Tackling the Computational Complexity of Understanding Coalition Formation in Multi-agent Systems

Thesis submitted in accordance with the requirements of the University of Liverpool for the degree of Doctor in Philosophy by Andrew James Dowell

April 12, 2010

Abstract

Tackling the Computational Complexity of Understanding Coalition Formation in Multi-agent Systems, Andrew Dowell

In its simplest metaphor, agent based computation is that undertaken *via* the interactions of autonomous computational entities (from [39]). Often, these interactions are cooperative and, in this context, a coalition describes any group of agents who may cooperate together. In any multi-agent system, to understand which coalitions will be formed by the agents, the system can be represented as a cooperative game and solution concepts from game theory can be employed. In particular, a coalition is *core stable* if no agent that belongs to this coalition can gain from forming another coalition instead. On the other hand, an *optimal coalition structure* consists of an exhaustive and disjoint collection of coalitions of agents that maximizes the welfare of the system. Given any coalition formation protocol, it is assumed that self-interested agents will always form core stable coalitions whereas fully cooperative agents will always partition themselves into an optimal coalition structure.

In a cooperative game representation, when the value obtained from forming every coalition is explicitly stated, existing research has shown that no algorithm is guaranteed to solve decision problems concerning the core and optimal coalition structure concepts with time complexity that is polynomial in the number of agents. Given this background, the research presented in this thesis aims to tackle this complexity by presenting:

(a) Algorithms that can compute these problems as efficiently as possible; and,

(b) Representations of cooperative games that can permit efficient computation of these problems.

With regard to point (a), the computational difficulties with generating an optimal coalition structure arise, in part, from the fact that the number of coalition structures grows exponentially in the number of agents. To this end, in this thesis, two optimal coalition structure generation algorithms are presented - each with the aim of efficiently generating an optimal coalition structure through analyzing only a fraction of all possible coalition structures. These algorithms develop upon the contributions of previous algorithms by considering both externalities from coalition formation and coalition value calculation processes.

In addition, with regard to point (b), existing research has shown that the complexity of computing if a given coalition belongs to the core of a game is polynomial in the size of the game itself. However, because the number of coalitions grows exponentially in the number of agents, the general representation will have size that is exponential in the number of agents. Thus, from a computational perspective, this is not positive. Given this insight, this thesis also contributes to the state-of-the-art understanding of multi-agent systems through developing a concise representation of coalition formation between self-interested agents. For certain, natural instances of this representation, a system user is able to solve problems concerning core stability with time complexity that is polynomial in the number of agents.

Acknowledgements

I am tremendously grateful to a number of people who have helped me throughout the course of my post-graduate study.

Firstly, I owe great thanks to Tomasz Michalak, arguably the most pro-active, motivated and enthusiastic person I have ever had the privilege to work with. I am particularly appreciative of his support and advice with respect to identifying research problems. Without his support, this thesis would certainly be considerably lighter than it is.

Joint firstly, I am indebted to the support provided by my supervisors Peter McBurney and Mike Wooldridge who, despite their heavy workloads, were always willing to read the work I submitted and provide constructive feedback (sometimes even when they were on their holidays). I am also thankful for the guidance they provided with respect to "keeping me on the straight and narrow" by helping me to focus on one particular research area. In addition, I owe many thanks to both Leslie Goldberg, who took time out of her hectic schedule to help me understand the fundamentals of complexity theory, and Thelma Williams, who (despite our conflicting football alliances) was always willing to help me complete the administrative side of the course. Also, I am thankful to all of the technical and administrative staff, including Ken Chan, Dave Shield, Helen Bradley and Janet Lowery for the expertise they offered when required.

Although brief, it was a pleasure to work with Talal Rahwan. I learned a lot from his ability to concisely write and present complex work, as well as his motivation and determination to ensure that the final work was to the highest standard it could be. Additionally, it is also important that I thank the various people on the MBC project for their feedback to the various ideas I have presented. I am especially thankful to Arthanis Papakonstantinou, Enrico Gerding and Nick Jennings from The University of Southampton and Peter Lewis, Edd Robinson, Xudong Luo and Xin Yao from The University of Birmingham.

Outside of work, special thanks go out to Dan Crosfield and Laura Beswick. Whilst I was investigating ways to extend the optimal coalition structure generation problem, they were taking care of less trivial matters such as ensuring I had somewhere to live and that bills were being paid. Also, I would like to thank the various people I have lived with, including Matthew Shaw and Andrew Shuttleworth. Their patience was particularly appreciated during times when working schedules (and laziness) often resulted in me abandoning housework.

Finally, thanks are owed to Tim Miller and Tony McCabe who organized weekly five a side football sessions.

Glossary of Terms

Ag	A set of agents.
a_i	An agent in Ag.
n	The number of agents.
C	A coalition of agents.
π	A coalition structure.
Π	The set of all coalition structures.
$\mathcal{E}$	The space of all embedded coalitions.
$\mathcal{P}$	A partition function game.
$P(C;\pi)$	the value of coalition C embedded in structure π.
$\mathcal{N}_t$	A characteristic function game with transferable utility.
$\mathcal{N}_{nt}$	A characteristic function game with non-transferable utility.
$v(C)$	The value of coalition C.
$\underline{\mathbf{x}}$	An imputation.
x_i	Agent a_i's allocation of $\underline{\mathbf{x}}$.
θ_i	Agent a_i's Shapley value allocation.
π^*	An optimal coalition structure.
G_W	A weighted graph game.
$w(i,j)$	The weighted value of the edge connecting agents a_i and a_j in G_W.
WCG	Weighted coalitional game.
$\mathcal{S}$	A synergy game.
MCN	A marginal contribution nets representation.
$\mathcal{H}$	A hedonic coalitional game.
$\succ_i$	The preference ordering of agent a_i in $\mathcal{H}$.
$C_i(\pi)$	The coalition in π to which agent a_i is a member.
$\mathcal{H}'$	A hedonic nets representation.
Γ	A qualtiative coalitional game representation.
G	The set of goals in Γ.
m	The number of goals in G.
G_i	The goals in G agent a_i wishes to accomplish.
Ψ	A formula of proposition logic that represents v_q.
Γ_{succ}	A representation of Γ that uses Ψ instead of v_q.
$\mathcal{B}_n$	The Bell number of n.
$\overline{G}$	The set of all integer partitions of n.
$\overline{g}$	An integer partition of n in $\overline{G}$.
$UB_{\overline{g}}$	The upper bound on the values of all coalition structures in $\overline{g}$.
$LB_{\overline{g}}$	The lower bound on the values of all coalition structures in $\overline{g}$.
$AV_{\overline{g}}$	The average bound on the values of all coalition structures in $\overline{g}$.
$UB_{\overline{G}}$	The upper bound on the values of all coalition structures in $\overline{G}$.
$LB_{\overline{G}}$	The lower bound on the values of all coalition structures in $\overline{G}$.
$AV_{\overline{G}}$	The average bound on the values of all coalition structures in $\overline{G}$.
max_S	The maximum value of all coalitions of size S.
min_S	The minimum value of all coalitions of size S.
av_S	The average value of all coalitions of size S.

$\overline{g}^*$	The most promising subspace in $\overline{G}$.
$\mathbf{M}_{i_j}$	A temporary array that can be used to cycle through all possible coalitions of size $i_j \in \mathbb{R}$.
$\mathcal{L}_s$	The list containing all coalitions of size s.
$\|\mathcal{L}_s\|$	The number of coalitions in $\mathcal{L}_s$.
$\mathcal{L}_{s,i}$	Agent a_i's allocation of the coalitions in $\mathcal{L}_s$ in 'basic' allocation method.
$\text{Index}_{s,i}$	The last coalition in the a_i's allocation in 'basic' allocation method.
$\mathcal{L}_{s,i}^{1/2}$	Agent a_i's allocation of the coalitions in $\mathcal{L}_s$ in refined allocation method.
$\text{Index}_{s,i}^{1/2}$	The last coalition in the a_i's allocation in refined allocation method.
$\mathcal{L}'_s$	The 'left over' coalitions after allocation.
$\mathcal{L}'_{s,i}$	Agent a_i's allocation of the 'left over' coalitions.
α	Index of agent who is assigned the first left over coalition in $\mathcal{L}_s$.
A'	The set of agents the left over coalitions in $\mathcal{L}_s$.
F	Set of all coalitions.
F_p	Set of all promising coalitions.
F_{np}	Set of all non-promising coalitions.
$\tilde{d}_s$	The domination value of all coalitions in $\mathcal{L}_s$.
FR1-**FR3**	The filter rules.
$\overrightarrow{Z}_s(j)$	All coalitions in $\mathcal{L}_s$ in which a_j is the first agent.
$\mathcal{P}_{sup}^{+}$	Partition function games with super-additivity and positive externalities.
$\mathcal{P}_{sup}^{-}$	Partition function games with super-additivity and negative externalities.
$\mathcal{P}_{sub}^{+}$	Partition function games with sub-additivity and positive externalities.
$\mathcal{P}_{sub}^{-}$	Partition function games with sub-additivity and negative externalities.
$v^{max}(C)$	The maximum value of coalition C.
$v^{min}(C)$	The minimum value of coalition C.
a	The synergy from super-additivity.
b	The loss from negative externalities.
$\Gamma_{\mathcal{H}}$	Hedonic qualitative coalitional games.
P1 - **P2**	Preference assumptions in $\Gamma_{\mathcal{H}}$.
$T = (C, G')$	A team in $\Gamma_{\mathcal{H}}$.
π_T	A team structure in $\Gamma_{\mathcal{H}}$.
Π_T	The set of all team structures in $\Gamma_{\mathcal{H}}$.
$T_j(\pi_T)$	The team in π_T to which agent a_j is a member.
$\mathcal{T}_i$	Agent a_i's preference over the teams they can form.
P^{seq}	The sequential coalition formation protocol.
h^t	The history of P^{seq} at stage t.
$\mathcal{R}$	The rule of order defining the turns of the agent in P^{seq}.
δ	A time period during which agents can propose or react to proposals in P^{seq}.
Ag^-	A set of agents who have already formed teams.
π_{T,Ag^-}	A team structure formed by the agents inAg^-.
$\hat{T} = (\hat{C}, G')$	An on-going proposal.
Ag^A	A set of agents who have accepted an on going proposal.
$\mathcal{L}_{reject}$	A list of rejected teams.
λ	A strategy profile containing strategies played by agents.
λ_i	the strategy played by agent a_i in λ.
λ^*	A sub-game perfect Nash equilibrium strategy profile.
$\pi(\lambda)$	The team structure formed by the agents playing λ.
$T_i(\pi(\lambda))$	the team to which agent a_i is a member in the structure formed from the agents playing λ.
$\mathcal{G}_{game}$	The game tree representation of P^{seq}.
$\mathcal{T}$	The set of all non-null teams that can be formed by the agents.
π_O	The structure formed from the agents participating in P^{seq}.

$\Gamma_{\mathcal{H}_U}$	Hedonic coalitional games with universal preferences.
$S(C)$	The space of all coalitions in $\Gamma_{\mathcal{H}}$ and in $\Gamma_{\mathcal{H}_U}$.
$S_i(C)$	The space of all coalitions of size i in $\Gamma_{\mathcal{H}}$ and in $\Gamma_{\mathcal{H}_U}$.
$C_{i,j}$	The j^{th} coalition in $S_i(C)$.
S^*	Sub-game perfect Nash equilibrium strategy for P^{seq} in $\Gamma_{\mathcal{H}_U}$.
π_T^*	An optimal team structure.

Part I

Background and Introduction

Contents

Chapter 1

Introduction

Advances in the theory of computing have resulted in the development of *multi-agent system technologies* [39]. Conceptually, an *agent* is interpreted as a computational entity that satisfies the following criteria (from [83]):

I1 *Autonomy, i.e.,* agents are capable of independent action and do not require human involvement with respect to making decisions;

I2 *Reactivity, i.e.,* agents are able to perceive their environment and respond, in a timely fashion, to changes that may occur;

I3 *Pro-activity, i.e.,* agents are able to exhibit goal directed behaviour by taking the initiative in order to accomplish their goals; and,

I4 *Social ability, i.e.,* agents are capable of interacting with others.

All of **I1-I4** provide a notion of intelligence and, against this criteria, an agent is formally defined as follows.

Definition 1.1 *(adapted from [83]) An agent is a computer system that is situated in some environment and, in this environment, is capable of all of **I1-I4** in order to meet its design objectives.*

Given Definition 1.1, a *multi-agent system* consists of a number of agents who are situated in a common environment and who carry out their individual activities within that environment. For the purposes of this thesis, a *coalition* simply refers to a group of agents who may cooperate together and, in a number of multi-agent systems, it has been illustrated that an agent's performance improves through cooperating with others as opposed to working alone [34]. For example:

- In *distributed sensor networks*, autonomous sensors form coalitions in order to monitor targets of interest [14];
- In *e-commerce* systems, buyer agents form coalitions to purchase a product in bulk and take advantage of price discounts [46];
- In systems where intelligent agents negotiate over meeting scheduling options on behalf of people for whom they work, agents form coalitions when they agree upon a schedule [69]; and,
- In *information gathering systems*, such as the RETSINA (REusable Task-based System of Intelligent Networked Agents) system considered in [73]. Here, due to the complexity of gathering information, as well as the specialist knowledge of the agents, the agents in this system form coalitions in order to efficiently gather information.

In the context of coalition formation, two broad classes of agent have been identified (from [77]):

- Those that are *fully cooperative*; and,
- Those that are *self-interested*.

These two agent types are characterized by how the agents work together. Typically, fully cooperative agents will share common goals and cooperate without any regard to individual payoff or utility. For example, agents concerned with *cooperative distributed problem solving* (that is, how loosely coupled networks of problem solvers work together to solve problems) can be interpreted as fully cooperative if they all have the same goal of solving the same problem [81]. In these systems, if the agents have different expertise or the problem is inherently complex to solve then these agents can form coalitions in order to solve the problem.

In contrast to fully cooperative agents, self-interested agents do not share common goals and are concerned with maximizing their own individual utility. For example, consider coalition formation between agents representing different transport centres that deliver packages [67]. Here, self-interested agents can form coalitions in order to reduce expenses. However, because the agents represent different companies, they will only form coalitions that are best for the company and not the system. Additionally, consider an electronic market populated with automated agents which represent different enterprises who buy and sell [34]. Here, buyers and sellers can form coalitions in order to establish strong business connections. However, buyers and sellers are only going to form coalitions if it is beneficial to themselves (*i.e.*, if sellers can improve profit while buyers get a good deal on what they purchase) .

