptable time. The algorithms is giving an average of 0% for medium problems.

# Chapter 5

## Our Contribution to Task Allocation via Coalition Formation in Selfish Settings

The main goal of this model is to assign agents to tasks while maximizing the payoff of each agent. First, we define the payoff of the agent as the payment of the task in proportion to how much resources the agent has compared to the members in the coalition it is a member of. This definition of payoff is highly inspired by game theory concepts explained in chapter one where the payoff of each agent depends on the strategies taken by other agents. The model in itself is a multi objective realization. In fact, the number of objectives is equal to the number of agents as the model aims at maximizing the payoff of each and every agent. This problem is very complex due to the fractional complexity in the objective function.

### 4.1 Assumptions

- Each task has a vector of requirements that needs to be satisfied before a task could be called done
- There is a finite number of tasks in the system
- Each task has a payment that is given at the end of its execution
- Each agent has a vector of resources that shall be utilized to perform tasks
- There is a finite number of agents in the system
- Each agent has a cost that is endured once it is assigned to a task
- Agents are selfish and want to increase their payoff more than increasing the efficiency of the system
- A single agent may not have enough resources to satisfy the requirements of a task

### 4.2 Notations

#### 4.2.1 Sets

- Tasks: Set of candidate tasks to be performed in the system  
Tasks= (tasks:1,2,3...J), where J is the number of tasks

- Requirements: set of requirements in the system  
Requirements= (Requirements:1,2,3...K), where K is the number of requirements
- Task-Requirements: set of requirements required by the task  
Task-Requirements= (Task-Requirements:1,2,3...K), where K is the number of requirements required by the task
- Agents: Set of agents in the system  
Agents= (Agents:1,2,3...I), where I is the number of agents
- Resources: Set of resources in the system  
Resources= (Resources:1,2,3...K), where K is the number of resources
- Agent-Resources: set of resources belonging to the agent  
Agent-Resources= (Agent-Resources:1,2,3...K), where K is the number of resources for the agent
- Agents-Tasks: Set of assignments between the agents and the tasks  
Agents-Tasks= (Agents-Tasks:1,2,3...I\*J), where I\*J is the size of the two dimensional matrix linking agents and tasks

#### 4.2.2 Input Parameters

- $T_j$ : Task j in the system
- $P_j$ : The payment per task j
- $TR_{jk}$ : The requirement k of task j
- $A_i$ : Agent i in the system
- $A_p$ : Participating agents in the coalition
- $C_i$ : The cost per agent i
- $AR_{ik}$ : The resource k of agent i

### 4.2.3 Binary Decision Variables

$$AT_{ij} = \begin{cases} 1 & \text{if agent } i \text{ is assigned to task } j \\ 0 & \text{otherwise} \end{cases} \quad (10)$$

$$Done_j = \begin{cases} 1 & \text{if task } j \text{ is performed} \\ 0 & \text{otherwise} \end{cases} \quad (11)$$

### 4.2.4 Continuous Decision Variables

$$payoff_i = \sum_{task} (AT_{ij} * done_j * P_j * (\sum_k AR_{ik} / (\sum_{pk} AR_{pk} * AT_{pj}))), \forall i, j, p, \text{ and } k \quad (12)$$

## 4.3 Constraints

There are two main constraints in the systems.

- Each agent is assigned to one and only one coalition. That is, each agent can only perform one task

$$\sum_j AT_{ij} \leq 1, \forall i \in 1, \dots, I \quad (13)$$

- The resources of the agents in the coalitions should be greater than or equal to the requirements of the performed task

$$\sum_i AT_{ij} * AR_{ik} \geq TR_{jk} * Done_j, \forall j, \text{ and } k \quad (14)$$

## 4.4 Objectives

The overall objective of this model is to maximize the payoff of all the agents.

$$\text{Maximize}(\sum_i payoff_i) \quad (15)$$

## 4.5 Illustrative Example

In this section, an illustrative example of 3 tasks and 6 agents is illustrated. Tables 6, 7, and results2 show the tasks, agents and coalitions formed in this example.

Table 6: Task's Structure

Tasks	Requirement 1	Requirement 2	Requirement 3	Payment
Task #1	173	469	33	18780
Task #2	204	294	20	5446
Task #3	436	489	104	15760

Table 7: Agent's Structure

Agents	Resource 1	Resource 2	Resource 3	Cost
Agent #1	237	204	225	9637
Agent #2	344	343	425	9869
Agent #3	-48	61	141	1921
Agent #4	301	-32	209	5909
Agent #5	435	425	449	5462
Agent #6	375	466	211	6874

Table 8: Coalitions Formed with selfish agents

Task 1	Task 2	Task 3
Agent 3	Agent 2	Agent 1
Agent 4		Agent 5
Agent 6		

#### 4.5.1 Sensitivity Analysis

In this section, we will apply sensitivity analysis on the example illustrated above. Specifically, we are interested on how the value of the efficiency changes when one of the following changes at a time: A task's payment, a task's requirement, an agent's cost, and an agent's resource. Table 9 below illustrates the simulations.

Table 9: Sensitivity Analysis for Selfish Behaviors

Parameter Changed	Parameter Value	Coalition Value	Coalition Structure	Change?
Task 2 Payment	5446	39962	Task 1,2,3 are Performed	Initial Case
Task 2 Payment	6000	40515.5	Task 1,2,3 are Performed	Yes

Parameter Changed	Parameter Value	Coalition Value	Coalition Structure	Change?
Task 1 Requirement	173	39962	Task 1,2,3 are Performed	Initial Case
Task 1 Requirement	200	39962	Task 1,2,3 are Performed	No
Task 1 Requirement	250	39962	Task 1,2,3 are Performed	No
Task 1 Requirement	300	39962	Task 1,2,3 are Performed	No
Task 1 Requirement	350	39962	Task 1,2,3 are Performed	No
Task 1 Requirement	400	39962	Task 1,2,3 are Performed	Yes
Agent 4 cost	5909	39962	Task 1,2,3 are Performed	Initial Case
Agent 4 cost	4000	39962	Task 1,2,3 are Performed	No
Agent 4 cost	3000	39962	Task 1,2,3 are Performed	No
Agent 4 cost	2000	39962	Task 1,2,3 are Performed	No
Agent 4 cost	1000	39962	Task 1,2,3 are Performed	No
Agent 5 resource	435	11397	Task 1,2,3 are Performed	Initial Case
Agent 5 resource	400	57960	Task 1,2 is Performed	Yes

In this simulation, we start with the initial case of the example illustrated above Task 1,2, and 3 are performed and the efficiency of the system is 39962. After that, we gradually change the payment of task 2 to understand if increasing its payment would give incentives for the agents to perform it. When the payment of task 2 is 6000, the coalition structure changes with a new value of 40515.5 with higher payoffs to agents. The same changes are applied to Task 1 requirement, Agent 4 cost and Agent 5 resource. As shown from the simulations above, it seems that the payments of tasks and the resources of the agents are the most sensitive parameters in this example.

## 4.6 Computational Results and Conclusion

Our formulation for the problem concerning the second objective which is maximizing the payoff of each agent is very complex and can not be solved in a polynomial time due to the decision variable that exists in the fraction in the objective function. Therefore, big problems could not be solved exactly and thus there was a need for meta-heuristics to give acceptable solutions in optimal times. This section will explain runs, experiments, and results from exact solutions.

