UNIVERSITY OF SOUTHAMPTON

FACULTY OF ENGINEERING AND PHYSICAL SCIENCES

Electronics and Computer Science

Optimising Social Welfare in Practical Cooperative Settings

by

Fatma R Habib

Thesis for the degree of Doctor of Philosophy

February 2020

UNIVERSITY OF SOUTHAMPTON

ABSTRACT

FACULTY OF ENGINEERING AND PHYSICAL SCIENCES
Electronics and Computer Science

Doctor of Philosophy

OPTIMISING SOCIAL WELFARE IN PRACTICAL COOPERATIVE SETTINGS

by Fatma R Habib

The coalition structure generation (CSG) problem is a fundamental topic in multi-agent systems and cooperative game theory. It treats scenarios in which cooperative agents strive to maximise the sum of their utilities known as the social welfare. Existing research addressing the CSG problem has focused on settings where an agent's participation is restricted to one coalition, while little research has been done on overlapping coalition formation games (OCF-Gs). OCF-Gs, introduced by Chalkiadakis et al. (2010), allow agents to join multiple coalitions. More specifically, Chalkiadakis et al. (2010) defined threshold task games (TTGs), to represent overlapping coalitions of agents in a task-based environment. In essence, every agent is endowed with a certain amount of a resource and is allowed to contribute to multiple tasks represented as coalitions. However, the TTG model makes the simplifying assumption that the environment consists of only one type of resource. As the first contribution in this thesis, we introduce MR-TTGs (multiple-resource threshold task games), an extension of TTGs that models environments with multiple resources. Furthermore, for our second contribution, we solve the CSG problem for MR-TTGs. To this end, we present two reductions of the CSG problem on MR-TTGs to two different variants of the knapsack problem. We then propose two anytime branch and bound algorithm for solving these reductions.

Moreover, work on CSG studies a very general setting where coalition structures of all sizes are feasible. However, in certain scenarios, it is desirable to specify the size (cardinality) of a coalition structure depending on the availability of some resource. The number of rooms or vehicles available, for example, influences the number of coalitions in a coalition structure. For our third contribution, we propose an algorithm to address the problem of cardinality constrained CSG. The running time of the algorithm is small for large coalition structure sizes. Moreover, the approximation ratio was less than 1.006 for all instances.

Furthermore, most of the literature on CSG focused on settings where the coalition values are given. However, in many settings, obtaining the optimal coalition values require

complex computations. As a result, in order to utilise existing CSG algorithm, one needs to calculate the values of all coalitions beforehand. We ran a couple of problems of 25 agents in this setting and it took 3 days to solve each problem using CPLEX. To circumvent expensive calculations, for our fourth contribution, we utilise interval cooperative games to substitute coalition values with approximate ones. More specifically, we prove that the resulting coalition structure is within β^2 of the optimal, where β is the approximation ratio of the values used in the interval model. Additionally, empirical evaluation of our proposed approach output solutions that are within β of the optimal.

Contents

List of Figures

Declaration of Authorship

I, Fatma R Habib , declare that the thesis entitled *Optimising Social Welfare in Practical Cooperative Settings* and the work presented in the thesis are both my own, and have been generated by me as the result of my own original research. I confirm that:

- this work was done wholly or mainly while in candidature for a research degree at this University;
- where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated;
- where I have consulted the published work of others, this is always clearly attributed;
- where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work;
- I have acknowledged all main sources of help;
- where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself;
- parts of this work have been published as: Fatma R. Habib, Maria Polukarov and Enrico Gerding. Optimising Social Welfare in Multi-Resource Threshold Task Games. In, PRIMA 2017: Principles and Practice of Multi-Agent Systems. Springer (Lecture Notes in Computer Science, 10621).

Signed:.....

Date:.....

Acknowledgements

I would like to express my sincerest gratitude to my supervisors, Dr Enrico H Gerding and Dr Maria Polukarov, for their guidance and academic support which was vital for accomplishing this work.

With many thanks to my mother, Dr Hind Alshudukhi, for her kindness and patience while I was away from home to continue this work. To my father, Dr Raad Habib, for his emotional and financial support.

I would also like to thank all members of my family and friends in Southampton for their kindness and support throughout this journey.

To my mother and father.

Nomenclature

$\mathbb{R}$	The set of real numbers.
$\mathbb{R}^+$	The set of positive real numbers.
$\mathbb{N}$	The set of natural numbers.
$\mathbb{N}_0$	$\mathbb{N} \cup \{0\}$.
ϕ	The empty set.
$\mathcal{P}(A)$	The power set of the set A.
Γ	A game.
A	The set of agents.
v	The characteristic function.
$\mathcal{C}$	A coalition.
CS	A coalition structure.
$v(CS)$	The value of a coalition structure.
$v(\mathcal{C})$	The value of a coalition.
$\mathcal{CS}$	The set of all feasible coalition structures.

Acronyms

BKP	bounded knapsack problem
BMKP	bounded multidimensional knapsack problem
CRG	coalitional resource game
CSG	coalition structure generation
CSV	coalitional skill vector
DCOP	distributed constraint optimisation problem
DP	dynamic programming
FPTAS	fully polynomial time approximation scheme
IDP	improved dynamic proframming
IP	integer partitioning
KP	knapsack problem
LB	lower bound
LP	linear programme
MAS	multi-agent systems
MCKP	multiple-choice knapsack problem
MC-nets	marginal contribution nets
MMKP	multi-dimensional multiple-choice knapsack problem
MR-TTG	multi-resource threshold task game
mTSP	multiple travelling salesman problem
OCF-G	overlapping coalition formation game
SCG	synergy coalition groups
TCSG	task count skill game
TSG	travelling salesman game
TSP	travelling salesman problem
TTG	threshold task game
UB	upper bound
WTSG	weighted task skill game
WSN	wireless sensor network

Chapter 1

Introduction

A multi-agent system (MAS) consists of a number of entities, known as agents, interacting with one another and their environment in order to achieve their goals (Wooldridge, 2009). Agents may compete to maximise their individual gains. Alternatively, they may cooperate in order to solve a specific problem, which is the focus of this thesis. Cooperation between agents plays a significant role in MASs. Introducing cooperation into a system could potentially increase the overall value of a system. For instance, through cooperation, more businesses could emerge when entrepreneurs pool their talents and resources. Furthermore, when allowed to cooperate, agents could increase their gains as opposed to competing with one another.

In a cooperative environment, agents are expected to join forces in groups known as coalitions. To illustrate this process, consider a wireless sensor network (WSN) that is required to cover a set of areas. For example, a WSN used to monitor some environmental factors in a forest for early fire detection needs to transmit readings from several areas in the forest. In order for the nodes in the WSN to perform a certain task, say send temperature, humidity, rainfall and smoke readings from 10 areas in the forest, they have to arrange themselves into groups with each group covering a specific area. In this example, nodes represent agents and groups represent coalitions. A coalition can consist of any combination of agents. It can be anywhere between the extreme cases: when all agents cooperate in a single coalition, and, when no one cooperates and each agent forms a singleton coalition. Moreover, every coalition has a worth or value. In the WSN example, a possible coalition value is the number of readings sent to the central server.

When forming coalitions, there is an exponential number of ways or layouts in which agents can divide themselves into. Each of these layouts or structures is known as a coalition structure (CS). A coalition structure is, typically, a set of disjoint coalitions of all the agents in the MAS. The value of a CS is obtained by summing up the value of every formed coalition. In this thesis, we are interested in cooperative scenarios in which the global reward is more important than the individuals' gains. More formally,

we are interested in maximising the social welfare of a system, by finding the coalition structure of the highest value, namely, the optimal coalition structure. This problem is known as coalition formation or the coalition structure generation (CSG) problem. Hence, in the WSN example, the optimal coalition structure is the one that covers the highest percentage of the given set of areas.

Early research on coalition formation investigated games with certain properties. In particular, in cooperative games, it was often assumed that the valuation function, a function used to assign values to coalitions, is superadditive. I.e., the value of a coalition is at least equal to the value of the sum of its disjoint partitions. As a result, the optimal coalition structure is always the grand coalition; the coalition of all agents in the game. The converse assumption is a subadditive game, in which the sum of the values that members of a coalition could obtain on their own is at least equal to the value of the coalition. In this case, the optimal coalition structure consists of singleton sets with cardinality equal to the number of agents. Indeed, these assumptions cannot capture many cooperative scenarios. E.g., a superadditive valuation function does not take into account the cost of communication which increases with the number of agents in a coalition.

More recent research in CSG focused on settings in which the valuation function is neither superadditive nor subadditive. Nonetheless, there are three main assumptions regarding the input of the problem and its outcome are still being made in the majority of research in cooperative game theory. The first assumption is that an agent can take part in only one coalition, i.e., a coalition structure is an exhaustive partition of the agents. The second assumption is that all coalition structures of all possible sizes are feasible. The third assumption is that the coalition values are given as input to the CSG algorithm. Nonetheless, these assumption do not apply to many practical scenarios. In this work, we study some practical scenarios that arise by dropping one of the aforementioned classic assumptions in the CSG literature.

The first assumption restricts the contribution of an agent to a single coalition. However, it is widely accepted for agents to join multiple coalitions. In the WSN example, agents (nodes) often have to cooperate to transmit data, since they have limited capabilities. Consider the forest monitoring system mentioned earlier, where the current temperature and humidity readings are required to be sent to the central sever. Assume that a group of nodes (coalition) accomplished the task of transmitting the temperature data, and there is a node in that coalition that has a surplus of battery. Then this node can join the coalition of nodes responsible for transmitting the humidity readings.

Similarly, when small cell networks cooperate to increase the coverage and capacity of wireless networks, disjoint coalitions limit the gain of cooperation as opposed to joining multiple coalitions (Zhang et al., 2014).

This motivates the need for a more general model that captures this behaviour of agents. Overlapping coalition formation games (OCF-Gs) were formally introduced by Chalkiadakis et al. (2010) along with threshold task games (TTGs), a subclass of OCF-Gs where a coalition's value depends on the tasks it completed. These models assume that agents are endowed with a single resource. In this work, we extend the TTG model to consider multiple resource types. This is a natural scenario where agents could distribute their computational resources and memory among different activities. Consider a task defined in a WSN. Here, a task may require a number of sensors and sufficient memory to log the readings from the sensors. In particular, we consider resources divisible into integral parts. I.e., resources cannot be divided into fractions such as sensors and bytes of memory.

The second assumption does not impose constraints on the cardinality of a coalition structure, i.e., the number of coalitions in a coalition structure. This constraint appears in many practical scenarios. For example, when a WSN is required to cover a specific number of areas. Here, the number of coalitions in a coalition structure is determined by the number of area. Another example is a variants of the classic multiple salesman problem such as delivering goods to customers and car sharing In these problems the number of vehicles dictates the number of coalitions and a feasible coalition structure. Another type of problems that can make use of cardinality constrained CSG is the task assignment problem, such as assigning tasks to employees in a firm or machines in a factory. Current CSG algorithms are not designed with this constraint in mind. As a result, bounding techniques and the optimality proofs in most CSG algorithms do not apply to cardinality constrained CSG.

The last assumption we drop is that coalition values are given. I.e., we consider the setting when obtaining the coalition values requires further computation. More specifically, we look at coalitions whose values are obtained by solving an optimisation problem. In the game theory literature, such settings are included under the headers: combinatorial optimisation games (Deng et al., 1997) and operation research games (Borm et al., 2001). Such scenarios appear in various domains, e.g., an online store should assign a schedule to its vehicles to deliver goods to customers. Indeed, the store aims at reducing the delivery costs while maintaining customer satisfaction. if we perceive the tasks assigned to each vehicle as a coalition, then determining the minimum running cost of a vehicle in its own is an optimisation problem with constraints regarding the convenience of customers, vehicle capacity and cost of travel. Another scenario arises from the WSNs example. In order to strengthen security in a WSN, a method is defined to find reliable routing paths between nodes in a coalition. A naïve way to utilise the current CSG algorithms for the examples described earlier is to compute all possible coalition values beforehand. However, this step is computationally expensive, especially for combinatorial optimisation games. The current CSG algorithms that assume that coalitions values are known a priori can only scale up to problems with a maximum of 25 agents

(Michalak et al., 2016). In addition to this computational complexity, the CSG problem is in itself $\mathcal{NP}$-hard (Sandholm et al., 1999). This step adds further to the computational challenge as the number of coalitions is exponential in the number of agents, e.g., there are $1,073,741,823$ coalitions in a problem that involve 30 agents. Furthermore, a CSG algorithm requires to search through all the possible permutations of these coalitions.

To circumvent the problem of storing all the coalition values, compact representations have been proposed to concisely describe the cooperation between agents. In more detail, one approach is to store a smaller set of rules that describe the marginal contributions of the agents as in marginal contribution nets (Ieong and Shoham, 2005). Another approach is to specify the value of a coalition only when there exists some positive synergy (Conitzer and Sandholm, 2006). These representations have some attractive computational properties, i.e., they are usually polynomial in the number of rules or synergies presented. However, such representations cannot accurately represent complex scenarios such as combinatorial optimisation games unless an exponential number of rules or synergies is defined. As a result, these representations are not efficient in this context. Section 2.1.6 discusses compact representations in more detail.

Against this background we discuss the research objectives of this thesis in the next section.

1.1 Research Objectives

TTGs are powerful in representing overlapping coalitions of agents in task-based environments. However, in many environments, we cannot assume the existence of a single resource which is adequate to accomplish all task types. It is natural for tasks to require different resources. Consider a task defined in a wireless sensor network environment. Typically, agents should have sufficient memory and battery levels. Hence, tasks' requirements can be defined depending on the computational complexity and the time needed to complete the task. Since TTGs can only represent a single resource, an extended model of TTGs is needed to capture multiple resources (objective 1). Moreover, this model needs to be investigated in order to solve the CSG problem. The CSG problem involve is finding the optimal coalition structure which is generally a hard problem. The complexity of the problem grows exponentially with the number of agents. Existing exact algorithms can only handle up to 30 agents (Rahwan et al., 2012). In order to address this, the design of efficient algorithms that could reach optimal or near-optimal solutions is needed (objective 2).

The problem of cardinality constrained CSG is seen in many settings. However, current CSG algorithms do not impose any restrictions on the size of a feasible coalition structure. These algorithms cannot be adopted to this special case due to the search space

representation or bounding technique. Hence, the design of a specialised algorithm is required to address the cardinality constrained CSG problem (objective 3).

The research on combinatorial optimisation games has focused on the stability of coalition structures and the fairness of payoff distribution rather than the CSG problem. A CSG defined on a combinatorial optimisation games can be seen as two $\mathcal{NP}$-hard problems on top of each other. As current CSG algorithms assume that coalition values are given as input, calculating the all coalition values prior to running one of these algorithms requires solving an exponential number of $\mathcal{NP}$-hard problems. Furthermore, combinatorial optimisation games cannot utilise compact representations of coalitional games without using an exponential amount of memory. Therefore, the design of a specialised algorithm is needed to effectively address the CSG problem with computationally expensive coalition values (objective 4).

The research objectives can be summarised as follows:

1. Constructing a model for threshold task games that can represent multiple resources.
2. Designing efficient algorithms for solving the CSG problem for the proposed model.
3. Designing efficient algorithms for solving the cardinality constrained CSG problem.
4. Designing efficient algorithms for solving the CSG problem in combinatorial optimisation games.

1.2 Research Contributions

The contributions in this thesis are as follows. In order to address research objective 1, we extend the threshold task games model and refer to it as the Multi-Resource Threshold Task Game (MR-TTG) model. In particular, tasks can require more than one resource type. Likewise, agents can possess more than one resource type. In addition, the contribution of agents is restricted to integral parts. Another restriction we impose is the availability of tasks in the system, we define task types and set a demand for every task. Following that, we formulate the coalition structure generation problem for the model.

Furthermore, we analyse the MR-TTG model to address objective 2. We approach the CSG problem by reducing it to two variations of the knapsack problem, the bounded multidimensional knapsack problem and the multidimensional multiple choice knapsack problem. The knapsack problem is extensively studied in mathematics and operational research, due to that, these reductions enable us to make use of existing techniques in

the literature. Besides, the computational complexity of the algorithms we develop for the reduced knapsack problems is independent of the number of agents.

To address objective 3, we start by analysing the current CSG algorithms and point out the limitations of these algorithms when applying them to CSG problems with fixed size coalition structures. Furthermore we developed a specialised algorithm that exploits the structure of the cardinality constrained CSG problem.

To address objective 4 we introduced a new approach to solve the CSG problem in settings with complex valuation functions based on approximate coalition values. The approach we propose utilises the interval cooperative model (Section 2.1.5) to capture approximate coalition values instead of exact ones. Furthermore, the coalition structure returned is guaranteed to be within a factor of the optimal. This guarantee is directly related to the quality of the approximation of the coalition values. Our main contributions are Theorem 6.1, that identifies the coalitions for which an exact value is crucial in finding the optimal coalition structure and Theorem 6.2, that proves bounds on the quality of the solution returned by our proposed approach.

Our work on addressing objectives 1 and 2 has been accepted in the following refereed publication:

Fatma R. Habib, Maria Polukarov and Enrico Gerding. Optimising Social Welfare in Multi-Resource Threshold Task Games. In, PRIMA 2017: Principles and Practice of Multi-Agent Systems. Springer (Lecture Notes in Computer Science, 10621).

1.3 Outline of the Thesis

The remainder of this thesis is organised as follows:

Chapter 2 provides a literature review on cooperative game theory and coalition structure generation. We also provide a detailed description of TTGs and give an overview of the existing research on OCF-Gs and interval cooperative games.

Chapter 3 presents the MR-TTG model and the CSG problem for the model. In addition, the reductions to BMKP and MMKP are provided along with their proofs.

Chapter 4 proposes algorithms for solving the BMKP and the MMKP and reports on the performance of these algorithm.

Chapter 5 presents the algorithm for cardinality constrained CSG.

Chapter 6 presents an interval based approach to solving CSG with complex valuation functions.

Chapter 7 concludes the report and highlights prospects for future work.

Chapter 2

Background and Related Work

This chapter introduces definitions and notation relevant to cooperative game theory and coalition formation. In addition, it highlights on current research in these topics. In Section 2.1 we give a general overview of cooperative game theory, providing definitions of specific games classes and compact representations of cooperative games. Section 2.2 introduces coalition structure generation (CSG) is describes current CSG algorithms. Finally, Section 2.3 introduces the travelling salesman and multiple travelling salesman problems which will be used in empirical evaluations later on.

2.1 Cooperative Game Theory

Cooperative game theory (Osborne, 2004) studies the behaviour of rational players[1] in strategic settings for which cooperation is possible. The rationality of agents, which is a fundamental assumption in game theory, suggests that agents seek to maximise their own payoffs or the sum of payoffs of a group of them. Through cooperation, individual agents in a group, or a coalition, may gain a payoff higher than what they could have achieved on their own (Osborne, 2004). Cooperation is seen in environments that allow binding agreements. Binding agreements suggest that agents are permitted or even advised to join forces and apply joint actions.

Two central problems studied in cooperative settings are the CSG problem and payment allocation. The objective of the CSG problem (detailed description in section 2.2) is to maximise the overall utility of the MAS. In payment allocation, on the other hand, the objective is to find a systematic way to distribute the payoffs of coalitions among the agents based on their contribution. In this context, stable allocations are desired. One of the strongest stability concepts is the core. An allocation is in the core (Gillies, 1959) if no agent or group of agents can be better off by deviating to another coalition. In some

[1]The terms agent and player are used interchangeably throughout this thesis.

games, however, the core can be empty. The emptiness of the core means that there is no stable budget balanced allocation for this game. Budget balancedness is an ideal revenue sharing scenario. An allocation is budget balanced when the value of a coalition is equal to the sum of the revenue shares of the agents in the coalition.

The remainder of this section provides definitions and notation for characteristic function games, which is the classic model of cooperative games; combinatorial optimisation games, where the coalition values are defined in terms of more complex characteristic functions; overlapping coalition formation games, where agents are allowed to join more than one coalition; and interval cooperative games, where the value of a coalition is represented as an interval.

2.1.1 Characteristic Function Games

A cooperative or coalitional game is defined by a set of players and a valuation function. The valuation function, also known as the characteristic function, defines the worth of each coalition. More formally, a coalitional game is defined as follows:

Definition 2.1. A coalitional game is defined as a pair $\Gamma = \langle A, v \rangle$, where A is the set of agents and v denotes the characteristic function.

Definition 2.2. A characteristic function for a coalitional game is defined as $v : 2^A \rightarrow \mathbb{R}$. Its purpose is to assign a value to each possible coalition in the game.

Definition 2.3. In a cooperative game defined over the set of agents A, a coalition C is a set of agents such that $C \subseteq A$.

The outcome of a cooperative game is defined in terms of a coalition structure. In a coalition structure (Aumann and Dreze, 1974), agents are divided into disjoint coalitions.

Definition 2.4. A coalition structure, in a given cooperative setting $\langle A, v \rangle$, is a partition of the set of agents A. A coalition structure CS is feasible if and only if $\cup_{C \in CS} C = A$ and $\forall\, C \in CS$ and $C' \in CS$ s.t. $C \neq C', C \cap C' = \phi$.

The value of a coalition structure is equal to the sum of values of its corresponding coalitions. It denoted by $v(CS)$, i.e., $v(CS) = \sum_{C \in CS} v(C)$. Moreover, the set $\mathcal{CS}$ holds all feasible coalition structures in the setting. I.e., all the possible outcomes of the game.

The next section deals with cooperative games for which the characteristic function is defined in terms of an optimisation problem.

2.1.2 Combinatorial Optimisation Games

The characteristic function given in Definition 2.2 provides no information about the cooperation of the players. That is, the value $v(C)$ of every possible set of players $C \subseteq A$ is stored in a table and does not explain how they earn this value. As a result, in order to represent a cooperative game in this form, one would need an exponential amount of memory. In many settings, the cooperation of players can be represented succinctly in terms of a mathematical function. Owen (1975) treated a cooperative game in which the characteristic function can be obtained by solving a linear program. He modelled a situation of linear production where players can pool resources to produce goods and sell them at a market price. For this class of games, Owen (1975) showed that the core is non-empty. In addition, the linear production game is useful in handling cooperative games in which the characteristic function is defined as an optimisation problem, however, not all combinatorial optimisation games can be formulated as a linear programme as shown by Granot (1986). Deng et al. (1999) extended Owen's model and explicitly defined the integrality constraint and showed useful applications of their model to games on graphs.

Definition 2.5. A combinatorial optimisation game (Deng et al., 1999) is a cooperative game in which the value of every subset of players is obtained via solving a combinatorial optimisation problem.

In combinatorial optimisation games, the problem considered is how to fairly divide the cost savings or revenue gains. The objective of payoff distribution is to find a systematic way to distribute the payoffs of coalitions among the agents involved. A well-know solution concept for payoff distribution is the Shapley value (Shapley, 1953). One of the criteria for ensuring fairness is symmetry. Symmetry states that agents with the same contribution receive the same reward or incur the same cost. For instance, in the travelling salesman game (TSG), a repairman has to visit a number of customers and the cost of the repairman's trip has to be paid by the customers (Kuipers, 1993). Considering that each trip should start and terminate at the repairman's home, the cost of visiting a number of customers in one trip might be less than the total cost of visiting the customers in separate trips. A TSG considers the problem of sharing the cost savings from the repairman's trip among the customers in a fair way. By symmetry, two customers living in the same building should pay the same amount for the transportation of the repairman.