To understand which coalitions will be formed by the agents, *cooperative game theory* can be used. In this discipline, the system can be represented as a *cooperative game* that can consist of:

(i) The agents;

(ii) The coalitions that can be formed; and,

(iii) The value obtained from forming each of these coalitions.

In cooperative games, it is generally assumed that:

CG1 The value obtained from forming coalitions is measured numerically;

CG2 The value is attributed to the coalition as a whole, *i.e.*, it is not allocated to the individuals who belong to it; and,

CG3 The value is not affected by co-existing coalitions, *i.e.*, there are *no externalities from coalition formation*.

It should be noted that, with regard to **CG3**, the term externality is used to describe the effect that the formation of a particular coalition may have on the value of other co-existing coalitions. Assumption **CG3** simply states that co-existing coalitions do not affect one another, meaning every formed coalition has the same value at any moment in time.

There are both advantages and limitations to using game theory. On the plus side, it is sufficiently flexible to represent many different multi-agent system domains (as will be evidenced by the various representations of cooperative games in Chapter 2) and it can enable mathematical proof. However, on the negative side, cooperative game theory makes a number of assumptions that are not always true in the real world. For example, it is often assumed that, at any moment in time, agents can belong to no more than one coalition and that agents have perfect information regarding the system. The former point is not always true in a number of real world multi-agent systems, such as in the distributed network sensor systems considered in [14] where the agents can belong to more than one coalition. Whereas one line of research has focused on addressing its real world limitations (see [78]), this thesis addresses the computational limitations of game theory.

1.1 Computational Limitations of Cooperative Game Theory

In cooperative games where all of **CG1**-**CG3** hold, three key issues have been researched in the multi-agent systems literature thus far (from [66]):

Key Issue 1 *Coalition structure generation, i.e.,* how the agents decide who to form coalitions with;

Key Issue 2 *Solving the optimization problem of each coalition, i.e.,* how the agents find the best way to maximize the utility the coalition receives; and,

Key Issue 3 *Allocating the utility obtained from forming each coalition, i.e.,* how the agents in the coalition share the utility among themselves.

Since the research in this thesis investigates coalition formation between both self-interested and fully cooperative agents, this thesis focuses on both **Key Issue 1** and **Key Issue 3**. This is because self-interested agents make their decisions regarding coalition formation based on the allocation of the gain they receive. In particular, the research presented in this thesis investigates the computational complexity of solving natural problems concerning coalition formation between self-interested and fully cooperative agents, *i.e.*, this research focuses on the following question:

How hard is it to compute which coalitions will be formed?

Firstly, consider fully cooperative agents. If a cooperative game satisfies all of **CG1**-**CG3** (that is, the cooperative game assumptions specified on the previous page) then, for any exhaustive and disjoint collection of coalitions of agents (referred to as a *coalition structure* from now on), the utilitarian metric is a natural measure from which to assess the welfare of the system should these coalitions be formed by the agents. In other words, the utility attributed to the coalition structure is equal to the combined values of all the coalitions that belong to that structure. In this context, the coalition structure with maximal utility is the one that represents the collection of coalitions that, if formed by the agents, maximizes the welfare of the system. This coalition structure is referred to as an *optimal coalition structure* and, in any multi-agent system represented as a cooperative game that satisfies all of **CG1**-**CG3**, (given sufficient time, processing capabilities and knowledge) fully cooperative agents will always partition themselves into optimal coalition structures, irrespective of whether the coalitions in these structures are necessarily the best ones for themselves.Thus, to determine which coalitions will be formed by fully cooperative agents, an optimal coalition structure must be generated. In any cooperative game representation that satisfies all of **CG1**-**CG3**, in order to guarantee that a given coalition structure is optimal, the utilitarian value of every coalition structure may have to be computed. From a computational perspective, this is problematic because the number of possible coalition structures that can be formed grows exponentially with the number of agents.

However, if the value obtained from forming every coalition is known then, as there are no externalities from coalition formation (as given by assumption **CG3**), it may be possible to identify *a priori* whether certain structures cannot be optimal. In this way, the computational difficulty of optimal coalition structure generation can be circumvented because the values of those structures that definitely cannot be optimal need not be computed. Consequently, in the multi-agent system literature, for cooperative games where all of **CG1**-**CG3** hold, one line of research has focused on developing algorithms that can efficiently generate an optimal coalition structure [84, 66, 55].

In contrast to the above, if the agents are self-interested then they will choose to form the coalitions that are best for themselves as individuals. If the cooperative game satisfies all of **CG1**-**CG3** then the coalitions formed by a self-interested agent are influenced by the allocation of the utility they receive. Intuitively, a self-interested agent is not going to form a coalition if it can be guaranteed a bigger utility share by forming another coalition instead. Coalitions in which none of the agents have incentive to defect from and form other coalitions instead are referred to as *stable*. Clearly, self-interested agents will only form stable coalitions, meaning the coalitions that they form is dependent upon the allocation of the gain they receive. As the next subsection will show, the complexity involved in determining which coalitions are formed by self-interested agents is related to the size of the cooperative game representation.

1.1.1 Representing Cooperative Games

It cannot be guaranteed that self-interested agents will partition themselves into coalition structures that maximize the welfare of the system. Instead, problems concerning stability can be answered to determine which coalitions will be formed by self-interested agents.

Existing research has shown that the difficulty in computing these problems is related to the size of the cooperative game representation [17, 13, 29, 20]. In the multi-agent systems literature, a number of representations of cooperative games that satisfy all of **CG1**-**CG3** have been proposed. To assess the quality of any representation, the following four criteria can be used (from [29]):

Expressivity: the breadth of the class of coalitional games covered by the representation;

Conciseness: the space requirements of the representation;

Efficiency: the efficiency of the algorithms that can be developed for the representation; and,

Simplicity: the ease of use of the representation by users of the system.

Given a cooperative game representation, it would be desirable that this representation can model all possible classes of coalitional game. Also, because the number of coalitions grows exponentially in the number of agents, it would be desirable that the representation is concise as possible and permits efficient computation of problems concerning coalition formation. In addition, it would also be desirable that the representation is such that a user of the system can easily study the game in order to solve these problems. To this end, against the above criteria, it is generally accepted that the ideal representation should:

1. Be fully expressive;
2. Use as little space as possible;
3. Enable efficient computation of coalition formation related problems; and,
4. Be easy to use by users of the system.

A natural way to represent a cooperative game would involve explicitly stating the value obtained from forming every coalition. These representations are fully expressive over the domains they represent. Also, problems related to coalition stability can be answered through analyzing the values obtained from forming every coalition. However, as there are 2^n coalitions that could potentially be formed in a system of n agents, these representations will have size that is exponential in the number of agents. This implies that, with respect to computing coalition stability, any positive results are neither meaningful nor computationally significant. Against both conciseness and efficiency, this is an undesirable feature of these representations.

A representation scheme is *succinct* if it has size polynomial in the number of agents. Clearly, succinct representations are desirable from a conciseness perspective but they are not guaranteed to be fully expressive as there can exist classes of cooperative games that cannot be captured within a succinct representation. Furthermore, despite the conciseness of the representation, they also do not guarantee that coalition formation related problems can be efficiently answered [1].

Against this reasoning, it is therefore desirable to develop *compact* representations of cooperative games that strike a useful balance between both conciseness and efficiency. In other words, it is desirable to develop fully expressive representations of cooperative games that are succinct for cases of interest yet still allow for coalition formation related problems to be efficiently computed.

1.1.2 Coalition Formation Protocols

Following previous discussion, fully cooperative agents will always form coalitions that maximize the welfare of the system whereas self-interested agents will always form stable coalitions. With respect to the latter case, as the work in later chapters will show, in a number of cooperative games, there is no guarantee that coalitions which meet the requirements of various stability criteria will exist. . In these games, as there does not exist a single coalition that is beneficial to all its members, if an agent desires to form a coalition then they may have to negotiate its formation with other agents who belong to that coalition.

In the context of negotiation, a coalition formation protocol is a set of rules that define how agents can interact whilst negotiating. For example, if the protocol is *sequential* then the agents take it in turn to interact (such as those sequential protocols considered in [5, 24, 19]) whereas if the protocol is *simultaneous* then the agents simultaneously announce the coalitions they want to form (such as those simultaneous protocols considered in [45, 60]). In coalition formation protocols, a *strategy* defines the choice made by an agent with respect to forming a coalition and whenever an agent plays a strategy, this is described as a *stage* of the protocol. Following previous discussion, given any protocol, self-interested agents will always play strategies that result in them forming stable coalitions. When there is no guarantee that stable coalitions exist, the agents are faced with following problem:

Which coalitions should be formed?

In any protocol, a *strategy profile* consists of a tuple of strategies played by every agent. In protocols that require a number of stages, the concept of a *sub-game perfect equilibrium* can be used to identify strategies for the agents to play. Informally, a strategy profile is said to be *Nash equilibrium* if the deviation of any agent from the strategy they play in the profile, given that all of the other agents do not change their strategies, does not result in the agent being better off as a consequence. With this in mind, a strategy profile is a *sub-game perfect equilibrium* if, at any stage of the negotiation process, no matter what the history is, no agent is motivated to deviate and play another strategy other than what is defined in the strategy profile (these definitions are taken from [19]). Against this insight, given any coalition formation protocol that may require a number of stages, the following question naturally arises:

Does there exist a sub-game perfect equilibrium profile and,
if so, how hard is it to compute this profile?

1.2 Research Contributions

Firstly, consider optimal coalition structure generation. For any cooperative game that satisfies all of **CG1-CG3** (that is, the cooperative game assumptions specified earlier), existing optimal coalition structure generation algorithms have been developed that can efficiently generate an optimal coalition structure. A number of these algorithms take, as input, the values obtained from forming every coalition (referred to as the coalition value from now on). Since there are no externalities from forming coalitions in these games, these algorithms attempt to identify *a priori* if groups of coalition structures cannot be optimal and, from this information, analyze only those structures that could potentially be optimal.

All of these algorithms commence from the moment all coalition values have been computed. This is surprising because, even for moderate numbers of agents, there are an exponential number of coalitions and the process of computing all coalition values is not trivial. To this end, the first contribution of this thesis is an optimal coalition structure generation algorithm that considers both coalition value calculation and optimal coalition structure generation. This algorithm consists of a heavily refined version of the sequential application of the *distributed coalition value calculation (DCVC)* and *Integer Partition (IP) optimal coalition structure generation* algorithms (as presented in both [56] and [59], respectively). Since the computational processes in the DCVC algorithm are distributed among the agents, whereas in the IP algorithm they are coordinated by a single entity, connecting the two algorithms is not trivial. Thus, pre-processing

techniques are developed which can be incorporated into this algorithm. These techniques are presented as filter rules that can identify coalitions that cannot belong to an optimal coalition structure. In the coalition value calculation stage, the values of these coalitions are removed from the algorithm and, in the optimal coalition structure generation stage, the values of the coalition structures containing these coalitions are not computed. These filter rules can reduce the number of coalition values an individual agent needs to transfer to the entity who is to execute the optimal coalition structure generation phase, as well as the number of coalition structure values that are computed during the coalition structure generation phase. In this way, the filter rules can overcome the computational difficulties associated with optimal coalition structure generation. Furthermore, this algorithm, combined with the filter rules, provides a foundation from which a distributed optimal coalition structure generation algorithm can be developed.

Now, cooperative games that satisfy all of **CG1**-**CG3** are sufficient to represent coalition formation in many real world multi-agent systems (particularly the distributed sensor network and e-commerce systems considered at the start of this chapter). This is because the coalitions either do not interact with each other while pursuing their own goals or because such interactions are small enough to be neglected. However, in a number of multi-agent systems, assumption **CG3** (that is, the assumption that there are no externalities from coalition formation) may not hold. For instance, as multi-agent system technologies advance, they can be applied to solve increasingly complex cooperative distributed problems. As these problems become increasingly complex, interdependencies between coalitions may also increase, meaning *ad hoc* coalition formation may need to allow for externalities from coalition formation. Against this insight, this thesis contributes to the state-of-the-art by investigating optimal coalition structure generation in cooperative games where **CG3** does not hold, *i.e.*, cooperative games where there exist externalities from coalition formation. In these representations, optimal coalition structure generation is particularly problematic since the value of every coalition is dependent upon the structure to which it belongs. In this way, it is not possible to predict the values of all coalition structures without having to actually compute them. Consequently, the value of every coalition structure will have to be computed in order to guarantee an optimal coalition structure. Nevertheless, in this thesis, for certain natural classes of these representations, an algorithm is developed that is able to generate an optimal coalition structure without having to analyze all possible coalition structures that can be formed. In particular, by analyzing only a fraction of all coalition structures, this algorithm is able to bound the maximum and minimum values of all possible coalition structures that could be formed. After doing this, the algorithm is then able to exploit this information and analyze all the remaining coalition structures whilst avoid those coalition structures that cannot be optimal. This contribution is particularly significant since it is the first optimal coalition structure algorithm to be developed for cooperative games where **CG3** does not hold.

As well as optimal coalition structure generation, this thesis also introduces a novel representation of coalition formation between self-interested agents. In this representation, agents form coalitions based on both the set of goals this coalition is able to accomplish and the agents who belong to the coalitions. This so-called *hedonic qualitative coalitional game* (HQCG) representation combines facets from both the existing hedonic and qualitative coalitional game representations (the latter two representations are presented in Section 2.4.5 and Section 2.4.6, respectively).

In the HQCG representation, various concepts of stability are formalized. For many of these concepts, there is no guarantee that coalitions which satisfy these stability criteria will exist. To this end, a sequential coalition formation protocol is developed such that if all the agents participate in the protocol then a coalition structure will be formed. With the exception of the negotiation protocol presented in [19], in contrast to most of the negotiation protocols developed in the multi-agent systems paradigm, this protocol considers negotiation among n agents (as opposed to bilateral negotiations) and assumes that the utility is non-transferable (as opposed to transferable).[1] Although stable coalitions are not guaranteed to exist in these representations, an equilibrium strategy is guaranteed to exist for this protocol. However, even if the representation is concise, there is no guarantee that this equilibrium strategy can be efficiently computed. Furthermore, insincere

[1] See [35] for more details.

agents may be able to manipulate the protocol so that they have an advantage over other agents. Against this insight, a natural class of hedonic qualitative coalitional games are studied in which a stable coalition structure is guaranteed to exist. In this class of games, if the representation is concise then this core stable structure can be efficiently generated.