### 4.6.1 Testing Environment

The optimization model was coded, run and tested in LINGO 12.0 in an Intel(R) Core(TM)2 Duo CPU processor, 2.10 GHz speed and 3.00 GB RAM memory.

The problem sets were generated randomly in the excel sheet and input to the LINGO program via a database connection. The following parameters were considered for testing:

- Small problems: 1-3 tasks, 3-6 agents, 1-3 resources
- Range of task's payments: 1:10,000 units
- Range of task's requirements: 1:500 units
- Range of agent's costs: 1:10,000 units
- Range of agent's resources: -100:500 units

The following conclusions could be made about the runs:

- The tasks with the maximum payments are being performed because they give maximum payoffs for agents
- Agents are assigned if and only if they will be participate in performing a task to avoid any wasted payments
- Agents prefer high payment over a complex task; that is, a capable agent would perform a high paying complex task rather than a low paying simple task
- Agent's costs are ignored in this case
- Most of the time, all agents are assigned to tasks even if they don't contribute to accomplishing the task just to increase their payoff

- The LINGO program didn't give any results for big problems. This is obvious due to the fractional complexity of the problem
- The payments of tasks and resources of agents are the most sensitive parameters in changing the coalitional structure and value

Table 16 in Appendix C show test results for small problems.

#### 4.6.2 Genetic Algorithms Application for selfish agents

The same structure of Genetic Algorithms applied for cooperative agents will be utilized for selfish agents. However, the only difference will be realized in the fitness function due to the different objective.

##### The fitness function

The fitness function is used to rank and rate individuals depending on the payoff each agent receives from being assigned to tasks. The outline of the fitness function is shown below in Algorithm 5.

---

##### Algorithm 5 The Fitness Function

---

**Require:** A chromosome

**Ensure:** The fitness of the chromosome

**if** The Chromosome satisfies constraint one and constraint two **then**

**while** There are tasks **do**

**if** the task is performed **then**

**while** There are agents **do**

**if** the agent is assigned to the task **then**

          Calculate the payoff of the agent participating in the task

          Set the payoff of the agent

$fitness = fitness + the\ payoff\ of\ the\ agent$

**else**

$fitness = 0$

**end if**

**end while**

**end if**

**end while**

**else**

$fitness = -\infty$

**end if**

---

##### Genetic Algorithm pseudo code

After explaining the structure of the different operators in genetic algorithm, the structure of the overall algorithm is stated below in algorithm 6:

---

**Algorithm 6** Genetic Algorithm

---

**Require:** An instance of satisfiability

**Ensure:** An assignment of agents to tasks maximizing the sum of payoffs in the system

**while** The algorithm didn't converge **do**

    Generate the initial population

    Select two parents

    Apply the cross over operator

$b \leftarrow \text{randomnumberbetween}0\text{and}1$

**if**  $r \leq 0.2$  **then**

        Apply the mutation operator

**end if**

    Add the children to the new population

**end while**

**return** the best individual A found

---

## 4.7 Performance Assessment

As there are no benchmark problems available in the literature to assess the quality of the solution, the genetic algorithm will be bench marked against the exact solutions obtained above. The problems were structured in Microsoft Excel and then connected to Lingo and Java for experimentation. The problem sets were randomly generated. The testing show that the genetic algorithms gave 0% deviation for the small problems. In addition, it resulted in an average time saving of 6.95 seconds. The genetic algorithm was performing very well giving 0% deviation for medium problems. In addition, the genetic algorithm is able to solve bigger problems than the exact solution. Figures 3 and 4 below show the performance of both the GA and the Lingo in terms of number of tasks and number of agents respectively. The data points are randomly selected. The blue line indicates where the Lingo stopped running. Therefore, all points after the blue line are Genetic Algorithm runs as the graphs show. The points in the graphs overlap because there is 0% deviation between the GA and the exact solution. The details of the testing are shown in table 17 in Appendix D.

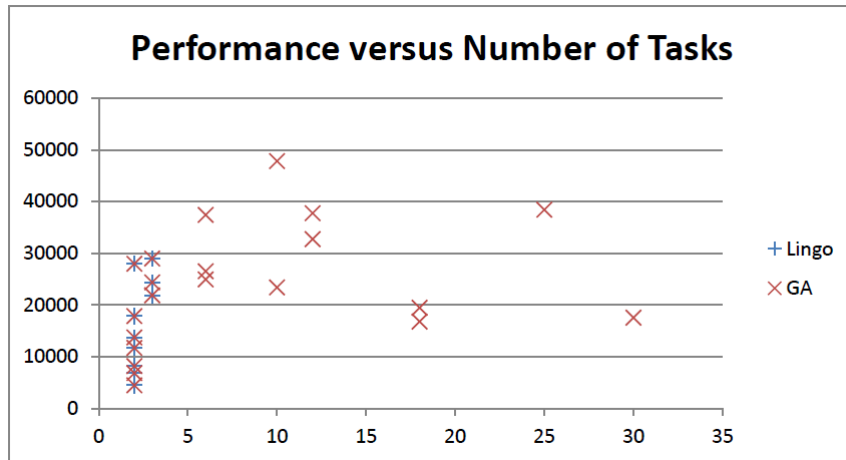


Figure 3: Performance Versus Number of Tasks in Selfish Settings

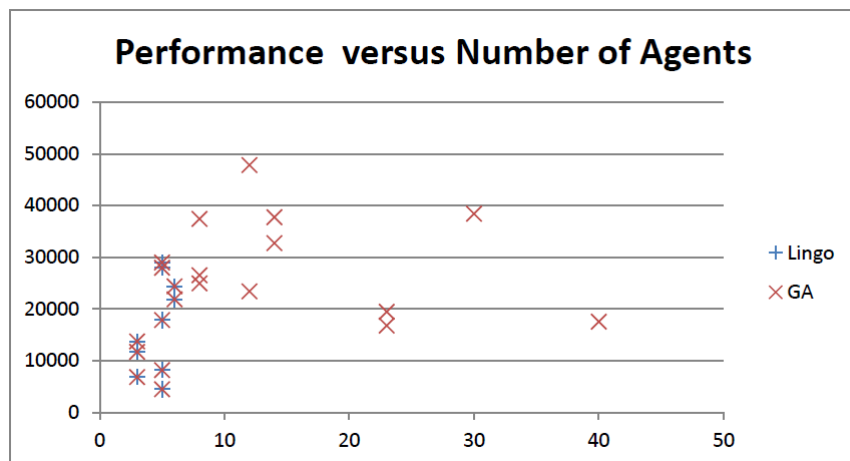


Figure 4: Performance versus Number of Agents in Selfish Settings

## 4.8 Chapter Summary

In this section, an optimization model was generated to solve for coalition formation in selfish environments. The coalitions were formed in order to increase the payoff of each agent which is defined as the payment of the task he is performing in proportion to how much resources it has compared to the resources the agents in his coalition have. Due to the NP and fractional complexity of the problem, genetic algorithms was developed to solve for big problems in order to get acceptable solutions in

an acceptable time. The algorithms is giving 0% deviation for small problems with an average time saved of 7 seconds.

## **Chapter 5**

### **Our Contribution to Task Allocation via Coalition Formation in Cooperative AND Selfish Settings**

In this section, we are interested in examining the behavior of the system when we combine both behaviors; cooperative and selfish. In order to accomplish this, we will examine the behavior of the system taking into account increasing both the efficiency of the system and the payoff of each and every agent.