The most relevant combinatorial optimisation games to our work are the travelling salesman and the vehicle routing games. Early work investigating the cost allocation problem in TSGs characterised the graphs for which the core is non-empty. Related the underlying structure of TSGs, Tamir (1989) showed that the triangular inequality[2] is not sufficient for the core to be non-empty. The triangular inequality property is the basis

[2]The triangle inequality theorem states that the sum of the lengths of any two sides of a triangle is always greater than or equal to the length of the remaining side.

for approximation algorithms of the travelling salesman problem (TSP). It is well-know that no polynomial time algorithm that approximates the general TSP within a constant factor exists unless P=NP. Nonetheless, the core is non-empty if there are less than 6 players in the game. Potters et al. (1992), Tamir (1989) and Kuipers (1993) proved the non-emptiness of the core for TSGs with 3, 4 and 5 players respectively.

In regard to the non-emptiness of the core in TSGs with more than 5 players, Tamir (1989) provided an example of an unbalanced TSG with 6 players. To combat this negative result, Potters et al. (1992) introduced a class of matrices defining TSGs with non-empty cores. In addition, Caprara and Letchford (2010) addressed the problem of finding good cost shares for games with an empty core. They present new techniques to find a γ-budget balanced allocation, which can be informally defined as a stable allocation with minimal loss. According to Caprara and Letchford (2010), γ-budget balanced allocations are ideal for subsidising public services. Furthermore, they apply their approach on the travelling salesman and the vehicle routing games.

2.1.3 Overlapping Coalition Formation Games

The coalitional game discussed in the previous section assumes that an agent can take part in only one coalition. However, this is not usually the case in many real-world settings. For instance, agents could distribute their time and computational and memory resources between different activities. This motivates the need for a more general model that captures the overlapped coalitions of agents. Shehory and Kraus (1996) introduced the concept of overlapping coalitions and applied it in distributed multi-agent environments where agents are expected to execute a set of tasks. In addition, they investigated settings when tasks may have a precedence order. Later on, they presented simple, distributed approximation algorithms for task execution via overlapping coalitions.

In OCF-Games, it is assumed that agents possess a certain amount of resources. Furthermore, in order to fulfil their goals, agents are expected to distribute their resources among several coalitions. In general, the overlapping setting allows agents to join as many coalitions as they wish. Since this is not always feasible, it is a natural way to restrict agents' participation in a coalitions depending on the resources they possess. For simplicity, the OCF-G model considers single-resource settings and it is assumed that agents have one unit of that resource. As agents partially contribute in coalitions, the notion of 'coalition' (see Definition 2.3) is replaced by 'partial coalition' in the non-overlapping setting, which is defined as follows:

Definition 2.6. In an OCF-G defined over a set on n agents A, a partial coalition is a vector of size n, $\pi = (\pi^1, \ldots, \pi^n)$, where $\pi^i \in [0, 1]$ specifies the contribution of agent $i \in A$ to the partial coalition.

As in the non-overlapping setting, a valuation function is defined to assign values to partial coalitions. Obviously, partial coalitions in which the contribution of all agents is equal to 0 have a value of 0.

Definition 2.7. For an OCF-G defined over the set A of n agents, the valuation function v is given as $v : [0, 1]^n \rightarrow \mathbb{R}$, such as $v(0)^n = 0$

In contrast to the non-overlapping setting, here, a coalition structure is defined as a list. This is because different partial coalition are allowed to have the same values of agents' contributions. It is formally defined as:

Definition 2.8. A coalition structure over a set of agents A is list of partial coalitions, $CS = (\pi_1, \ldots, \pi_\varrho)$ where $\pi_\rho \in [0, 1]^n$ and $\sum_{\rho=1}^{\varrho} \pi_\rho^i \leq 1$.

Threshold task games, introduced by Chalkiadakis et al. (2010), provide a simple, yet, expressive representation for cooperative games with overlapping coalitions. Here, agents aggregate their resources, by joining a coalition, in order to accomplish a task. A TTG is defined considering a single-resource environment and every agent has a specific weight of that resource. A task type is defined by a resource threshold and a value. The threshold specifies the minimum resource amount needed to complete a task of that type. Besides, the value of a task type is the gain obtained upon completing a task copy of that type. It is assumed that there is an infinite number of copies of every task type and agents working on completing a certain task can contribute any amount of the resource they possess. A TTG is formalised in the following definition:

Definition 2.9. A threshold task games, Γ, is defined by a tuple $\Gamma = \langle A, \omega, T \rangle$, where:

1. $A = \{1, \ldots, n\}$ is the set of agents.
2. $\omega = (\omega^1, \ldots, \omega^n)$ is a vector of agents weights such that $\omega^i \in \mathbb{R}^+$.
3. $T = \{\langle \tau_1, v_1 \rangle, \ldots, \langle \tau_q, v_q \rangle\}$ is the set of task types, where each task type is described by a threshold $\tau_k \geq 0$ and value v_k.

Chalkiadakis et al. (2010) work on OCF-Gs focused on the stability of these games. They generalise the concept of the core to for the overlapping setting by defining 3 cores for the game in regards to the utility of deviating players. However, their results are focused on the conservative core, where a deviating agent does not get any reward from a coalition upon leaving it. Similarly, the work of Zick et al. (2019) focused on the core of OCF-Gs. However, in their model, Zick et al. (2019) utilised arbitrator function to assign utilities for coalitions upon deviations. Also, to reduce the computational complexity, their model does not allow agents to form arbitrarily large coalitions. However, in this thesis, in line with maximising the social welfare, we focus on the CSG problem for OCF-G rather than the stability of the game.

Overlapping coalition formation has been effective in solving networked multi-agent systems problems. Dang et al. (2006) investigated the problem of widearea surveillance in multi-sensor networks. They utilised the structure of the problem in forming overlapping coalitions and developed an exact algorithm and a greedy approximate algorithm to find the optimal coalition structure.

Wang et al. (2013) investigated the problem of collaborative spectrum sensing in cognitive networks by forming overlapping coalitions. They concluded that the overlapping coalitions structures algorithm they proposed has an improved performance over non-cooperative algorithms and the state-of-art cooperative algorithm with non-overlapping coalitions. Similarly, Di et al. (2013) investigated overlapping coalition formation for collaborative smartphone sensing.

Moreover, Zhang et al. (2014) investigated the problem of cooperative interference management in small cell networks. They utilised overlapping coalitions with negative externalities in solving the problem by devising a decentralised algorithm. likewise, their approach has a notable performance over non-cooperative approaches and coalition structures with non-overlapping coalitions.

The next section reviews some coalitional models defined in resource-based settings and task-based environments.

2.1.4 Coalitional Games in Resource-Based and Task-Based Environments

Researchers have introduced interesting models of coalitional games in the non-overlapping setting. Some of these models are similar to our work, in particular, the ones which represent resource-based and task-based environments. In this section, we give an overview of these models and compare them to TTGs and the extended model we propose in Chapter 3, which we refer to as the multi-reource threshold task game model (MR-TTG)

One of the models considering a resource-based environment is the coalitional resource games (CRG) model (Wooldridge and Dunne, 2006). This work is similar to the MR-TTG model for which agents possess an amount of different resources. However, agents are associated with a set of goals and supposed to achieve one of them. The set of goals might overlap and agents are indifferent between the goals available to them, while in the TTG setting, the available tasks and their valuations is the same for all agents. In CRGs, as in TTGs, the ability of a coalition to achieve a set of goals depends on the collective sum of the agents' resources. In contrast to our work, the researchers considered the complexity of solving CRGs in environments comprising self-oriented agents.

One of the early models of coalitional games in a task-based environment was given by Dang and Jennings (2006). They approached a very general model making no assumptions on the coalition value, or restrictions on the number of agents in a coalition. Furthermore, they defined the CSG problem for this model and studied some properties of the coalitions. However, the problem they had to address is harder than finding the optimal coalition structure in characteristic function games since a task is not associated with a value or resource requirement. The value of a coalition depends on the tasks associated with it and the identity of the agents involved. In their work, Dang and Jennings (2006), gave an algorithm for the CSG problem in task-based non-overlapping settings. In addition, the authors have shown that the least number of coalitions to be searched in order to establish a bound from the optimal solution is exponential in the number of agents.

Another class of coalitional games defined in a task-based environment is the coalitional skill game introduced by Bachrach and Rosenschein (2008). Here, there is a set of tasks to be completed, each task needs several skills. In addition, each agent has a set of skills. Agents join coalitions in order to complete tasks. A set of tasks is accomplished if the required skills can be covered by agents in a coalition. In this model, skills are not quantitative. Furthermore, to determine the value of a coalition, two games were defined. Firstly, the task count skill game (TCSG), where the value of a coalition is defined as the number of tasks it can accomplish. Secondly, the weighted task skill game (WTSG), where there is a weight associated to each task, as in TTGs. Here, the value of a coalition is equal to the sum of weights of the tasks accomplished by the coalition.

In (Bachrach and Rosenschein, 2008) the focus was on questions related to stability and fairness. In terms of stability, they analysed the complexity of deciding whether the core is non-empty. In terms of fairness, they studied the complexity of calculating Shapley value and Banzhaf's power index (Banzhaf, 1965). In addition, they determined the complexity of finding specific charachterisitcs of players such as the veto and dummy players.

Bachrach et al. (2010) studied the complexity of finding the optimal coalition structure in skill games. They proved hardness results for single-task skill games. However, they give positive results when reformulating the problem as a constraint satisfaction problem on a hyper graph. Moreover, they provide a polynomial time algorithm for finding the optimal coalition structure for instances with bounded tree width and number of tasks.

Finally, Tran-Thanh et al. (2013) introduced coalitional skill vectors (CSVs), an extension for coalitional skill games. An agent's set of skills is represented as a vector to encompass the agent's level in each skill. Similarly, in order to complete a task, agents are required to satisfy a certain minimum threshold represented by the aggregate level of agents in a skill. The vector representation of skills is similar to ours of resources. This representation is expressive and concise since the valuation of coalitions does not depend on the identity

of agents. Moreover, it is efficient to compute the upper bound for problem instances of up to 500 agents.

2.1.5 Cooperation under Interval Uncertainty

The cooperative setting discussed in the previous section assumes that the payoffs of coalitions are known with certainty as it is determined by the characteristic function. However, this representation does not account for modelling errors arising from domain experts or uncertainty stemming from the nature of the problem. Uncertainty can take several forms, and can affect any of the game components. In this work, we are interested in interval uncertainty. It can be seen when agents do not know the worth of coalitions, however, they can still infer the best and worst possible outcomes of their cooperation. These figures can be encapsulated in an interval with the upper and lower bounds representing the optimistic and pessimistic outcomes of the coalitions respectively.

Interval cooperative games were introduced by Branzei et al. (2004) as an extension to the classic characteristic function games to model interval uncertainty in bankruptcy games. Interval cooperative games, like the cooperative games described in section 2.1, can be defined by a set of agents and a characteristic function.

Definition 2.10. A cooperative interval game is defined as a pair $\Gamma = \langle A, w \rangle$ where A in the set of agents and w is the interval characteristic function.

However, the characteristic function is defined as a closed interval indicated by the upper and lower bounds on the values for each coalition.

Definition 2.11. A characteristic function for an interval cooperative game is defined as $w : 2^A \rightarrow \mathbb{R} \times \mathbb{R}$ for every coalition $C \subseteq A$. The function $w(C)$ assigns upper and lower bounds for every coalition. We denote $w(C) = [\overline{w}(C), \underline{w}(C)]$ where $\overline{w}(C)$ is the maximum gain of the agents cooperating in C, and $\underline{w}(C)$ is the minimum gain of these agents.

The interval cooperative model is useful in many domains

Branzei et al. (2004) introduced the concept of interval cooperative games in 2004 and applied it to division problems in bankruptcy games. Bankruptcy games involve the division of a fixed amount of good among a group of claimants. However, this amount is insufficient to cover the claims. In the interval setting, the claimants are unsure about the exact claims but they can specify upper and lower bounds on its amount. In addition, the available amount of good is less than the cumulative lower bounds of the claims.

The majority of the work on interval cooperative games focused on allocating or reallocating collective gains or costs. Alparslan-Gök et al. (2010) studied the properties of the

interval Shapley value on size monotonic games. Moreover, they proved its uniqueness considering the axioms additivity, symmetry, efficiency and the dummy-player. Subsequently, Hwang and Chen (2012) proposed a new axiomatisation for the Shapley value using the axioms, efficiency, symmetry and coalitional strategical equivalence. Furthermore, some classes of interval cooperative games, such as convex interval games, were studied in Alparslan-Gök et al. (2009) and new characterisations of the Shapley value were defined on special subclasses of cooperative interval games by Alparslan-Gök (2014).

Other work in cooperative interval games investigated existing games to account for the interval uncertainty. Specifically, Alparslan-Gök et al. (2008) studied an application to the interval cooperative games in airport games. Their goal was to assign a cost to the airplanes for using the runway when the cost of a piece of the runway is an interval. They also extended some results from classical game theory to the convex model they proposed.

Alparslan-Gök et al. (2013) studied sequencing games with interval data. Classical sequencing scenarios include scheduling jobs on a machine and serving patients at an accident and emergency department. Alparslan-Gök et al. (2013) extended the sequencing game for jobs on one-machine to account for the intervals of the job processing time and the cost per unit.

Another interesting application for cooperative interval games was the work of Kiekintveld et al. (2013) in modelling security games as an interval game. In security games, the defender has a list of targets and a number of resources and is supposed to assign resources to protect the targets given the attacker's interval payoff for each target.

2.1.6 Representation of Cooperative Games

The traditional representation of cooperative games assumes that the value of a coalition is given by the characteristic function. Here, the characteristic function presented in Definition 2.2, acts as a blackbox function. Hence, an exponential amount of memory is required in order to store the value of every coalition. This, in turn, adversely affects the scalability of CSG algorithms. Alternative representations for cooperative games were proposed to reduce the memory requirement for a CSG problem. In Section 2.1, we looked at a succinct representation of the characteristic function in combinatorial optimisation games.

Ieong and Shoham (2005) introduced marginal contribution nets (MC-nets), a fully expressive representation scheme to model cooperative settings. An MC-nets representation consists of a set of rules that describe the marginal contributions of the agents. Moreover, Ieong and Shoham (2005) developed efficient algorithms for the computation of the Shapley value (a payment allocation method) and problems regarding the core. In the context of MC-nets, Elkind et al. (2008) presented read-once MC-nets, a more

compact representation than the MC-nets introduced in (Ieong and Shoham, 2005), yet retains its attractive computational properties. In our work, we considered representing the ride sharing problem using MC-nets. However, we concluded that this scheme would require a vast amount of memory to precisely represent our problem.

Distributed constraint optimisation problems (DCOP) (Sultanik et al., 2007) provide a compact representation of cooperative games. In a DCOP, a set of agents are responsible for assigning the values of their respective variables subject to specific constraints. Ueda et al. (2010) considered the CSG problem when the value of a coalition is the optimal solution of a DCOP. Their work resulted in an efficient approximation algorithm for the CSG problem with quality guarantees. Subsequently, Ueda et al. (2011) introduced a type-based representation for cooperative games. The idea behind agent types is that agents can be classified into groups according to their capabilities. Given a cooperative setting with a fixed number of possible agent types, the characteristic function can be represented in terms of these types. This advantage of this representation is that the CSG problem can be solved in polynomial time in the number of agents (Ueda et al., 2011).

Conitzer and Sandholm introduced two compact representations, namely, multi-issue domains (Conitzer and Sandholm, 2004) and synergy coalition groups (SCGs) (Conitzer and Sandholm, 2006). In multi-issue domains, the characteristic function is decomposed over a number of independent issues. In this context, a coalition is assigned a certain value under each issue. Here, the characteristic function for a coalition is defined as the sum of its corresponding value in every issue. In SCGs, on the other hand, the value of a coalition is represented only when there exists some positive synergy.

Ohta et al. (2009) reformulated the CSG problem in order to exploit compact representation schemes, namely, MC-nets, SCGs and SCGs in multi-issue domains. Their results show that, using generic constraint optimisation solvers, the CSG problem can be solved more efficiently compared to the state-of-the-art algorithm (Rahwan et al., 2009) at that time.

In the next section, we look at the coalition structure generation problem in coalitional games in order to maximise the overall value of the game.

2.2 Coalition Structure Generation

Maximising the social welfare is a desirable outcome in many cooperative environments. In such systems, the overall value of the system is far more important than the utilities of individual agents. The CSG problem addresses this objective in coalitional games. Specifically, it intends to find the coalition structure of maximal value, namely, the

optimal coalition structure. For characterisitic function and combinatorial optimisation games the CSG problem is defined as:

Definition 2.12. CSG is the problem of partitioning the agents in a given coalitional game Γ into disjoint sets to find the optimal coalition structure. This can be formulated as:

$$argmax_{CS \in \mathcal{CS}}\, v(CS) \tag{2.1}$$

The CSG problem has not been defined formally for the interval cooperative model, however, one can choose a substitute $v(C)$ in the above definition with $\underline{w}(C)$ for example in order to maximise the minimum possible value of the problem.

Considering the specifications of OCF-Gs in Section 2.1.3, there will be an infinite number of feasible partial coalition in the set $\mathcal{CS}$. Hence, it might not be possible to define the coalition structure generation problem for OCF-Gs as in the non-overlapping setting. We address this issue in Chapter 3 by introducing a discrete model for OCF-Gs

2.2.1 Restricted CSG

The general definition of CSG, as presented above, does not assume a structure or relationship between the agents in the set A. However, spatial and social relationships are very common between entities in a MAS. Such relationships can restrict the formations of certain coalitions. A natural way to capture such relationships is by using an undirected graph. The edges can indicate the availability of a route between two locations. In a Steiner TSG[3] for example, the repairman cannot visit customer a immediately after customer b if there is no edge connecting these customers in the graph. Therefore, a graph can restrict the formation of coalitions. Given a CSG problem with an underlying graph $\mathcal{G}$, a coalition is feasible if and only if all its members are connected[4] by $\mathcal{G}$ (Myerson, 1977).

Voice et al. (2012) proposed an algorithm for graph coalition structure generation assuming a characteristic function that satisfies the independence of disconnected members (IDM) property. The IDM property states that, given any two disconnected members a and b, the marginal contribution of a to a coalition C such that $b \notin C$ is equal to its marginal contribution when b joins the coalition C, i.e., $C \cup \{b\}$.

In addition, spatial and organisational constraints can limit the number of coalitions in a coalition structure. For instance, when assigning n buyers to m houses as in the well-known assignment game, the size of a feasible coalition structure is implicitly limited by the number of houses (Shapley and Shubik, 1971). Skibski et al. (2016) formally

[3] In a Steiner travelling salesman problem, one is given an incomplete graph and a cost for each edge of the graph. The task is to find a minimum-cost tour that passes through each node at least once.

[4] In graph theory, connectedness, means that there is a path from any point to any other point in the graph.

introduced k-coalitional cooperative games, where the size of a feasible coalition structure is limited by a constant k. Moreover, they extend the Shapley value for their model and analyse its computational properties.

More recently, Sless et al. (2018) introduced the CSG problem on k-coalitional games on a social network. In contrast to the work Skibski et al. (2016), the coalition structure size is exactly k. Their model uses an organiser to introduce the members of the social network to each other. In addition, they solve the CSG problem in polynomial time if there is a small number of negative edges in the network.

2.2.2 CSG Algorithms

CSG algorithms can be classified into three classes, heuristic, dynamic programming and anytime algorithms. Heuristic algorithms, such as the algorithm by Sen and Dutta (2000), although effective, do not provide any guarantee on the quality of the solution. On the other hand, dynamic programming and anytime algorithms guarantee the output of an optimal solution when run is completed. However, anytime algorithms (Rahwan et al., 2015) are more robust to failure since they provide a solution that gradually improves during the search. Our review focuses on algorithms in the second and third classes since we are interested in algorithms that can provide quality guarantees on the solutions they return. Furthermore, we relate the most important algorithms to the CSG problem considering coalition structures of a fixed size.

2.2.2.1 Heuristic Algorithms

A number of heuristic approaches were investigated for solving the CSG problem. One of the early CSG heuristic algorithms based on stochastic search was proposed by Sen and Dutta (2000). More recently, algorithms based on swarm intelligence (Dos Santos and Bazzan, 2012) and hierarchical clustering (Farinelli et al., 2013) were developed for searching for the optimal coalition structure.

2.2.2.2 Anytime Algorithms

The first anytime CSG algorithm was presented by Sandholm et al. (1999). The search space is represented as a coalition structure graph. For a CSG problem considering n agents, a coalition structure graph consists of n levels. The graph represents the whole search space such that the coalition structures of a certain size are contained in the corresponding level, see Figure 2.1. The top level, $LV1$, holds the coalition of all agents, i.e., the grand coalition. The edges in the graph, when followed from top to bottom, represent a split of one of the coalitions. Whereas, when following the edges from bottom to top, a merge of 2 coalitions takes place.

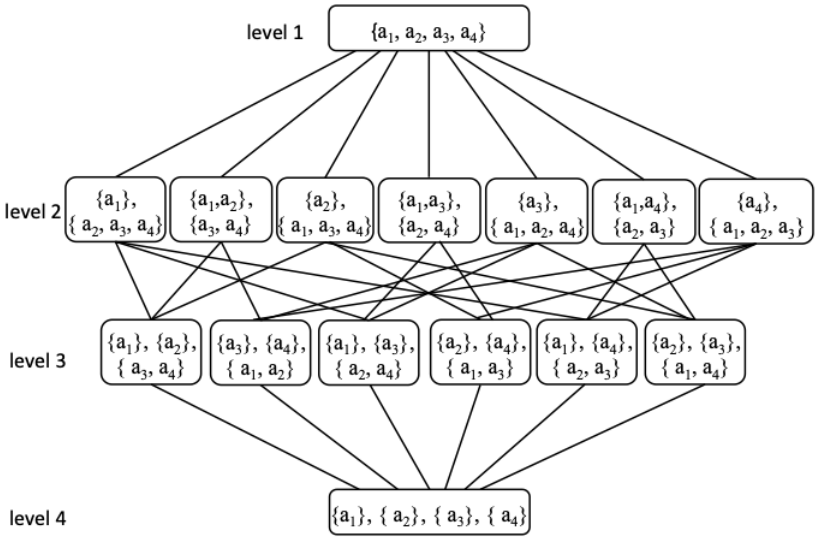

Figure 2.1: The coalition structure graph proposed by Sandholm et al. (1999).

Sandholm et al. (1999) established a bound β, such that $\beta = n$, on the quality of the solution when searching the top 2 levels. The bound continues to improve by searching through the remaining levels one by one starting from the bottom level. It is worth noting that, searching through the bottom level improves the bound to $\beta = \frac{n}{2}$.

The downside to this algorithm is that it cannot be applied to the CSG problem when the coalition structure size is fixed. The main reason is that the search algorithm is given in terms of levels. There is no detailed process for traversing the coalition structures in a given level. In addition, bounds are established from searching coalition structures of specific sizes, that do not necessarily match the optimal coalition structure size. For the sake of completeness, it is worth mentioning here that a subsequent algorithm based on the coalition structure graph was proposed by Dang and Jennings (2004).