1.2.1 Thesis Structure

The remainder of this thesis is divided into three parts. The first part is divided into two chapters which provide a thorough background to the study of coalition formation in the multi-agent system paradigm. Specifically:

In Chapter 2, an overview of some of the key concepts from cooperative game theory is provided. In addition, an overview of the representations of cooperative games that have been developed in the multi-agent system paradigm is also provided; and,

In Chapter 3, the state-of-the-art optimal coalition structure generation algorithms that have been developed in the multi-agent system paradigm are presented.

Given this background, the second part is divided into three chapters that each describe the research contributions made by this thesis. To be precise:

In Chapter 4, optimal coalition structure generation is considered as a two stage process consisting of a coalition value calculation stage and an optimal coalition structure generation stage;

In Chapter 5, optimal coalition structure generation is studied in natural classes of cooperative games where there are externalities from coalition formation; and,

In Chapter 6, hedonic qualitative coalitional games are studied.

It is worth pointing out that the findings presented in Chapter 4 and Chapter 5 were first published in the following refereed proceedings, respectively:

> T. Michalak, A. Dowell, P. McBurney and M. Wooldridge [2009]: Pre-processing techniques for anytime optimal coalition structure generation. In *J.-J. Ch. Meyer and J. Broersen (eds.), Knowledge Representation for Agents and Multi-Agent Systems (In Proceedings of KRAMAS 2008), LNAI 5605, Springer Berlin / Heidelberg, 2009*. [44].

> T. Michalak, A. Dowell, P. McBurney and M. Wooldridge [2008]: Optimal coalition structure generation in partition function games. In *M. Ghallab and C.D. Spyropoulos (Editor): Proceedings of the 18th European Conference on Artificial Intelligence (ECAI 2008)*, Patras, Greece: July 2008. [43].

In addition to the above, the third part of this thesis consists of one chapter which provides a conclusion to the research that is presented in Chapters 4-6. In more detail:

In Chapter 7, the contributions and significance of the work presented in Chapters 4 - 6 are summarized and possible avenues for further research are presented.

Finally, because computational complexity features heavily in this work, as a point of reference, key concepts from complexity theory are presented in Appendix A.

Chapter 2

Cooperative Games

Game theory is a branch of applied mathematics that aims to understand the best course of action for those involved in *strategic situations*, *i.e.*, situations where an individual's success in making choices depends on the choices of others [49, 4]. In this discipline, the strategic situation is represented as a *game*. These games can be divided into two main types: *cooperative* and *non-cooperative*. In both non-cooperative and cooperative games, self-interested agents desire to maximize their individual gains. However, the fundamental difference between the two is that, in cooperative games, agents can make binding agreements and form *coalitions* whereas, in non-cooperative games, this is not possible. As this thesis is concerned with coalition formation, in this chapter, an overview of cooperative game theory is presented. To be precise:

- In Sections 2.1 and 2.2, the partition function and characteristic function game representations of cooperative games are presented;
- In Section 2.3, solution concepts are presented for characteristic function game representations. These concepts can help identify coalitions that will be formed and, in this section, the core and optimal coalition structure concepts are formally defined; and,
- In Section 2.4, an overview of existing representations of characteristic function games is presented. In particular, the following representations are presented:
    1. Weighted graph games;
    2. Weighted coalitional games;
    3. Synergy games;
    4. Marginal contribution nets;
    5. Hedonic Coalitional Games; and,
    6. Qualitative Coalitional Games.

2.1 Partition Function Games

Cooperative games were first considered in [80]. Generally, the value obtained from forming a coalition can depend upon the other coalitions that are formed simultaneously. Therefore, when given a set of agents Ag, to determine if a given coalition $C \subseteq Ag$ is a good one to join, all other co-existing coalitions $C' \subseteq Ag \setminus C$ must be considered as well. To this end, consider the following definition.

Definition 2.1 *A* coalition structure *(π) is a partition of the agents in Ag.*

Any coalition $C \subseteq Ag$ that belongs to a structure π is said to be *embedded* in π. If the utility attributed to a coalition $C \subseteq Ag$ is affected by co-existing coalitions then every coalition C may have different values in each coalition structure to which it is embedded. *Partition function game* representations, first proposed in [38], measure the value of forming every coalition $C \subseteq Ag$ in every coalition structure to which the coalition is embedded.

Definition 2.2 *Let:*

- $\mathcal{E}$ *denote the set of all embedded coalitions* $(C; \pi)$*; and,*
- Π *denote the space of all coalition structures.*

A partition function game with transferable utility $\mathcal{P} = \langle Ag, P \rangle$ *consists of:*

- *A set of agents* $Ag = \{a_1, \ldots, a_n\}$*; and,*
- *A* partition function P *which takes, as input, an embedded coalition* $(C; \pi) \in \mathcal{E}$ *and outputs a real number value reflecting the value obtained from forming coalition* C *given that the other coalitions in* π *have also formed,* i.e.*,*

$$P : \mathcal{E} \to \mathbb{R}.$$

Intuitively, partition function games account for *externalities from coalition formation* where the formation of coalitions may affect the utility attributed to co-existing coalitions. Externalities from coalition formation can exist in many real world multi-agent systems. For example, consider fisheries on the oceans [52]. If the agents represent different fishing companies and, upon forming a coalition, the agents in the coalition decide to reduce fishing activities then this may have a positive impact on co-existing coalitions as these coalitions may gain more when this additional competition is reduced. Similarly, consider coalition formation between agents representing research and development firms [10]. In this system, the formation of a particular coalition may have a negative impact on the value of co-existing coalitions. This is because the market positions of some firms could be hindered by the increased competitiveness resulting from a collusion between other firms. In this way, partition function games can model coalition formation in multi-agent systems where agents represent either research and development or fishing firms.

2.2 Characteristic Function Games

In many natural systems, co-existing coalitions either do not interact with each other while pursuing their own goals or the interactions are insignificant enough to be neglected. In such systems, the value obtained from forming a coalition is independent of co-existing coalitions, *i.e.*, there are no externalities from coalition formation. Thus, for every coalition, the utility obtained from forming the coalition is the same in every structure to which it is embedded.

The manner in which the utility is attributed to a coalition gives rise to two natural classes of characteristic function game:

1. Characteristic function games with transferable utility; and,
2. Characteristic function games with non-transferable utility.

Consider Characteristic function games with transferable utility first.

Definition 2.3 Characteristic function games with transferable utility $\mathcal{N}_t = \langle Ag, v \rangle$ *consist of:*

- *A set of agents* $Ag = \{a_1, \ldots, a_n\}$*; and,*
- *A function* $v : 2^{Ag} \to \mathbb{R}$ *that takes, as input, a coalition* $C \subseteq Ag$ *and outputs a real number value* $v(C) \in \mathbb{R}$.[1]

[1] Although it will not always be explicitly stated, in this work, it is always assumed that $v(\emptyset) = 0$.

Observe that the characteristic function game representation in Definition 2.3 satisfies all of **CFG1**-**CFG3**, *i.e.*, it satisfies all the criteria which were inherent to the cooperative games considered in the previous chapter. Because the utility from forming coalitions is attributed to the coalition as a whole, it can be freely distributed among all of the agents who belong to the coalition. Clearly, the characteristic function game representation in Definition 2.3 is a restrictive version of the the partition function game representation presented in Definition 2.2. Despite this restriction, a number of real world multi-agent system scenarios can be represented as characteristic function games, including:

- *Distributed sensor networks*, where autonomous sensors may cooperate to monitor targets of interest [14]; and,
- *E-commerce*, where buyers can form coalitions to purchase a product in bulk and take advantage of price discounts [46].

Now, in some systems, the utility from forming a coalition may not be attributed to the coalition as a whole but to the individual agents who belong to the coalition. When this is the case, characteristic function games with non-transferable utility can be used to represent the system.

Definition 2.4 Characteristic function games with non-transferable utility $\mathcal{N}_{nt} = \langle Ag, v \rangle$ *consist of:*

- *A set of agents* $Ag = \{a_1, \ldots, a_n\}$*; and,*
- *A function* v *that takes, as input, a coalition* $C \subseteq Ag$ *and outputs a set,* $v(C) \subseteq \mathbb{R}^{|C|}$*, that is interpreted as the set of payoffs coalition* C *can achieve for its members.*

Intuitively, in these games, the set of payoffs can represent the utilities attributed to the individual agents depending upon the choices made or set of actions collectively undertaken by the agents in the coalition. Note that, when the utility is non-transferable, the set of payoffs can be of a size that is exponential in the number of agents. Thus, characteristic function games with transferable utility may be exponentially more concise than characteristic function games with non-transferable utility. Additionally, in characteristic function games with transferable utility, the manner in which the utility is allocated to the coalition as a whole gives rise to a number of natural classes of this representation. Consequently, unless stated otherwise, whenever the term 'characteristic function game' is used through out the rest of this document, it is in reference to a characteristic function game with transferable utility.

2.2.1 Classes of Characteristic Function Games

Consider characteristic function games with non-transferable utility. Due to the diverse environments in which multi-agent systems technology can be utilized, for those that can be represented as characteristic function games, it may be possible to make the function less general by ensuring that it satisfies certain criteria. For instance, consider super-additivity:

Definition 2.5 *In any characteristic function game with transferable utility* $\mathcal{N}_t$*, the function* v *is* super-additive *if, for every pair of disjoint coalitions, the combined value of each disjoint coalition is not less than the value of the union,* i.e.,$\forall C, C' \subseteq Ag : C \cap C' = \emptyset$*,*

$$v(C \cup C') \geq v(C) + v(C').$$

Intuitively, for every coalition $C \subseteq Ag$, the super-additivity condition states that a coalition of agents cannot collectively gain more if they partition themselves into two separate coalitions. An important class of super-additive games are *convex games*.

Definition 2.6 *In any characteristic function game with transferable utility* $\mathcal{N}_t$*, the function* v *is* convex *if* $\forall C, C' \subseteq Ag$*,*

$$v(C \cup C') + v(C \cap C') \geq v(C) + v(C').$$

Clearly, all convex games are super-additive but the converse is not true. Thus, in this context, convex games are stronger than super-additive ones. Following [74], convex characteristic function games can represent a typical scheduling application. For example, suppose J is a set of jobs and K is a set of machines such that each job can only be scheduled on certain machines but not others. If there is a value to each scheduled job then the cooperative game $\langle J \cup K, v \rangle$, where v is the maximal value of the jobs, is convex.

In contrast to super-additivity, in certain settings, it may be that coalitions of smaller size gain no less than bigger ones. To this end, the notion of *sub-additivity* is considered.

Definition 2.7 *In any characteristic function game with transferable utility $\mathcal{N}_t$, the function v is* sub-additive *if, for every pair of disjoint coalitions, the value of the union is not greater than the combined value,* i.e.*, $\forall C, C' \subseteq Ag : C \cap C' = \emptyset$,*

$$v(C \cup C') \leq v(C) + v(C').$$

If the characteristic function game is sub-additive and any two coalitions merge to form a new coalition then this new coalition can gain no more than the combined values of any two coalitions whose merge created this coalition. This constraint may be applicable in representations of systems where there is a cost incurred from cooperation within large groups. Thus, as agents join coalitions, the cost incurred increases and, therefore, the utility obtained from cooperating may be reduced.

In contrast to the above, in a number of settings, it may not always be possible to measure the performance of a coalition in terms of a real number utility value but rather whether the coalition is 'good' or 'bad' instead. In such cases, the following constraint may be useful.

Definition 2.8 *In any characteristic function game with transferable utility $\mathcal{N}_t$, the function v is said to be* simple *if $v : 2^{Ag} \rightarrow \{0, 1\}$,* i.e.*, v outputs either '1' or '0'.*

For simple games, any coalition $C \subseteq Ag$ such that $v(C) = 1$ can be interpreted as 'winning' whereas any coalition $C \subseteq Ag$ such that $v(C) = 0$ can be interpreted as 'losing'. This constraint is particularly useful in modeling voting situations with the intuition being that if all the agents in a winning coalition vote in the same way then the motion they vote for will be passed whereas, if they all vote the same way in a losing one, it will not be passed [76].

2.3 Solution Concepts

To understand which coalitions will be formed, *solution concepts* from game theory can be used. Generally, it is assumed that the *grand coalition* (that is, the coalition $C = Ag$) will form, meaning the challenge is to allocate the utility $v(Ag)$ among the agents in Ag. It should be observed that this assumption is not restrictive since, even if agents deviate from the grand coalition and form smaller coalitions, solution concepts are still applicable to the sub-games defined by whatever coalitions actually form. For instance, in a five agent characteristic function game where $Ag = \{a_1, \ldots, a_5\}$, if coalition $C = \{a_1, a_2, a_3\}$ forms then the sub-game would involve all coalitions which exclusively consist of the agents $\{a_1, a_2, a_3\}$. In this manner, the intuition behind allocating $v(Ag)$ among all of the agents in Ag can be directly applied to allocating $v(C)$ among all of the agents in any coalition $C \subseteq Ag$.

For every coalition $C \subseteq Ag \setminus \{a_i\}$, a self-interested agent a_i will join the grand coalition if they receive an allocation of $v(Ag)$ which is greater than any allocation they would receive from $v(C \cup \{a_i\})$. To this end, various solution concepts have been formulated to determine if a given allocation of $v(Ag)$ ensures that Ag will be formed by rational agents. In this section, an overview of these concepts is provided.

Formally, a *solution concept* is defined as follows.

Definition 2.9 *For any characteristic function game $\mathcal{N}_t$, a solution concept is a set of* payoff vectors *(where each vector is denoted by $\underline{\mathbf{x}} \in \mathbb{R}^n$) that represent the allocation of the utility from forming the grand coalition.*

Specifically, $\forall i = 1, \ldots, n$, the i^{th} element (x_i) of vector $\underline{\mathbf{x}}$ represents agent a_i's allocation of $v(Ag)$. Clearly, there exist infinitely many payoff vectors for any $v(Ag)$. Consequently, an interesting question arises:

What makes a given payoff vector $\underline{\mathbf{x}}$ a suitable or fair one?