#### **5.1 Assumptions**

The main assumptions in this model are:

- Each task has a vector of requirements that needs to be satisfied before a task could be called done
- There is a finite number of tasks in the system
- Each task has a payment that is given at the end of its execution
- Each agent has a vector of resources that shall be utilized to perform tasks
- There is a finite number of agents in the system
- Each agent has a cost that is endured once it is assigned to a task
- Agents are cooperative and selfish. Therefore, they want to increase both the efficiency of the system and their payoffs.
- A single agent may not have enough resources to satisfy the requirements of a task

#### **5.2 Notations**

##### **5.2.1 Sets**

- Tasks: Set of candidate tasks to be performed in the system  
Tasks= (tasks:1,2,3...J), where J is the number of tasks
- Requirements: set of requirements in the system  
Requirements= (Requirements:1,2,3...K), where K is the number of requirements

- Task-Requirements: set of requirements required by the task  
Task-Requirements= (Task-Requirements:1,2,3...K), where K is the number of requirements required by the task
- Agents: Set of agents in the system  
Agents= (Agents:1,2,3...I), where I is the number of agents
- Resources: Set of resources in the system  
Resources= (Resources:1,2,3...K), where K is the number of resources
- Agent-Resources: set of resources belonging to the agent  
Agent-Resources= (Agent-Resources:1,2,3...K), where K is the number of resources for the agent
- Agents-Tasks: Set of assignments between the agents and the tasks  
Agents-Tasks= (Agents-Tasks:1,2,3...I\*J), where I\*J is the size of the two dimensional matrix linking agents and tasks

### 5.2.2 Input Parameters

- $T_j$ : Task j in the system
- $P_j$ : The payment per task j
- $TR_{jk}$ : The requirement k of task j
- $A_i$ : Agent i in the system
- $A_p$ : Participating agents in the coalition
- $C_i$ : The cost per agent i
- $AR_{ik}$ : The resource k of agent i

### 5.2.3 Binary Decision Variables

$$AT_{ij} = \begin{cases} 1 & \text{if agent } i \text{ is assigned to task } j \\ 0 & \text{otherwise} \end{cases} \quad (16)$$

$$Done_j = \begin{cases} 1 & \text{if task } j \text{ is performed} \\ 0 & \text{otherwise} \end{cases} \quad (17)$$

### 5.2.4 Continuous Decision Variables

$$payoff_i = \sum_{task} (AT_{ij} * done_j * P_j * (\sum_k AR_{ik} / (\sum_{pk} AR_{pk} * AT_{pj}))), \forall i, j, p, \text{ and } k \quad (18)$$

### 5.3 Constraints

There are two main constraints in the systems.

- Each agent is assigned to one and only one coalition. That is, each agent can only perform one task

$$\sum_j AT_{ij} \leq 1, \forall i \in 1, \dots, I \quad (19)$$

- The resources of the agents in the coalitions should be greater than or equal to the requirements of the performed task

$$\sum_i AT_{ij} * AR_{ik} \geq TR_{jk} * Done_j, \forall j \in 1, \dots, J \forall k \in 1, \dots, K \quad (20)$$

### 5.4 Objectives

The overall objective of this model is to maximize the payoff of all the agents.

$$Maximize(\sum_i payoff_i) + ((\sum_j DONE_j * P_j) - (\sum_{ij} AT_{ij} * C_i)) \quad (21)$$

### 5.5 Illustrative Example

In this section, an illustrative example of 3 tasks and 6 agents is illustrated. Tables 10, 11, and 12 show the tasks, agents and coalitions formed respectively.

#### 5.5.1 Sensitivity Analysis

In this section, we will apply sensitivity analysis on the example illustrated above. Specifically, we are interested on how the value of the efficiency changes when one of the following changes at a time: A task's payment, a task's requirement, an agent's cost, and an agent's resource. Table 13 below illustrates the simulations.

Table 10: Tasks

Tasks	Requirement 1	Requirement 2	Requirement 3	Payment
Task #1	173	469	33	18780
Task #2	204	294	20	5446
Task #3	436	489	104	15760

Table 11: Agents

Agents	Resource 1	Resource 2	Resource 3	Cost
Agent #1	237	204	225	9637
Agent #2	344	343	425	9869
Agent #3	-48	61	141	1921
Agent #4	301	-32	209	5909
Agent #5	435	425	449	5462
Agent #6	375	466	211	6874

Table 12: Coalitions Formed for Cooperative and Selfish behaviors combined

Task 1	Task 2	Task 3
Agent 3	Agent6	Agent1
Agent 5		Agent2

Table 13: Sensitivity Analysis for Hybrid Behaviors

Parameter Changed	Parameter Value	Coalition Value	Coalition Structure	Change?
Task 2 Payment	5446	46182	Task 1,2,3 are Performed	Initial Case

Parameter Changed	Parameter Value	Coalition Value	Coalition Structure	Change?
Task 2 Payment	6000	47289.6	Task 1,2,3 are Performed	Yes
Task 1 Requirement	173	46182	Task 1,2,3 are Performed	Initial Case
Task 1 Requirement	200	46182	Task 1,2,3 are Performed	No
Task 1 Requirement	250	46180	Task 1,2,3 are Performed	Yes
Agent 4 cost	5909	46182	Task 1,2,3 are Performed	Initial Case
Agent 4 cost	4000	46182	Task 1,2,3 are Performed	No
Agent 4 cost	3000	46182	Task 1,2,3 are Performed	No
Agent 4 cost	2000	46182	Task 1,2,3 are Performed	No
Agent 4 cost	1000	46182	Task 1,2,3 are Performed	No
Agent 5 resource	435	46181	Task 1,2,3 are Performed	Initial Case
Agent 5 resource	400	46181	Task 1,2,3 are Performed	No

Parameter Changed	Parameter Value	Coalition Value	Coalition Structure	Change?
Agent 5 resource	350	46181	Task 1,2,3 are Performed	No
Agent 5 resource	300	46180	Task 1,2,3 are Performed	No
Agent 5 resource	250	46180	Task 1,2,3 are Performed	No
Agent 5 resource	200	46180	Task 1,2,3 are Performed	No

In this simulation, we start with the initial case of the example illustrated above Task 1,2, and 3 are performed and the efficiency of the system is 46182. After that, we gradually change the payment of task 2 to understand if increasing it's payment would give incentives for the agents to perform it. When the payment of task 2 is 6000, the coalition structure changes with a new value of 47289.6 with higher payoffs to agents. This methodology is also applied to Task 1 requirement, Agent 4 cost and Agent 5 resource. As shown above from the simulation runs, it seems the payment is the most sensitive parameter in this example.

## 5.6 Computational Results and Conclusion

Our formulation for the problem concerning both objectives is complex due to the nature of the second objective and therefore, can not be solved in a polynomial time. Therefore, big problems could not be solved exactly and thus there was a need for meta-heuristics to give acceptable solutions in optimal times. This section will explain the runs, experiments, and results from exact solutions.

### 5.6.1 Testing Environment

The optimization model was coded, run and tested in LINGO 12.0 in an Intel(R) Core(TM)2 Duo CPU processor, 2.10 GHz speed and 3.00 GB RAM memory. The

problem sets were generated randomly in excel sheets and input to the LINGO program via a database connection. The following parameters were considered for testing:

- Medium problems: 1-3 tasks, 3-6 agents, 1-3 resources
- Range of task's payments: 1:10,000 units
- Range of task's requirements: 1:500 units
- Range of agent's costs: 1:10,000 units
- Range of agent's resources: -100:500 units

Table 18 in Appendix E show test results for medium problems. In addition, we compared both objectives in terms of both objectives. That is, we calculated the payoffs for agents while executing objective one, and the efficiency of the system while executing objective two, and both the efficiency and the payoffs while executing both objectives. This is important to see which objective results in more good in the system. The results are shown in table 20 in Appendix G.