Subsequently, Rahwan et al. (2009) designed a branch and bound algorithm based on integer partitioning (IP) to solve the CSG problem. The IP algorithm divides the search space with respect to the integer partitions of the agents. An integer partition of a number is all the possible ways of splitting the number into parts which sum up to that integer. The set of all partitions of a number n is denoted as $P(n)$.

For example, there are 5 integer partitions for the number 4, as shown in Figure 2.2, i.e., $P(4) = \{P[4], P[3,1], P[2,2], P[2,1,1], P[1,1,1,1]\}$. The search space is divided based on these partitions. There is a unique subspace associated with every partition where the partition acts as a pattern for the coalition structures in that subspace. For example, the subspace which corresponds to the partition $(2,2)$ contains the coalition structures $\{\{1,2\}\{3,4\}\}, \{\{1,3\},\{2,4\}\}, \{\{1,4\},\{2,3\}\}$.

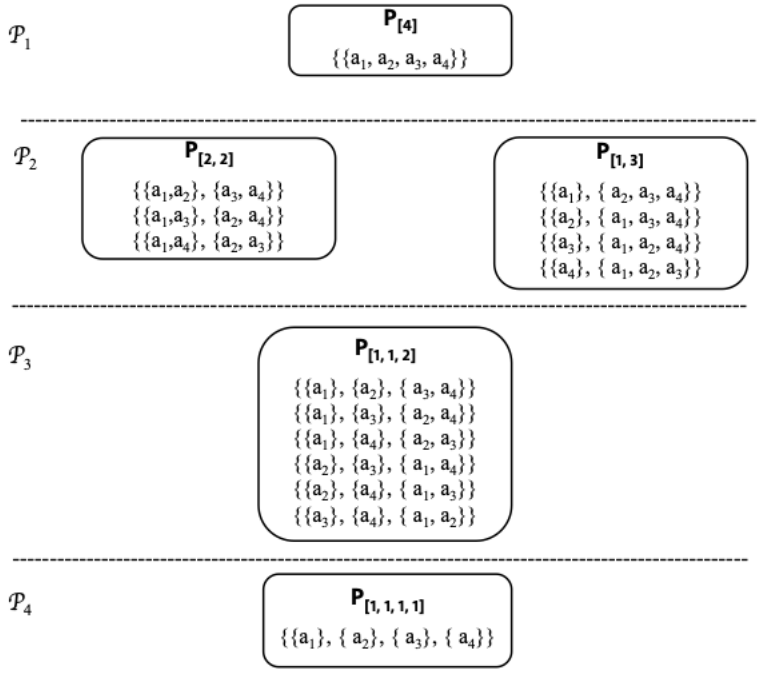

Figure 2.2: The integer partition graph proposed by Rahwan et al. (2009).

When scanning the input, the IP algorithm searches through the coalition structures of sizes 1 and 2 as proposed by Sandholm et al. (1999) to establish a bound on the solution. However, as explained earlier, it is not possible to make use of this bound due to the special structure of our problem. Then IP partitions the number of agents to form the subspaces and upper and lower bounds are computed for every subspaces. These bounds are computed using simple operations that can be performed while scanning the input.

Having computed the subspaces' bounds, it is now possible to prune entire subspaces by comparing their upper bounds to the highest lower bound. The next step is to choose a subspace and search through it. The search is based on a depth first search approach with a simple branch and bound applied. The advantage of this algorithm is that it provides an efficient technique to search through all the coalition structures without repetition. The IP representation allows us to choose the subspaces based on the coalition structure size which makes the algorithm suitable for our problem structure. However, apart from the bounds proposed by Sandholm et al. (1999), searching within a subspace does not improve the bound. Hence, the decision of using the algorithm depends on the size of the search space we are able to eliminate by comparing the subspaces' bounds. Moreover, the IP algorithm was efficient for problems of characteristic function drawn from certain distributions. However, Service and Adams (2011) constructed a distribution in which the performance of the IP algorithm deteriorated significantly.

2.2.2.3 Dynamic Programming Algorithms

A DP algorithm for solving the complete set partitioning problem, which is equivalent to the CSG problem, was presented by Yun Yeh (1986). The algorithm evaluates all the possible splittings of non-singelton coalitions and compares them to the coalition's value. Upon making a decision on whether it is beneficial to split the coalition and the best split, these entries are stored in a table.

Subsequently, these entries are used to evaluate all the possible ways to replace the grand coalitions with a coalition structure of size 2. Then every coalition in the resultant split is evaluated in the same way. This splitting process can be visualised by moving through the levels of the coalition structure graph from top to bottom.

The downside to this algorithm is that we cannot isolate coalition structures of a specific size and apply the 'splitting' without seeing coalition structures of smaller sizes. Rahwan and Jennings (2008) proposed an improved dynamic programming (IDP) algorithm which uses less memory and makes fewer calculations than DP. However, since IDP uses the same splitting method in DP, it is not efficient to apply these algorithms when searching for coalition structures with a specific size. The same holds for the anytime dynamic programming algorithm and hybrid algorithm proposed by Rahwan et al. (2012) and Michalak et al. (2016) respectively.

2.3 The Travelling and Mutilple-Travelling Salesman Problems

In this section, we present the travelling salesman problem (TSP) and the multiple-travelling salesman problem (mTSP). The TSP is used as a valuation function for the CSG simulations in Chapter 5. In addition, The mTSP is used as a model problem to evaluate the approach we present in Chapter 6 for solving CSG problems combinatorial optimisation games. The mTSP is a generalisation of the classic TSP which is $\mathcal{NP}$-hard. Sections 2.3.1 and 2.3.2 give the definitions and mathematical formulations of the TSP and mTSP respectively. Moreover, Section 2.3.1 presents Christofides's algorithm to approximate the TSP along with some necessary definitions from graph theory.

2.3.1 The Travelling Salesman Problem

The travelling salesman problem (TSP) is the problem of finding a minimal cost tour visiting a set of prescribed cities exactly once, where the tour starts and ends at a point usually called the depot.

We now give the formulation of the TSP for a complete undirected graph G of $m+1$ cities, where city 0 is the depot and the cost of travelling from city i to j, i.e., along the edge (i,j), is denoted as $c_{i,j} \in \mathbb{R} \forall i \neq j$. A binary variable $x_{i,j}$ can be used for each ordered pair of cities to denote whether or not city j follows city i in the tour. In addition, every city apart from the depot, $i \neq 0$, is associated with a variable u_i that specifies the order in which the city is visited. Hence, the TSP can be formulated as follows:

$$min \sum_{i=0}^{m} \sum_{i \neq j, j=0}^{m} c_{i,j} x_{i,j} \tag{2.2a}$$

$$\sum_{i=0}^{m} x_{i,j} = 1 \qquad \forall i \neq j \tag{2.2b}$$

$$\sum_{j=0}^{m} x_{i,j} = 1 \qquad \forall i \neq j \tag{2.2c}$$

$$u_i - u_j + m x_{i,j} \leq m - 1 \qquad i, j = 1, \ldots, m, i \neq j \tag{2.2d}$$

$$u_i \in \mathbb{Z} \qquad \forall i = 1, \ldots, m \tag{2.2e}$$

$$x_{i,j} \in \{0, 1\} \qquad \forall i, j = 0, \ldots, m. \tag{2.2f}$$

Constraints 2.2b and 2.2c ensure that the salesman visits all the cities; by forcing the salesman to enter and leave every city exactly. Constraint 2.2d is called the subtour eliminating constraint (Miller et al., 1960); it guarantees that the formulation results in one tour.

The TSP can be approximated using Christofides (1976) algorithm which provides the best approximation guarantee $\beta = \frac{3}{2}$ for problems on graphs that obey the triangular inequality. The triangle inequality theorem states that the sum of the lengths of any two sides of a triangle is always greater than or equal to the length of the remaining side. The first step in the algorithm is to find a minimum spanning tree T for the graph. It is defined as follows:

Definition 2.13. For a connected undirected graph $G = (V, E)$, a spanning tree is a subgraph that covers all the vertices in V without forming a cycle. A minimum spanning tree (MST) is lowest weight tree that connects the vertices in V without any cycles.

A MST can be generated using Kruskal's algorithm explained later in this section. Then a subgraph is induced from T containing the vertices that have odd degree. We need to find a minimum weight perfect matching for the induced subgraph. A matching is defined as:

Definition 2.14. Given a graph G = (V, E), a matching M in G is a set of edges such that no two edges share a common vertex.

Afterwards, we combine the edges of the matching M and the minimum spanning tree T. This will form a multigraph, defined as:

Definition 2.15. A multigraph is a graph that permits more than one edge to connect the same vertices.

Then we form an Eulerian circuit in the multigraph. An Euler path is defined as:

Definition 2.16. An Euler path is a path that visits every edge of a graph exactly once. If the path ends at the initial vertex then it is an Euler circuit.

And finally, form a Hamiltonian circuit by skipping repeated vertices. A Hamiltonian path is defined as:

Definition 2.17. A Hamiltonian path is a path that passes through each vertex exactly once. If the path ends at the initial vertex then it is a Hamiltonian circuit.

To construct an MST, G_{MST}, for a graph $G = (V, E)$ using Kruskal's algorithm (Kruskal, 1956) we start with an empty graph $G_{MST} = (\phi, \phi)$. Then we add the smallest edge in G to G_{MST}. Next, we add the second smallest edge to G_{MST} given that it does not form a cycle. We repeat this step until all the vertices in V are covered by E_{MST}. A pseudocode for Kruskal's algorithm is given in Algorithm 1.

Algorithm 1: Kruskal's Algorithm (Kruskal, 1956)

```
1: sort all edges in E in ascending order
2: G_MST = (V_MST, E_MST)
3: V_MST = φ and E_MST = φ
4: while V_MST ≠ V do
5:    d_{i,j} = min(E)
6:    if G' = (V_MST ∪ {i, j}, E_MST) ∪ {d_{i.j}}) is a tree then
7:       V_MST = V_MST ∪ {i, j}
8:       E_MST = E_MST ∪ {d_{i.j}})
9:    end if
10:   E = E \ d_{i.j}
11: end while
```

2.3.2 The multiple-Travelling Salesman Problem

Given a set A of $m+1$ cities and a set of k travelling salesmen, we consider the problem of minimising the total cost for the salesmen to collectively travel to these n cities. A salesman is supposed to leave the home city 0, travel to a unique set of cities and return to the home city. The cost for travelling between the given cities is represented as a complete undirected graph using an adjacency matrix, such that $c_{i,j} = c_{j,i}$ denotes the

cost of travelling from i to j and vice versa. In addition, all the costs are positive, It can be formulated as follows: while ensuring that the salesman visits all the cities.

$$min \sum_{i=0}^{m} \sum_{i \neq j, j=0}^{m} c_{i,j} x_{i,j} \tag{2.3a}$$

$$s.t. \sum_{j=1}^{m} x_{0,j} = k, \tag{2.3b}$$

$$\sum_{j=1}^{m} x_{j,0} = k, \tag{2.3c}$$

$$\sum_{i=0}^{m} x_{i,j} = 1, \qquad j = 1, \ldots, m, \tag{2.3d}$$

$$\sum_{j=0}^{m} x_{i,j} = 1, \qquad i = 1, \ldots, m, \tag{2.3e}$$

$$u_i - u_j + m x_{i,j} \leq m - 1 \qquad i, j = 1, \ldots, m, i \neq j \tag{2.3f}$$

$$u_i \in \mathbb{Z} \qquad \forall i = 1, \ldots, m \tag{2.3g}$$

$$x_{i,j} \in \{0, 1\}, \qquad i, j = 0, \ldots, m \tag{2.3h}$$

As in the TSP, the objective function aims to minimise the travel cost of the salesmen. Furthermore, the constraints (3.2b) and (3.2c) impose that all the salesmen leave and return to the depot. The constraints (3.2d) and (3.2e) guarantee that each city is visited and departed exactly once. Finally, the subtour eliminating constraint (3.2f) is due to Miller et al. (1960)An illustrative example of the mTSP is given below:

Example 2.1. *Consider an mTSP with 2 salesmen and 3 cities. The distance between the cities and depot is given in Figure 2.3 where node 0 is depot. There are 3 possible routes for the salesmen to visit each city only once starting from the depot and returning to it* $\{\{1, 2\}, \{3\}\}$ *,* $\{\{1, 3\}, \{2\}\}$ *and* $\{\{2, 3\}, \{1\}\}$ *regardless of the identity of the salesmen. The values of these routes are* $15 + 4 = 19$*,* $14 + 10 = 24$ *and* $13 + 10 = 23$ *respectively. Hence, the optimal route is the first one since it has the lowest value.*

2.4 Summary

In this chapter, we introduced the key notions of cooperative game theory and CSG. To do so, we gave an overview of general cooperative games, combinatorial optimisation games, overlapping coalition formation games, threshold task games, interval cooperative games which we use to model bounds of combinatorial optimisation games (objective 4). Furthermore, we gave an overview of compact representation schemes for cooperative

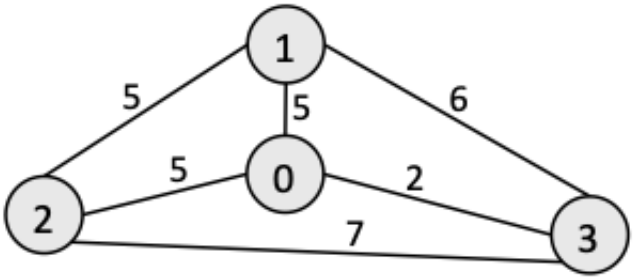

Figure 2.3: An illustration of the mTSP.

games and the complexity of the CSG problem when utilising these representations. In addition, we presented some models of non-overlapping coalitional games related to the extended model of TTGs we present in the next chapter.

Later on, we presented formal definitions of the coalition structure generation problem. We mention that the CSG problem cannot be defined on the OCF-G model, we address this issue in the next chapter (objective 2). We also reviewed a number of the algorithms designed to solve the CSG problem and analyse their applicability in solving the CSG problem for fixed size coalition structures (objective 5). In addition, graph constrained CSG was introduced due to its applicability to represent constraints that can restrict the cooperation between agents. Finally, we presented the TSP and mTSP problem which we use to evaluate the algorithms we propose (objectives 3 & 4).

Chapter 3

Multi-Resource Threshold Task Games and Coalition Structure Generation

In this chapter, we present a discrete model for overlapping coalition formation games. We refer to it as the multi-resource threshold task game (MR-TTG) model, which is a generalisation of the model of threshold task games introduced by Chalkiadakis et al. (2010). We then define the coalition structure generation Problem for MR-TTGs. As a step towards solving the CSG problem for MR-TTGs, we show two possible mappings for the problem to well-known knapsack problems: the Bounded Multidimensional knapsack problem (BMKP) and the multiple-choice multidimensional knapsack problem (MMKP). Through these mappings, we could make use of the existing literature on knapsack problems.

3.1 Model

A multi-resource threshold task game (MR-TTG) is defined by a set of n agents $A = \{1, \ldots, n\}$, a set of m resource types $R = \{1, \ldots, m\}$ and a set $T = \{1, \ldots, q\}$ of q task types. For each task type $k \in \{1, \ldots, q\}$, its demand d_k, indicates the number of copies of the task of type k. Each agent $i \in A$ is associated with a vector of resources $r^i = (r^i_1, \ldots, r^i_m)$, where $r^i_j \in \mathbb{N}_0$ is the integer weight that agent i possesses of each resource $j \in R$. Each task $k \in T$ is described by a value $v_k \in \mathbb{N}$ and a vector of thresholds $\tau_k = (\tau_{1k}, \ldots, \tau_{mk})$, where $\tau_{jk} \in \mathbb{N}_0$ denotes the weight of resource j needed to complete a task of type k. For a copy $l = 1, \ldots, d_k$ of a task type $k = 1, \ldots, q$, a partial coalition C_{kl} is given by a vector of integers indicating the amount of each resource that the agents contribute towards the completion of this task. If this amount

meets the requirement given by the threshold vector τ_k, the value of the coalition is v_k, and is 0 otherwise.

Definition 3.1. A partial coalition C_{kl} is represented as a vector of size m that details the contribution of each agent to the task kl, $C_{kl} = (\bar{w}_{1kl}, \ldots, \bar{w}_{mkl})$, where $\bar{w}_{jkl} = (w^1_{jkl}, \ldots, w^n_{jkl})$; w^i_{jkl} is the integer weight that agent i allotted of his resource j to C_{kl}. Thus $\forall\, i \in A$ and $\forall\, j \in R$, $\sum_{k=1}^{q} \sum_{l=1}^{d_k} w^i_{jkl} \leq r^i_j$.

Definition 3.2. The value of a partial coalition $v(C_{kl}) = v_k$ if and only if $\sum_{i=1}^{n} w^i_{jkl} \geq \tau_{jk}$, $\forall\, j \in R$ and $v(C_{kl}) = 0$ otherwise.

3.2 CSG in Multi-Resource Threshold Task Games

In this section, we give necessary definitions on coalition structures for the MR-TTG model. Moreover, we formulate the CSG problem for the proposed model and analyse its complexity.

A coalition structure for an MR-TTG is defined as follows:

Definition 3.3. A Coalition Structure CS is a set of partial coalitions which satisfies $\sum_{C_{kl} \in CS} w^i_{jkl} \leq r^i_j, \forall\, i \in A, j \in R, k \in T$. Let $CS_k \subseteq CS$ be a set that contains all the partial coalitions working on task type k, then $\cup_{k=1}^{q} CS_k = CS$ and $|CS_k| \leq d_k$, implying $|CS| \leq \sum_{k=1}^{q} d_k$.

Accordingly, the set of all feasible coalition structures for an MR-TTG is defined as:

Definition 3.4. The set of feasible coalition structures is denoted by $\mathcal{CS}$. Thus, $CS \in \mathcal{CS}$ if and only if $\sum_{C_{kl} \in CS} w^i_{jkl} \leq r^i_j, \forall\, i \in A, j \in R, k \in T$ and $|CS_k| \leq d_k$.

The coalition structure generation problem is the problem of finding a coalition structure of a maximal value. Using Definition 3.3 of a coalition structure, it can be formally defined as follows:

Definition 3.5. The coalition structure generation problem for an MR-TTG is the problem of finding the coalition structure $CS \in \mathcal{CS}$ which maximises the sum of the values of all partial coalitions $C_{kl} \in CS$:

$$max_{CS \in \mathcal{CS}} \sum_{C_{kl} \in CS} v(C_{kl}) \tag{3.1}$$

We analyse the complexity of the CSG problem defined on a discrete single resource TTG instead of an MR-TTG to simplify the analysis and proof. Moreover, the KP we consider has one copy of every item, i.e., there are no item types. We first prove that the problem is $\mathcal{NP}$-hard then compare the complexity of the problem had it been defined on an MR-TTG.

Theorem 3.6. *The CSG problem on a discrete TTG is weakly $\mathcal{NP}$-hard.*

Proof. We reduce from the 0-1 knapsack problem (KP), which is weakly $\mathcal{NP}$-hard (Hans et al., 2004), formally defined as:

$$max \qquad \sum_{k=1}^{q} p_k \cdot x_k \qquad x_k = 0, 1 \tag{3.2}$$

$$s.t. \quad \sum_{k=1}^{q} w_k \cdot x_k \leq C \tag{3.3}$$

The proof proceeds as follows: (1) The KP is mapped to an instance of a TTG; (2) a solution for the KP is constructed from the optimal solution for the CSG on a TTG; (3) we show that the optimality of the solution to the CSG guarantees the optimality of the KP from step (2).

First, given an instance of a KP, a TTG of 1 player is constructed as follows: the knapsack dimension c is mapped directly to the TTG resource r and the KP items are mapped to tasks $T = \{1, \ldots, q\}$. Each task $k \in T$ is described by a value $v_k = p_k$ and a threshold τ. The knapsack capacity c is assigned to the player's resource. I.e., $r^1 = c$

Secondly, let CS^* be an optimal coalition structure for the constructed TTG. A solution, $\mathbf{x}$, to the KP can be derived from CS^* as: the number of items to be packed in the knapsack is equal to the size of the optimal coalition structure $|CS^*|$, more specifically, there are $|CS^*_k|$ copies of each item k, i.e., $x_k = |CS^*_k|$. This solution respects the KP constraints eq (3.2) since $|CS^*_k|$, there is one copy ot every item k.

Finally, we prove by contradiction that $\mathbf{x}$ is optimal. Let us assume that the derived solution, $\mathbf{x}$, is not optimal. This assumption implies the existence of another solution, $\mathbf{x}'$, to the KP with a greater payoff than $\mathbf{x}$, i.e., $\sum_{k=1}^{q} p_k \cdot x'_k > \sum_{k=1}^{q} p_k \cdot x_k$ and $\sum_{k=1}^{q} w_k \cdot x'_k \leq c$. Since $c = r^1$ and $w_k = \tau_k$ we can write the last inequality as $\sum_{k=1}^{q} \tau_k \cdot x'_k \leq r^1$. This means that there exist a coalition structure CS such that players 1 can accomplish the tasks x'_k, i.e., $|CS_k| = x'_k$, $k = 1, \ldots, q$. The value of this coalition structure can be calculated as $\sum_{k=1}^{q} v_k \cdot |CS_k|$ which is equal to is $\sum_{k=1}^{q} p_k \cdot x'_k$ since $p_k = v_k$. This leads to a contradiction since $v(CS) = \sum_{k=1}^{q} p_k \cdot x'_k > \sum_{k=1}^{q} p_k \cdot x_k = v(CS^*)$.

□

Having proved that the CSG problem on a TTG in $\mathcal{NP}$-hard, it is easy to see that the CSG problem on an MR-TTG is also $\mathcal{NP}$-hard since the MR-TTG is a generalisation of the TTG. However, the problem defined on an MR-TTG is strongly $\mathcal{NP}$-hard since it is reduced to the bounded multidimensional knapsack problem (BMKP), discussed in the next section, which is strongly $\mathcal{NP}$-hard at 2 dimensions (Hans et al., 2004). The

KP is considered weakly $\mathcal{NP}$-hard because it has a fully polynomial time approximation schemes (FPTAS).

3.3 Reduction to BMKP

In this section, we show a mapping of the problem to a bounded multidimensional knapsack problem (BMKP). BMKP is informally defined as: There is a knapsack with m dimensions and a set of q item types. Each item type $k = 1, \ldots, q$ is characterised by a profit p_k and a vector of weights w_k to specify its dimensions, where w_{jk}, $j = 1, \ldots, m$ is the weight of the of j'th dimension of item type k. Besides, there is a limited number of copies of each item type k, denoted by b_k, the bound of k. The problem is to maximise the profit of items to be packed in the knapsack by packing at most b_k copies of item type k while adhering to the capacity constraints $c_j, j = 1, \ldots, m$.

Definition 3.7. The bounded multidimensional knapsack problem is formally defined as:

$$\begin{aligned} max \sum_{k=1}^{q} p_k \cdot a_k \qquad & x_k = 0, \ldots, b_k \\ s.t. \sum_{k=1}^{q} w_{jk} \cdot x_k \leq c_j \quad & j = 1, \ldots, m. \end{aligned} \tag{3.4}$$

The CSG problem for an MR-TTG can be easily mapped to a BMKP. The task types are mapped directly to item types along with their attributes; the resource thresholds, demand and value correspond to the weight vector of an item, its value and bound consecutively. Although, in our model, each agent has his own possession of the various resources, the value gained by completing a task is independent of the contributing agents. The only constraint enforced on resource consumption, as shown Definition 3.3, is the sum of all agents' possessions of that certain resource. This sum is mapped to the knapsack capacity such that each resource type corresponds to one of the dimensions. When inferring the partial coalitions in the optimal coalition structure from the solution of the BMKP, we directly re-map the packed items' copies to successful tasks. However, that gives as no information regarding the identity of the agents involved in each task. In order to satisfy Definition 3.1 of a partial coalition, we need re-distribute the agents' resources among completed tasks. This transformation is presented in more detail in Section 3.3.