To convey a notion of suitability and fairness, many criteria have proposed for characteristic function games. For instance, to provide a notion of fairness, the following criteria have been proposed:

Efficiency: The efficiency criterion states that the total allocation to each agent should be exactly equal to the utility allocated to the coalition, *i.e.*,

$$\sum_{i=1}^{n} x_i = v(Ag);$$

Symmetry: Any two agents $a_i, a_j \in Ag$ are said to be symmetric if, for every coalition $C \subseteq Ag \setminus \{a_i, a_j\}$, the marginal contribution of both a_i and a_j is identical, *i.e.*, $\forall C \subseteq Ag \setminus \{a_i, a_j\}$,

$$v(C \cup \{a_i\}) = v(C \cup \{a_j\}).$$

The symmetry criterion states that if a_i and a_j are symmetrical agents then they must receive exactly the same payoff, *i.e.*, $x_i = x_j$; and,

Dummy: An agent a_i is a *dummy* if their marginal contribution in every coalition that they belong to does not add any value to it, *i.e.*, $\forall C \subseteq Ag$,

$$v(C \cup \{a_i\}) - v(C) = v(\{a_i\}).$$

The dummy criteria states that all dummy agents $a_i \in Ag$ must receive exactly what they can accomplish on their own, *i.e.*, $x_i = v(\{a_i\})$.

In addition, to create a notion of suitability, the following criteria have been proposed:

Individual Rationality: The individual rationality criterion states that each agent should gain more through cooperating than if they acted alone, *i.e.*, for every x_i in the vector $\underline{\mathbf{x}}$,

$$x_i > v(\{a_i\});$$

Uniqueness: The uniqueness criterion states that there should be exactly one 'fair' allocation; and,

Additivity: The additivity criterion states that for any two characteristic function games $\mathcal{N}_t = \langle Ag, v \rangle$, $\mathcal{N'}_t = \langle Ag, v' \rangle$, the distributed gains in the combined game $\mathcal{N}_t \cup \mathcal{N'}_t$ should correspond to the gains derived from $\mathcal{N}_t$ and the gains derived from $\mathcal{N'}_t$. This means that were every agent $a_i \in Ag$ to receive:

1. $x_i \in \mathbb{R}$ in $\mathcal{N}_t \cup \mathcal{N}'_t$;
2. $x'_i \in \mathbb{R}$ in $\mathcal{N}_t$; and,
3. $x''_i \in \mathbb{R}$ in $\mathcal{N}_t$;

then $x_i = x'_i + x''_i$.

Whereas various solution concepts have been formulated for cooperative game theory, in this chapter, only those that imply stability or welfare maximization are considered.

2.3.1 The Core

The concept of the core was developed independently in both [70] and [26]. Essentially, the core is the set of payoff vectors $\underline{\mathbf{x}} \in \mathbb{R}^n$ that cannot be improved upon by any agent forming coalitions other than Ag.

Definition 2.10 *For any characteristic function game $\mathcal{N}_t$, the core is the set of imputations $\underline{\mathbf{x}}$ such that $\forall x_i$ in $\underline{\mathbf{x}}$:*

- $\sum_{i=1}^{n} x_i = v(Ag)$*; and,*
- $\forall C \subseteq Ag$, $\sum_{i \in C} x_i > v(C)$.

In words, the core is the set of individually rational and efficient payoff vectors that cannot be improved upon, *i.e.*, no agent can improve upon their core allocation by forming a coalition other than Ag. Formally, all payoff vectors which satisfy this criterion are referred to as *imputations*. Intuitively, every imputation $\underline{\mathbf{x}}$ in the core is said to *block* every vector $\underline{\mathbf{x}}'$ that does not belong to the core. Example 2.1 shows how the core may be computed for a particular characteristic function game $\mathcal{N}_t$.

Example 2.1 *Consider a system with two agents a_1, a_2 and suppose that:*

1. $v(\{a_1\}) = v(\{a_2\}) = 1$*; and,*
2. $v(\{a_1, a_2\}) = 3$.

The vector $\underline{\boldsymbol{x}} = \langle 1.5, 1.5 \rangle$ is in the core since, were any of the agents to deviate from the grand coalition and form any other coalition then they would receive less than 1.5. In fact, following this reasoning, all vectors $\underline{\mathbf{x}} = \langle x_1, x_2 \rangle$ such that:

$$x_1 \in [1, 1.5], x_2 = 3 - x_1; \textit{or},$$

$$x_2 \in [1, 1.5], x_1 = 3 - x_2,$$

are in the core.[2]

Clearly, the core conveys *stability*: no agent $a_i \in Ag$ has incentive to leave the grand coalition Ag since, were they to do so, they would receive no more than they are allocated in each imputation in the core. In this context, computing if there exists a set of imputations in the core or computing if a given payoff vector is in the core of any characteristic function game can aid a system designer to understand whether self-interested agents will form the grand coalition. However, it can be shown through example that:

1. The core may not be unique, *i.e.*, there may exist more than one allocation which meets the requirements of Definition 2.10 (as in Example 2.1); and,
2. The core may be empty, *i.e.*, there do not exist any allocations which meet the requirements of Definition 2.10 (see Example 2.2).

Example 2.2 *Consider a system with two agents a_1, a_2 and suppose that $v(\{a_1\}) = v(\{a_2\}) = 1$ and $v(\{a_1, a_2\}) = 1.1$. All efficient allocations of the value 1.1 between the two agents will result in at least one of them receiving less than '1' meaning, all $\underline{\boldsymbol{x}} \in \mathbb{R}^n$ such that $\sum_{i=1}^{n} x_i = v(Ag)$ will not be individually rational. Thus, the core is empty.*

Whereas emptiness of the core is not desirable, there does exist a natural class of characteristic function games in which the core is guaranteed to be non-empty.

[2]Note that $x_1 \in [1, 1.5]$ ($x_2 \in [1, 1.5]$) means that x_1 (x_2) can take all values between, and including, 1 and 1.5

Lemma 2.1 *(Proven in [9]) If any characteristic function game $\mathcal{N}_t$ is* convex *then it has a non-empty core.*

Note that the characteristic function game $\mathcal{N}_t$ in Example 2.1 is convex as,

$$v(\{a_1\} \cup \{a_2\}) + v(\emptyset) = 3 > v(\{a_1\}) + v(\{a_2\}) = 2,$$

whereas the characteristic function game $\mathcal{N}_t$ in Example 2.2 is not convex since,

$$v(\{a_1\} \cup \{a_2\}) + v(\emptyset) = 1.1 < v(\{a_1\}) + v(\{a_2\}) = 2.$$

For those class of characteristic function games that are not convex, one attempt to circumvent the problem of core emptiness has involved reformulating the core concept itself. For example, the *strong epsilon core* (ϵ-core) was formulated in [72]. For a chosen value $\epsilon \in \mathbb{R}$, if an agent chooses to leave the grand coalition then it must incur a cost of value ϵ which, in turn, will affect the amount it is allocated in the coalition it chooses to join. In this manner, although the core presented in Definition 2.10 may be empty, for large ϵ, the ϵ-core will not be.

Example 2.3 *Recall Example 2.2 where the core is empty. Now suppose $\epsilon = \frac{2}{3}$, meaning should either a_1 or a_2 deviate from the grand coalition and act alone then each will only receive $1 - \frac{2}{3} = \frac{1}{3}$ rather than 1. In such a case, the $\frac{2}{3}$-core is non-empty as, for example, the payoff vector $\underline{\boldsymbol{x}} = \langle 0.6, 0.5 \rangle$ belongs to it.*

2.3.2 The Shapley Value

Since there can exist infinitely many possible pay-off vectors, computing an appropriate pay-off is not trivial. In this section, the Shapley value concept, which allocates $v(Ag)$ among the agents in Ag, is analyzed.

The Shapley value determines the allocation of $v(Ag)$ to a given agent $a_i \in Ag$ by assessing the marginal contribution of every agent in every coalition they could join [71]. For every coalition $C \subseteq Ag : a_i \in C$, this is achieved by computing,

$$v(C) - v(C \setminus \{a_i\}).$$

For each coalition, the marginal contribution value is then averaged over all the ways in which the coalition could be formed by the agents who belong to it. There are,

$$\frac{(n - |C|)!(|C| - 1)!}{n!}$$

ways in which coalition C can be formed by the agents in Ag. The sum of these averaged out marginal values of agent a_i in every coalition $C \subseteq Ag \setminus \{a_i\}$ is then computed and this is the Shapley value for a_i. It is formally defined as follows.

Definition 2.11 *For any super-additive characteristic function game $\mathcal{N}_t$, the Shapley value allocated to every agent $a_i \in Ag$ in coalition Ag (denoted θ_i) is defined as follows:*

$$\theta_i = \sum_{\forall C \subseteq Ag : a_i \in C} \frac{(n - |C|)!(|C| - 1)!}{n!} [v(C) - v(C \setminus \{a_i\})].$$

Example 2.4 shows how this value is computed.

Example 2.4 *Consider a three agent characteristic function game $\mathcal{N}_t$ where $Ag = \{a_1, a_2, a_3\}$ and v is as follows:*

- $v(\{a_1, a_2, a_3\}) = 8$*;*
- $v(\{a_1, a_2\}) = v(\{a_1, a_3\}) = v(\{a_2, a_3\}) = 5$*; and,*

- $v(\{a_1\}) = v(\{a_2\}) = v(\{a_3\}) = 2$.

Consider agent a_1 first. In the coalition $\{a_1\}$, since $v(\emptyset) = 0$, the value is

$$\frac{(3-1)!(1-1)!}{3!} \times (2-0) = \frac{2!0!}{6} \times 2 = \frac{2}{3}.$$

Now, in the coalitions $\{a_1, a_2\}$ and $\{a_1, a_3\}$ the value is

$$\frac{(3-2)!(2-1)!}{3!} \times (5-2) = \frac{1}{6} \times 3 = \frac{1}{2},$$

and, finally, in the grand coalition, the value is

$$\frac{(0)!(2)!}{3!} \times (8-5) = \frac{1}{3} \times 3 = 1.^3$$

Thus,

$$\theta_1 = \frac{2}{3} + (2 \times \frac{1}{2}) + 1 = \frac{8}{3}.$$

Since all coalitions of the same size have equal value, it follows that $\theta_2 = \theta_3 = \frac{8}{3}$ as well.

Let $\underline{\mathbf{x}} = \langle \frac{8}{3}, \frac{8}{3}, \frac{8}{3} \rangle$ denote the Shapley value allocation of $v(\{a_1, a_2, a_3\})$. Since $\frac{8}{3} + \frac{8}{3} + \frac{8}{3} = 8 = v(Ag)$ then the allocation is efficient. Also, observe that $\frac{8}{3} + \frac{8}{3} > 5$ and $\frac{8}{3} > 2$ and so is individually rational as well. Following Definition 2.10, $\underline{\boldsymbol{x}}$ is an imputation and so belongs to the core.

In contrast to the core, the Shapley value is unique and always exists. However, what makes this concept particularly important is the following result.

Theorem 2.1 *(From [72]) There exists a unique value satisfying the efficiency, symmetry, dummy, and additivity axioms: it is the Shapley value given in Definition 2.11.*

Theorem 2.1 states that any payoff vector $\underline{\mathbf{x}}$ which satisfies the efficiency, symmetry, dummy and additivity criteria must be a Shapley value as this, and only this, satisfies all of these criteria. Furthermore, if the characteristic function game is also convex then the Shapley value is guaranteed to be in the core of the game [12]. Nevertheless, there are two major issues with both the core and Shapley value, namely:

1. Both are dependent upon the representation satisfying certain restrictive requirements, *e.g.*, if the game is not convex then the core may be non-empty, whereas the Shapley value only offers meaningful results if the characteristic function is super-additive; and,

2. Both Definition 2.10 and Definition 2.11 were formulated for characteristic function games where there are no externalities from coalition formation, meaning both generalizing and extending these solutions to partition function games is not trivial [18, 16, 40, 47, 8, 28, 33].

Against this reasoning, in the next section, an *optimal coalition structure* concept is considered that can be easily formulated for both partition function game and characteristic function game representations.

2.3.3 Optimal Coalition Structures

Rather than trying to allocate the utility obtained from forming the grand coalition among the agents, the optimal coalition structure problem is concerned with finding a partition that maximizes the welfare of the system. In the literature thus far, an optimal coalition structure has been formally defined as follows.

Definition 2.12 *(from [59]) Let* Π *denote the space of all coalition structures that could be formed by a set of agents* Ag*. An* optimal coalition structure *is a coalition structure* $\pi^* \in \Pi$ *such that the combined value of all the coalitions which make up this structure is no less than the combined values in every other structure. Given any characteristic function game* $\mathcal{N}_t$*, an optimal structure is one such that:*

$$\pi^* = \text{arg}max_{\pi \in \Pi} \sum_{C \in \pi} v(C).$$

Since a utilitarian metric is used to assess the performance of a coalition structure, the optimal structure is the one that maximizes the welfare of the system as a whole. This means that although π^* is the best structure for the system as a whole, it may not be the best structure for certain individual agents. Therefore, following previous discussion, although fully cooperative agents are guaranteed to partition themselves into this structure, there is no guarantee that self-interested agents will. However, as is reasoned in [55], once π^* is known, it may be possible to reward the entities to form π^* or penalize them if they do not. Significantly large rewards and penalties can ensure that all agents will be worse off if they partition themselves into a structure $\pi \neq \pi^*$. In the spirit of the ϵ-core concept, rewards and penalties ensure *stability* of π^*, meaning both fully cooperative and self-interested agents will have incentive to form a known optimal solution. With respect to multi-agent system applications, it is worth noting that these reward and penalty constraints do not contradict any autonomy assumptions as physically distributed agents are able to make their own decisions with respect to forming coalitions (and therefore structures) but must adhere to the penalty or reward constraints.[4]

Example 2.5 *Recall Example 2.4. For* $Ag = \{a_1, a_2, a_3\}$*, there are exactly five possible coalition structures in* Π*:*

$$\begin{aligned}
\pi_\alpha &= \{\{a_1\}, \{a_2\}, \{a_3\}\} \quad v(\{a_1\}) + v(\{a_2\}) + v(\{a_3\}) = 6; \\
\pi_\beta &= \{\{a_1, a_2\}, \{a_3\}\} \quad v(\{a_1, a_2\}) + v(\{a_3\}) = 7; \\
\pi_\gamma &= \{\{a_1, a_3\}, \{a_2\}\} \quad v(\{a_1, a_3\}) + v(\{a_2\}) = 7; \\
\pi_\delta &= \{\{a_2, a_3\}, \{a_1\}\} \quad v(\{a_2, a_3\}) + v(\{a_1\}) = 7; and, \\
\pi_\epsilon &= \{a_1, a_2, a_3\} \quad v(\{a_1, a_2, a_3\}) = 8.
\end{aligned}$$

Since π_ϵ *has maximal value,* $\pi_\epsilon = \pi^*$*.*

Example 2.5 shows that, for a general characteristic function v, the value of every possible coalition structure must be computed in order to determine the optimal one. However, Theorem 2.2 shows that, for certain natural classes of characteristic function games, the optimal structure can be immediately identified.