The following conclusions could be made about the runs:

- The results of the second objective are the ones dominating when the number of agents is large
- Combining Both Objectives seems to be more beneficial to the system in terms of efficiency and payoff
- The number of coalitions created with selfish agents are more than the number of coalitions created with cooperative agents

## **5.7 Overview of Genetic Algorithm**

While combining both objectives, we retained the structure of the genetic algorithms except for changing the fitness function as shown in Algorithm 7:

### **5.7.1 The fitness function**

The fitness function is used to rank and rate individuals depending on the efficiency and the sum of the payoffs that result in the system from assigning agents to tasks. The outline of the fitness function is shown below in Algorithm 7.

---

**Algorithm 7** The Fitness Function

---

**Require:** A chromosome

**Ensure:** The fitness of the chromosome

**if** The Chromosome satisfies constraint one and constraint two **then**

**while** There are members in the chromosome **do**

**if** There is an assignment between agent  $i$  and task  $j$  **then**

            Calculate the cost of all agents participating in the task

            Calculate the payoff of the agent participating in the task

**if** The resources of the participating agents satisfy the requirements of the task **then**

$fitness1 = fitness1 + (\text{the payment of the task} - \text{the cost of the participating agents in this task})$

            Set the payoff of the agent

$fitness2 = fitness2 + \text{thepayoffoftheagent}$

**else**

$fitness1 = 0$

$fitness2 = 0$

**end if**

**end if**

**end while**

**else**

$fitness = -\infty$

**end if**

$fitness = fitness + fitness1$

---

### 5.7.2 Genetic Algorithm Pseudo Code

The overall structure of the genetic algorithms is shown below in Algorithm 8.

---

**Algorithm 8** Genetic Algorithm

---

**Require:** An instance of satisfiability

**Ensure:** An assignment of agents to tasks maximizing the overall efficiency and payoff of the system

**while** The algorithm didn't converge **do**

    Generate the initial population

    Select two parents

    Apply the cross over operator

$b \leftarrow \text{randomnumberbetween0and1}$

**if**  $r \leq 0.2$  **then**

        Apply the mutation operator

**end if**

    Add the children to the new population

**end while**

**return** the best individual A found

---

### 5.8 Performance Assessment

As there are no benchmark problems available in the literature to assess the quality of the solution, the genetic algorithm will be bench marked against the exact solutions obtained above. The problems were structured in Microsoft Excel and then connected to Lingo and Java for experimentation. The problem sets were randomly generated. The testing show that the genetic algorithms is giving exact solution as lingo with time savings potential towards bigger sized problems starting problem 14 onwards. However, problem 16 gave a deviation of 1.6% due to unknown reasons. In addition, the genetic algorithm is able to solve bigger problems than the exact solution. Figures 5 and 6 below show the performance of both the GA and the Lingo in terms of number of tasks and number of agents respectively. The data points are randomly selected. The blue line indicates where the Lingo stopped running. Therefore, all points after the blue line are Genetic Algorithm runs as the graphs show. The details of the testing are shown below in table 19 in Appendix F.

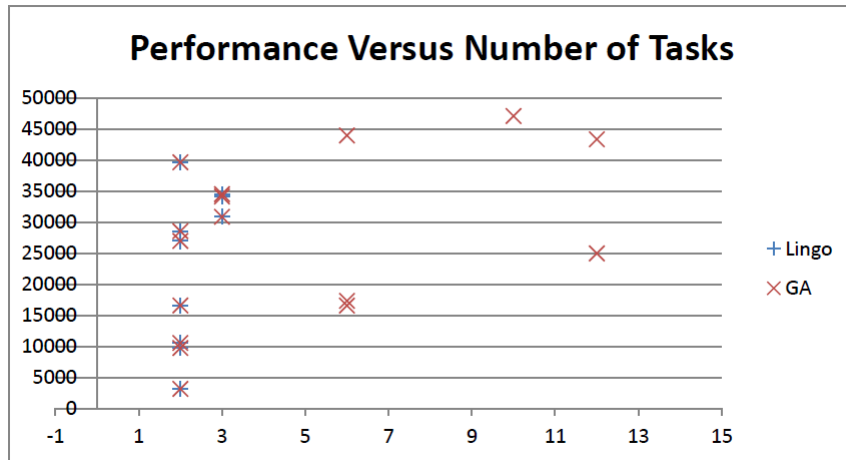


Figure 5: Performance Versus Number of Tasks in Hybrid Settings

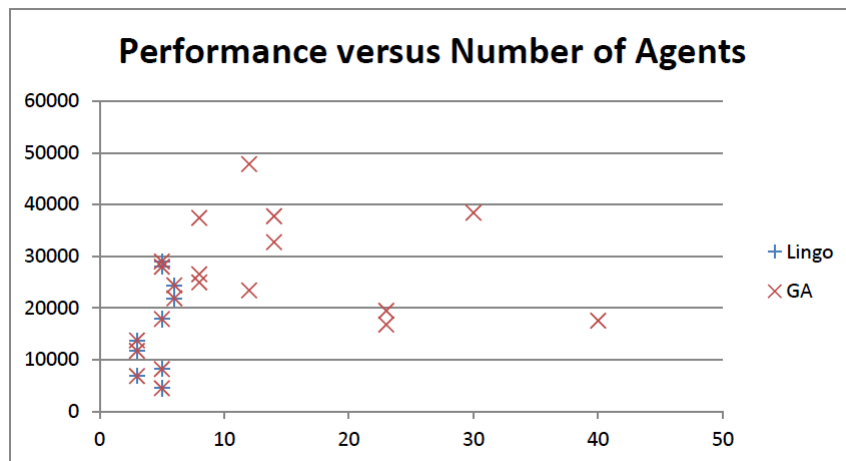


Figure 6: Performance versus Number of Agents in Hybrid Settings

## 5.9 Chapter Summary

In this section, an optimization model was generated to solve for coalition formation in cooperative and selfish environments. Due to the complexity of the problem, genetic algorithms was developed to solve for big problems in order to get acceptable solutions in an acceptable time. The algorithms is giving 0% deviation for medium problems with time savings towards bigger size problems. In addition, and after com-

paring results, it is worth noting the cooperative setting is outperforming the selfish setting. However, combining both objectives together provided better results overall.

# Chapter 8

## Conclusion and Future Work

### 6.1 Conclusion

First, we considered cooperative agents that want to maximize the efficiency of the system. Our model took into account the number of tasks, their payments and requirements, the number of agents, their costs and resources. The main objective was to maximize the payments of all performed tasks and minimize the costs of all assigned agents. The model was tested for medium problems over a set of randomly generated sets. The main conclusions about the runs indicate that tasks with maximum payments were performed using the cheapest capable agents. This was further verified through sensitivity analysis simulation proving that the task's payment is the most sensitive parameter.

Moreover, due to the NP complexity of the problem, a genetic algorithm approach was addressed aiming at providing good solutions in an acceptable time. The algorithm is giving 0% deviation for medium problems. In addition, it provided results for bigger problems where exact solutions could not be found.