In this section we prove that the CSG problem for an MR-TTG can be reduced to a BMKP. In addition, we give an algorithm to generate the optimal coalition structure from the outcome of the reduction.

Theorem 3.8. *The Coalition Structure Generation problem for an MR-TTG can be reduced in a polynomial time to a BMKP.*

Proof. The proof proceeds as follows: (1) The CSG problem is mapped to an instance of the BMKP; (2) a coalition structure is constructed from the optimal solution to the BMKP; (3) we show that the optimality of the solution to the BMKP guarantees the optimality of the coalition structure from step (2).

First, a BMKP is constructed from a CSG problem for an MR-TTG. Given the input to the CSG problem, each task type k can be mapped to an item type k in the corresponding BMKP. Item type k is described by a profit $p_k = v_k$ and the weights vector $w_k = \tau_k$. In addition, the demand d_k corresponds to the bound b_k (number of copies available of each item). Finally, the capacity of the knapsack is calculated as $c_j = \sum_{i=1}^{n} r_j^i, \forall\, j = 1, \ldots, m$.

Secondly, we give a polynomial time algorithm of complexity $O(\sum_{k=1}^{q} a_k \cdot m \cdot n)$ to translate an outcome of a BMKP to a coalition structure for an MR-TTG (see algorithm 2). The outcome has a_k copies of item k and a profit of $\sum_{k=1}^{q} p_k \cdot a_k$. The input of the algorithm is the set of agents A, agents' possessions r^i, the set of task types T, and the outcome of the BMKP. The algorithm distributes agents' resources among partial coalitions so that each partial coalition is satisfied; it assigns a value to the variable w_{jkl}^i (line 7). From the loops in lines 3 and 4 we can see that the number of partial coalitions formed is $\sum_{k=1}^{q} a_k$, and specifically the number of partial coalitions formed of type k is a_k. Therefore, $|CS_k| = a_k \leq b_k = d_k$, from the definition of the BMKP. In addition, the condition in line 9 ensures that an agent i does not contribute more than a total of $r_j^i, \forall\, j \in R$ in all the partial coalitions i has joined. Hence, the outcome of a BMKP satisfies the properties of a coalition structure after algorithm 2 has run.

Finally, we prove by contradiction that the generated coalition structure CS is optimal. Let us assume that CS is not optimal, this assumption leads to the existence of another coalition structure CS' with a value greater than the value of CS; $\exists\, CS' s.t. \sum_{C_{kl} \in CS'} v(C_{kl}) > \sum_{C_{kl} \in CS} v(C_{kl})$. Suppose that the number of partial coalitions working on a task of type k in CS' is $a'_k (|CS'_k| = a'_k)$ and for CS we know from algorithm 2 that $|CS_k| = a_k$ then the previous inequality can be written as $\sum_{k=1}^{q} v_k \cdot a'_k > \sum_{k=1}^{q} v_k \cdot a_k$. Also, from the feasibility of CS', the resource constraint $\sum_{C_{kl} \in CS'} w_{jkl}^i \leq r_j^i$ can be generalised to $\sum_{C_{kl} \in CS'} \sum_{i=1}^{n} w_{jkl}^i \leq \sum_{i=1}^{n} r_j^i$. Since it is optimal to allocate exactly τ_k to each partial coalition C_{kl} due to monotonicity $(\sum_{i=1}^{n} w_{jkl}^i = \tau_{jk}, \forall\, j \in R)$, the resource constraint can be re-written as $\sum_{k=1}^{q} \tau_{jk} \cdot a'_k \leq \sum_{i=1}^{n} r_j^i, \forall\, j \in R$. Since $\tau_k = \bar{w_k}, d_k = b_k$ and $v_k = p_k \,\forall\, k \in T$ and $c_j = \sum_{i=1}^{n} r_j^i \,\forall\, j \in R$ then there is a feasible solution to the BMKP with a'_k copies of item k, resource requirement $w_{jk} \cdot a'_k \leq c_j, \forall\, j = 1, \ldots, m$ and value $\sum_{k=1}^{q} p_k \cdot a'_k > \sum_{k=1}^{q} p_k \cdot a_k$ - contradiction since it outvalues the optimal solution.

□

Algorithm 2: Redistribute agents' resources

$w^i_{jkl} = 0, \forall\, i \in A, j \in R, k \in T, l \leq a_k$
$i = 1$
for all tasks types $k \in T$ **do**
  for all copies $l \leq a_k$ of k **do**
    for all $j \in R$ **do**
      repeat
        $w^i_{jkl} = \min\,(r^i_j, \tau_{jk} - \sum_{i=1}^{n} w^i_{jkl})$
        $r^i_j = r^i_j - w^i_{jkl}$
        if $r^i_j = 0$ **then**
          $i = (i+1) \mod n$
        end if
      until $\sum_{i=1}^{n} w^i_{jkl} = \tau_{jk}$
    end for
  end for
end for

Using the mapping from Theorem 3.8, solving the CSG problem for MR-TTG would consist of two steps: (1) Solving a bounded multidimensional knapsack problem, discussed in Section 4.1 and (2) distributing the agents' resources among successful partial coalitions as given in Algorithm 2.

3.4 Reduction to MMKP

Another method to address the CSG problem for an MR-TTG is to transform it to a multiple-choice multidimensional knapsack problem (MMKP) informally defined as: There is a knapsack with m dimensions and a number of classes, each of which corresponds to a set of items. Each item is associated with a profit and and a vector of m weights to specify the item's dimensions. The problem is to maximise the values of items to be packed in the knapsack by choosing exactly one item from each class while adhering to the knapsack constraints. The MMKP is formally defined as:

$$\begin{aligned} max \quad & \sum_{y=1}^{\upsilon} \sum_{g=1}^{h^y} p^y_g \cdot x^y_g && y = 1, \ldots, \upsilon \\ s.t. \quad & \sum_{y=1}^{\upsilon} \sum_{g=1}^{h^y} w^y_{jg} \cdot x^y_g \leq C_j && j = 1, \ldots, m \\ and \quad & \sum_{g=1}^{h^y} x^y_g = 1 && g = 1, \ldots, h^y \\ & x^y_g \in \{0, 1\} \end{aligned} \tag{3.5}$$

This section shows how to construct an MMKP from a given BMKP. From theorem 3.8, a reduced CSG problem for an MR-TTG can be used as input to the following mapping to achieve the transformation to an MMKP.

Let C be a multiset such that $k \in C$, $\forall\, k = 1, \ldots, q$ and the recurrence of item $k \in C$ (multiplicity of k) is denoted by $m_{k \in C} = b_k$, $\forall\, k \in C$. Thus, $|C| = \sum_{k=1}^{q} b_k$.

Suppose that C is partitioned into an arbitrary number υ of multisets C^y, $y = 1, \ldots, \upsilon$ and let $Y = \cup_{y=1}^{\upsilon} \{\mathcal{P}(C^y)\}$, where $\mathcal{P}(C^y)$ denotes the power set of C^y. For an arbitrary multiset $S \in \mathcal{P}(C^y)$, assuming all the items it holds $k \in S$ are required to be packed at a time, the profit and vector of weights for S are $p^S = \sum_{k \in S} p_k$ and $w^S = \sum_{k \in S} w_k$ respectively.

The set Y represents a multiple-choice multidimensional knapsack problem with υ classes. Furthermore, the power set $\mathcal{P}(C^y)$ constitutes the classes $y = 1, \ldots, \upsilon$. The objective function, as shown below, restricts the number of sets of items to be packed in the knapsack out of each class to exactly one. Formally, the transformed MMKP problem is defined as:

$$\begin{aligned} max & \sum_{y=1}^{\upsilon} \sum_{S \in \mathcal{P}(C^y)} p^S \cdot x^S \\ s.t. & \sum_{y=1}^{\upsilon} \sum_{S \in \mathcal{P}(C^y)} w_j^S \cdot x^S \leq c_j \quad \forall\, j = 1, \ldots, m \\ and & \sum_{S \in \mathcal{P}(C^y)} x^S = 1 \qquad\qquad \forall\, y = 1, \ldots, \upsilon \end{aligned} \tag{3.6}$$

Lemma 3.9. *The union of the selected set of each class $y = 1, \ldots, \upsilon$ of the outcome of the MMKP $\cup_{y=1}^{\upsilon} S$ s.t. $x^S = 1$ is the optimal solution for the BMKP.*

Proof. The union set is a valid outcome for the BMKP since it is a subset of the multiset C holding the maximum number of items that could be packed in the knapsack. In addition, the union set could be any subset of the multiset C, since it is formed of the union of the power sets of all partitions of C. Hence, the mapping does not affect the possible outcomes of the original problem. Finally, the maximisation in the objective function guarantees that optimality of the solution. □

The following example demonstrates the construction of an MMKP from a BMKP:

Example 3.1. *Given a BMKP with two item types, where $w_1 = (2, 3), p_1 = 2, b_1 = 3$ and $w_2 = (4, 1), p_2 = 3, b_2 = 2$, we show 3 different constructed MMKPs. As we can see the multiset $C = \{1, 1, 1, 2, 2\}$ can be partitioned in different ways, we construct the MMKPs of 3 of these partitions:*

First partition: $C^1 = \{1, 1, 1\}$ and $C^2 = \{2, 2\}$. It results in the power sets $\mathcal{P}\left(C^1\right) = \{\{1, 1, 1\}, \{1, 1\}, \{1\}, \phi\}$ and $\mathcal{P}\left(C^2\right) = \{\{2, 2\}, \{2\}, \phi\}$.

Second partition: $C^1 = \{1,2\}, C^2 = \{1,2\}$ *and* $C^3 = \{1\}$. *It results in the power sets* $\mathcal{P}\left(C^1\right) = \mathcal{P}\left(C^2\right) = \{\{1,2\},\{1\},\{2\},\phi\}$ *and* $\mathcal{P}\left(C^3\right) = \{\{1\},\phi\}$.

Third partition: $C^1 = \{1,1,2\}$ *and* $C^2 = \{1,2\}$. *It results in the power sets* $\mathcal{P}\left(C^1\right) = \{\{1,1,2\},\{1,1\},\{1,2\},\{1\},\{2\},\phi\}$ *and* $\mathcal{P}\left(C^2\right) = \{\{1,2\},\{1\},\{2\},\phi\}$.

When the sets in each partition are enumerated and their corresponding weight vectors and profits are computed, three different multidimensional multiple-choice knapsack problems are generated as shown in following tables:

Table 3.1: MMKP resulting from partition 1

Class 1	Class 2
$w_1^1 = (6,9), p_1^1 = 6$	$w_1^2 = (8,2), p_1^2 = 6$
$w_2^1 = (4,6), p_2^1 = 4$	$w_2^2 = (4,1), p_2^2 = 3$
$w_3^1 = (2,3), p_3^1 = 2$	$w_3^2 = (0,0), p_3^2 = 0$
$w_4^1 = (0,0), p_4^1 = 0$	

Table 3.2: MMKP resulting from partition 2

Class 1	Class 2	Class 3
$w_1^1 = (6,4), p_1^1 = 5$	$w_1^2 = (6,4), p_1^2 = 5$	$w_1^3 = (2,3), p_1^3 = 2$
$w_2^1 = (2,3), p_2^1 = 2$	$w_2^2 = (2,3), p_2^2 = 2$	$w_2^3 = (0,0), p_2^3 = 0$
$w_3^1 = (4,1), p_3^1 = 3$	$w_3^2 = (4,1), p_3^2 = 3$	
$w_4^1 = (0,0), p_4^1 = 0$	$w_4^2 = (0,0), p_4^2 = 0$	

Table 3.3: MMKP resulting from partition 3

Class 1	Class 2
$w_1^1 = (8,7), p_1^1 = 7$	$w_1^2 = (6,4), p_1^2 = 5$
$w_2^1 = (4,6), p_2^1 = 4$	$w_2^2 = (2,3), p_2^2 = 2$
$w_3^1 = (6,4), p_3^1 = 5$	$w_3^2 = (4,1), p_3^2 = 3$
$w_4^1 = (2,3), p_4^1 = 2$	$w_4^2 = (0,0), p_4^2 = 0$
$w_5^1 = (4,1), p_5^1 = 3$	
$w_6^1 = (0,0), p_6^1 = 0$	

3.5 Summary

In this chapter we introduced a discrete model for overlapping coalitional games with multiple resources and task types, the MR-TTG model. Subsequently, we formulated the CSG problem the proposed model. Furthermore, we proved the complexity of the CSG problem and reduced it to two well-known knapsack problems, the BMKP and MMKP. These reduction allow the use of algorithms that are independent of the number of agents.

Chapter 4

Algorithms for the CSG Problem in MR-TTGs

Having presented two transformations of the coalition structure generation problem in multi-resource threshold games, in this chapter we solve the resultant knapsack problems of these transformations. We present algorithms for both problems. In addition, we report the numerical results for the proposed algorithms.

4.1 Solving the BMKP

We propose a branch and bound algorithm based on a best first search to solve the MR-TTG CSG problem as a BMKP. Branch and bound is a well-known problem solving paradigm which is found effective in solving $\mathcal{NP}$-hard problems. It consists mainly of two steps: (1) branching, where a search tree is constructed to divide the problem into subproblems and (2) bounding, which involves finding upper and lower bounds for the subproblems to reduce the size of the search space. In this section, we describe in detail our branch and bound algorithm to find an optimal solution to the BMKP.

4.1.1 The Search Tree

A search tree is constructed to explore all the possible solutions for the reduced problem. The number of levels of the tree is equal to the number of item types q. Each developed node in the tree corresponds to a partial solution. A node is identified by its level λ and $a_k, k = 1, \ldots, q$; the number of copies packed of item k. Also, at a given level λ, a node cannot have any items packed of the next levels. Thus, $a_k = 0, \ldots, d_k, \forall\, k = 1, \ldots, \lambda$ and $a_k = 0, \forall\, k = \lambda + 1, \ldots, q$. A node is feasible if $c_j \geq \sum_{k=1}^{q} a_k \cdot w_{jk}, \forall\, j = 1, \ldots, m$ and it is infeasible otherwise. Furthermore, the value of a feasible node is calculated as

$\sum_{k=1}^{q} a_k \cdot p_k$, and in any given set L, the best node $\in L$, is the node with the greatest value. A child of a node at a given level is a node, in the next level, with a_k equal to its parent $\forall\, k = 1, \ldots, \lambda, \lambda + 2, \ldots, q$ and $a_k = 0, \ldots, b_k$, $k = \lambda + 1$.

4.1.2 Lower and Upper Bounds

No lower bound is calculated before running the algorithm. Since a best first search approach is adopted, the quality of the solution rapidly improves during the early steps. Moreover, because the number of tree levels is limited a reasonable lower bound is reached once a leaf node is developed; in $\sum_{k=1}^{q} d_k$ steps maximum.

An upper bound is calculated for each developed node in order to prune the search space. The dimensions of the BMKP are aggregated into a single dimension as in (Dobson, 1982) and the integrality constraints are removed. The resultant problem is a BKP with the capacity $\sum_{j=1}^{m} c_j$ and each item $k = 1, \ldots, q$ has the dimension $\sum_{j=1}^{m} w_{jk}$ and the bound b_k. The problem is formally defined as follows:

$$\begin{aligned} max & \sum_{k=1}^{q} p_k \cdot x_k \\ s.t & \sum_{j=1}^{m} w_{jk} \cdot x_k \leq \sum_{j=1}^{m} c_j \\ & 0 \leq x_k \leq b_k \end{aligned} \tag{4.1}$$

The above linear programme outcome serves as an upper bound to the BMKP and it can be solved using Dantzig's approach described in (Dantzig, 1957). The approach consists of two steps. Firstly, items are ordered descendingly with respect to their efficiency; the efficiency of an item k is calculated by $e_k = \frac{p_k}{\sum_{j=1}^{m} w_{jk}}$. Secondly, items are packed into the knapsack, in the order generated by 1, until the capacity $\sum_{j=1}^{m} c_j$ is reached.

In order to calculate the upper bound for any node at a given level λ, a subproblem of the BMKP is considered with the items $k = \lambda + 1, \ldots, q$ and the corresponding bounds $(b_{\lambda+1}, \ldots, b_q)$. The capacity of each dimension is calculated as $c_j = \sum_{i=1}^{n} r_j^i - \sum_{k=1}^{q} a_k \cdot w_{jk}, \forall\, j \in m$. Afterwards, the subproblem is mapped to a LP BKP and solved using Dantzig's approach described above.

4.1.3 Procedure

A psuedocode of the algorithm is given in Algorithm 3. Furthermore, the algorithm is summarised in the following steps:

Algorithm 3: Solving the reduced BMKP

```
1: node = root node
2: solution = node
3: L = node
4: while L ≠ φ do
5:    best = best(L)
6:    repeat
7:       k = level(best) + 1
8:       a_k = 0
9:       node = child(best, a_k) {develop child of best with a_k copies of item k}
10:      if feasible(node) then
11:         if value(node) > value(solution) then
12:            update solution
13:         end if
14:         if level(node) < q then
15:            calculate UB(node)
16:            if UB(node) > value(solution) then
17:               L = L ∪ node
18:            end if
19:         end if
20:      end if
21:      a_k = a_k + 1
22:   until a_k > b_k or not feasible(node)
23:   L = L \ best
24: end while
25: return solution
```

Initialisation The root node is developed (a node at level 0 with $a_k = 0, \forall\, k = 1, \ldots, q$), and the solution is set to the root node. Throughout the algorithm, the list L is used to keep track of the nodes whose children are to be developed; leaf nodes are not added to the list (line 14). To start with, the root node is added to L.

Branching The best node in L is selected and all its feasible children are developed in the order $a_k = 0, \ldots, b_k$ (lines 8, 21 & 22), where k is the level of the child nodes in the tree. The best node is discarded (line 23) afterwards. For each developed node, its value is calculated and the solution is updated accordingly (lines 11 & 12). Furthermore, the upper bound (UB) is calculated as shown in section 4.1.2 and only the nodes whose upper bound is greater than the incumbent solution are added to the list (lines 14 to 19).

Termination The algorithm terminates once there are no further nodes to be developed and the solution is returned, this is achieved when L is empty.

Example 4.1 illustrates the BMKP algorithm steps.

Example 4.1. *Consider a BMKP with 3 item types and 1 copy of each item. I.e.,* $q = 3, b_1 = b_2 = B_3 = 1$. *Figure 4.1 shows the order in which the search tree is expanded*

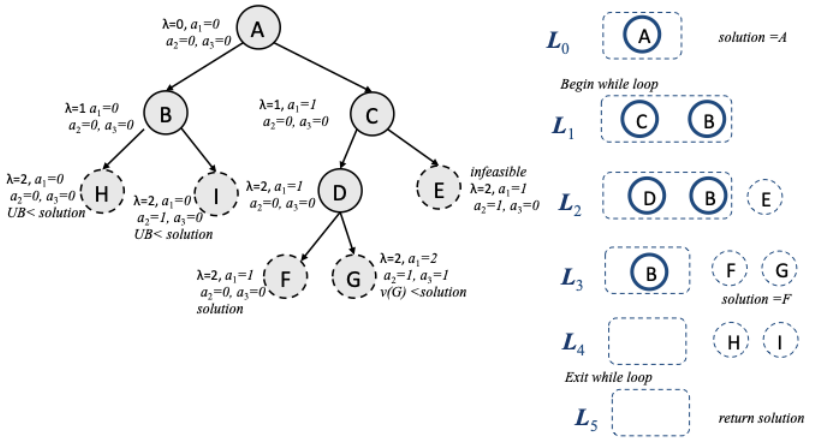

Figure 4.1: An illustration of the BMKP Algorithm.

and the nodes in list L at each iteration. For clarity, a subscript is added to L to indicate the iteration number and nodes are generated alphabetically. At the initialisation step, node A is created and added to L_0*. At iteration 1, the children of node A are created. I.e., nodes B and C and A is removed from* L_1*. The list L is sorted with the nodes of highest upper bound placed at the front of the list. At iteration 2, the children of the best node in* L_1 *are developed. I.e., the children of C. Note that the infeasible node E was not added to the list* L_2*. At iteration 3, nodes F and G are developed and the solution was updated. They were not added to* L_3 *since they are leaf nodes. At iteration 4, nodes H and I are developed. They were not added to* L_4 *since they have an upper bound lower than the solution. At iteration 5, the algorithm temrinates as* L_5 *is empty.*

4.2 Solving the MMKP

In this section, we modify the EMKP algorithm proposed by Sbihi (2007) to optimally solve the MMKP. We first present the EMKP algorithm and highlight some problems regarding it. Later on, we provide our modified version of the algorithm.

4.2.1 The Original EMKP Algorithm

The EMKP algorithm is based on a branch and bound best first search approach. In this section, the elements of the search tree are described and the branching and pruning strategies are explained.

4.2.1.1 The Search Tree

Prior to constructing the search tree, items in each class are sorted in descending order with respect to their profits. When items have the same profit, the item with the lowest $\sum_{j=1}^{m} \frac{w_{jg}^{y}}{c_j}$ is considered first.

The solution is developed gradually by selecting an item from each class in the order $1, \ldots, \upsilon$. A node with the last item from class $\gamma = 1, \ldots, \upsilon$ would consist of an item $g = 1, \ldots, h^y$ from each class $y = 1, \ldots, \gamma$. A node is feasible if the total weight of its corresponding items is less than the capacity, $\sum_{y=1}^{\gamma} w_{jg}^{y} \leq c_j, \forall\, j = 1, \ldots, m$. The profit of a node is equal to $\sum_{y=1}^{\gamma} p_g^y$ and the best node in the tree is the node with the highest profit. The root node of the tree is assigned to the first item in the first class. The search tree is constructed by developing child and sibling nodes. For a node with the last item from class $\gamma < \upsilon$, a child node is developed by adding the first item of the class $\gamma + 1$ to the parent node. On the other hand, a sibling node is developed if the item selected from class γ is $g < h^{\gamma}$. This item is replaced by the item $g + 1$ in the same class.

4.2.1.2 Bounding

For the lower bound (LB), a heuristic solution developed by the MTLS algorithm described in (Hifi et al., 2006). The upper bound (UB) is calculated from the auxiliary problem to the MMKP ($MMKP_{aux}$) defined as :

$$
\begin{aligned}
& max \sum_{y=1}^{\upsilon} \sum_{g=1}^{h^y} p_g^y \cdot x_g^y && y = 1, \ldots, \upsilon \\
& s.t. \sum_{y=1}^{\upsilon} \sum_{g=1}^{h^y} w_g^y \cdot x_g^y \leq c && c = \sum_{j=1}^{m} c_j \\
& and \sum_{g=1}^{h^y} x_g^y = 1 && g = 1, \ldots, h^y \\
& x_g^y \in \{0, 1\}
\end{aligned}
\tag{4.2}
$$

From each class $y = 1, \ldots, \upsilon$, the most profitable item g_{max} is chosen. It is the item with the highest profit to weight ratio, $g_{max} = max \frac{p_g^y}{w_g^y}, \forall\, g = 1, \ldots, h^y$. The weights of the most profitable items are summed up and noted as $w_{max} = \sum_{y=1}^{\upsilon} w_{g_{max}}^{y}$. Now, there are two possibilities:

1. If $w_{max} > c$ then the upper bound is calculated as $UB = \sum_{y=1}^{\upsilon} p_g^y \cdot \frac{c}{\sum_{y=1}^{\upsilon} w_{g_{max}}^{y}}$.
2. If $w_{max} \leq c$ then a supplementary knapsack problem is formed of union of the items of all classes except g_{max}, and the knapsack capacity is set to $c - w_{max}$.