Theorem 2.2 *For any characteristic function game* $\mathcal{N}_t = \langle Ag, v \rangle$*, if* v *is super-additive then* $\pi^* = Ag$ *is an optimal structure. On the other hand, if* v *is sub-additive then* $\pi^* = \{\{a_1\}, \ldots, \{a_n\}\}$ *is an optimal structure.*

[4] Although the reasoning presented in this section suggest that it is possible to use π^* to identify coalitions that will be formed by self interested agents, for the rest of this work, it will be exclusively employed to identify coalitions that will be formed by fully cooperative agents.

Proof: Assume super-additivity holds first. This means that $\forall S, T \subseteq Ag$, such that $S \cap T = \emptyset$ and $S \cup T = Ag$,

$$v(Ag) \geq v(S) + v(T).$$

In turn, this also means that for all exhaustive and disjoint coalitions $S', T' \subseteq S$, as well as all disjoint and exhaustive coalitions $S'', T'' \subseteq T$,

$$v(S) + v(T) \geq v(S') + v(T') + v(S'') + v(T'').$$

Clearly, iteratively repeating this mathematics for all coalitions of Ag shows that no structure can have value greater than $v(Ag)$, meaning Ag is an optimal structure.

Now, assume sub-additivity instead. This means that $\forall C \subseteq Ag$ and $\forall S, T \subseteq C$, such that $S \cap T = \emptyset$ and $S \cup T = C$,

$$v(C) \leq v(S) + v(T).$$

In turn, this also means that for all exhaustive and disjoint coalitions $S', T' \subseteq S$, as well as all exhaustive and disjoint coalitions $S'', T'' \subseteq T$,

$$v(S) + v(T) \leq v(S') + v(T') + v(S'') + v(T'').$$

Clearly, iteratively repeating this mathematics for all coalitions of Ag shows that no structure can have value greater than $\{\{a_1\}, \ldots, \{a_n\}\}$, meaning $\{\{a_1\}, \ldots, \{a_n\}\}$ is an optimal structure. ∎

It is worth noting that if the function is either strictly super-additive or strictly sub-additive then $\pi^* = Ag$ and $\pi^* = \{\{a_1\}, \ldots, \{a_n\}\}$ are the only optimal structures, respectively. Clearly, if it is known *a priori* that the characteristic function is either super- or sub-additive then π^* can be immediately identified without computing the values of any structures.

As well as the fairness criteria presented in this chapter, the *computational complexity* of these concepts also provides a useful measure of their suitability. For instance, if decision problems concerning these solutions are *intractable* then, under the assumption that NP $\neq$ P, this means that, in practice, computationally limited agents may not be able to use them. To this end, in the next section, the computational complexity of problems relating to stability of coalitions is considered.

2.4 Representations of Cooperative Games

To understand coalition structure generation in cooperative game representations of systems containing self-interested agents, the following core-related decision problems can be answered:

Core Non-emptiness Is the core of the game empty?

Core Membership Is a given payoff vector in the core of the game?

Observe that, to fully define any characteristic function game with non-transferable utility, for every coalition $C \subseteq Ag$, the utility obtained from forming C must be given. When the values of all coalitions are stated, decision problems concerning core non-emptiness and core membership have time complexity that is polynomial in the size of the input. However, since 2^n coalitions can be formed in any system that contains n agents, the input to these decision problems can contain a number of values that are exponential in the number of agents. Therefore, although these decision problems have polynomial time complexity, this complexity is still exponential in the number of agents. Thus, to achieve meaningful and significant results, one line of work has focused on developing compact representations of cooperative games that enable decision

problems concerning membership and non-emptiness of the core to be computed with low time complexity whilst still maintaining a balance between all of the expressivity, conciseness, efficiency and simplicity criteria presented in Section 1.1.2.

Various representations of characteristic function games have been developed for modeling coalition formation in a number of multi-agent systems. This section provides an overview of a number of these representations. The quality of these representations is assessed against all of the expressivity, conciseness, efficiency and simplicity criteria.

2.4.1 Weighted Graph Games

Within the field of computer science, the first study of succinct representations of characteristic function games was presented in [17]. Here, the authors study a game in which the agents are represented as vertices in a graph with weights on the edges. In these games, the value of a coalition is determined by the total weight of the edges contained in the subgraph induced by that coalition.

Definition 2.13 *A* weighted graph game *is defined by a undirected graph $\mathcal{G}_W = (V, E)$, where V is a set of vertices and E is a set of weighted edges between the vertices, i.e., for every $(i, j) \in E$, a weight $w(i, j) \in \mathbb{R}$ is attributed to the edge. In this class of games:*

- $V = Ag$*; and,*
- $\forall C \subseteq Ag$, $v(C) = \sum_{i,j \in C} w(i, j)$.

In terms of conciseness, this graphical representation is guaranteed to be concise in the number of agents since it can be defined by no more than $\frac{n(n-1)}{2}$ weights. Furthermore,the value of any coalition $C \subseteq Ag$ can be easily computed through summing the weights of the edges containing vertices that represent those agents in C. However, in terms of expressivity, weighted graph games cannot capture all classes of characteristic function games. For example, in some systems, the pairwise contribution of every pair of agents may vary in different coalitions and, although the framework $\mathcal{N}_t$ may be able to represent these systems, $\mathcal{G}_W$ cannot.

With respect to efficiency, computing decision problems concerning both the Shapley value and the core have been analyzed in this representation.

Lemma 2.2 *The Shapley value of an agent a_i in a weighted graph game $\mathcal{G}_W$ is given by:*

$$\theta_i = \frac{1}{2} \sum_{j \neq i} w(i, j).$$

Example 2.6 *Consider a three agent system, where*

- $Ag = V = \{a_1, a_2, a_3\}$*; and,*
- $W = \{\ w(1,1)$*,* $w(2,2)$*,* $w(3,3)$*,* $w(1,2)$*,*$w(1,3)$*,*$w(2,3)\ \}$*, in which*
  - $w(1,2) = 6$*;*
  - $w(1,3) = 1$*;*
  - $w(2,3) = 5$*; and,*
  - *for* $i = 1, \ldots, 3$*,* $w(i,i) = 0$.

In the grand coalition Ag, the Shapley value allocation to each agent is given as follows:

- $\theta_1 = \frac{1}{2} \times (w(1,2) + w(1,3)) = 3.5$*;*

- $\theta_2 = \frac{1}{2} \times (w(1,2) + w(2,3)) = 5.5$*; and,*
- $\theta_2 = \frac{1}{2} \times (w(1,3) + w(2,3)) = 3.0$.

Following Lemma 2.2, the Shapley value can be computed in $O(n^2)$ operations which is polynomial in both the number of weights and agents. However, with respect to the core solution, for a general weighted graph representation $\mathcal{G}_W$, the following problems are NP-complete:

- Given $\underline{\mathbf{x}}$, is $\underline{\mathbf{x}}$ not in the core of $\mathcal{G}_W$?
- Is the Shapley value of $\mathcal{G}_W$ not in the core of $\mathcal{G}_W$? and,
- Is the core of $\mathcal{G}_W$ empty?

Thus, given any $\mathcal{G}_W$, no algorithm can be guaranteed to compute core-related problems with time complexity that is polynomial in the input size. However, there do exist certain instances of these problems that can be computed with polynomial time complexity, *e.g.*, when all the weights are non-negative.

In summary, weighted graph games are succinct, easy to use and enable efficient computation of the Shapley value solution. However, on the negative side, they are not fully expressive representations of characteristic function games and do not guarantee efficient computation of problems concerning the core.

2.4.2 Weighted Coalitional Games

Weighted coalitional games are a representation of simple characteristic function games and are formally defined as follows.

Definition 2.14 *A weighted game is a* $(n+2)$*-tuple* $WCG = \langle Ag, w_1, \ldots, w_n, q \rangle$ *where:*

- $Ag = \{a_1, \ldots, a_n\}$ *is a set of agents;*
- $w_i \in \mathbb{R}^+$ *is a real number value called the 'weight' of agent* $a_i \in Ag$*; and,*
- $q \in \mathbb{R}^+$ *is the quota of the game.*

A coalition is 'winning' if the sum of the weights of the agents in the coalition is greater than or equal to the value of q and is 'losing' otherwise. Clearly, these representations are guaranteed to be succinct because, in order to fully define them, no more than n weights and one quota value q are required. Also, determining if a coalition $C \subseteq Ag$ is winning can be easily computed - simply sum up the weights of the agents in the coalition and verify if they are greater than or equal to q.

Regarding expressivity, since the weighted coalitional games defined in Definition 2.14 are inherently monotonic, they cannot represent every simple characteristic function game. Also, with respect to the efficiency criterion, under the assumption that $\sum_{a_i \in Ag} w_i = 1$, an *imputation* is defined as follows.

Definition 2.15 *In any* WCG*, an* imputation *is a vector of non-negative rational numbers* $\underline{\mathbf{x}} = \langle x_1, \ldots, x_n \rangle$ *such that,*

$$\sum_{j=1}^{n} x_j = 1,$$

i.e.*,* $v(Ag) = 1$ *is allocated efficiently among all of the agents.*

Given an imputation $\underline{\mathbf{x}}$, the excess $e(\underline{\mathbf{x}}, C)$ of a coalition $C \subseteq Ag$ under $\underline{\boldsymbol{x}}$ is defined as,

$$e(\underline{\mathbf{x}}, C) = \sum_{j \in C} x_j - \sum_{j \in C} w_j.$$

Given these definitions, the core and Shapley value are formally defined as follows.

Definition 2.16 *(From [20]) The* core *of a weighted voting game is the set of imputations* $\underline{\mathbf{x}}$ *such that, for every* $C \subset Ag$*,*

$$e(\underline{\mathbf{x}}, C) \geq 0.$$

The Shapley value is defined as in Definition 2.11 where $x_i = \theta_i, \forall a_i \in Ag$*.*

In words, $\underline{\mathbf{x}}$ is in the core if each agent cannot gain more than they are allocated in $\underline{\mathbf{x}}$ through forming a coalition $C \subset Ag$.

Example 2.7 *Consider a weighted coalitional game* WCG *where,*

- $Ag = \{a_1, a_2\}$*; and,*
  - $w_1 = 5$*;*
  - $w_2 = 4$*; and,*
  - $q = 6$*.*

Clearly, Ag *is the only winning coalition. Now, consider imputation* $\underline{\mathbf{x}} = \langle 0.5, 0.5 \rangle$*. Observe that,*

$$e(\langle 0.5, 0.5 \rangle, \{a_1, a_2\}) = 1 - 1 = 0,$$

$$e(\langle 0.5, 0.5 \rangle, \{a_1\}) = 1 - 0 = 1,$$

and,

$$e(\langle 0.5, 0.5 \rangle, \{a_2\}) = 1 - 0 = 1.$$

Therefore, $\underline{\mathbf{x}} = \langle 0.5, 0.5 \rangle$ *is in the core of* WCG*.*

Following Definition 2.11, this is also the Shapley value payoff vector to Ag *as, for* $i = 1, 2$*,*

$$\theta_i = \frac{(2-1)!(1-1)!}{2!} = 0.5 = x_i \in \underline{\mathbf{x}}.$$

The authors of [20] investigated a number of natural problems concerning both the core and Shapley value in weighted coalitional games[5], including:

1. Given a WCG, is the core non-empty?
2. Given a WCG and a value ϵ, is the ϵ-core non-empty? and,
3. Given a WCG, what is the Shapley value allocation to an individual agent a_i?

With regards to the first problem, the authors proved that computing if the core of any WCG is non-empty can be done in time polynomial of the input size. The following Lemma is fundamental to this result.

Lemma 2.3 *(From [20]) The core of a weighted coalitional game is non-empty if and only if there is an agent who is present in all winning coalitions,* i.e.*, at least one agent is a* veto agent.

[5]Specifically, [20] refer to weighted threshold games, however, essentially a weighted threshold game is a weighted coalitional game.

As is noted in [20]. it is easy to check if there is an agent that is present in all winning coalitions. Specifically, for every $a_i \in Ag$, compute if,

$$\sum_{a_j \in Ag \setminus \{a_i\}} w_j < q.$$

Because weighted coalitional games are inherently monotonic, if this is true then a_i belongs to every winning coalition. Thus, for this succinct representation, determining non-emptiness of the core can be computed in a number of steps that are polynomial in the input size.

In contrast to this positive result, the second problem was proven to be NP-hard meaning it cannot be guaranteed that an algorithm will compute this problem with time complexity that is polynomial in the size of WCG. However, this problem can be solved in polynomial time if certain restrictions are imposed, *e.g.*, if the size of the weights are bounded so that they are, at most, polynomially large in n.

With respect to the third problem, following results in [54, 41], it is #P-hard to compute the Shapley value of a given agent and NP-complete to determine whether this value is zero in any weighted voting game. This means that computing the Shapley value solution to a weighted coalitional game is as hard as solving an enumeration problem which is concerned with an NP-complete decision problem. Thus, it cannot be guaranteed that an algorithm will compute this problem in a number of steps that are polynomial in the size of WCG.

In conclusion, weighted coalitional games are succinct representations that are simple to use and can enable efficient computation of certain core-related problems. However, they cannot fully represent every simple characteristic function domain and do not guarantee efficient computation of the Shapley value.

2.4.3 Synergy games

A synergy game representation of characteristic function games was formally developed in [13]. Specifically, synergy games are defined as follows.

Definition 2.17 *A synergy game is a representation $\mathcal{S} = \langle Ag, S \rangle$ where*

- *Ag is the set of agents; and*
- *S is a set of values $(C, v(C))$, such that;*
  - *$C \subseteq Ag$; and,*
  - *$v(C)$ is computed as follows:*

$$v(C) = (max_{\{C_1,\ldots,C_k\} \in \Pi(C)} \sum_{i=1}^{k} v(C_i)).$$

Here, $\Pi(C)$ is the set of all partitions of a coalition $C \subseteq Ag$.

Intuitively, the value of a coalition $v(C)$ is only stated if it is strictly super-additive, *i.e.*, if $v(C)$ is greater than the combined value of all partitions of C. This representation is fully expressive over characteristic function domains but, in some games, this representation may require space that is exponential in the number of agents. Therefore conciseness is guaranteed if and only if the synergies among coalitions are sparse (*i.e.*, the number of coalitions whose value is greater than the combined values of its partitions is low). In such cases, only a fraction of all coalition values $(C, v(C))$ will be specified as the values of the coalitions which are not stated can be computed from the specified values. Given concise representations of synergy games, the authors proved the following lemma.