This thesis addressed a Multi Objective approach for the Task Allocation via Coalition Formation problem. First, we considered cooperative agents that want to maximize the efficiency of the system. Our model took into account the number of tasks, their payments and requirements, the number of agents, their costs and resources. The main objective was to maximize the payments of all performed tasks and minimize the costs of all assigned agents. The model was tested for medium problems over a set of randomly generated sets. The main conclusions about the runs indicate that tasks with maximum payments were performed using the cheapest capable agents. This was further verified through sensitivity analysis simulation proving that the task's payment is the most sensitive parameter.

Moreover, due to the NP complexity of the problem, a genetic algorithm approach was addressed aiming at providing good solutions in an acceptable time. The algorithm is giving 0% deviation for medium problems. In addition, it provided results for bigger problems where exact solutions could not be found.

Second, we considered selfish agents that want to maximize their payoff based on the strategies of other players. Our model took into account the number of tasks, their payments and requirements, the number of agents, their costs and resources. The main objective was to maximize the payoffs of all assigned agents. The model was tested for

medium problems over a set of randomly generated sets. The main conclusions about the runs indicate that tasks with maximum payments were performed using as many agents as possible as each agent is trying to maximize their payoff. This was further verified through sensitivity analysis simulation proving that the task's payment and the agent's resources are the most sensitive parameter.

Moreover, due to the NP complexity of the problem, a genetic algorithm approach was addressed aiming at providing good solutions in an acceptable time. The algorithm is giving 0% deviation for medium problems. In addition, it provided results for bigger problems where exact solutions could not be found.

Finally, we considered hybrid agents that want to maximize the efficiency of the system and their payoff as well. Our model took into account the number of tasks, their payments and requirements, the number of agents, their costs and resources. The main objective was to increase the payoffs of all assigned agents. In addition, the model aimed at maximizing the payments of all performed tasks and minimize the costs of all assigned agents. The model was tested for medium problems over a set of randomly generated problems. The main conclusions about the runs indicate hybrid behaviors is acting similarly to cooperative agents when the number of agents is small. By increasing the number of agents, the hybrid system tends to behave like selfish agents. In addition, hybrid behaviors tend to perform better than cooperative behaviors and cooperative behaviors tend to behave better than selfish behaviors. Moreover, a sensitivity analysis simulation was conducted proving that the task's payment is the most sensitive parameter.

Moreover, due to the NP complexity of the problem, a genetic algorithm approach was addressed aiming at providing good solutions in an acceptable time. The algorithm is giving 0% deviation for medium problems. In addition, it provided results for bigger problems where exact solutions could not be found.

## **6.2 Future Work**

This field of research is attracting the interest of many researchers due to the applications it has. In the future, we recommend considering negotiation and communication among agents where they can bargain. This will be highly realized in the selfish behaviors where they can negotiate on the tasks they will be performing. Moreover, we recommend an empirical study for the agile application to further verify the results provided by this thesis.

## References

- [1] S. J. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, 3rd ed. Prentice Hall, 2009.
- [2] S. d. Jong, K. Tuyls, and K. Verbeeck, “Fairness in multi-agent systems,” *The Knowledge Engineering Review*, vol. 23, no. 2, pp. 153–180, 2008. [Online]. Available: <http://ezproxy.aus.edu/login?url=http://search.proquest.com/docview/217523231>
- [3] E. Rasmusen, *Games and information An introduction to Game Theory*, 4th ed. Blackwell Publishing, 2007.
- [4] J. P.Kahan, *Theories of Coalition Formation*. LAWRENCE ERLBAUM ASSOCIATES, PUBLISHERS, 1984.
- [5] B. M. Dunin-Keplicz and R. Verbrugge, *Teamwork in Multi-Agent Systems: A Formal Approach*, 1st ed. Wiley Publishing, 2010.
- [6] P. C. Pendharkar, “Game theoretical applications for multi-agent systems,” *Expert Systems with Applications*, vol. 39, no. 1, pp. 273 – 279, 2012. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S0957417411009791>
- [7] P. D. Straffin, *Game Theory and Strategy*, N. M. Library, Ed. The Mathematical Association of America, 1993.
- [8] S. Stahl, *A Gentle Introduction to GAME THEORY*. American Mathematical Society, 1999.
- [9] C. D. AliPrantis and S. K. Chakarabarti, *Games and Decision Making*. Oxford University Press, 2000.
- [10] J. V. Neumann and O. Morgenstern, *Theory of Games And Economic Behavior*. Princeton University Press, 1944.
- [11] O. Shehory and S. Kraus, “Feasible formation of coalitions among autonomous agents in non-super-additive environments,” *Computational Intelligence*, vol. 15, pp. 218–251, 1999.

- [12] W. F. de la Vega and M. Lamari, “The task allocation problem with constant communication,” *Discrete Applied Mathematics*, vol. 131, no. 1, pp. 169 – 177, 2003. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S0166218X02004237>
- [13] “Science and technology: Np or not np?; game theory,” *The Economist*, vol. 365, pp. 104–79, 2002. [Online]. Available: <http://ezproxy.aus.edu/login?url=http://search.proquest.com/docview/224048421>
- [14] T. Michalak, J. Sroka, T. Rahwan, M. Wooldridge, P. Mcburney, and N. Jennings, “A distributed algorithm for anytime coalition structure generation,” in *Autonomous Agents And MultiAgent Systems (AAMAS 2010)*, May 2010, pp. 1007–1014. [Online]. Available: <http://eprints.ecs.soton.ac.uk/18491/>
- [15] A. E.-S. Mohammad Khajehzadeh, Mohd Raihan Taha and M. Eslami, “Survey on meta-heuristic global optimization algorithms,” *Research Journal of Applied Sciences, Engineering and Technology*, vol. 3, pp. 569–578, 2011.
- [16] O. Shehory and S. Kraus, “Task allocation via coalition formation among autonomous agents,” in *Proc. of IJCAI*, 1995, pp. 655–661.
- [17] S. K. N. R. J. Efrat Manisterski, Esther David, “Forming efficient agent groups for completing complex tasks,” in *AAMAS '06: Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*, 2006, pp. 834–841.
- [18] V. D. Dang and N. R. Jennings, “Generating coalition structures with finite bound from the optimal guarantees,” *International joint conference on Autonomous Agents and Multiagent Systems*, vol. 3, pp. 564–571, 2004.
- [19] J. Yang and Z. Luo, “Coalition formation mechanism in multi-agent systems based on genetic algorithms,” *Applied Soft Computing*, vol. 7, no. 2, pp. 561 – 568, 2007. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S1568494606000421>
- [20] W. Gruszczyk and H. Kwasnicka, “Coalition formation in multi-agent systems - an evolutionary approach.” in *IMCSIT*. IEEE, 2008, pp. 125–130.
- [21] M. Ahmadi, M. Sayyadian, and H. R. Rabiee, “Coalition formation for task allocation via genetic algorithms,” 2002.