The upper bound of the knapsack problem, UB_{KP}, is computed using Dantzig method explained in section 4.1.2. Subsequently, the upper bound is calculated as $UB = \sum_{y=1}^{\upsilon} p_g^y + UB_{KP}$.

4.2.1.3 Procedure

The EMKP algorithm can be summarised in the following steps:

Initialisation The lower bound is calculated using a heuristic algorithm. The items of each class are sorted in decreasing order of their corresponding profits. The root node, consisting of the first item in the first class, is developed.

Branching The best node in the tree is selected, and a child node is developed if the best node was feasible. Also, if exists, the sibling of the best node is developed and added to the tree. The child node is only added to the tree if its upper bound was greater than the lower bound.

Termination If the developed child node is a leaf node and is feasible.

4.2.2 Problems with the EMKP Algorithm

Bing et al. (2010) pointed out several problems in the EMKP algorithm, we have encountered most to them during the implementation of the algorithm. Ineffectiveness of the pruning strategy, because a feasible node may be developed from infeasible ones, infeasible nodes are not eliminated from the tree. This produces the problem of computing the upper bound of infeasible nodes, which is not stated in the algorithm. Moreover, the algorithm attempts to reduce the search space by excluding the nodes whose upper bound is lower than the lower bound from the search tree. As a result, the algorithm might omit the subspace containing the optimal solution from the search.

Another problems not mentioned by Bing et al. (2010) are the order of developed nodes and the optimality of the solution found at the proposed stopping condition. Proposition 3 in (Sbihi, 2007) proves that the first obtained feasible solution is the optimal solution. It is based on Lemma 1 in (Sbihi, 2007) which states that the solutions obtained by the EMKP are developed in decreasing order of their profit regardless of their feasibility state. Here, we give a counter example to falsify the proposition.

Example 4.2. *For simplicity we give an example of a multiple-choice knapsack problem (MCKP), which has one dimension, and assume that the weight of each item is equal to its profit. Consider the following MCKP, with the capacity 38, given in the following table:*

Table 4.1: An MCKP that shows the invalidity of the EMKP stopping condition and the ordering of the solutions developed.

Class 1	Class 2	Class 3
$w_1^1 = (20), p_1^1 = 20$	$w_1^2 = (12), p_1^2 = 12$	$w_1^3 = (10), p_1^3 = 10$
$w_2^1 = (17), p_2^1 = 17$	$w_2^2 = (7), p_2^2 = 7$	$w_2^3 = (3), p_2^3 = 3$
$w_3^1 = (16), p_3^1 = 16$		

For clarity, we write the nodes in terms of their profits when tracing the algorithm. Initially, the list L will include the first item of the first class, $L = \{(20)\}$. *At each step, we develop a child and a sibling for the item with the highest value. Upon the first iteration,* $L = \{(20, 12), (17)\}$. *Upon the second iteration,* $L = \{(20, 12, 10), (20, 7), (17)\}$. *Now, we could develop a sibling for, (20,12,10),the best node in L. The sibling of the best node, (20,12,3), is the first feasible solution, we could stop now according to the claim that nodes are developed in decreasing order of profit. We skip this node since it is not clear from the algorithm that we could exit even if the last item in the node is not the first item of the last class. Now,* $L = \{(20, 7), (17)\}$. *In the next iteration, the node (20, 7,10) is developed. According to the algorithm, (20, 7,10) is the optimal solution. However, it is clear that (16, 12, 10) is the optimal solution. In fact, in this example, the optimal solution is developed lastly.*

4.2.3 The Modified EMKP

Here, we present our new version of the EMKP algorithm. To reduce the execution time, we added two preprocessing steps before running the algorithm. Furthermore, we address the problems in the EMKP algorithm explained in the previous section. A pseudocode of the modified algorithm is given in Algorithm 4

4.2.3.1 Removing Dominated Items

As a preprocessing step, dominated items are removed from each class $y = 1, \ldots, \upsilon$. An item is dominated if there is another item in the same class that yields a greater profit while having less weight for each dimension.

Definition 4.1. In an MMKP, it is said that an item g is dominated by item g' if and only if $g \in y$, $g' \in y$, $w_{jg} \geq w_{jg'}, \forall\, j = 1, \ldots, m$ and $p_{g'} > p_g$.

4.2.3.2 Reducing Duplicate States

Another preprocessing step is proposed in Lemma 4.2 to reduce the number of items in each class due to the special structure of the MMKP constructed. Since classes are created by deriving power sets and the original set we partition is a multiset, many of the

classes in the MMKP might be identical. Due to that, identical nodes might be developed in the search process. In addition, in each class, there is an item which corresponds to the element $\phi \in \mathcal{P}(C^y)$. We refer to this item as the fictitious item. The existence of the fictitious items adds to the number of duplicate states that can be derived. As a result, the processing time of the algorithm would be adversely affected. This situation can be demonstrated by the following MMKP:

Consider the second partition in Example 3.1, in the resulting MMKP, selecting $\{1, 2, \}$ from $\mathcal{P}\left(C^1\right)$ and ϕ from $\mathcal{P}\left(C^2\right)$ is identical to selecting $\{1\}$ from $\mathcal{P}\left(C^1\right)$ and $\{2\}$ from $\mathcal{P}\left(C^2\right)$.

Storing all the developed nodes in a list and searching through the list to determine if a developed node has a duplicate is expensive. However, we reduce the effect of duplicates by eliminating some of the sets in classes which has duplicates. This is shown in the following lemma:

Lemma 4.2. *Suppose that the multiset C was partitioned, such that there are partitions which are identical. Then for every partition $C^{y'}$ which is identical to C^y, we can safely eliminate the sets with cardinality $\left|C^{y'}\right|$ from the power set of $C^{y'}$. Note that we preserve $\mathcal{P}(C^y)$.*

As a result, in Example 3.1, the MMKP formed by the set Y as shown in the transformation in Section 3.4 is identical to $Y' = \{\{\{1, 2, \}, \{1\}, \{2\}, \phi\}, \{\{1, 2, \}, \phi\}, \{\{1\}, \phi\}\}$.

4.2.3.3 Pruning the Search Space

No lower bound is calculated prior to running the modified algorithm since the tree nodes can serve as an incumbent solution. For feasible nodes, the upper bound is calcuated as in the EMKP algorithm. In the modified algorithm, if a node is infeasible then its upper bound is set to its parent's upper bound.

A sibling node is only developed if the upper bound of its parent is greater than the incumbent solution. When developing a sibling, instead of keeping infeasible nodes in the tree, we keep on developing sibling of a sibling nodes until a feasible one is encountered. A feasible sibling is added to the search space if it does not have an item of the class v. As in the EMKP algorithm, a child node is developed for nodes whose upper bounds are greater than the incumbent solution.

4.2.3.4 Termination Condition

Although the stopping condition of the EMKP is does not yield the optimal solution of the MMKP, we ran the algorithm according to that stopping condition. Removing the

stopping condition would produce optimal results, however, the run time is greater than the time required for solving the BMKP optimally using Algorithm 3 as our preliminary experiments suggested. The modified algorithm was run again and stopped at 0.5 milliseconds to record the incumbent (suboptimal) solution.

Algorithm 4: The Modified EMKP

```
1: node = item 1 in class 1
2: parent_UB(node) = Σ_{y=1}^{υ} p_g^y, g = 1
3: value(solution) = 0
4: L = node
5: while L ≠ φ do
6:    best = best(L)
7:    L = L \ best
8:    if last_class(best) ≠ υ and feasible(best) and UB(best) ≥ value(solution)
      then
9:       child = child(best)
10:      L = L ∪ child
11:   end if
12:   if feasible(child) then
13:      if value(child) > value(solution) then
14:         solution=child
15:      end if
16:      if last_class(child) = υ then
17:         return solution
18:      end if
19:   end if
20:   if parent_UB(best) ≥ value(solution)and has_sibling(best) then
21:      sibling=sibling(best)
22:      while notfeasible(sibling) do
23:         sibling=sibling(sibling) {there will always exist a feasible sibling due to the
            fictitious item}
24:      end while
25:      if feasible(sibling) and value(sibling) > value(solution) then
26:         solution=sibling
27:      end if
28:      if not (feasible(sibling)andlast_class(sibling) = υ) then
29:         L = L ∪ sibling
30:      end if
31:   end if
32: end while
```

4.3 Empirical Evaluation

We evaluate the performance of Algorithm 3 and Algorithm 4 in solving BMKPs. The algorithms were programmed in C++ and run on a Mac 2.9 GHz Intel Core i5 processor

and 20 GB memory. This section describes the generated data sets and presents the numerical results of the experiments performed.

4.3.1 Instance Generation

Two types of data sets were generated; uncorrelated and strongly correlated data sets with the latter being considered hard to solve (Pisinger, 2005). In uncorrelated instances, where the profit of an item is independent of its dimensions, the profits were drawn randomly from the interval [1, 100]. On the other hand, in strongly correlated instances, the profit of an item is a linear function of its weight (Pisinger, 1998). The profits were calculated as $p_k = \sum_{j=1}^{m} w_{jk} + 10$.

The number of item types was fixed throughout the experiments (q= 6), and the number of dimensions of the knapsacks was varied in the BMKP (m=5, 7, 10) and fixed in the MMKP (m=5). For each item, the weight of each dimension was drawn randomly from the interval [0, 10]. The bound b_k of item types $k = 1, \ldots, 6$ was randomly drawn in the first set of the BMKP experiments and fixed in the second set of the BMKP and all the MMKP experiments. The random intervals and bounds are discussed in each experiment.

The generated items sets were tested for knapsacks with different capacities. Zanakis (1977) introduced the term, the degree of constraint slackness. We used the formula $s_j = c_j / \sum_{k=1}^{q} w_{jk} \cdot b_k$, where, s_j is the slackness ratio of the constraint j from (Akçay et al., 2007). The slackness $s_j, \forall\, j = 1, \ldots, m$ was drawn from the intervals [0.40, 0.60], [0.60, 0.80], [0.80, 1] and [0.40, 1].

4.3.2 The BMKP Algorithm

Algorithm 3 was tested on the data sets generated in the previous section. The bound b_k was drawn randomly in these experiments from the interval [1, 50]. As a result, the total number of copies of all items was not determined earlier. Ihe number of items varied from 103 to 221. The execution times are shown in Figures 4.2, 4.3 and 4.4 in the form of boxplots. A boxplot is a diagram that shows how the data is distributed and identifies outliers in the data. A rectangle is drawn to mark the first quartile Q_1 and third quartile Q_3 of the data with the line in the middle of the rectangle representing the median. The top and bottom whiskers represent the maximum and minimum values in the dataset excluding outliers. Outliers are represented by a circle or an asterisk in relation to the interquartile range calculated as $Q_3 - Q_1$. A circle marks the outliers which are 1.5 times the interquartile range away from the first and third quartiles. In addition, an asterisk marks the outliers that are 3 times the interquartile range away from the first and third quartiles.

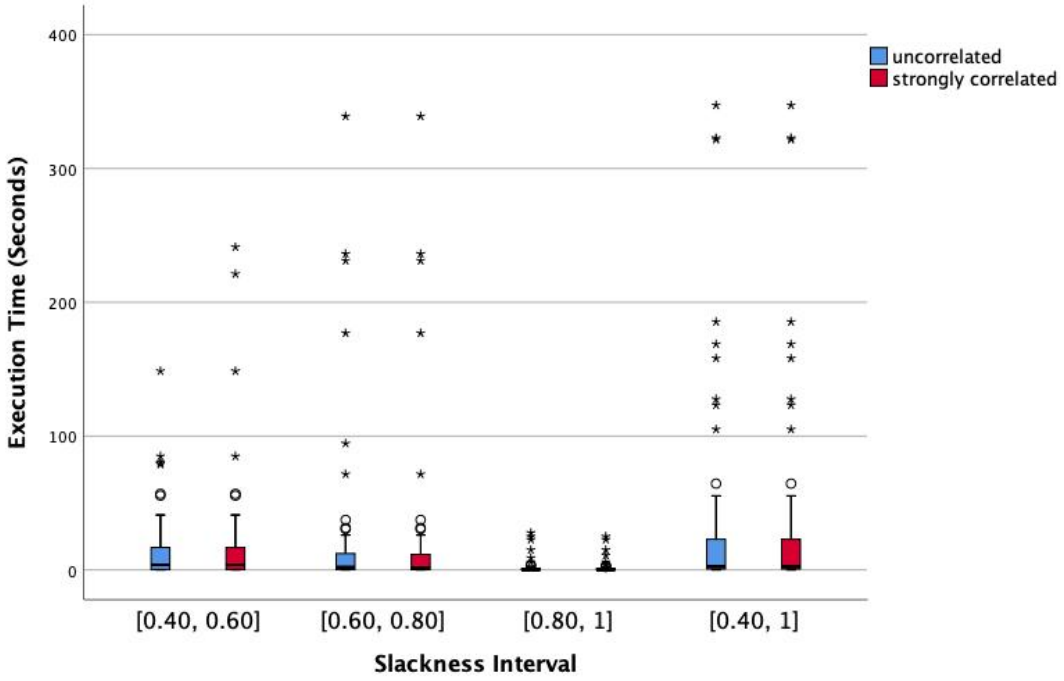

Figure 4.2: A boxplot of the execution time of 50 uncorrelated and strongly correlated instances of the BMKP, m=5 and varying number of items.

We observe from the previous figures that the execution time increases with the number of knapsack dimensions regardless of the BMKP correlation. In addition, instances with slackness interval [0.80, 1] needed the least time to execute for all dimensions and correlations. Moreover, strongly correlated instances needed the most time to execute for all slackness intervals and dimensions.

Referring to the outliers in the previous figures, we observe that the execution time changes significantly when the total number of items is above 200. In our second set of experiments, we constructed 100 instances of a BMKP with number of item types $q = 6$ and bounds $b_k, k = 1, \ldots, 6$ to the values: $b_1 = 41$, $b_2 = 5$, $b_3 = 43$, $b_4 = 16$, $b_5 = 49$ and $b_6 = 53$. As a result, the total number of items was 207 in all the instances. Also, we ran the experiments on instances of 5 dimensions. A box plot of the runtime of Algorithm 3 for 100 instances is presented in Figure 4.5

4.3.3 The Modified EMKP Algorithm

We executed the modified EMKP algorithm on the second set of experiments of the BMKP and slackness interval [0.80, 1]. In order to construct an MMKP, the multiset C is partitioned to 5 partitions of $\{1, 2, 3, 3\}$, 16 partitions $\{3, 4, 5, 5\}$, 18 partitions $\{1, 1, 6, 6\}$ and 17 partitions $\{3, 5, 6\}$ Since the number of power sets of each partition is exponential to the number of elements, we reduce the number of power sets by repetition of elements in partitions. As an example, $|\mathcal{P}\{1, 2, 3, 3\}| = 12$ while $|\mathcal{P}\{1, 2, 3, 4\}| = 16$.

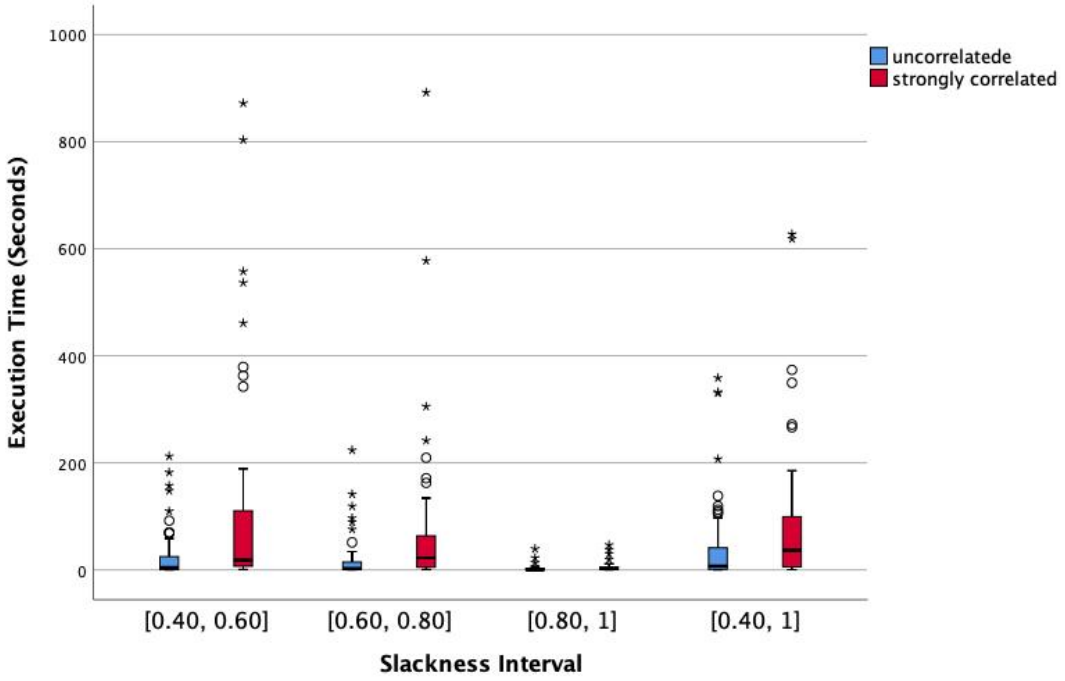

Figure 4.3: A boxplot of the execution time of 50 uncorrelated and strongly correlated instances of the BMKP, m=7 and varying number of items.

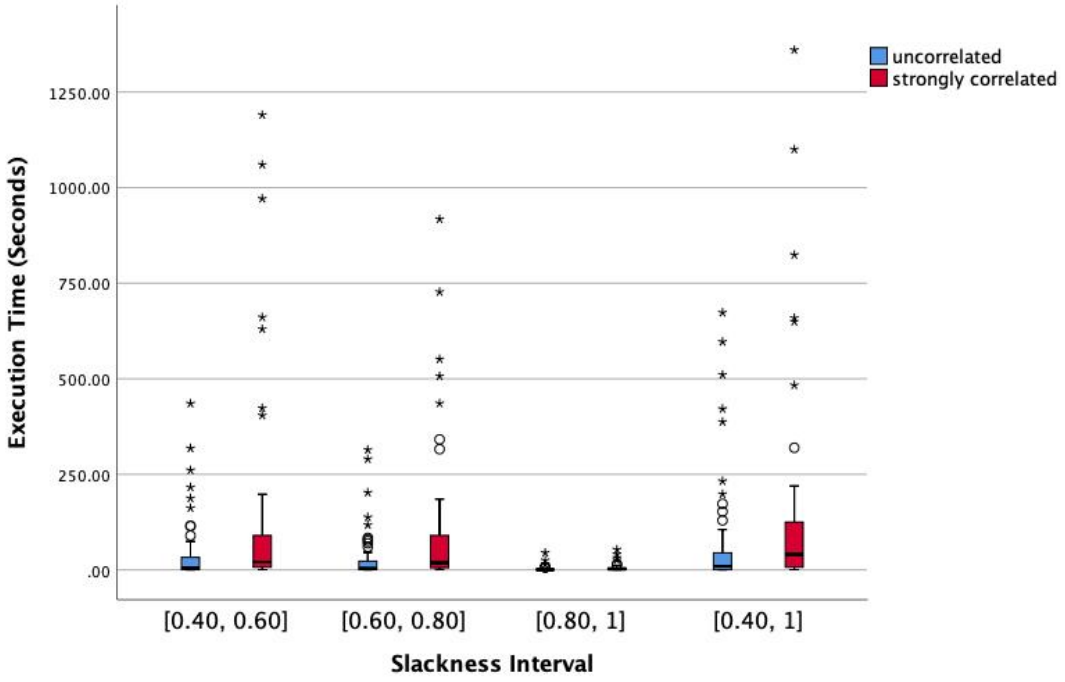

Figure 4.4: A boxplot of the execution time of 50 uncorrelated and strongly correlated instances of the BMKP, m=10 and varying number of items.

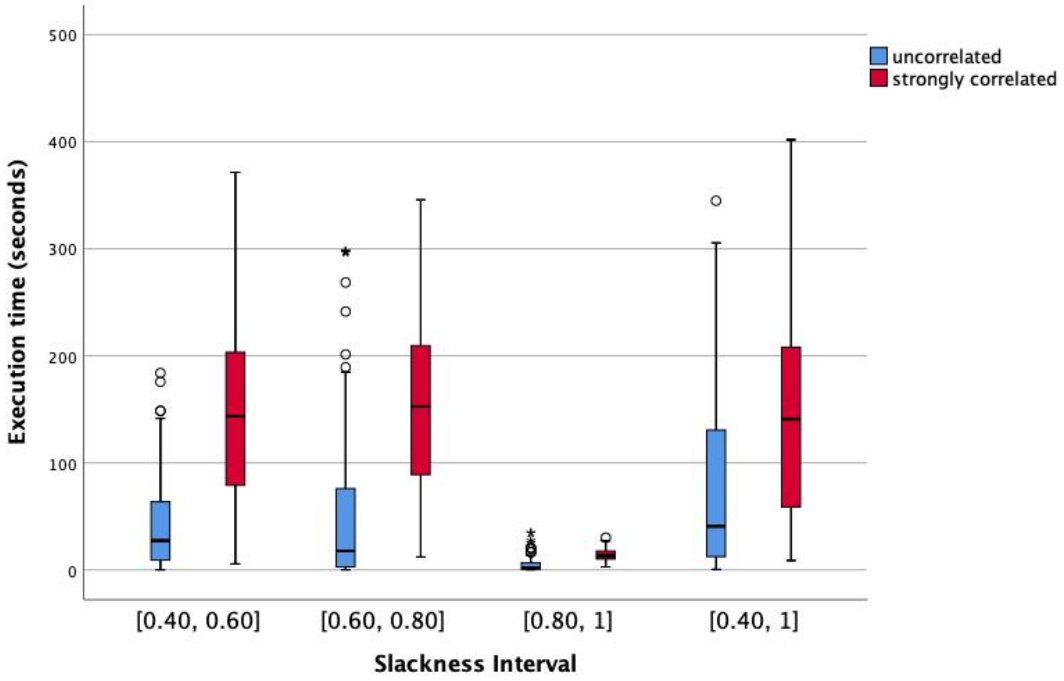

Figure 4.5: A boxplot of the execution time of 100 uncorrelated and strongly correlated instances of the BMKP, m=5 and varying number of items.