Lemma 2.4 *Given $\mathcal{S}$, it is NP-hard to compute the value of the grand coalition.*

This lemma shows that it may be difficult for either a system designer or a user of the system to compute coalition values. Thus, it could be argued that this representation is not easy to use.

With respect to solution concepts, the authors of [13] focus upon the core solution. If the convexity criterion is satisfied, the authors prove that computing core membership can be done in $O(nl^2)$ time, where n is the number of agents and l is the number of synergies, *i.e.*, the number of $(C, v(C)) \in S$. Thus, in compact instances, core related problems can be computed in a number of steps that are polynomial in the number of agents. More generally, for any $\mathcal{S}$, if the value of the grand coalition is given as input, non-emptiness of the core can be determined in polynomial time. Clearly, this is a positive result against efficiency.

Consequently, it can be concluded that synergy games may be compact and can represent any characteristic function game. Furthermore, when they are compact, they enable efficient computation of the core solution. However, they may be difficult to use by users of the system since it cannot be guaranteed that the values of certain coalitions can be efficiently computed.

2.4.4 Marginal Contribution Nets Games

Marginal contribution nets (MCN) representations were first presented in [29]. This representation consists of rules that have the syntactic form:

$$\textit{pattern} \rightarrow \textit{value},$$

where *value* denotes a real number $\mathbb{R}$ attributed to coalitions depending upon if they satisfy the requirements of *pattern.*

In the 'basic' representation, these patterns consist of conjunctions of literals that represent agents. A coalition C meets the requirement of the given pattern if all of the agents who are represented by the literals belong to C. The value C is then computed by summing the values of all rules that C meets the requirement of.

Example 2.8 *Consider a two agent system $Ag = \{a_1, a_2\}$ where the rules are as follows:*

$$\begin{aligned} a_1 &\rightarrow 5 \\ a_2 &\rightarrow 4 \\ a_1 \wedge a_2 &\rightarrow 3 \end{aligned}$$

Observe that there is only one rule that contains only $\{a_1\}$ and no others. This is $a_1 \rightarrow 5$. Thus, the value of $\{a_1\}$ is 5. Similarly, there is only one rule which exclusively contains only $\{a_2\}$ ($a_2 \rightarrow 4$) and so the value of $\{a_2\}$ is 4. On the other hand, all of the rules are supersets of $\{a_1, a_2\}$, meaning the value of $\{a_1, a_2\}$ is $5 + 4 + 3 = 12$.

By introducing negated literals, MCNs can represent any characteristic function game and are therefore fully expressive representations. Nevertheless, to represent a coalitional game in characteristic form, in the worst case, all 2^n coalition values would have to be specified. Thus, conciseness is not guaranteed. However, conciseness can be guaranteed in many naturally arising games, including, the *recommendation* game presented in [29].

Generally, conciseness may be ensured by introducing negated literals into *pattern*. Here, the interpretation is that the agents represented by non-negated literals contributed the value to all coalitions of which they are subsets in the absence of those negated agents. This is useful for expressing concepts such as substitutability

or default values and, in representations where these phenomena naturally occur, the introduction of negated agent literals can result in an exponentially more conciseness representation.

Given a MCN representation. if there are no negated literals in the pattern of the rules then it is very easy to compute the Shapley value, simply:

1. Identify all rules that contain agent a_i;
2. For each rule divide the value of the rule by number of agents in the rule; and,
3. Sum up all obtained values. The sum will the Shapley value for agent a_i.

Clearly, the Shapley value can be computed in a number of steps that are polynomial in the size of the input which means that, in cases where the number of rules is compact, this can be done in a number of steps that are polynomial in the number of agents.

Example 2.9 *Recall Example 2.8. There are two rules that contain agent* a_1. *The first one has only one agent and so* $\frac{5}{1} = 5$. *The second one is* $a_1 \wedge a_2 \rightarrow 3$ *and so* $\frac{3}{2} = 1.5$. *Therefore, the Shapley value of* a_1 *is* 6.5. *Repeating this for* a_2 *gives* $4 + 1.5 = 5.5$ *as its Shapley value.*

With regards to both membership and non-emptiness of the core, the authors in [29] prove that the problem of computing if a pay-off vector is in the core is coNP-complete whereas the problem of computing if the core is non-empty is coNP hard. Recall that, from the previous section, a payoff vector $\underline{\mathbf{x}}$ is in the core if, $\forall C \subseteq Ag$ and, $\forall a_i \in C$,

$$\sum_{x_i \in \underline{\mathbf{X}}} x_i \geq v(C) \Rightarrow \sum_{x_i \in \underline{\mathbf{X}}} x_i - v(C) \geq 0.$$

If $\sum_{x_i \in \underline{\mathbf{X}}} x_i - v(C)$ represents the *excess* of a coalition $C \subseteq Ag$ then a naive approach to compute if $\underline{\mathbf{x}}$ belongs to the core involves checking that the excesses of all coalitions are non-negative. To circumvent this difficulty, an algorithm is proposed which can compute core membership through computing the excesses of only a number of coalitions.

Specifically, this algorithm represents the marginal contribution nets game as a tree. The nodes in this tree represent coalitions which are connected by edges. The tree is represented in decomposition form (for more details see [29]). This representation has the advantage that, for a given payoff vector $\underline{\mathbf{x}}$, it is possible for an algorithm to infer that certain coalitions have non-negative excesses due to the excesses computed elsewhere in the graph. In this way, the problem of computing if $\underline{\mathbf{x}}$ belongs to the core can be computed without having to compute the excess of every coalition $C \subseteq Ag$.

Despite this insight, core membership algorithms may still run in time that is exponential in the width of the tree which, in turn, may have size that is exponential in the number of agents. Additionally, core non-emptiness can be computed by solving the following linear programme *via* the ellipsoid method using solutions to the core-membership problem as an oracle:[6]

$$\text{minimize}_{x \in \mathbb{R}^n} \sum_{i=1}^{n} x_i$$
$$\text{subject to} \sum_{a_i \in C} x_i \geq v(C), \forall C \subseteq Ag.$$

In this way, core non-emptiness can be answered in time complexity that is polynomial in the running time of core-membership algorithms which, in turn, can run in time that is polynomial in the width of the tree. Clearly, this result is significant if and only if the tree has bounded width.

[6] An oracle is formally defined in Appendix A.

'Read Once' Marginal Contribution Nets

The positive results presented in this section are dependent upon the conciseness of the MCN representation which cannot always be guaranteed. To this end, building upon the work presented in [29], Elkind *et al.* showed that the general Marginal contribution net representation leads to intractability with respect to computing the Shapley value [21]. In more detail, they proved that, in the general Marginal contribution nets representation, the problem of computing if the Shapley value of a given agent was exactly 0 is coNP-complete. In turn, this result implies that approximating the Shapley value is a NP-hard problem which undermines the positive results presented in [29]. Against this result, the authors proposes a 'Read Once' MCN representation in which *pattern* was expressed as a *'read-once' boolean formula*.

Definition 2.18 *(from [21]) A* read-once Boolean formula *is a binary rooted tree in which each internal node is labeled with a Boolean connective, such as '*$\wedge$*','*$\vee$*',* e.t.c. *and the leaves are labeled with literals (*i.e.*, variables or their negations) subject to the constraint that each variable appears in at most one leaf.*

For a number of classes of games, the read-once representation was proven to exponentially more concise than the one proposed in [29]. Consequently, computing the Shapley value in a 'read once' MC-nets representation can be exponentially less complex than in the general MCN representation. Thus, without any loss in expressivity, the read-once representation provides exponentially more compact instances than the general MCN representation.

Of course, since the tree decomposition size is not related to the rules, this representation does not guarantee improvements with respect to solving problems concerning non-emptiness and membership of the core.

2.4.5 Hedonic Coalitional Games

Hedonic coalitional games are a class of cooperative games that can represent systems where agents have preferences over the coalitions they can join. Hedonic coalitional games were first formalized into the following framework in both [3] and [7].

Definition 2.19 *A hedonic game is a tuple* $\mathcal{H} = \langle Ag, \{\succ_i\}_{\forall i \in Ag} \rangle$ *where:*

- $Ag = \{a_1, \ldots, a_n\}$ *are the set of agents; and,*
- *Every* $\succ_i$ *is a rational preference relation over coalitions* $C \subseteq Ag$ *such that* $a_i \in C$.

For notation, $\succ_i$ will read "a_i strictly prefers", whereas $\succeq_i$ will read "a_i strictly prefers or is indifferent between". Note that to fully define $\mathcal{H}$, the preference orderings of all the agents must be specified. Since every agent can have preferences over every coalition they can join, this representation may be of a size that is exponential in the number of agents. However, there can exist compact instances of this case when the preference orderings may be succinct, *e.g.*:

Pref1 When the agents preferences are *individually rational coalitional lists*, *i.e.*, for every $a_i \in Ag$, instead of listing a complete list of coalitions in $\succ_i$, only those coalitions which are preferred to $\{a_i\}$ are considered;

Pref2 When the agents have *anonymous* preferences, *i.e.*, they have preference over the size of coalitions rather than the individuals who belong to them;

Pref3 If the game is *additively separable*, *i.e.*, if there exists a $n \times n$ matrix of real values v such that $C_1 \succ_i C_2 \iff \sum_{a_j \in C_1} v[i,j] > \sum_{a_j \in C_2} v[i,j]$, where $v[i,j]$ is the jth value in row i and reflects the value of agent a_j to agent a_i; and,

Pref4 When the agents have $\mathcal{B}-$ and $\mathcal{W}-$Preferences. In these preferences, for every agent $a_i \in Ag$, $\succ_i$ represents the preferences of a_i over the other agents in the system. For any two coalitions C_1 and C_2, in $\mathcal{B}-$Preferences they will prefer C_1 to C_2 if and only if they prefer the best member of C_1 (according to $\succ_i$) to the best member of C_2 (according to $\succ_i$). In contrast, $\mathcal{W}-$Preferences, they will prefer C_1 to C_2 if and only if they prefer the worst member of C_1 (according to $\succ_i$) to the worst member of C_2 (according to $\succ_i$).

All of these examples are taken from [22]. Since the utility obtained from forming coalitions is not measured using a real number value, the notion of a core, as defined in Definition 2.10, is not applicable to these domains. Instead, a notion of stability is conveyed in the following definition.

Definition 2.20 *For any coalition structure π and for $j = 1, \ldots, n$, let $C_j(\pi)$ denote the coalition in π to which agent a_j belongs. Any coalition $C \subseteq Ag$* blocks *π if and only if $\forall a_j \in C$,*

$$C \succ_j C_j(\pi).$$

In words, a coalition C blocks a coalition structure π if every agent in C prefers C to the coalition they belong to in π. This conveys a notion of instability in π since these agents would happily deviate from the coalitions they belong to in π and join another structure that contains C. Against this intuition, the concept of a core solution in $\mathcal{H}$ is formally defined as follows.

Definition 2.21 *(From [2] and [22]) Coalition structure π is* core stable *if there does not exist a coalition C that blocks π.*

As well as core stability, other notions of stability have been proposed for $\mathcal{H}$.

Definition 2.22 *(From [2] and [22]) Coalition structure π is* individually rational *if $\forall a_i \in Ag$,*

$$C_i(\pi) \succeq_i \{a_i\}.$$

If a structure is individually rational then every agent does at least as well in that structure than they would do alone. Trivially, core stability implies individual rationality.

Definition 2.23 *(From [2] and [22]) Coalition structure π is* Nash stable *if, $\forall a_i \in Ag$ and $\forall C \in \pi \setminus \{C_i(\pi)\}$,*

$$C_i(\pi) \succ_i C \cup \{a_i\}.$$

In words, a structure π is Nash stability if no agent would want to join any other coalition in π, assuming the other coalitions in π did not change. As with core stability, Nash stability also implies individual rationality.

Definition 2.24 *(From [2] and [22]) Coalition structure π is* individually stable *if there do not exist $a_i \in Ag$ and $C \in \pi$ such that,*

$$C \cup \{a_i\} \succ_i C_i(\pi), \forall a_i \in Ag$$

and

$$C \cup \{a_i\} \succ_j C, \forall a_j \in C.$$

Intuitively, individual stability implies that no agent a_i could join another coalition in the structure that they preferred to $C_i(\pi)$ without making some member of the coalition they joined unhappy.

Definition 2.25 *(From [2] and [22]) Coalition structure π is* contractually individually stable *if there do not exist $a_i \in Ag$ and $C \in \pi$ such that,*

$$C \cup \{a_i\} \succ_i C_i(\pi), \forall a_i \in Ag$$

and,

$$C \cup \{a_i\} \succ_j C, \forall a_j \in C,$$

and,

$$C_i(\pi) \setminus \{a_i\} \succ_k C_i(\pi), \forall a_k \in C_i(\pi) \setminus \{a_i\}.$$

A structure is contractually individually stable if no agent can move to a coalition it prefers more in that structure without rendering the agents in both the coalitions it joins and leaves worse off. Clearly, Nash stability implies individual stability which, in turn, implies contractual individual stability.

Example 2.10 *Consider a four agent system where $Ag = \{a_1, a_2, a_3, a_2\}$ where the indivudally rational preference orderings of the agents are as follows:*

- *For agent a_1, $\{a_1, a_2\} \succ_1 \{a_1, a_2, a_3, a_4\} \succ_1 \{a_1\} \ldots$;*
- *For agent a_2, $\{a_1, a_2\} \succ_2 \{a_2\} \ldots$;*
- *For agent a_3, $\{a_1, a_3\} \succ_3 \{a_3, a_4\} \succ_3 \{a_2, a_3\} \succ_3 \{a_3\} \ldots$; and,*
- *For agent a_4, $\{a_2, a_4\} \succ_4 \{a_1, a_4\} \succ_4 \{a_3, a_4\} \succ_4 \{a_4\} \ldots$;*

In this example, the structure $\pi = \{\{a_1, a_2\}, \{a_3, a_4\}\}$ is core stable since coalitions $\{a_2, a_4\}$, $\{a_1, a_4\}$, $\{a_1, a_3\}$, $\{a_1, a_2, a_3, a_4\}$, $\{a_1\}$, $\{a_2\}$, $\{a_3\}$ and $\{a_4\}$ do not block π. Also, π is Nash stable since:

1. *$\{a_1, a_2\} \succ_1 \{a_1, a_3, a_4\}$ and $\{a_1, a_2\} \succ_1 \{a_1\}$;*
2. *$\{a_1, a_2\} \succ_2 \{a_2, a_3, a_4\}$ and $\{a_1, a_2\} \succ_2 \{a_2\}$;*
3. *$\{a_3, a_4\} \succ_3 \{a_1, a_2, a_3\}$ and $\{a_3, a_4\} \succ_3 \{a_3\}$; and,*
4. *$\{a_3, a_4\} \succ_4 \{a_1, a_2, a_4\}$ and $\{a_3, a_4\} \succ_4 \{a_4\}$.*

Because Nash stability implies individual and contractual individual stability, π also satisfies these stability requirements.