- [22] T. Rahwan and N. R. Jennings, “An algorithm for distributing coalitional value calculations among cooperating agents,” *Artif. Intell.*, vol. 171, no. 8-9, pp. 535–567, Jun. 2007. [Online]. Available: <http://dx.doi.org/10.1016/j.artint.2007.03.002>
- [23] P. Caillou, S. Aknine, and S. Pinson, “A multi-agent method for forming and dynamic restructuring of pareto optimal coalitions,” in *Proceedings of the first international joint conference on Autonomous agents and multiagent systems: part 3*, ser. AAMAS '02. New York, NY, USA: ACM, 2002, pp. 1074–1081. [Online]. Available: <http://doi.acm.org/10.1145/545056.545075>
- [24] S. Arib and S. Aknine, “A plan based coalition formation model for multi-agent systems,” in *Proc. IEEE/WIC/ACM Int Web Intelligence and Intelligent Agent Technology (WI-IAT) Conf*, vol. 2, 2011, pp. 365–368.
- [25] M. Hinojosa, A. Marmol, and L. Thomas, “A multi-objective model for bank atm networks,” *Naval Research Logistics (NRL)*, vol. 52, no. 2, pp. 165–177, 2005. [Online]. Available: <http://dx.doi.org/10.1002/nav.20040>
- [26] M. J. Seyyed Mohammad Sayyadi Kenari, Majid Vafaei Jahan, “Multi-skill agents coalition formation under skill uncertainty,” in *International Symposium on Artificial Intelligence and Signal Processing*, 2011.
- [27] J.-G. JIANG, Z.-P. SU, M.-B. QI, and G.-F. ZHANG, “Multi-task coalition parallel formation strategy based on reinforcement learning,” *Acta Automatica Sinica*, vol. 34, no. 3, pp. 349 – 352, 2008. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S1874102908600178>
- [28] X. Tong, H. Huang, and W. Zhang, “Agent long-term coalition credit,” *Expert Syst. Appl.*, vol. 36, pp. 9457–9465, July 2009. [Online]. Available: <http://dx.doi.org/10.1016/j.eswa.2008.12.048>
- [29] M. Hyodo, T. Matsuo, and T. Ito, “An optimal coalition formation among buyer agents based on a genetic algorithm,” in *Proceedings of the 16th international conference on Developments in applied artificial intelligence*, ser. IEA/AIE'2003. Springer Springer Verlag Inc, 2003, pp. 759–767. [Online]. Available: <http://dl.acm.org/citation.cfm?id=1113914.1113991>

- [30] A. F. . K. Z. . N. Nematbakhsh, “Adaptive team-based multi-agent organizational model: A case in rescue systems,” *International Journal of Computer Science & Information Technology*, vol. 3, pp. 165–175, 2011.
- [31] Y. Jin, J. Jin, A. Gluhak, K. Moessner, and M. Palaniswami, “An intelligent task allocation scheme for multihop wireless networks,” *Parallel and Distributed Systems, IEEE Transactions on*, vol. 23, no. 3, pp. 444 –451, march 2012.
- [32] H. D. Dalila Boughaci, Belaid Benhamou, “Scatter search and genetic algorithm for max-sat problems,” *J Math Model Algor*, vol. 7, pp. 101–124, 2008.

## A Summary of Exact Solution for Cooperative agents for Medium Problems

Table 14: Summary of Efficiency Results for medium problems

Problem	Lingo	CPU Time	CPU Mem-ory(K)	Variables	Constraints	Coalitions
#1	3705	01	25	14	8	1
#2	0	01	25	14	8	0
#3	0	01	25	14	8	0
#4	0	01	26	22	10	1
#5	0	01	26	22	10	1
#6	31	00	27	22	12	1
#7	3043	00	27	22	12	1
#8	0	00	27	22	12	0
#9	608	01	27	22	12	1
#10	0	00	27	22	12	0
#11	11687	00	27	22	12	1
#12	12925	01	28	33	15	1
#13	11555	00	28	33	15	1
#14	8742	01	28	33	15	1
#15	795	01	29	39	16	1
#16	1409	00	29	39	16	1
#17	0	01	29	39	16	0
#18	11397	00	29	39	16	1
#19	13253	00	29	39	16	2
#20	15849	00	29	39	16	1

Problem (cont)	Lingo (cont)	CPU Time (cont)	CPU Mem- ory(cont)	Variables (cont)	Constraints (cont)	Coalitions (cont)
#21	13279	00	25	11	8	1
#22	0	01	25	11	8	0
#23	3998	00	25	11	8	1
#24	0	00	25	11	8	0
#25	4868	00	25	11	8	1
#26	24160	00	26	27	16	2
#27	23567	00	26	27	16	2
#28	4981	00	26	27	16	1
#29	15412	00	26	27	16	2
#30	17704	00	26	27	16	2

## B Comparison Between GA Solution versus Exact Solution for Cooperative agents for Medium Problems

Table 15: Genetic Algorithms Solution versus Exact Solutions for medium problems

Problem	Lingo	CPU Time	GA solutions	CPU Time	Deviation
#1	3705	01	3705	16	0%
#2	0	01	0	12	0%
#3	0	01	0	10	0%
#4	0	01	0	13	0%
#5	0	01	0	13	0%
#6	31	00	31	23	0%
#7	3043	00	3043	25	0%
#8	0	00	0	25	0%
#9	608	01	608	23	0%
#10	0	00	0	12	0%
#11	11687	00	11687	35	0%
#12	12925	01	12925	29	0%
#13	11555	00	11555	26	0%
#14	8742	01	8742	28	0%
#15	795	01	795	32	0%
#16	1409	00	1409	32	0%
#17	0	01	0	6	0%
#18	11397	00	11397	16	0%
#19	13253	00	13253	36	0%
#20	15849	00	15849	26	0%
#21	13279	01	13279	05	0%
#22	0	00	0	04	0%

Problem (cont)	Lingo (cont)	CPU Time (cont)	GA so- lution (cont)	CPU Time(cont)	Deviation (cont)
#23	3998	00	3998	09	0%
#24	0	01	0	63	0%
#25	4868	01	4868	03	0%
#26	24160	00	24160	09	0%
#27	23567	01	23567	11	0%
#28	4981	00	4981	12	0%
#29	15412	00	15412	15	0%
#30	17704	02	17704	05	0%

## C Summary of Exact Solution for Selfish agents for Medium Problems

Table 16: Summary of Payoff Results for Medium Problems

Problem	Lingo	CPU Time	CPU Mem-ory(K)	Variables	Constraints	Coalitions
#1	6862.4	02	26	17	35	1
#2	1030.22	00	26	17	35	1
#3	0	00	26	17	35	0
#4	10440.3	04	26	17	35	1
#5	7262.04	01	26	17	15	1
#6	4505.66	05	27	17	17	2
#7	8205.42	03	27	17	17	2
#8	8325.71	04	27	17	17	2
#9	5898.22	03	27	17	17	2
#10	15912.7	01	27	17	17	1
#11	27978.3	02	27	17	17	2
#12	17878.3	13	28	23	20	2
#13	17249.8	07	28	23	20	1
#14	28999.9	16	28	23	20	2
#15	16881.4	10	29	27	22	2
#16	11596.8	26	29	27	22	1
#17	0	01	29	27	22	0
#18	39962	38	29	27	22	3
#19	21895.8	28	29	27	22	3
#20	24396.2	11	29	27	22	2

Problem	Lingo (cont)	CPU Time (cont)	CPU Memory (cont)	Variables (cont)	Constraints (cont)	Coalitions (cont)
#21	13748.3	01	27	17	35	2
#22	2875.4	00	27	17	35	1
#23	11197.9	01	27	17	35	2
#24	9500.05	00	27	17	35	1
#25	11731.2	00	27	17	35	1
#26	41479.7	35	29	27	22	3
#27	37543.5	26	29	27	22	3
#28	17556.8	51	29	27	22	2
#29	32400.1	19	29	27	22	2
#30	36823.9	54	29	27	22	2