Initially, 15 instances were executed to get an idea about the algorithm's run time which excludes the time for creating the classes and the preprocessing. The algorithm terminated in a reasonable time for the interval [0.80, 1] when terminating the algorithm using the original EMKP stopping condition. However, the average execution time for strongly correlated instances was about 150 times faster than uncorrelated instances. In addition, the runtime and accuracy were more stable in strongly correlated instances. The minimum runtime in strongly correlated instances for these 15 instances is 0.037 seconds and the max is 0.0382 seconds, and the approximation ratio is 1.03 in all instances. However, in uncorrelated instances, the runtime ranged between 0.435 and 21 seconds. Likewise, the approximation ratio ranged between 1.16 and 1.011 with average ratio of 1.06. Subsequently, we ran the modified EMKP algorithm on 100 instances of 5 dimensions and stopped the algorithm after 0.s milliseconds. Figure 4.6 shows the approximation ratio of the solutions obtained after 0.5 ms. In addition, running a 2 sample t-test on the approximation ratios of the correlated and uncorrelated instances problems, we find that the mean of the uncorrelated instances (1.4388) is significantly higher than the mean of correlated instances (1.2723) with p value of 0.000008.

4.4 Summary

In this chapter, we solve the reduced problems for the CSG on MR-TTGs. For the BMKP reduction, we presented a branch-and-bound algorithm and evaluated its efficiency for

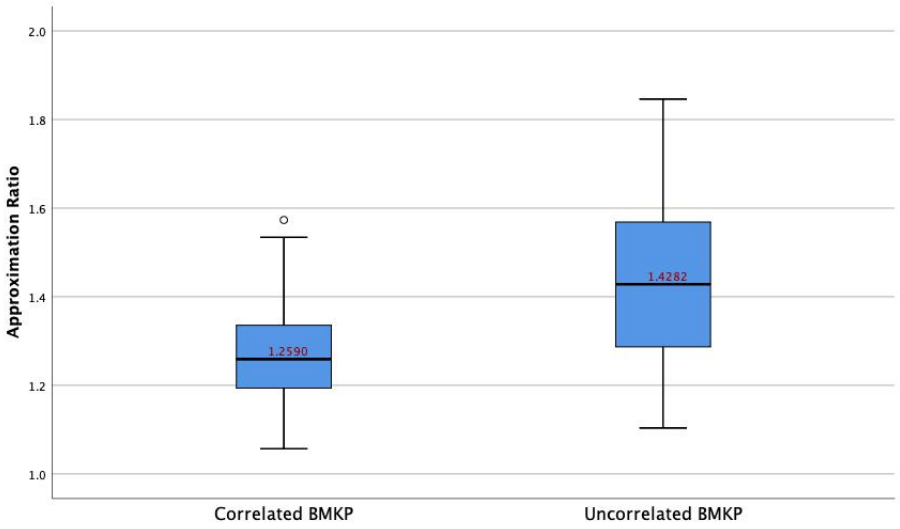

Figure 4.6: Accuracy of the modified EMKP Algorithm after 0.5 milliseconds on 100 instances and m=5.

instances of different resource numbers and availabilities. For the MMKP reduction, we modified an existing algorithm, the EMKP algorithm, and evaluated its efficiency for problems with 5 resource types as they map to a WSN. For both reduction, we tested instances with a strong correlation between the resource requirement of tasks and value and instances with no correlation. For the BMKP, as in the existing literature, strongly correlated instances were the hardest to solve in all settings. However, when using the MMKP reduction, we obtain results for the strongly correlated instances of average approximation ratio of 1.27 in 0.5 milliseconds.

Chapter 5

Cardinality Constrained CSG

This chapter starts by giving a dynamic programming formulation for the cardinality constrained CSG problem. In addition, we present a near-optimal algorithm for solving the problem. Moreover, the algorithm is evaluated in terms of efficiency and accuracy (Section 5.3).

5.1 Problem Formulation

The cardinality constrained CSG problem k can be defined by restricting the set of all feasible coalition structures $\mathcal{CS}$ in Definition 2.12 to $\{\mathcal{CS} = \{CS : |CS| = k\}$. We can view the CSG problem defined over the set A for a fixed size coalition structure k as the union of the set $\{C\}, \forall C \subset A$ with the optimal coalition structure of size $k-1$ defined over $A \setminus C$. Let $CS^*_k(A)$ denote the optimal coalition structure of size k defined over the set A. Then the problem can be formulated as follows:

$$CS^*_k(A) = argmax_{C \subset A}\{v(C) + v(CS^*_{k-1}(A \setminus C))\}, 1 \leq |C| \leq |A| - k + 1 \qquad (5.1)$$

The smallest coalition size is 1 and that the largest coalition size is $|A| - k + 1$ for k partitions. The following example demonstrates Eq. 5.1.

Example 5.1. *For $A = \{a, b, c, d\}$ find the optimal coalition structure with cardinality of 3, i.e., $|CS| = 3$. The search space for $CS^*_3(A)$ is shown in Table 5.1.*

The formulation in Eq. 5.1 can introduce repetition. In Example 5.1, for instance, when solving $\{\{a\}\} \cup CS^*_2(\{b, c, d\})$, $CS^*_2(\{b, c, d\})$ can be one of the coalition structures $\{\{d\}, \{b, c, \}\}$, $\{\{b\}, \{d, c\}\}$ or $\{\{c\}, \{b, d\}\}$. That makes $\{\{a\}\} \cup CS^*_2(\{b, c, d\})$ equivalent to one of the problems $\{\{b, c\}\} \cup CS^*_2(\{a, d\})$, $\{\{c, d\}\}) \cup CS^*_2(\{a, b\})$ or $\{\{b, d\}\} \cup CS^*_2(\{a, c\})$. We can avoid this repetition by considering $C \subset A$ with sizes

Table 5.1: The CSG problem for 4 agents and $|CS| = 3$, Eq. 5.1

C	$CS_2^*(A \setminus C)$	C	$CS_2^*(A \setminus C)$
{a}	CS_2^*({b, c, d})	{a, c}	CS_2^*({b, c})
{b}	CS_2^*({a, c, d})	{a, d}	CS_2^*{c, d})
{c}	CS_2^*({a, b, d})	{b, c}	CS_2^*({a, d})
{d}	CS_2^*({a, b, c})	{b, d}	CS_2^*({a, c})
{a,b}	CS_2^*({c, d})	{c, d}	CS_2^*({a, b})

Table 5.2: The CSG problem for 4 agents and $|CS| = 3$, Eq. 5.2

C	$CS_2^*(A \setminus C)$
{a, c}	CS_2^*({b, c})
{a, d}	CS_2^*({c, d})
{b, c}	CS_2^*({a, d})
{b, d}	CS_2^*({a, c})
{c, d}	CS_2^*({a, b})
{a, b}	CS_2^*({c, d})

greater than or equal to the largest coalition in $CS_{k-1}(A \setminus C)$. To this end, we set the size of smallest coalition to $\left\lceil \frac{|A|}{k} \right\rceil$. Therefore, we rewrite Eq. 5.1 as:

$$CS_k^*(A) = argmax_{C \subset A}\{v(C) + v(CS_{k-1}^*(A \setminus C))\}, \left\lceil \frac{|A|}{k} \right\rceil \leq |C| \leq |A| - k + 1. \quad (5.2)$$

Table 5.2 shows the search space for Example 5.1 following Eq 5.2. As the problem is defined recursively, we need to solve the highlighted area in the table first before finding $CS_3^*(A)$.

The next section explains how we incorporate the aforementioned problem formulation into an algorithm.

5.2 The Algorithm

The algorithm starts by forming coalition structures of cardinality 2 then add a coalition at each iteration until we reach k coalitions. The algorithms consists of $k-1$ iterations. For every iteration, i, we find the optimal coalition structures of specific sets, s.t., $|CS| = i+1$ until we reach $|CS| = k$. To determine these specific sets, we generate the integer partitions of size k for $|A|$. As mentioned in Section 2.2.2.2, an integer partition of a number is all the possible ways of splitting the number into parts which sum up to that integer.

We demonstrate how the algorithm works using the variables $A = \{a, b, c, d, e, f, g, h, i, j\}$ and $|CS| = 4$. Firstly, we generate the integer partitions as $P_4(10) = \{P[7, 1, 1, 1], P[6, 2, 1, 1],$

$P[5,3,1,1], P[4,4,1,1], P[5,2,2,1], P[4,3,2,1], P[4,2,2,2], P[3,3,2,2], P[3,3,3,1]\}$ with each partition listed in non-increasing order.

As mentioned earlier, we start by generating coalition structures of size 2, hence, we consider the 2 right-most (smallest) parts of each partition. For this setting we get $P_2(2)$, $P_2(3)$ and $P_2(4)$. Since there are two ways to partition 4 agents into 2 coalitions, $P_2(4) = \{P[2,2], P[3,1]\}$, we calculate the upper and lower bounds of these partitions using Rahwan et al. (2009). The upper and lower bounds for $P[3,1]$ are calculated respectively as $max(v(C')) + max(v(C''))$ and $average(v(C')) + average(v(C''))$ where $|C'| = 3$ and $|C''| = 1$. Moreover, to avoid repetition when a coalition is split into equal parts, we ensure that we generate coalitions such that they follow a lexicographical order. This can be easily satisfied by splitting the set C s.t. the first coalition C' holds the smallest element in C. Hence, $CS_2^*(C) = \{\{C' : C \setminus C''\}, \{C'' : a \notin C'' \, s.t. \, a < b \forall a, b \in C\}\}, |C'| = |C''|$. E.g., let $C = \{c, e, g, j\}$,to satisfy the partition $P[2,2]$ the possible values for $CS_2^*(C)$ are $\{\{c,e\},\{g,j\}\}$, $\{\{c,g\},\{e,j\}\}$ and $\{\{c,j\},\{e,gj\}\}$.

The result of this step is a list of all coalitions $C \subset A$, s.t. $|C| = 2, 3, 4$ and each coalition C is associated with $CS_2^*(C)$ and its value $v(CS_2^*(C))$. Table 5.3 shows a possible list generated by the first iteration, note that the size of the coalitions in $CS_2^*(C)$ can vary as in the third column.

Table 5.3: The first list generated 10 agents

C	$CS_2^*(C)$	v	C	$CS_2^*(C)$	v	C	$CS_2^*(C)$	v
{a, b}	{a}, {b}	v	{a, b, c}	{{b, c}, {a}}	v	{a, b, c, d}	{{a, d}, {b, c}}	v
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
{i, j}	{i}, {j}	v	{h, i, j}	{{h, i}, {j}}	v	{g, h, i, j}	{{h, i, j}, {g}}	v

Upon generating the initial list of coalition structures, we build upon them by generating coalition structures of size 3. Hence, we set i = 3. To determine the size of the sets to be added to each coalition structure, we look at the second $(k - i + 1)^{th}$ part in the integer partitions list. Table 5.4 shows how to generate the coalition structure for i=3. Note that we consider the size of largest coalition to ensure that we add coalitions in an increasing order. Otherwise, by looking at the size of the set C only, we can introduce repetition. E.g., by treating coalition structures of the forms $P[3,1]$ and $P[2,2]$ equaly, we would generate a coalition structure of the form $P[2,3,1]$ which does not follow Definition 5.2, and is a duplicate of $P[3,2,1]$ which is generated in the same iteration.

Before adding any coalition following Table 5.4, we rearrange the coalition structures according to the resulting number of agents in a coalition structure in order to calculate bounds as shown in Table 5.5. To do so, we calculate the average and maximum values of the coalition structures of the same number of agents from the previous iteration, i.e., Table 5.3. Later on, we calculate bounds specific to each resulting coalition structure

Table 5.4: Guide table for $|CS| = 3$

$\|C\|$	size of largest coalition	sizes of coalitions to be added
2	1	1, 2, 3, 4
3	2	2, 3
4	2	2, 3
4	3	3

Table 5.5: Resulting Table From Iteration 3.

Number of agents in resulting CS	Number of agents in previous CS	Size of coalition to be added
3	2	1
4	2	2
5	3	2
	2	3
6	2	4
	4	2
	3	3
7	4	3

category. For example, in Table 5.5, there are 2 ways to form a coalition structure of 3 coalitions with total of 5 agents. Either by adding a coalition of size 2 to a coalition structure of size 3 or vice versa. Hence, we calculate upper and lower bounds for both ways as $max(v(CS)) + max(v(C))$ and $average(v(CS)) + average(v(C))$ respectively, where CS is the coalition structure from the previous iteration. Subsequently, we compare the bounds of both ways. This step is performed on resulting coaltion structures of size 6 as well.

Finally, one bound is calculated when going over the list of previous coalition structures and before adding any coalition to it. We compute a bound by adding the value of that coalition structure to the maximum value of the coalition of that size. Later on, we compare the bound to the best value achieved by coalition structures of the same number of members. The same steps are repeated for the last iteration.

5.3 Empirical Evaluation

Algorithm 5 was evaluated for performance and accuracy by solving the CSG problem for 25 agents while varying the cardinality of the coalition structures from 4 to 24. The algorithms were programmed in Java and run on a Mac 2.9 GHz Intel Core i5 processor and 20 GB memory using default heap size. This section describes the CSG problem, test instances and presents the numerical results of the experiments performed. We use IBM ILOG CPLEX, an industry standard software used for solving optimisation problems, as

Algorithm 5: Cardinality Constrained CSG Algorithm

1: $input : A, k$
2: $coalitionList = \{\langle C, v(C)\rangle | C \subset A and 1 \leq |C| \leq |A| - k + 1\}$
3: $prevProblemList = null, curentProblemList = null, integerPartitions$
4: $i = 2$
5: $S = findSetSizes(integerPartitions, i)$
6: $curentProblemList = SolveCSG(coalitionList, S, i)$
7: **while** $i \leq k$ **do**
8: $\quad prevProblemList = curentProblemList$
9: $\quad S = findSetSizes(integerPartitions, i)$
10: $\quad curentProblemList = SolveCSG(coalitionList, prevProblemList, S, i)$
11: $\quad i = i + 1$
12: **end while**
13: $CS_k^*(A) = argmax(curentProblemList)$
14: **return** $CS_k^*(A)$

a benchmark for Algorithm 5. Moreover, the number of CPLEX threads was restricted to 1 since multithreading does not improve the efficiency of CPLEX.

5.3.1 The CSG Problem

We considered a cooperative game with a characteristic function that approximates the TSP problem, $v(C)$ is calculated using Chrisofides's algorithm, presented in Section 2.3.1. Moreover, we only considered coalition structures with coalitions of size less than or equal to 9.

5.3.2 Test Data

We used the New York taxi trips' data in 2013 taken from NYC Taxi & Limousine Commission[1] to evaluate Algorithm 5. For these experiments, we considered problems of 25 agents and randomly selected 26 locations from the dataset with the first location being the depot. Subsequently, we created an adjacency matrix for these location by calculating the distance between every pair of cities rounded up to the nearest metre.

5.3.3 The Algorithm

The coalition values were calculated as needed and their execution times are included in the running times shown in the figures. To compute the value of a coalition C, a subgraph $|C| + 1$ cities is extracted from the adjacency matrix then Christofides's algorithm is run on the subgraph. The value $v(C)$ is set to the negative of the result of Christofides's

[1] Dataset is available from: `https://chriswhong.com/open-data/foil_nyc_taxi/`

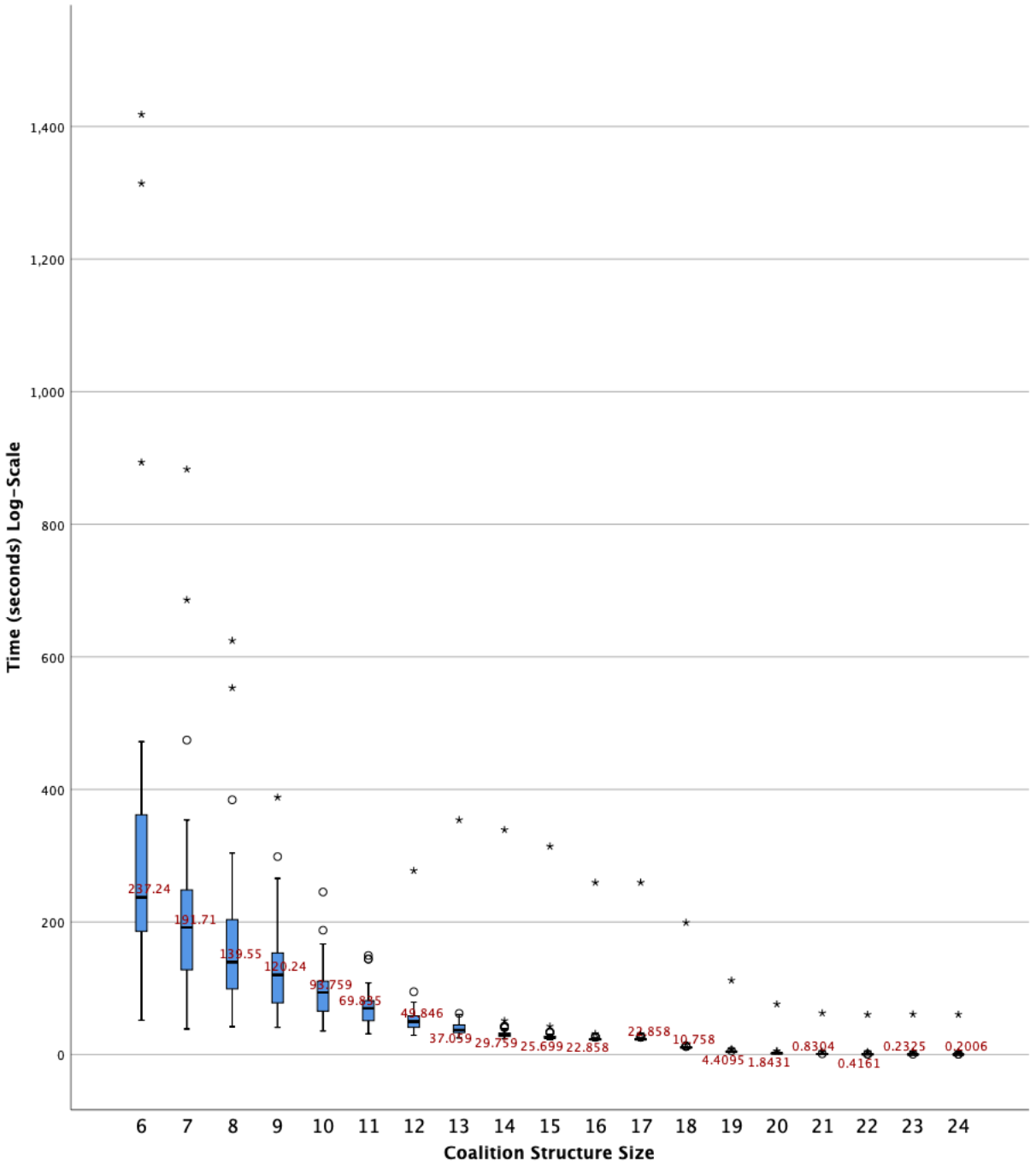

Figure 5.1: Running time vs. coalition structure size for instances of 25 agents (Algorithm 5)

algorithm presented in Section 2.3.1. Figure 5.1 shows Algorithm 5 running times vs coalition structure sizes. The experiments were run on 60 instances of the problem for coalition structure cardinalities 6 through 24. However, we ran out of memory for one instance with $|CS| = 6$. It is clear that the coalition structure size affects the running time. I.e., the running time increases when the coalition structure size decreases.

5.3.4 CPLEX

The value of all coalitions $C \subset A, |C| \leq 9$ are computed before calling CPLEX. The value $v(C)$ is set to the result of Christofides's algorithm. However, the problems solved are equivalent since CPLEX returns the coalition structure of minimal value. The experiments were run on 60 instances for coalition structure cardinalities 6 through 25. Figure 5.2 shows a boxplot of CPLEX running times. As in Algorithm 5 running time inversely correlated to coalition structure size.

5.3.5 Accuracy

The output of Algorithm 5 was compared against the output from running CPLEX for 60 problems for $6 \leq |CS| \leq 24$ respectively. The percentage of optimal solutions for coalition structure size is shown in Figure 5.4. Furthermore, to evaluate the quality of the solutions obtained from Algorithm 5, we calculated the approximation ratio as $\frac{output from Algorithm 5}{output from CPLEX}$. A boxplot of the approximation ratios is depicted in Figure 5.3.

5.3.6 Statistical Analysis

A paired-sample t-test was conducted to compare the execution times of Algorithm 5 and CPLEX. The sample consisted of 60 instances of the CSG problem of 25 agents and the coalition structure size was varied from 6 through 24. The t-test shows a significant difference in the running times for Algorithm 5 and CPLEX for coalition structure of size 8 through 24. A summary of the statistical analysis is shown in Table 5.6.

5.4 Summary

In this chapter we introduced an algorithm that addresses the cardinality constrained CSG problem. The algorithm was evaluated for accuracy and efficiency. Algorithm 5 output the optimal solution in at least 60% of the problems. In fact, the percentage of optimal solutions increases with the coalition structure size; it exceeds 80% for $|CS| \geq 13$ and exceeds 98% for $|CS| = 24$. Moreover, the approximation ratio is less that 1.006 for all tested coalition structure sizes. More specifically, the approximation ratio decreases when the coalition structure size increases. It is less than 1.001 for $|CS| \geq 13$. In regards to efficiency, paired-sample t-test was conducted to compare the algorithm's execution time to CPLEX. The statistical analysis shows that Algorithm 5 significantly faster than CPLEX for coalition structure sizes 8 through 24.

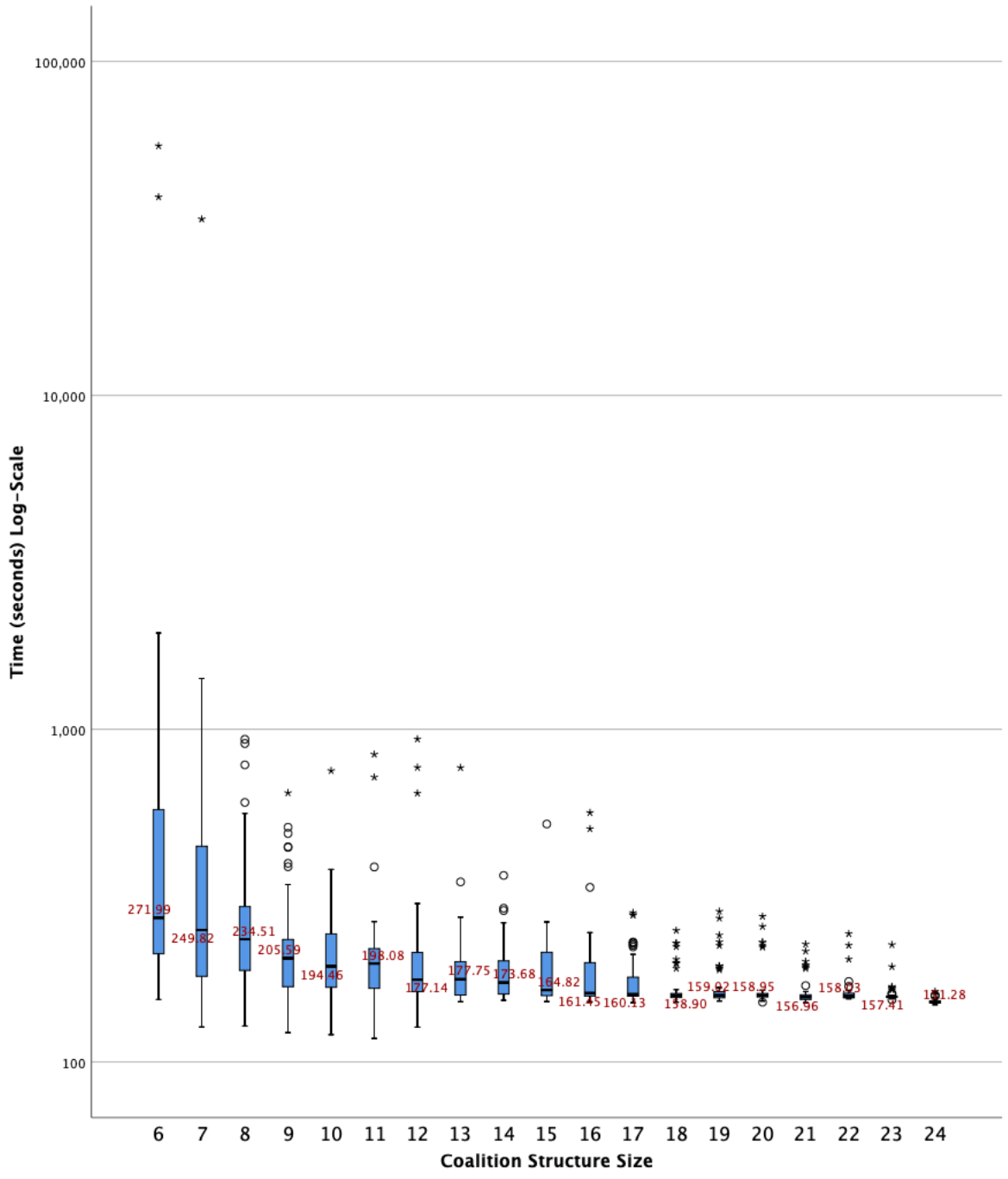

Figure 5.2: Running time vs. coalition structure size for instances of 25 agents (CPLEX)

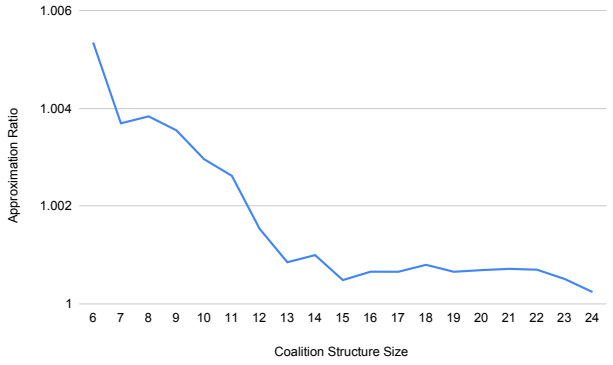

Figure 5.3: Approximation ratio for Algorithm 5.