From an efficiency perspective, the quality of the representation can be assessed with respect to the complexity of computing stability related problems. To this end, consider Proposition 2.1.

Proposition 2.1 *(proven in [2]) Every hedonic game $\mathcal{H}$ has an individually and contractually individually stable solution.*

Proposition 2.1 was proven in [2]. Consequently, non-emptiness problems for individually and contractually individually stable solutions are redundant as they have a definite 'yes' answer. In contrast, no such guarantees can be made for the other stability concepts.

If the preferences are represented as individually rational coalitional lists then problems concerning non-emptiness of the core and Nash stable solutions are NP-hard, meaning no algorithm can guarantee to answer this question with time complexity that is polynomial in the size of $\mathcal{H}$ [2]. On the other hand, computing if a given coalition structure is either individually rational, Nash stable, individually stable or contractually individually stable can be answered in polynomial time. This is because these problems are concerned with individual deviations among the agents who belong to the coalitions in these structures and there can be no

more than n^2 such deviations.

Conversely, if the preferences are represented as individually rational coalitional lists then computing if a coalition structure belongs to the core can be done with polynomial time complexity [2]. However, if the preferences are additively seperable then the same problem is coNP-complete [75]. Of course, if the preferences are represented as individually rational coalitional lists then this does not guarantee succinctness in the representation and so the former insight is significant if and only if the representation is concise.

Hedonic Coalition Nets

In the spirit of marginal contribution nets representations, a succinct, rule-based representation for hedonic games has been developed [22]. Formally, these games are represented by a framework $\mathcal{H}' = \langle Ag, R_1, \ldots, R_n \rangle$ where:

- $Ag = \{a_1, \ldots, a_n\}$ is the set of agents; and,
- R_i is a set of rules for every agent $a_i \in Ag$.

Intuitively, these rules define the preferences of the agents over the coalitions and are represented as formulas of propositional logic using the conventional Boolean operators ("$\wedge$", "$\vee$", "$\Rightarrow$", "$\iff$", and "$\neg$"), as well as the truth constants "$\top$" (for truth) and "$\bot$" (for falsity). The pattern of these rules contain variables that represent every agent. In this representation, a rule for agent a_i is a pair (φ, B), where:

- φ is a formula of propositional logic; and,
- $B \in \mathbb{R}$.

From all these rules, the utility attributed to a coalition $C \subseteq Ag$ ($v(C)$) is computed from summing the values of all rules (all B) where $\varphi[C] = \top$, *i.e.*, through summing the values of all rules φ which are satisfied by a truth assignment where all of the variables that represent an agent in C have value '$\top$' and all variables representing an agent in $Ag \setminus C$ have value '$\bot$'.

In this context, for any two coalitions C, C' and, $\forall a_i \in C \cap C'$,

$$C \succ_i C' \iff v(C) > v(C').$$

In terms of both expressivity and conciseness, the following was proven to hold for every hedonic nets representation:

1. Hedonic nets are just as compact as all the representations whose preferences are represented as any of **Pref1**-**Pref4**;
2. Hedonic nets are strictly more expressive than the representations whose preferences are annonymous, additively seperable or of form β-$I\mathcal{W}$; and,
3. For some games, hedonic nets are exponentially more compact than hedonic representations whose preferences are represented as individually rational coalitional lists.

In the general hedonic nets representation, computing if a structure belongs to the core of $\mathcal{H}'$ is coNP-complete, whereas computing if the core is non-empty is Σ_2^P-complete. However, imposing certain restrictions on the rules enables the development of algorithms which can guarantee answers to decision problems concerning non-emptiness of the core with time complexity that is polynomial in the size of the representation (*e.g.*, if the rules contain no more than a number of variables and connectives that are no more than polynomial in the size of n).

2.4.6 Qualitative Coalitional Games

While, in many domains, a real number value offers a feasible indication of the utility obtained from forming coalitions, in others it does not. For example, in multi-agent system resource allocation domains, it may not be possible to measure the exact utility of the agents, but only whether they are 'satisfied' or 'unsatisfied' [11]. In such cases, *qualitative*, and not quantitative, characteristic functions should be used and *qualitative coalitional games* (QCG) are a particular example of representations that employ such functions [82].

Definition 2.26 *(from [82]) A QCG is a* $(n+3)$*-tuple* $\Gamma = \langle G, Ag, G_1, \ldots, G_n, v \rangle$*, where:*

- $G = \{g_1, \ldots, g_m\}$ *is a set of possible goals;*
- Ag = $\{a_1, .., a_n\}$ *is a set of agents;*
- $G_i \subseteq G$ *is a set of goals for each agent* $a_i \in Ag$*; and,*
- $v : 2^{Ag} \rightarrow 2^{2^G}$ *is a function which takes, as input, a coalition and outputs a set of subsets of the goals in* G*.*

In Γ, the function v determines a set $v(C)$ of choices of goals for a coalition $C \subseteq Ag$ with the interpretation being, for any $G' \subseteq G$, if $G' \in v(C)$ then one of the choices available to C is to accomplish *all* the goals in G'. In this representation:

- Every agent $a_i \in Ag$ is indifferent between the goals in his own set G_i, meaning they would be satisfied if they accomplished *any* of these goals; and,
- There are no externalities from coalition formation, which in this particular setting, has the interpretation that an agent is satisfied if and only if it belongs to a coalition that can accomplish any of its goals.

To convey a notion of stability in QCGs, consider the following definition.

Definition 2.27 *In any QCG* Γ*:*

- *A set of goals* G' *satisfies agent* a_i *if* $G' \cap G_i \neq \emptyset$ *(where* $\emptyset$ *is the empty set);*
- G' *satisfies coalition* C *if it satisfies every agent in* C*; and*
- G' *is feasible for coalition* C *if* $G' \in v(C)$*.*

Against Definition 2.27, In Γ, a coalition $C \subseteq Ag$ is:

- *successful* if $\exists G' \subseteq G$ which is feasible for C and satisfies every agent in C; and,
- *minimal* if it is successful and all coalitions $C' \subset C$ are not successful.

In words, a coalition C is successful if it can accomplish a set of goals $G' \subseteq G$ that contains any of the goals in the individual sets of every agent in C. Intuitively, no self-interested agent will form a coalition that is not successful because none of their individual goals will be accomplished. Thus, in QCGs, it can be assumed that only successful coalitions will be formed. However, success alone does not ensure stability of the coalition. From a game theoretical point of view, a set of goals which satisfies a minimal coalition C_{min} can be interpreted as being in the *core* of C_{min} since all coalitions $C' \subset C_{min}$ are not successful and, therefore, no agent has incentive to deviate from C_{min} and form any coalition C' instead. In this context, computing if a set of goals G' is in the core of any coalition $C \subseteq Ag$ is equivalent to computing if:

1. $G' \in v(C)$;
2. $\forall a_i \in C, G' \cap G_i \neq \emptyset$; and,

3. C is minimal.

Thus, the problem of computing if the core of a coalition $C \subseteq Ag$ is non-empty is equivalent to verifying if it is both minimal and successful.

Example 2.11 *Consider a QCG Γ where $Ag = \{a_1, a_2, a_3\}$ $G = \{g_1, g_2, g_3\}$ and $\forall i = 1, \ldots, 3$, $G_i = \{g_i\}$. Also, suppose that:*

$$\begin{aligned} v(\{a_1, a_2\}) &= \{\{g_1, g_2\}\}; \\ v(\{a_1, a_2\}) &= \{\{g_1, g_2, g_3\}\}; \\ v(\{a_2, a_3\}) &= \{\{g_2, g_3\}\}; and, \\ v(\{a_1, a_2, a_3\}) &= \{\{g_1, g_2, g_3\}\}. \end{aligned}$$

and $v(C) = \emptyset$ for all other coalitions $C \subseteq Ag$. Observe that all of the coalitions $\{a_1, a_2\}, \{a_1, a_3\}, \{a_2, a_3\}$ and $\{a_1, a_2, a_3\}$ are successful since there is exactly one set of goals that these coalitions can accomplish that contains any of the individual goals in the individual sets of the agents. Also, observe that the coalitions $\{a_1, a_2\}, \{a_1, a_3\}, \{a_2, a_3\}$ are minimal, whereas the coalition $\{a_1, a_2, a_3\}$ is not. Consequently, the core of $\{a_1, a_2, a_3\}$ is empty, whereas the set of goals that the coalitions $\{a_1, a_2\}, \{a_1, a_3\}, \{a_2, a_3\}$ can accomplish is in the core of these coalitions.

As well as stability, *dependency* can also be used to understand coalition formation in QCGs. In QCGs, dependencies are captured in the following concept.

Definition 2.28 *In any Γ, agent a_j is a* veto agent *for agent a_i if it is the case that, for every set of goals $G' \subseteq G$ such that $G' \cap G_i \neq \emptyset$, then a_j belongs to every coalition which can accomplish any G',* i.e*, $\forall C \subseteq Ag$, $\forall G' \subseteq G : G' \cap G_i \neq \emptyset$,*

$$G' \in v(C) \Rightarrow a_j \in C.$$

If a_j is a veto agent for a_i then this characterizes a *dependency* between a_i and a_j since the co-operation of a_j is essential for a_i to accomplish any of a_i's goals. Therefore, a_i must cooperate with a_j if a_i is to achieve any of their goals.

Example 2.12 *Consider a QCG Γ where $Ag = \{a_1, a_2, a_3\}$, $G = \{g_1, g_2, g_3\}$ and $\forall i = 1, \ldots, 3$, $G_i = \{g_i\}$. Also, suppose that,*

$$\begin{aligned} v(\{a_1\}) &= \{\{g_1\}\} \\ v(\{a_3\}) &= \{\{g_3\}\} \\ v(\{a_1, a_3\}) &= \{\{g_1, g_2, g_3\}\} \\ v(\{a_2, a_3\}) &= \{\{g_2, g_3\}\} \\ v(\{a_1, a_2, a_3\}) &= \{\{g_1, g_2, g_3\}\} \end{aligned}$$

and $v(C) = \emptyset$ for all other coalitions $C \subseteq Ag$. Clearly, against Definition 2.28, in this QCG, agent a_1 is a veto agent for themselves, since they belong to every coalition which can accomplish g_1. Also, a_3 is a veto agent for both themselves and agent a_2 since they belong to every coalition that can accomplish both g_2 and g_3. No other veto agent relations exist.

In the QCG representation Γ, as presented in Definition 2.26, every set of goals that can be accomplished by every coalition must be explicitly specified. Since the number of sets of goals and coalitions are exponential in the number of goals and agents, respectively, this means that Γ will not be concise. To this end, a compact representation of QCGs was presented in [82].

A Representation Based on Logic

Let $\Gamma_{succ} = \langle Ag, G, G_1, \ldots, G_n, \Psi \rangle$ denote the compact representation developed in [82]. The agents in Ag and the goals in G are represented as propositional variables and the function Ψ is a formula of propositional logic over these variables. This formula consists of the conventional Boolean operators ("$\wedge$", "$\vee$", "$\Rightarrow$", " $\iff$ ", and "$\neg$"), as well as the truth constants "$\top$" (for truth) and "$\bot$" (for falsity). In this formula, *literals* represent the individual agents and goals. Specifically, Ψ takes, as input, both a coalition $C \subseteq Ag$ and a goal set $G' \subseteq G$. Intuitively, when C and G' are input this means that the literals that represent these individual goals and agents have assignment '$\top$', whereas the literals representing the agents in $Ag \setminus C$ and goals in $G \setminus G'$ have assignment '$\bot$'. Given this representation,

$$\Psi[C, G'] = \top \iff G' \in v(C),$$

i.e., $\Psi[C, G']$ evaluates to'$\top$' if and only if the agents in C can co-operate to achieve *all* of the goals in G'. Thus, $\Gamma_{succ} = \langle Ag, G, G_1, \ldots, G_n, \Psi \rangle$.

Γ_{succ} is no less expressive than Γ as any function v can be expressed as a formula of propositional logic. Additionally, although not always guaranteed, propositional logic formulae are capable of describing concise presentations of propositional functions when such are possible. In this context, Γ_{succ} may be concise in both the number of agents and goals and all complexity results for this representation are given under the assumption that Γ_{succ} is concise. Furthermore, given $C \subseteq Ag$, $G' \subseteq G$ and Ψ, determining if $\Psi[C, G'] = \top$ can be done in $|\Psi|$ steps, where $|\Psi|$ is the number of literals in Ψ. In this context, the processes involved in computing if a given coalition can exhaustively accomplish a given set of goals can be answered with polynomial time complexity.

Now, in QCGs, the utility (*i.e.*, 'satisfaction') obtained by individual agents in the grand coalition, as well as in any other coalition, are already given two important core-related questions in this representation are:

(Q1) Is the core of coalition C non-empty? and,

(Q2) Is a set of goals $G' \subseteq G$ in the core of a coalition $C \subseteq Ag$?

Following previous discussion, (Q1) is equivalent to computing if C is both minimal and successful.

Lemma 2.5 *Given Γ_{succ} and a coalition $C \subseteq Ag$, computing if C is minimal and successful is DP-complete.*[7]

Lemma 2.5 shows that decision problems concerning (Q1) are intractable. For a given coalition C, this implies that no algorithm can be guaranteed to solve a decision problem concerning (Q1) in a number of steps that are polynomial in the size of the representation. Conversely, (Q2) is equivalent to computing if G' is both a feasible choice for and satisfies C. Since the latter can be with polynomial time complexity, this intuition with respect to the complexity of (Q1) is also applicable to (Q2).

As well as stability, a natural question regarding dependency in QCGs can be expressed as follows.

(Q3) Is an agent a_i a veto agent for a_j?

Lemma 2.6 *Given Γ and any two agents $a_i, a_j \in Ag$, the problem of computing if a_i is a veto agent for a_j is coNP-complete.*

Lemma 2.6 shows that decision problems concerning (Q3) are complement to an intractable problem. Therefore, no algorithm can be guaranteed to solve these decision problems in a number of steps that are polynomial in the size of the representation.