## D Comparison Between GA Solution versus Exact Solution for selfish agents for Medium Problems

Table 17: Genetic Algorithms versus Optimal Solutions for Medium Problems

Problem	Lingo	CPU Time	GA solutions	CPU Time	Time Saved	Deviation
#1	6862.4	02	6867	03	02	0%
#2	1030.22	00	1031	05	00	0%
#3	0	00	0 0	03	-03	0%
#4	10440.3	04	10445	07	04	0%
#5	7262.04	01	7269	06	01	0%
#6	4505.66	05	4506	08	05	0%
#7	8205.42	03	8209	08	03	0%
#8	8325.71	04	8330	10	04	0%
#9	5898.22	03	5900	08	03	0%
#10	15912.7	01	15917	08	01	0%
#11	27978.3	02	27997	09	02	0%
#12	17878.3	13	17889	10	13	0%
#13	17249.8	07	17259	08	07	0%
#14	28999.9	16	29013	10	16	0%
#15	16881.4	10	16892	11	10	0%
#16	11596.8	26	11599	09	17	0%
#17	0	01	0	4	00	0%
#18	39962	38	39984	12	26	0%
#19	21895.8	28	21912	13	28	0%
#20	24396.2	11	24409	11	00	0%
#21	13748.3	11	13777	05	06	0%
#22	2875.4	11	2877	04	07	0%

Problem	Lingo (cont)	CPU Time (cont)	GA so- lution (cont)	CPU Time (cont)	Time Saved (cont)	Deviation (cont)
#23	11197.9	11	11210	05	06	0%
#24	9500.05	11	9512	05	06	0%
#25	11731.2	11	11743	03	08	0%
#26	41479.7	35	41512	11	24	0%
#27	37543.5	26	37570	24	02	0%
#28	17556.8	51	17562	03	48	0%
#29	32400.1	19	32415	03	16	0%
#30	36823.9	54	36841	03	51	0%

## E Summary of Exact Solution for Cooperative and Selfish agents for Medium Problems

Table 18: Summary of Efficiency and Payoff Results for Medium Problems

Problem	Lingo	CPU Time	CPU Mem-ory(K)	Variables	Constraints	Coalitions
#1	10563.7	01	26	17	35	1
#2	0	01	26	17	35	0
#3	0	00	26	17	35	0
#4	1955.67	03	27	17	15	1
#5	1784.85	01	27	17	15	1
#6	3165.81	02	27	17	17	1
#7	9752.52	01	26	17	17	2
#8	0	02	27	17	17	0
#9	4825.98	02	27	17	17	2
#10	10795.9	01	27	17	17	1
#11	39660.7	03	27	17	17	2
#12	28574.7	01	29	23	20	1
#13	26048.5	02	29	23	20	1
#14	30881.31	14	29	23	20	2
#15	9662.6	04	29	27	22	1
#16	13002.7	04	29	27	22	1
#17	0	01	29	27	22	0
#18	46180.6	28	29	27	22	3
#19	34529.4	12	29	27	22	2
#20	34141.7	02	29	27	22	1

Problem	Lingo (cont)	CPU Time (cont)	CPU Memory (cont)	Variables (cont)	Constraints (cont)	Coalitions (cont)
#21	26990.2	00	26	13	13	2
#22	0	00	26	13	13	0
#23	12323.2	00	26	13	13	1
#24	0	00	26	13	13	2
#25	16597	00	26	13	13	1
#26	63179.9	14	30	29	24	3
#27	57960	15	30	29	24	2
#28	22525.6	02	30	29	24	2
#29	47804.2	02	30	29	24	2
#30	54525	40	30	29	24	2

## F Comparison Between GA Solution versus Exact Solution for cooperative and selfish agents for Medium Problems

Table 19: Genetic Algorithms versus Optimal Solutions for Medium Problems

Problem	Lingo	CPU Time	GA solution	CPU Time	Deviation
#1	10563.7	01	10572	04	0%
#2	0	01	0	01	0%
#3	0	00	0	01	0%
#4	1955.67	03	1963	06	0%
#5	1784.85	01	1801	04	0%
#6	3165.81	02	3166	07	0%
#7	9752.52	01	9758	06	0%
#8	0	02	0	10	0%
#9	4825.98	02	4828	06	0%
#10	10795.9	01	10802	50	0%
#11	39660.7	03	39685	06	0%
#12	28574.7	01	28588	07	0%
#13	26048.5	02	26060	06	0%
#14	30881.31	14	30896	07	0%
#15	9662.6	04	9500	04	1.6%
#16	13002.7	04	13008	141	0%
#17	0	01	0	01	0%
#18	46180.6	28	46207	09	0%
#19	34529.4	12	34544	08	0%
#20	34141.7	02	34158	06	0%

Problem (cont)	Lingo (cont)	CPU Time (cont)	GA so- lution (cont)	CPU Time (cont)	Deviation (cont)
#21	26990.2	02	27019	04	0%
#22	0	02	0	01	0%
#23	12323.2	02	12334	03	0%
#24	0	02	0	0	0%
#25	16597	02	16611	04	0%
#26	63179.7	15	63212	02	0%
#27	57960	02	57984	08	0%
#28	22525.6	04	22539	09	0%
#29	47804.2	02	47829	09	0%
#30	54525	40	54546	09	0%

## **G Summary of Cooperative versus Selfish versus Hybrid Solutions**

Table 20: More Comparisons

Problem No	Efficiency			Payoff			Both Objectives			Summary
	Obj1	Obj2	SUM	Obj1	Obj2	SUM	Obj1	Obj2	Sum	
1	3705	6867.9	10572.9	3705	6262.4	10567.4	3705	6858.7	10563.7	Obj2
2	0	0	0	-33264	1030.22	-32233.78	0	0	0	Obj1,3
3	0	0	0	0	0	08	0	0	0	Obj1,2,3
4	0	0	0	-53960	10440.3	-43519.7	0	0	0	Obj1,3
5	0	0	0	-61367	7262.04	-54104.96	0	0	0	Obj1,3
6	31	3135.9	3166.9	-33738.27	4505.66	-29232.61	31	3134.8	3165.8	Obj1,3
7	3043	6276	9319	-25249	8205.42	-17043.58	1549	8203.5	9752.5	Obj3
8	0	0	0	-82455.55	8325.7	-74129.85	0	0	0	Obj1,3
9	608	3574	4182	-23022	5898.22	-17123.78	608	3572.02	4180.02	Obj1,3
10	0	0	0	-42876.54	15912.7	-26963.84	0	0	0	Obj1,3
11	11687	28000	39687	-29864.2	27978.3	-1885.9	11687	27973.7	39660.7	Obj1,3
12	12925	15663	28588	-55151	17878.3	-37272.7	12925	15649	28574	Obj1,3
13	11555	12738	24293	-76879.01	17249.8	-59629.21	8801	17247.6	26048.6	Obj3
14	8742	16856	25598	-61344.4	28999.9	-32344.5	1883	28998.3	30881.3	Obj3
15	795	8874	9669	-73988.89	16881.4	-57107.49	795	8867.6	9662.6	Obj1,3
16	1409	11600	13009	-129366.7	11596.8	-117769.9	1409	11593.6	13002.6	Obj1,3
17	0	0	0	0	0	0	0	0	0	NA
18	11397	18780	30177	-97567.9	39962	-57605.9	6223	39959.14	46182.14	Obj3
19	13253	21275.1	34528.1	-51597.3	21895.8	-29701.5	13253	21276.4	34529.4	Obj3
20	15849	18292	34141	-72288.9	24396.2	-47892.7	15849	18292.7	34141.7	Obj3
21	13279	13512	26791	8376.5	13748.3	22124.9	13242	13748.2	26990.1	Obj3
22	0	0	0	-26790	2875.4	-23914.6	0	0	0	Obj1,3
23	3998	8336	12334	-16413.6	11197.9	-47892.7	3998	8325.2	12323.2	Obj1,3
24	0	0	0	-34655.5	9500.05	-25155.5	0	0	0	Obj1,3
25	4868	11743	16611	-12865.43	11731.2	-1134.2	4868	11728.9	16596	Obj1,3
26	24160	32301	56461	-22133	41479.7	19346.7	21700	41479.7	63179.7	Obj3
27	23567	34419	57986	-42992.6	37543.5	-5449.1	23567	34393.9	57960	Obj1,3
28	4981	12581	17562	-77423	17556.8	-59866.2	4975	17550.62	22525.6	Obj3
29	15412	32418	47830	-91270.4	32400.1	-58870.3	15412	32392	47804.2	Obj1,3
30	17704	36844	54548	-33333.3	36823.9	3490.6	17704	36820.9	54524.9	Obj1,3