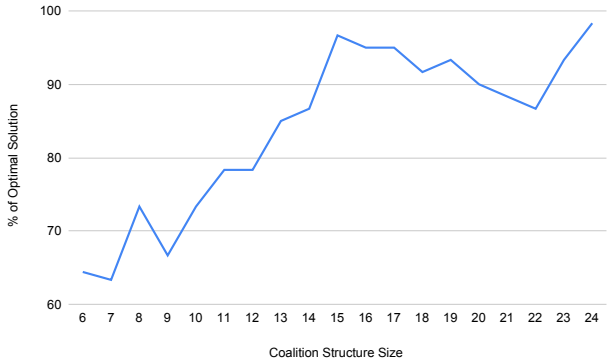

Figure 5.4: Percentage of optimal solutions for Algorithm 5

Table 5.6: Paired T-Test

CS size	CPLEX Mean	Algorithm 5 Mean	N	Sig. (2-tailed)
6	1987.0941	297.4712	59	0.1422
7	907.8737	240.2160	60	0.2358
8	292.9669	178.8313	60	1.5265E-4
9	232.5324	134.4194	60	2.2465E-8
10	217.3891	96.6172	60	6.6091E-14
11	215.1167	72.2704	60	1.1709E-13
12	219.9050	55.0131	60	1.2039E-12
13	193.8661	45.0682	60	5.4910E-18
14	188.3587	36.2371	60	1.9961E-27
15	190.0568	31.5909	60	1.3438E-25
16	189.4296	27.1473	60	1.1294E-22
17	175.9747	27.1473	60	1.1160E-32
18	166.2578	14.1589	60	1.1674E-41
19	169.4180	6.3698	60	1.9495E-44
20	166.4432	3.1639	60	8.5644E-48
21	162.0126	1.9917	60	9.9250E-58
22	162.1375	1.54051	60	6.2829E-59
23	159.8444	1.3753	60	5.9779E-66
24	152.2452	1.3261	60	5.7207E-76

Chapter 6

Solving the CSG Problem via Intervals

In this chapter, we utilise the interval cooperative model discussed in Section 2.1.5 to reduce the computational cost of solving the CSG problem for a combinatorial optimisation game. To represent these problems using the interval model, we use approximation and/or heuristic algorithms to bound the coalition values as shown in Section 6.1. Subsequently, we give detailed explanation on solving the problem optimally with necessary proofs. On the other hand, when a near-optimal solution is desired, we provide bounds for substituting optimal coalition values with approximate values. Section 6.2 maps the mTSP to a CSG problem in order to use it as a model problem in empirical evaluations presented in Section 6.4.

6.1 The Interval Approach

Approximation and heuristic techniques have been widely used to obtain suboptimal solutions to intractable problems. These techniques attracted the attention of mathematicians and operational researchers due to their capability of reaching solutions of high quality while requiring much less time and resources compared to exact methods. This has resulted in an enormous number of algorithms addressing combinatorial optimisation problems. Such algorithms have yet to be exploited in the context of CSG in combinatorial optimisation games. To this end, we utilise the interval cooperative model to capture the upper and lower bounds of the coalition values.

Solving the CSG problem for one of the bounds on the valuation function v, together with a factor that reflects the relation of the bound to the optimal solution, can put us a step closer to the solution of the CSG problem for v. This procedure can be deemed efficient as it helps avoid the impracticality of optimally computing the values of an enormous

set of complex problems. On the one hand, the optimal solution for the CSG problem considering the chosen bound is within a factor of the optimal solution considering v. On the other hand, it can reduce the size of the search space in case an optimal solution considering v is desired.

To construct the interval cooperative game $\langle A, w\rangle$ to be used in the proposed approach, we firstly define an interval function $w : 2^A \rightarrow \mathbb{R} \times \mathbb{R}$ to encapsulate the coalition values of the given cooperative game $\langle A, v\rangle$. This section follows the notation previously presented in Section 2.1.5. In addition, it is assumed that the problem at hand is a maximisation problem[1]. Either approximation or heuristic methods can be used to define w. Approximation algorithms, unlike heuristics, return a solution guaranteed to be within a certain factor from the optimal. The approximation factor[2], β, also known as the approximation ratio, represents the performance of the algorithm. Given an upper bound function[3] $\overline{w}$ such that $\overline{w}(C) \geq v(C)$ and $\beta = \frac{\overline{w}C}{v(C)}$, w is defined as:

$$w(C) = [\frac{1}{\beta}\overline{w}(C), \overline{w}(C)]\, \forall C \subseteq A \tag{6.1}$$

On the other hand, when adopting a heuristic approach to solve the combinatorial problem, two functions are needed to generate the bounds as there is no guarantee on the quality of the solution. Let $\underline{w}$ and $\overline{w}$ represent the lower and upper bounds of v respectively, then $\underline{w}(C) \leq v(C) \leq \overline{w}(C)$, $\forall C \subseteq A$. Thus, w is defined as:

$$w(C) = [\underline{w}(C), \overline{w}(C)]\, \forall C \subseteq A \tag{6.2}$$

Consequently, the approximation ratio β can be calculated as $\beta = max_{C \in A}\frac{\overline{w}(C)}{\underline{w}(C)}$. Thus, w can be rewritten as Equation 6.1.

$$w(C) = [\frac{1}{\beta}\overline{w}(C), \overline{w}(C)]\, \forall C \subseteq A$$

Having defined w, we now look at the coalition structures associated with it. We write $\mathcal{CS}$ to refer to the set of all feasible coalition structures. Note that we do not differentiate here between the set of all feasible coalition structures $\mathcal{CS}$ for v and w since both problems are defined on the same set A. For any $CS \in \mathcal{CS}$ we define $\overline{w}(CS) = \sum_{C \in CS} \overline{w}(C)$. Therfore, the optimal coalition structure of $\langle A, \overline{w}\rangle$, denoted as $CS^*_{\overline{w}}$, is defined as:

$$CS^*_{\overline{w}} = argmax_{CS \in \mathcal{CS}} \sum_{C \in CS} \overline{w}(C) \tag{6.3}$$

[1] This approach can be easily extended to cover minimisation problems as we show by example in Section 6.4.

[2] β is calculated as the ratio between the approximate solution and the optimal solution in whichever direction that makes $\beta > 1$.

[3] In case of a lower bound function $\underline{w}$, $\beta = \frac{v(C)}{\underline{w}(C)}$ and $w(C) = [\underline{w}(C), \beta\underline{w}(C)]\, \forall C \subseteq A$.

The remaining of this section demonstrates how $CS^*_{\overline{w}}$ can be used to find CS^*_v and analyses the quality of $CS^*_{\overline{w}}$ compared to the optimal solution CS^*_v.

6.1.1 Optimal CSG

We show, in the following theorem, that $CS^*_{\overline{w}}$ can help narrow our search for the optimal coalition structure CS^*_v, which in turn reduces the number of the coalitions required to be solved for optimal solution.

Theorem 6.1. *Let $\langle A, v\rangle$ and $\langle A, w\rangle$ be 2 cooperative settings where $w(C) = [\underline{w}(C), \overline{w}(C)]$ and $\underline{w}(C) \leq v(C) \leq \overline{w}(C)\ \forall C \subseteq A$. Then $CS^*_v \in \tilde{\mathcal{CS}}$ where $\beta = max_{C\subseteq A}\frac{\overline{w}(C)}{\underline{w}(C)}$ and $\tilde{\mathcal{CS}} = \{CS | CS \in \mathcal{CS}\ and\ \overline{w}(CS) = [\frac{1}{\beta}\overline{w}(CS^*_{\overline{w}}), \overline{w}(CS^*_{\overline{w}})]\}$.*

Proof. Given the approximation ratio β, it can be deduced that $v(C) = [\frac{1}{\beta}\overline{w}(C), \overline{w}(C)]$. Accordingly, $v(CS) = [\frac{1}{\beta}\overline{w}(CS), \overline{w}(CS)]\ \forall CS \in \mathcal{CS}$. Upon finding $CS^*_{\overline{w}}$, the optimal coalition structure for $\langle A, \overline{w}\rangle$, the set $\mathcal{CS}$ can be divided into two disjoint sets $\{CS | \overline{w}(CS) \geq \frac{1}{\beta}\overline{w}(CS^*_{\overline{w}})\}$ and $\{CS | \overline{w}(CS) < \frac{1}{\beta}\overline{w}(CS^*_{\overline{w}})\}$. We now prove that CS^*_v, the optimal coalition structure for $\langle A, v\rangle$, belongs to $\tilde{\mathcal{CS}}$.

Let us assume for the sake of contradiction that $CS^*_v \notin \tilde{\mathcal{CS}}$. That implies that, $CS^*_v \in \{CS | \overline{w}(CS) < \frac{1}{\beta}\overline{w}(CS^*_{\overline{w}})\}$. Accordingly, $v(CS^*_v) < \frac{1}{\beta}\overline{w}(CS^*_{\overline{w}})$. This is a contradiction since $CS^*_{\overline{w}}$ guarantees the existence of a coalition structure $CS \in \mathcal{CS}$ such that $v(CS) = [\frac{1}{\beta}\overline{w}(CS^*_{\overline{w}}), \overline{w}(CS^*_{\overline{w}})]$. □

Having proved that $CS^*_v \in \tilde{\mathcal{CS}}$, we define the set of coalitions we need to solve optimally as $\Pi = \{C | C \in \cup_{CS\in\tilde{\mathcal{CS}}} CS\ and\ C \notin \cap_{CS\in\tilde{\mathcal{CS}}} CS\}$. The condition $C \notin \cap_{CS\in\tilde{\mathcal{CS}}} CS$ means that we do not solve the coalitions that appear in every coalition structure in $\tilde{\mathcal{CS}}$. Upon calculating the values of coalitions in Π, the CSG problem for $\langle A, v\rangle$ can be written as follows:

$$CS^*_v = argmax_{CS\in\tilde{\mathcal{CS}}} \sum_{C\in CS} \omega(C)$$

$$\text{where :} \tag{6.4}$$

$$\omega(C) = \begin{cases} v(C),\ if C \in \Pi \\ \overline{w}(C),\ if C \notin \Pi. \end{cases}$$

The interval approach to CSG we propose here can be summarised in the following steps:

1. Define w to bound the values of v, such that, $w(C) = [\underline{w}(C), \overline{w}(C)]$ and $\underline{w}(C) \leq v(C) \leq \overline{w}(C)$, $\forall C \subseteq A$.
2. Solve $\overline{w}(C), \forall C \subseteq A$.

Table 6.1: The interval values of the coalitions in Example 6.1

C	$w(C)$	C	$w(C)$
{a}	[2.5, 3]	{a, c}	[3.4, 4]
{b}	[0.9, 1]	{a, d}	[2.9, 3.6]
{c}	[2.7, 3]	{b, c}	[2, 2.2]
{d}	[1.7, 2]	{b, d}	[2.8, 3]
{a, b}	[4, 5]	{c, d}	[1.3, 1.5]

3. Construct the set $\mathcal{CS}$ by enumerating all feasible coalitions structures.
4. Find the optimal coalition structure considering $\overline{w}$, $CS^*_{\overline{w}} = argmax_{CS \in \mathcal{CS}} \sum_{C \in CS} \overline{w}(C)$.
5. Calculate β, if not given, as $\beta = max_{C \in A} \frac{\overline{w}(C)}{\underline{w}(C)}$.
6. Find the set $\tilde{\mathcal{CS}} = \{CS | \overline{w}(CS) \geq \frac{1}{\beta} \overline{w}(CS^*_{\overline{w}})\}$.
7. Find the set $\Pi = \{C | C \in \cup_{CS \in \tilde{\mathcal{CS}}} CS \; and \; C \notin \cap_{CS \in \tilde{\mathcal{CS}}} CS\}$.
8. Solve $v(C)$ for all the coalitions $C \in \Pi$.
9. Define $\omega : 2^A \to \mathbb{R}$ as: $\omega(C) = v(C) \; if C \in \Pi$ and $\omega(C) = \overline{w}(C) \; if C \notin \Pi$.
10. Find the optimal coalition structure CS that belongs to $\tilde{\mathcal{CS}}$, assuming $\omega(CS) = \sum_{C \in CS} \omega(C)$.

Furthermore, we use the following example to demonstrate the interval approach:

Example 6.1. *Let $A = \{a, b, c, d\}$ be the set of agents in a given cooperative setting where the interval value of a coalition $C \subseteq A$ is given in Table 6.1. Suppose that only coalition structures of size 3 are feasible, then the optimal coalition structure CS^*_v can be obtained through the following:*

Steps 1 and 2: The interval values of the coalitions are given by Table 6.1.

Step 3: The set of all feasible coalition structures $\mathcal{CS}$ is constructed as shown in Table 6.2.

Table 6.2: The set $\mathcal{CS}$ in Example 6.1

CS	$\overline{w}(CS)$	CS	$\overline{w}(CS)$
{{a}, {b}, {c, d}}	5.5	{{b}, {c}, {a, d}}	7.6
{{a}, {c}, {b, d}}	9	{{b}, {d}, {a, c}}	7
{{a}, {d}, {b, c}}	7.2	{{c}, {d}, {a, b}}	10

Step 4: The optimal coalition structure $CS^*_{\overline{w}}$ is $\{\{c\}, \{d\}, \{a, b\}\}$ and $\overline{w}(CS^*_{\overline{w}}) = 10$.

Step 5: The approximation factor β is computed as $\beta = \frac{5}{4}$.

Step 6: By substituting the value of $\frac{1}{\beta}\overline{w}(CS^*_{\overline{w}})$, it follows that, $\tilde{\mathcal{CS}} = \{CS \mid \overline{w}(CS) \geq 8\}$. Hence $\tilde{\mathcal{CS}} = \{\{\{c\},\{d\},\{a,b\}\},\{\{a\},\{c\},\{b,d\}\}\}$.

Step 7: $\Pi = \{\{a\},\{a,b\},\{b,d\},\{d\}\}$, note that $\{c\} \notin \Pi$ since it is a member of all the coalition structures in $\tilde{\mathcal{CS}}$.

Step 8: The values of the coalitions in Π are given by Table 6.3.

Table 6.3: The optimal values of the coalitions in the set Π in Example 6.1

C	$v(C)$	C	$v(C)$
{a}	2.5	{b, d}	3
{a, b}	4.5	{d}	2

Step 9: ω is defined as given in Table 6.4.

Table 6.4: ω as defined in Example 6.1

C	$\omega(C)$	C	$\omega(C)$	C	$\omega(C)$
{a}	2.5	{b, d}	3	{d}	2
{a, b}	4.5	{c}	3		

Step 10: From Table 6.4, it can be deduced that $CS^*_v = \{\{c\},\{d\},\{a,b\}\}$, since, $\omega(\{\{c\},\{d\},\{a,b\}\}) = 9.5$ while $\omega(\{\{a\},\{c\},\{b,d\}\}) = 8.5$.

The size of Π determines the percentage of reduction in the calculation of the optimal coalition values. By using the interval approach, the number of coalitions for which an optimal value is calculated is reduced by 60%.

6.1.2 Near-Optimal CSG

Enumerating all the feasible coalition structures to find $CS^*_{\overline{w}}$ requires a vast amount of memory due to the inherit complexity of the CSG problem. However, using any CSG algorithm to find $CS^*_{\overline{w}}$ can guarantee a near-optimal solution.

Theorem 6.2. *Given an upper bound function $\overline{w}$ for v and an approximation factor β, $CS^*_{\overline{w}}$ is within β^2 from CS^*_v.*

Proof. For a maximisation problem, we have:

$$\frac{1}{\beta}\overline{w}(CS^*_{\overline{w}}) \leq v(CS^*_{\overline{w}}) \leq v(CS^*_v) \leq \overline{w}(CS^*_{\overline{w}}) \leq \beta\overline{w}(CS^*_{\overline{w}}) \tag{6.5}$$

From Equation 6.5 we get:

$$\frac{1}{\beta}\overline{w}(CS^*_{\overline{w}}) \leq v(CS^*_v) \leq \beta\overline{w}(CS^*_{\overline{w}}) \tag{6.6}$$

From Equation 6.6, we conclude that the approximation ratio for the CSG problem is β^2 where β is the approximation ratio for the approximate coalition values used to define the interval.

□

Since β is usually small, the approximation ratio β^2 for the CSG problem is considered sufficient, especially that the optimal value is not computed for any coalition when using $CS^*_{\overline{w}}$. For example, the CSG problem reduced from an mTSP in the next section has $\beta^2 = \frac{9}{4}$.

6.2 The mTSP as a CSG problem

As mentioned earlier, the mTSP is used as a model problem to evaluate our approach in solving CSG problems with complex valuation function. The reason for choosing mTSP is that the values of coalitions can be obtained by solving the classic TSP, which is extensively studied in mathematics and operational research.

In this section we show a mapping of the multiple travelling salesman problem to a CSG problem defined in a cooperative setting. In order to use the mTSP as a test problem, we reduce it to a CSG problem where the coalition values are defined as a TSP.

Theorem 6.3. *The multiple travelling salesman problem can be reduced to a coalition structure generation problem in polynomial time.*

Proof. The proof begins as follows: first, we map the components of the mTSP to a cooperative setting. Secondly, we define the CSG problem associated with the setting. Finally, we prove that the CSG problem is equivalent to the mTSP.

Let the set A be the set of agents. Also, in the reduced problem, the home city 0, the adjacency matrix and the number of salesmen are defined as in the original problem. Now, we define the CSG problem. A valid coalition structure partitions the set of cities into k coalitions. In addition, the cost of a coalition $c(C)$ is the shortest route a salesman follows leaving the home city 0 and visiting all the cities in C before More formally, the coalition structure generation problem is defined as follows:

$$\begin{aligned} &argmin_{CS \in \mathcal{CS}} \sum_{C \in CS} c(C) \\ &such\ that |CS| = k \end{aligned} \tag{6.7}$$

Having defined the CSG problem with k coalitions in a feasible coalition structure, we map each coalition to a tour in the mTSP solution.

From the first and second constraints, we can see that a solution for the mTSP with k salesmen is given as k tours. In addition, every tour starts from the home city and visits at least one city before returning to the home city.

Moreover, from the mTSP's third and fourth constraints, every city is visited and it is visited only once. This implies that every tour consists of distinct cities. More formally, suppose that T_i is the tour corresponding to the i's salesman then $\cap_{i=1}^{k} T_i = \phi$ and $\cup_{i=1}^{k} T_i = A$. By definition, the structure of a mTSP resembles a coalition structure.

Finally, we prove by contradiction that the tours generated by the mTSP are equivalent to the coalitions in the optimal coalition structure.

Assume that the cost of the optimal coalition structure is not equal to the total cost of the tours from the mTSP. A cost of a tour T_i is denoted as $ct(T_i)$

This implies that $\sum_{C \in CS} c(C) \neq \sum_{i=1}^{k} ct(T_i)$. Therefore, either $\sum_{C \in CS} c(C) < \sum_{i=1}^{k} ct(T_i)$ or $\sum_{C \in CS} c(C) > \sum_{i=1}^{k} ct(T_i)$.

$\sum_{C \in CS} c(C) < \sum_{i=1}^{k} ct(T_i)$ implies that there exists another coalition structure CS' such that $\sum_{C \in CS'} c(C) < \sum_{C \in CS} c(C)$ which is a contradiction.

Similarily, $\sum_{C \in CS} c(C) > \sum_{i=1}^{k} ct(T_i)$, implies that there is another solution to the mTSP problem with less cost. A contradiction $\sum_{i=1}^{k} ct(T'_i) < \sum_{i=1}^{k} ct(T_i)$

Hence, an optimal solution for the mTSP is optimal for the reduced CSG problem where $c(C)$ is defined as a TSP.

□

6.3 Solving the CSG Problem for the mTSP using Intervals

The first step in solving the CSG problem using the interval approach is to define the function w. In the CSG problem associated with a mTSP, the cost of every coalition is defined as the optimal tour of a TSP. The cost function $c : 2^A \to \mathbb{R}$ is defined using Equation 2.2 and the optimal CSG is denoted as CS_c^*. As described in Section 2.3.1, for a graph that obeys the triangular inequality, Christofides' algorithm approximates the solution of a TSP with a guarantee $\beta = \frac{3}{2}$ of the optimal. Hence, the interval w can be defined as:

$$w(C) = [\frac{2}{3}\overline{w}(C), \overline{w}(C)] \forall C \in A \tag{6.8}$$

where the upper bound $\overline{w}$ is obtained by Christofides' algorithm. This can be further demonstrated using the following example.

Example 6.2. *Given a set of cities* $A = \{1, 2, 3, 4, 5\}$ *and a depot located at city* 0 *as given by the distance matrix in Table 6.2. Consider the CSG problem for* 2 *salesmen at the depot.*

Table 6.5: Adjacency matrix for Example 6.2

	0	1	2	3	4	5
0	0	4,894	5,784	1,635	4,703	5,294
1	4,894	0	4,514	9,473	4,042	10,050
2	5,784	4,514	0	7,225	8,001	10,695
3	1,635	9,473	7,225	0	5,547	3,660
4	4,703	4,042	8,001	5,547	0	8,263
5	5,294	10,050	10,695	3,660	8,263	0

The second step is to solve, $\overline{w}$, the TSP using Christofides' algorithm for the coalitions as given in Table 6.6.

Table 6.6: The upper bounds of the coalitions in Example 6.2

C	$\overline{w}(C)$	C	$\overline{w}(C)$	C	$\overline{w}(C)$
{1}	9,788	{2, 4}	18,488	{1, 4, 5}	24,089
{2}	11,568	{2, 5}	21,773	{2, 3, 4}	20,967
{3}	3,270	{3, 4}	11,885	{2, 3, 5}	21,774
{4}	9,406	{3, 5}	10,589	{2, 4, 5}	29,445
{5}	10,588	{4, 5}	18,260	{3, 4, 5}	18,261
{1, 2}	15,192	{1, 2, 3}	18,268	{1, 2, 3, 4}	22,119
{1, 3}	16,002	{1, 2, 4}	19,043	{1, 2, 3, 5}	25,398
{1, 4}	13,639	{1, 2, 5}	25,397	{1, 2, 4, 5}	29,248
{1, 5}	20,238	{1, 3, 4}	19,853	{1, 3, 4, 5}	24,090
{2, 3}	14,644	{1, 3, 5}	20,239	{2, 3, 4, 5}	30,389

Table 6.7: The upper bounds of $CS \in \mathcal{CS}$ and the optimal costs of $C \in \Pi$ and $CS \in \tilde{\mathcal{CS}}$ in Example 6.2.

#	$CS = \{C_1, C_2\}$	$\overline{w}(CS)$	$c(C_1)$	$c(C_2)$	$c(CS)$
1	{{1}, {2, 3, 4, 5}}	40,177	9,788	27,343	37,131
2	{{2}, {1, 3, 4, 5}}	35,658	11,568	22,494	34,062
3	{{3}, {1, 2, 4, 5}}	32,518	3,270	27,897	31,167
4	{{4}, {1, 2, 3, 5}}	34,804	9,406	25,398	34,804
5	{{5}, {1, 2, 3, 4}}	32,707	10,588	21,522	32,110
6	{{1, 2}, {3, 4, 5}}	33,453	15,192	18,261	33,453
7	{{1, 3}, {2, 4, 5}}	45,447	-	-	-
8	{{1, 4}, {2, 3, 5}}	35,413	13,639	21,774	35,413
9	{{1, 5}, {2, 3, 4}}	41,205	20,238	20,967	41,205
10	{{2, 3}, {1, 4, 5}}	38,733	14,644	22,493	37,137
11	{{2, 4}, {1, 3, 5}}	38,727	18,488	20,239	38,727
12	{{2, 5}, {1, 3, 4}}	41,626	21,773	16,118	37,891
13	{{3, 4}, {1, 2, 5}}	37,282	11,885	25,397	37,282
14	{{3, 5}, {1, 2, 4}}	29,632	10,589	19,043	29,632
15	{{4, 5}, {1, 2, 3}}	36,528	18,260	18,268	36,528

In Table 6.7, we enumerate all the feasible coalition structures (Step 3), in order to find the optimal CS of $\overline{w}$ (Step 4). As we can see, $CS^*_{\overline{w}}$ is the 14^{th} CS. Since we are

considering a minimisation problem, $\tilde{\mathcal{CS}}$ is defined as $\tilde{\mathcal{CS}} = \{CS \mid \overline{w}(CS) \leq \beta \overline{w}(CS^*_{\overline{w}})\}$. By substituting $\overline{w}(CS^*_{\overline{w}}) = 29,632$, we deduce that $\tilde{\mathcal{CS}}$ holds the coalition structure with $\overline{w}(CS) \leq 44,448$. That is, the 7^{th} coalition structure in Table 6.7 can be eliminated from $\tilde{\mathcal{CS}}$. This results in 6.67% reduction in the number of coalitions to be optimally computed.

6.4 Empirical Evaluation

We evaluate the accuracy of the near-optimal interval approach for the NYC taxi trip data instances from Section 5.3 for 85 instances with coalition structure cardinalities of 9, 10, 14, 15, 19 and 20. To do so, we optimally solve the CSG problem with coalition values equal to Christofides's TSP upper bound and its equivalemt mTSP problem optimally using CPLEX. Then we calculate the optimal coalition values of the output of the CSG problem. I.e., we find the optimal value of the resultant coalition structure by solving the TSP for all coalitions in the coalition structure. Finally, we compare the optimal solution of the CSG problem to the optimal solution of its equivalent mTSP. The approximation ratio of the interval approach is calculated as $\frac{optimalmTSP}{optimalCSG}$. Out of the 85 CSG problems solved using the interval approach, 24 problems (28.24%) were solved optimally. The mean, median and standard deviation of approximation ratios are 1.0123, 1.0057 and 0.0166 respectively. Moreover, the largest approximation ratio is 1.0799 which is less than β^2. Hence, our result agrees with Theorem 6.2. In fact, the largest approximation ratio is less than $\beta = 1.5$

6.5 Summary

This chapter has presented an interval based approach for solving the CSG problem for cooperative environments with complex valuation functions. In this regard, the interval approach can be exploited for optimal and approximate solutions with performance guarantee. However, an optimal solution requires the enumeration of a large set of coalition structures which increases the amount of resources needed. Consequently, we mapped the mTSP problem to a CSG problem as a step for evaluating the interval approach proposed in this chapter. We evaluate near-optimal solutions obtained by solving the CSG problem using an upper bound instead of exact values compared to the optimal mTSP solutions. Empirical evaluation gave an approximation ratio smaller than that of the upper bound used in the algorithm in place for the optimal coalition values.

Chapter 7

Conclusions and Future Work

This chapter summarises the work presented in this thesis with emphasis on the empirical results from the experiments. In addition, it draws conclusions form these results. Moreover, we point out the limitations of the research and highlight prospects for future work.

7.1 Conclusions

This thesis investigated CSG in specific cooperative settings, namely: OCF-Gs, cardinality constrained CSG and combinatorial optimisation games. In regard to overlapping coalition structures, we extended the threshold task games model to represent environments with multiple resources. Furthermore, we solved the coalition structure generation problem for the extended model which we refer to as the multi-resource threshold task game (MR-TTG) model. To do so, we reduced the problem to two variants of the knapsack problem, the BMKP and the MMKP. These reductions allow for the design of algorithms that are independent of the number of agents. Later on, we designed a branch and bound algorithm to solve the BMKP and evaluated the BMKP reduction. We found that the run time of the BMKP algorithm depends on four factors: The correlation of the value of task types with the resource requirements, the total number of tasks, the number of dimensions, and the slackness of the resources available in the environment compared to the ones required by all tasks.

Furthermore, to solve the MMKP, we modified the EMKP algorithm for the MMKP. When evaluating the MMKP reduction, we aimed to address instances with total number of tasks greater than 200 and 5 resource types. We fixed the number of task types and built the MMKP from one possible partitioning of tasks. The algorithm's run time was smaller for strongly correlated instances compared to uncorrelated instances. Moreover, when exiting the algorithm after 0.5 milliseconds, the strongly correlated instances had significantly better approximation ratio than the uncorrelated instances.

In regard to cardinality constrained CSG, the current CSG algorithms were designed considering no restrictions on the coalition structure size. As a result, these algorithms do not exploit the special structure of the problems we consider, as analysed in Section 2.2.2. One of the reasons is that upper bounds in most of these algorithms are derived from coalition structures of specific sizes; these sizes do not necessarily match the size of feasible coalition structures in our problems. In addition, the DP (Section 2.2.2.3) and anytime (Section 2.2.2.2) CSG algorithms we reviewed, apart from the IP algorithm, jump between coalition structures of different sizes when traversing the search space. As a result, these algorithms traverse coalition structures that are not part of the search space considered in our problem. We designed a specialised algorithm that addresses these problems. Although, the algorithm does not guarantee optimality, more that 85% of the problems were solved optimally for $|CS| \geq 13$. Empirical evaluation shows that our algorithm is considerably faster than CPLEX for instances with large coalition structure sizes. For instances of 25 agents and $|CS| = 24$ the mean of the the statistical significance between the running times of and Algorithm 5 is 5.72E-76. Moreover, the mean of Algorithm 5 execution time was smaller than that of CPLEX for all the tested coalition structure sizes.

In regard to combinatorial optimisation games, we utilised the cooperative interval model to use approximate coalition values as a substitute to optimal ones. In case an optimal solution is required, we showed, in Theorem 6.1, that the interval approach can reduce the number of coalitions that need to be optimally solved in order to find the optimal coalition structure. However, we relax our requirement and consider a near-optimal coalition structure since this approach requires a huge amount of memory. In addition, we analysed the quality of the solution reached by our proposed approach in case that only approximate coalition values are used. As proved in Theorem 6.2, given the approximate ratio β for the coalition values, the worst-case coalition structure obtained by the interval approach is within β^2 of the optimal. The accuracy of this approach was evaluated for the CSG problem equivalent to the mTSP. The largest approximation ratio was 1.0799 which is less than $\beta = \frac{3}{2}$.

7.2 Future Work

Further studies should investigate the following:

1. Chalkiadakis et al. (2010) defined three stability concepts for OCF-Gs. Our model should be analysed further to assess its stability. In addition, to guarantee fairness, an extension for the Shapley value is required. A possible research direction is characterising the computational complexity of finding stable, fair outcomes and developing efficient algorithms to achieve these outcomes.

2. Enhancing Algorithm the cardinality constrained CSG algorithm to address small coalition structure sizes.

3. Designing algorithms for cardinality constrained CSG that exploits the properties of the valuation function, such as, superadditivity and subadditivity since the grand coalition and singleton coalitions are no longer feasible.

4. Designing a machine learning algorithm that predicts which coalition values to compute and subspaces to search within a predefined execution time. Such algorithm is known as a design-to-time algorithm (Bonissone and Halverson, 1990) and (D' Ambrosio and Fehling, 1989).

Bibliography

Akçay, Y., Li, H., and Xu, S. H. (2007). Greedy algorithm for the general multidimensional knapsack problem. *Annals of Operations Research*, 150(1):17–29.

Alparslan-Gök, S. Z. (2014). On the interval shapley value. *Optimization*, 63(5):747–755.

Alparslan-Gök, S. Z., Branzei, R., Fragnelli, V., and Tijs, S. H. (2013). Sequencing interval situations and related games. *Central European Journal of Operations Research*, 21(1):225–236.

Alparslan-Gök, S. Z., Branzei, R., and Tijs, S. H. (2008). Cooperative interval games arising from airport situations with interval data. *CentER Discussion Paper*, 2008.

Alparslan-Gök, S. Z., Branzei, R., and Tijs, S. H. (2009). Convex interval games. *Journal of Applied Mathematics and Decision Sciences*, 2009:14.

Alparslan-Gök, S. Z., Branzei, R., and Tijs, S. H. (2010). The interval shapley value: an axiomatization. *Central European Journal of Operations Research*, 18(2):131–140.

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

Bachrach, Y., Meir, R., Jung, K., and Kohli, P. (2010). Coalitional structure generation in skill games. In *Twenty-Fourth AAAI Conference on Artificial Intelligence, Atlanta, Georgia*, volume 10, pages 703–708.

Bachrach, Y. and Rosenschein, J. S. (2008). Coalitional skill games. In *Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems-Volume 2*, pages 1023–1030. International Foundation for Autonomous Agents and Multiagent Systems.

Banzhaf, J. F. (1965). Weighted voting doesn't work: A mathematical analysis. *Rutgers Law Review*, 19:317–343.

Bing, H., Leblet, J., and Simon, G. (2010). Hard multidimensional multiple choice knapsack problems, an empirical study. *Computers & Operations Research*, 37(1):172–181.

Bonissone, P. P. and Halverson, P. C. (1990). Time-constrained reasoning under uncertainty. *Real-Time Systems*, 2(1):25–45.

Borm, P., Hamers, H., and Hendrickx, R. (2001). Operations research games: A survey. *Top*, 9(2):139–199.

Branzei, R., Dimitrov, D., Pickl, S., and Tijs, S. (2004). How to cope with division problems under interval uncertainty of claims? *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 12(02):191–200.

Caprara, A. and Letchford, A. N. (2010). New techniques for cost sharing in combinatorial optimization games. *Mathematical programming*, 124(1-2):93–118.

Chalkiadakis, G., Elkind, E., Markakis, E., Polukarov, M., and Jennings, N. R. (2010). Cooperative games with overlapping coalitions. *Journal of Artificial Intelligence Research*, 39(1):179–216.

Christofides, N. (1976). Worst-case analysis of a new heuristic for the travelling salesman problem. Technical Report RR-388, Management Sciences Research Group, Carnegie-Mellon University, Pittsburgh, PA, USA.

Conitzer, V. and Sandholm, T. (2004). Computing shapley values, manipulating value division schemes, and checking core membership in multi-issue domains. In *Proceedings of the Nineteenth National Conference on Artificial Intelligence and Sixteenth Innovative Applications of Artificial Intelligence Conference*, pages 219–225.

Conitzer, V. and Sandholm, T. (2006). Complexity of constructing solutions in the core based on synergies among coalitions. *Artificial Intelligence*, 170(6):607–619.

D' Ambrosio, B. and Fehling, M. (1989). Resource bounded-agents in an uncertain world. In *Proceedings of the Workshop on Real-Time Artificial Intelligence Problems*.

Dang, V. D., Dash, R. K., Rogers, A., and Jennings, N. R. (2006). Overlapping coalition formation for efficient data fusion in multi-sensor networks. *Twenty-First National Conference on Artificial Intelligence (AAAI-06), Boston, USA*, pages 635–640.

Dang, V. D. and Jennings, N. R. (2004). Generating coalition structures with finite bound from the optimal guarantees. In *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems-Volume 2*, pages 564–571. IEEE Computer Society.

Dang, V. D. and Jennings, N. R. (2006). Coalition structure generation in task-based settings. In *Proceedings of the 2006 Conference on ECAI 2006: 17th European Conference on Artificial Intelligence, Riva Del Garda, Italy*, pages 210–214. IOS Press.

Dantzig, G. B. (1957). Discrete-variable extremum problems. *Operations Research*, 5(2):266–288.

Deng, X., Ibaraki, T., and Nagamochi, H. (1997). Combinatorial optimization games. In *Proceedings of the Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 720–729.

Deng, X., Ibaraki, T., and Nagamochi, H. (1999). Algorithmic aspects of the core of combinatorial optimization games. *Mathematics of Operations Research*, 24(3):751–766.

Di, B., Wang, T., Song, L., and Han, Z. (2013). Incentive mechanism for collaborative smartphone sensing using overlapping coalition formation games. In *Global Communications Conference (GLOBECOM), 2013 IEEE*, pages 1705–1710.

Dobson, G. (1982). Worst-case analysis of greedy heuristics for integer programming with nonnegative data. *Mathematics of Operations Research*, 7(4):515–531.

Dos Santos, D. S. and Bazzan, A. L. (2012). Distributed clustering for group formation and task allocation in multiagent systems: A swarm intelligence approach. *Applied Soft Computing*, 12(8):2123–2131.

Elkind, E., Goldberg, L. A., Goldberg, P. W., and Wooldridge, M. (2008). A tractable and expressive class of marginal contribution nets and its applications. In *Proceedings of the Seventh International Joint Conference on Autonomous Agents and Multi-Agent Systems*, pages 1007–1014. IFAAMS.

Farinelli, A., Bicego, M., Ramchurn, S., and Zuchelli, M. (2013). C-link: A hierarchical clustering approach to large-scale near-optimal coalition formation. In *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence*, pages 106–112.

Gillies, D. B. (1959). Solutions to general non-zero-sum games. In Kuhn, H. W. and Tucker, A. W., editors, *Contributions to the Theory of Games*, volume 4, pages 47–85. Princeton University Press.

Granot, D. (1986). A generalized linear production model: A unifying model. *Mathematical Programming*, 34(2):212–222.

Hans, K., Ulrich, P., and David, P. (2004). Knapsack problems.

Hifi, M., Michrafy, M., and Sbihi, A. (2006). A reactive local search-based algorithm for the multiple-choice multi-dimensional knapsack problem. *Computational Optimization and Applications*, 33(2-3):271–285.

Hwang, Y.-a. and Chen, M.-c. (2012). A new axiomatization of the shapley value under interval uncertainty. *Economics Bulletin*, 32(1):799–810.

Ieong, S. and Shoham, Y. (2005). Marginal contribution nets: a compact representation scheme for coalitional games. In *Proceedings of the Sixth ACM conference on Electronic commerce*, pages 193–202. ACM.

Kiekintveld, C., Islam, T., and Kreinovich, V. (2013). Security games with interval uncertainty. In *Proceedings of the Twelfth International Conference on Autonomous Agents and Multiagent Systems*, pages 231–238, Richland, SC. IFAAMS.

Kruskal, J. B. (1956). On the shortest spanning subtree of a graph and the traveling salesman problem. *Proceedings of the American Mathematical society*, 7(1):48–50.

Kuipers, J. (1993). A note on the 5-person traveling salesman game. *Zeitschrift für Operations Research*, 38(2):131–139.

Michalak, T., Rahwan, T., Elkind, E., Wooldridge, M., and Jennings, N. R. (2016). A hybrid exact algorithm for complete set partitioning. *Artificial Intelligence*, 230:14–50.

Miller, C. E., Tucker, A. W., and Zemlin, R. A. (1960). Integer programming formulation of traveling salesman problems. *Journal of the ACM*, 7(4):326–329.

Myerson, R. B. (1977). Graphs and cooperation in games. *Mathematics of operations research*, 2(3):225–229.

Ohta, N., Conitzer, V., Ichimura, R., Sakurai, Y., Iwasaki, A., and Yokoo, M. (2009). Coalition structure generation utilizing compact characteristic function representations. In *Principles and Practice of Constraint Programming*, pages 623–638. Springer.

Osborne, M. J. (2004). *An introduction to game theory*. Oxford University Press, New York.

Owen, G. (1975). On the core of linear production games. *Mathematical Programming*, 9(1):358–370.

Pisinger, D. (1998). A fast algorithm for strongly correlated knapsack problems. *Discrete Applied Mathematics*, 89(1-3):197–212.

Pisinger, D. (2005). Where are the hard knapsack problems? *Computers & Operations Research*, 32(9):2271–2284.

Potters, J. A., Curiel, I. J., and Tijs, S. H. (1992). Traveling salesman games. *Mathematical Programming*, 53(1-3):199–211.

Rahwan, T. and Jennings, N. R. (2008). An improved dynamic programming algorithm for coalition structure generation. In *Proceedings of the Seventh International Joint Conference on Autonomous Agents and Multi-Agent Systems*, pages 1417–1420. IFAAMS.

Rahwan, T., Michalak, T. P., and Jennings, N. R. (2012). A hybrid algorithm for coalition structure generation. In *Twenty-Sixth Conference on Artificial Intellegence (AAAI-12), Toronto, Canada*, pages 1443–1449.

Rahwan, T., Michalak, T. P., Wooldridge, M., and Jennings, N. R. (2015). Coalition structure generation: A survey. *Artificial Intelligence*, 229:139–174.

Rahwan, T., Ramchurn, S. D., Jennings, N. R., and Giovannucci, A. (2009). An anytime algorithm for optimal coalition structure generation. *Journal of Artificial Intelligence Research*, pages 521–567.

Sandholm, T., Larson, K., Andersson, M., Shehory, O., and Tohmé, F. (1999). Coalition structure generation with worst case guarantees. *Artificial Intelligence*, 111(1):209–238.

Sbihi, A. (2007). A best first search exact algorithm for the multiple-choice multidimensional knapsack problem. *Journal of Combinatorial Optimization*, 13(4):337–351.

Sen, S. and Dutta, P. S. (2000). Searching for optimal coalition structures. In *Proceedings of the Fourth International Conference on MultiAgent Systems*, pages 287–292. IEEE.

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

Shapley, L. S. (1953). A value for n-person games. In Kuhn, H. W. and Tucker, A. W., editors, *Contributions to the Theory of Games*, volume 2, pages 307–317. Princeton University Press.

Shapley, L. S. and Shubik, M. (1971). The assignment game i: The core. *International Journal of game theory*, 1(1):111–130.

Shehory, O. and Kraus, S. (1996). Formation of overlapping coalitions for precedence-ordered task-execution among autonomous agents. *ICMAS-96 Proceedings. Second International Conference on Multi-Agent Systems*, pages 330–337.

Skibski, O., Matejczyk, S., Michalak, T. P., Wooldridge, M., and Yokoo, M. (2016). k-coalitional cooperative games. In *Proceedings of the 2016 International Conference on Autonomous Agents & Multiagent Systems*, pages 177–185. International Foundation for Autonomous Agents and Multiagent Systems.

Sless, L., Hazon, N., Kraus, S., and Wooldridge, M. (2018). Forming k coalitions and facilitating relationships in social networks. *Artificial Intelligence*, 259:217–245.

Sultanik, E., Modi, P. J., and Regli, W. C. (2007). On modeling multiagent task scheduling as a distributed constraint optimization problem. In *Proceedings of the Twentieth International Joint Conference on Artificial Intelligence*, pages 1531–1536.

Tamir, A. (1989). On the core of a traveling salesman cost allocation game. *Operations Research Letters*, 8(1):31–34.

Tran-Thanh, L., Nguyen, T.-D., Rahwan, T., Rogers, A., and Jennings, N. R. (2013). An efficient vector-based representation for coalitional games. In *Proceedings of the Twenty-Third international joint conference on Artificial Intelligence*, pages 383–389. AAAI Press.

Ueda, S., Iwasaki, A., Yokoo, M., Silaghi, M. C., Hirayama, K., and Matsui, T. (2010). Coalition structure generation based on distributed constraint optimization. In *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence*, pages 197–203.

Ueda, S., Kitaki, M., Iwasaki, A., and Yokoo, M. (2011). Concise characteristic function representations in coalitional games based on agent types. In *The Tenth International Conference on Autonomous Agents and Multi-Agent Systems*, pages 1271–1272. IFAAMS.

Voice, T., Polukarov, M., and Jennings, N. R. (2012). Coalition structure generation over graphs. *Journal of Artificial Intelligence Research*, 45:165–196.

Wang, T., Song, L., Han, Z., Saad, W., and Ieee (2013). Overlapping coalitional games for collaborative sensing in cognitive radio networks. *IEEE Wireless Communications and Networking Conference (WCNC), Shanghai, China*, pages 4118–4123.

Wooldridge, M. (2009). *An introduction to multiagent systems*. John Wiley & Sons, Chichester, 2nd edition.

Wooldridge, M. and Dunne, P. E. (2006). On the computational complexity of coalitional resource games. *Journal of Artificial Intelligence*, 170(10):835–871.

Yun Yeh, D. (1986). A dynamic programming approach to the complete set partitioning problem. *BIT Numerical Mathematics*, 26(4):467–474.

Zanakis, S. H. (1977). Heuristic 0-1 linear programming: An experimental comparison of three methods. *Management Science*, 24(1):91–104.

Zhang, Z., Song, L., Han, Z., and Saad, W. (2014). Coalitional games with overlapping coalitions for interference management in small cell networks. *Wireless Communications, IEEE Transactions on*, 13(5):2659–2669.

Zick, Y., Chalkiadakis, G., Elkind, E., and Markakis, E. (2019). Cooperative games with overlapping coalitions: Charting the tractability frontier. *Artificial Intelligence*, 271:74–97.