## H Summary of Solution for Cooperative agents for Big Problems

Table 21: Summary of Efficiency Results for big problems

Problem	Agents	Tasks	GA Sol	CPU Time
#1	12	10	21844	160
#2	12	10	29180	140
#3	14	12	30418	180
#4	12	10	13787	210
#5	23	18	6949	250
#6	30	25	17304	320
#7	30	25	9856	310
#8	30	25	16007	290
#9	30	25	22828	330
#10	30	25	12509	320

# I Summary of Solution for Selfish agents for Big Problems

Table 22: Summary of Payoff Results for big problems

Problem	Agents	Tasks	GA Sol	CPU Time
#1	12	10	19295	60
#2	12	10	20519	55
#3	12	10	23418	43
#4	12	10	47824	67
#5	14	12	32744	70
#6	14	12	37753	72
#7	23	18	16812	100
#8	23	18	19479	110
#9	30	25	38423	90
#10	30	25	17567	88

## J Summary of Solution for Hybrid agents for Big Problems

Table 23: Summary of Payoff and Efficiency Results for big problems

Problem	Agents	Tasks	GA Sol	CPU Time
#1	8	6	17341	33
#2	8	6	16591	36
#3	8	6	43975	45
#4	12	10	39841	54
#5	12	10	47102	53
#6	14	12	43362	78
#7	14	12	24987	86

## K Lingo Code for Cooperative Agents

SETS:

task: payment, done;

agent: cost, payoff;

requirement;

resource;

LINKS(agent, task): assigned;

TD(task, requirement): demand;

AS(agent, resource): supply;

numberOfAgents;

ENDSETS

!The Objective Function;

Max =@SUM(task(J):(done(J)\*payment(J))) –  
@SUM(LINKS(I,J):assigned(I,J)\*cost(I));

!Constraint 1: Each agent can  
participate in only one task;

@FOR(agent(I):

@SUM(task(J): assigned(I,J)) <=  
1);

!Constrain 2: The resources of a coalition should be greater than  
or equal to the requirements of the task;

@FOR(task(J):@FOR(requirement(K):

@SUM(agent(I): assigned(I,J) \* supply (I,K)) >=  
demand (J,K)\*done(J));

!Binary decision variables constraints;

@FOR(task(J):@BIN(done(J)));

@FOR(LINKS(I,J): @BIN(assigned(I,J)));

DATA:

task, requirement, payment, agent, resource, cost, demand, supply=

```
@OLE('FileName',  
'task', 'requirement', 'payment', 'agent', 'resource', 'cost', 'demand',  
'supply');  
ENDDATA
```

## L Lingo Code for Selfish Agents

SETS:

task: payment, done;

agent: cost, payoff;

requirement;

resource;

LINKS(agent, task): assigned;

TD(task, requirement): demand;

AS(agent, resource): supply;

ENDSETS

!Objective Function;

Max = @SUM(agent(I): payoff(I));

@FOR(agent(I): payoff(I) = @SUM(task(J): assigned(I, J) \* done(J) \* payment(J) \* (((@SUM(resource(K): supply(I, K)))) / (1 + (@SUM(AS(P, K): supply(P, K) \* assigned(P, J)))))));

!Each agent is allowed to work in only one task;

@FOR(agent(I):

@SUM(task(J): assigned(I, J)) <=

1 );

!The resources of the agents should be more than the done task;

@FOR(task(J): @FOR(requirement(K):

@SUM(agent(I): assigned(I, J) \* supply(I, K)) >= demand(J, K) \* done(J));

!Binary constraints;

@FOR(task(J): @BIN(done(J)));

@FOR(LINKS(I, J): @BIN(assigned(I, J)));

DATA:

```
task , requirment , payment , agent , resource , cost , demand , supply=  
@OLE( ' FileName ' ,  
' task ' , ' requirment ' , ' payment ' , ' agent ' , ' resource ' , ' cost ' , ' demand ' ,  
' supply ' );  
ENDDATA
```

## M Lingo Code for Hybrid Agents

SETS:

task: payment, done;

agent: cost, payoff;

requirement;

resource;

LINKS(agent, task): assigned;

TD(task, requirement): demand;

AS(agent, resource): supply;

ENDSETS

!Objective Function;

Max = super + eff;

@FOR(agent(I): payoff(I)=@SUM(task(J): assigned(I,J)\*done(J)

\*payment(J)\*(((@SUM(resource(K): supply(I,K))))/

(1+(@SUM(AS(P,K): supply(P,K)\*assigned(P,J))))));

eff=(@SUM(task(J):(done(J)\*payment(J)) -

@SUM(LINKS(I,J): assigned(I,J)\*cost(I)));

super=(@SUM(agent(I): payoff(I)));

!Each agent is allowed to work in only one task;

@FOR(agent(I):

@SUM(task(J): assigned(I,J)) <=

1 );

!The resources of the agents should be more than the done task;

@FOR(task(J):@FOR(requirement(K):

@SUM(agent(I): assigned(I,J) \* supply(I,K)) >=

demand(J,K)\*done(J));

!Binary constraints;

@FOR(task(J):@BIN(done(J)));

@FOR(LINKS(I,J): @BIN(assigned(I,J)));

DATA:

```
task , requirment , payment , agent , resource , cost , demand , supply=  
@OLE( ' FileName ' ,  
' task ' , ' requirment ' , ' payment ' , ' agent ' , ' resource ' , ' cost ' , ' demand ' ,  
' supply ' );  
ENDDATA
```

## **Vita**

Noha Tarek Amer was born in 1990, in Cairo, Egypt. She was educated in private schools and graduated from the Oxford School with honors. She received admissions and scholarships from the American University of Sharjah from 2006 to 2010. Her degree was a Bachelor of Science in Computer Science. In 2010, Ms. Noha began a Master's program in Engineering Systems Management at the American University of Sharjah. She was awarded the Master of Science degree in Engineering Systems Management in 2012. During these two years, she was granted as the graduate research and teaching assistant where she taught Statistics for Engineers and assisted Dr.Fouad Ben AbdelAziz in his research work. Ms.Noha published two papers in the EMCIS conference and CORS/MOPGP conference. Moreover, she presented a talk in the UAE math day Conference held in the American University of Sharjah. Also, Ms.Noha is the founder and head of the Engineering Systems Management student society during Spring 2012.