In conclusion, although Γ_{succ} can enable conciseness, it cannot guarantee that decision problems concerning coalition formation can be answered in time complexity that is polynomial in the size of the representation.

[7] See Appendix A for a formal definition of this complexity class.

2.5 Summary

In this chapter, an overview of cooperative game theory was provided. In particular, both partition and characteristic function games were formally defined. Essentially, the former representation accounts for externalities from coalition formation, whereas the latter does not.

For characteristic function games, when the gain from forming the coalition is transferable among all of the agents in the coalition, a number of solution concepts from game theory can be employed to determine which coalitions should be formed by either self-interested or fully cooperative agents. Generally, the core solution can be used to determine if the grand coalition is stable, whereas an optimal coalition structure concept can be used to identify disjoint and exhaustive coalitions of agents that maximize the welfare of the system.

With respect to understanding coalition formation, questions concerning non-emptiness and membership of the core need to be answered. The time complexity of decision problems concerning these questions is polynomial in the size of the representation. Therefore, one line of work has focused on developing representations that are fully expressive but, for case of interest, are of size that is polynomial in the number of agents. In this chapter, an overview of such representations was provided.

Chapter 3

Optimal Coalition Structure Generation Algorithms

Following discussions presented in Chapter 1, fully cooperative agents will always partition themselves into coalitions that are best for the system as whole. Thus, when given a characteristic function game representation of a system containing fully cooperative agents, the optimal coalition structure concept can aid a system designer to understand which coalitions will be formed by the agents. To this end, when given any characteristic function game, the following problem naturally arises,

For any characteristic function game, is a given coalition
structure π an optimal coalition structure?

In [66] it was proven that, even if all coalition values are given as input, computing if π is an optimal coalition structure is NP-hard. Thus, in contrast to the core-related problems that were considered in the previous chapter, even if all coalition values are known then no algorithm can be guaranteed to solve the above problem with time complexity that is polynomial in the number of coalition values. Furthermore, this result implies that, in the worst case, every coalition structure will have to be analyzed in order to solve this problem.

Proposition 3.1 *(Taken from [42]) In a system of n agents, there are,*

$$\mathcal{B}_n = \sum_{i=1}^{n-1} \binom{(n-1)}{i} \mathcal{B}_i,$$

coalition structures, where $\mathcal{B}_0 = \mathcal{B}_1 = 1$.

More formally, $\mathcal{B}_n$ is *the Bell number* (named after Edward Temple Bell) for the system and can be iteratively computed, beginning with $\mathcal{B}_0 = \mathcal{B}_1 = 1$. As n linearly increases, $\mathcal{B}_n$ exponentially increases. For example, if $n = 14$ then $\mathcal{B}_n = 190,899,322$ whereas if $n = 15$ then $\mathcal{B}_n = 1,382,958,545$. Thus, even for a moderate number of agents, billions of potential structures could be formed.

In characteristic function games, because there are no externalities from coalition formation, it is assumed that every coalition $C \subseteq Ag$ has the same value $v(C)$ in every structure in which it is embedded. This means that, given the values of every coalition $C \subseteq Ag$, it may be possible to determine *a priori* if certain coalition structures are not optimal. Against this insight, in the multi-agent systems paradigm, algorithms have been developed that can output an optimal structure through computing only the values of those structures that belong to a subset $\Pi' \subset \Pi$. This chapter provides an overview of the state-of-the-art optimal coalition structure generation algorithms. To be precise:

- In Section 3.1 ,an overview of the state-of-the-art optimal coalition structure generation algorithms that consider *ex-post* information assumptions is provided; and,
- In Section 3.2, an overview of the state-of-the-art optimal coalition structure generation algorithms that consider *ex-ante* information assumptions is provided

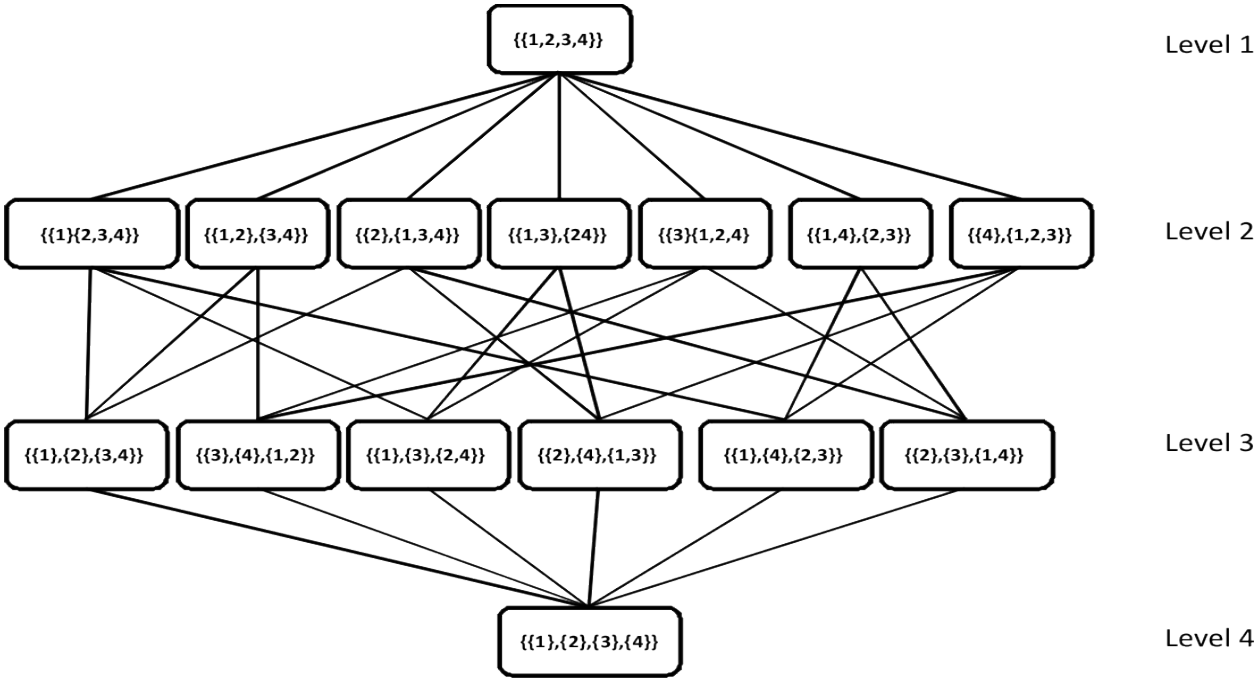


Figure 3.1: A graphical representation of all coalition structures in a four agent system (here, the number represents the index of the agent. For example, '1' denotes agent a_1)

3.1 *Ex-post* Optimal Coalition Structure Generation

Typically, *ex-post* optimal coalition structure generation is relevant when it is either impossible or infeasible to obtain all coalition values. Arguably, in the multi-agent system paradigm, it is the work of [66] which first fully discussed the issues associated with *ex-post* optimal coalition structure generation. Inherent to this work is the representation of the space of all structures. Specifically, Π is represented as a graph with n levels where, $\forall i \in [1, \ldots, n]$, Level i contains all structures of size i (that is, all structures which contain exactly i coalitions). For $i = 1, \ldots, n-1$, edges in the graph connect structures in Level i to structures in Level $i+1$. Here, an edge represents the fact that the structure in Level i was formed by the merge of two coalitions in the structure to which it is connected in Level $i+1$. The graphical representation of a multi-agent system with $Ag = \{a_1, a_2, a_3, a_4\}$ is presented in Figure 3.1.

Given this representation, Sandholm *et al.* proposed a procedure for generating an optimal coalition structure which is presented in Algorithm 3.1 [66]. This algorithm begins by first computing the value of the structure in Level 1 and storing both the structure and its value in memory. Upon doing this, an exhaustive search of Level 2 is undertaken and, after the value of every individual structure in this level has been computed, if a structure is found with value strictly greater than the current optimal value then the system is updated, *i.e.*, both this structure and its value are stored in memory as the new optimal. After analyzing these coalition structures, Levels $n, n-1, \ldots, 3$ are then also sequentially searched in this manner. Upon analyzing all of Π, the structure that is stored in memory is output as the optimal coalition structure.

Lemma 3.1 *(From [66]) For $L = n, n-1, \ldots, 3$, after all of the coalition structure values in Level L have been computed, let $v(\pi^*_L)$ denote the current optimal value and let $v(\pi^*)$ denote the value of the structure which maximizes the welfare of the entire system.*

If $h = \lfloor \frac{n-L}{2} \rfloor + 2$, where L denotes the level that has just been exhaustively searched by the algorithm then,

$$\frac{v(\pi^*)}{v(\pi^*_L)} \leq k,$$

Input: $\mathcal{N}_t = \langle Ag, v \rangle$

1. Firstly, the algorithm computes the value of the single structure in Level 1 and sets this to be the optimal structure.
2. The algorithm then exhaustively searches Level 2 of the graph representation, computing the values of all coalition structures in this level. Whilst doing this, if a coalition structure is encountered that has value greater than the current optimal value then this structure, and its value, are stored in memory as the current optimal.
3. The levels n, $n-1$,...,3 are then sequentially searched as in 2. until the running time of the algorithm has expired or the entire space has been analyzed. At his point, the current optimal value is output.

Output: π^*.

Algorithm 3.1: The optimal coalition structure generation algorithm presented in [66]

where:

1. $k = \lceil \frac{n}{h} \rceil$ *if* $n \equiv h - 1 (mod\ h)$ *and* $n \equiv L (mod\ 2)$*; and,*
2. $k = \lfloor \frac{n}{h} \rfloor$*, otherwise.*

In numbers, suppose $n = 10$ and Level 8 has just been exhaustively searched (after Levels 1,2,10,9). In this instance:

(i) $h = \lfloor \frac{10-8}{2} \rfloor + 2 = 3$; but,

(ii) $10 \equiv 1 (\text{mod } 3)$.

Consequently, $k = \lfloor \frac{10}{3} \rfloor = 3$ and so the real optimal value is no greater than three times the value of the current optimal value. On the other hand, suppose that $n = 11$ and all of the structures in Level 3 have been analyzed (after Levels 1,2,11,10,9,8,7,6,5,4). This time:

(i) $h = \lfloor \frac{11-3}{2} \rfloor + 2 = 6$;

(ii) $11 \equiv 5 (\text{mod } 6) \equiv (6-1) (\text{mod } 6)$; and,

(iii) $11 \equiv 1 (\text{mod } 2) \equiv 3 (\text{mod } 2)$.

Thus, $k = \lceil \frac{11}{6} \rceil = 2$, meaning the optimal value is no greater than double the current optimal value.

Observe that after Levels 1 and 2 have been exhaustively analyzed, for every coalition $C \subseteq Ag$, $v(C)$ has been computed exactly once. As every coalition value must be computed in order to determine all structure values then, clearly, this is the minimum number of structure values that must be computed in order to obtain a bound from the optimal. Since there are 2^{n-1} coalition structures in Levels 1 and 2, the following lemma holds.

Lemma 3.2 *Given any characteristic function game* $\langle Ag, v \rangle$*, no less than* 2^{n-1} *coalition structures must be analyzed to obtain a bound* k*.*

Clearly, this algorithm only guarantees to output an optimal structure if the value of every coalition structure is computed and, following Proposition 3.1, this may not be feasible for large n. Furthermore, the bounds obtained from this algorithm may not be large enough to justify the amount of computational effort undertaken. For example, if $n = 12$ then, to obtain a bound of $\frac{n}{4}$, Level 1, 2, 12, 11, 10, 9 of the tree must be searched. It can be argued that, the number of structures analyzed in these levels is too large to justify this relatively small bound.

Following on from this work, another *ex-post* optimal coalition structure generation algorithm was developed in [15]. After performing the same first two steps as Algorithm 3.1, this algorithm then exclusively searches the remaining space focusing on computing the values of particular coalition structures. In more detail, after analyzing all of the coalition structures in Levels 1 and 2, rather than sequentially analyze all of the coalition structures in levels n to 3, this algorithm then computes the values of particular coalition structures instead.

To describe how this is done, let $SL(n, k, c)$ denote the set of all structures in a n agent system that contain exactly k coalitions with at least one coalition of size greater than c. For example,

$$SL(4, 2, 2) = \{\{\{a_1\}, \{a_2, a_3, a_4\}\}, \{\{a_2\}, \{a_1, a_3, a_4\}\}, \{\{a_3\}, \{a_1, a_2, a_4\}\},$$

$$\text{and } \{\{a_4\}, \{a_1, a_2, a_3\}\}\}.$$

Against this notation, let $SL(n, c)$ denote the set of all structures in levels 3 to $n - 1$ which contain at least one coalition of size greater than c, *i.e.*,

$$SL(n, c) = \bigcup_{k=3}^{n-1} SL(n, k, c).$$

After exhaustively searching Levels 1 and 2, the algorithm computes the values of all coalition structures in $SL(n, c)$, where $c = \frac{n(q-1)}{q}$ and q is arbitrarily chosen. Every time a structure is found with value greater than the current optimal value then both the structure and its value are stored in memory as the new optimal. The structure that is in memory when the algorithm terminates is output as the optimal structure. In contrast to the Algorithm 3.1, the following bounds were computed.

Lemma 3.3 *(From [15]) After searching $SL(n, \frac{n(q-1)}{q})$, let $v(\pi^*_q)$ denote the currently optimal structure. If $v(\pi^*)$ is the optimal value of the entire system then,*

$$\frac{v(\pi^*)}{v(\pi^*_q)} \leq 2q - 1.$$

In this way, desirable bounds can be determined by choosing an appropriate value of q.[1]

Observe that, as with Algorithm 3.1, this algorithm is *anytime, i.e.,* the bound improves as the running time of the algorithm increases. However, empirical evidence suggests this algorithm may be much faster than the approach of Algorithm 3.1 with respect to generating an optimal structure. For example, when $n = 1000$ and q is chosen so that the bound is small, this algorithm was shown to be 10^{379} times faster than Algorithm 3.1 with respect to generating a coalition structure which has value within this bound.

The two algorithms considered thus far satisfy both positive and negative criteria. For example, both guarantee to output an optimal and both are anytime which means that, were they to stop midway (*e.g.*, through technical failure) then they could still output a solution which was within a particular bound of the optimal value, *i.e.*, the quality of solution produced monotonically increases with computation time. However, to guarantee that an optimal coalition structure is output, both algorithms must compute the values of every coalition structure. From a computational perspective, this is undesirable.

[1]Note that, to give similar bounds to [66], in [15], q runs from $\lfloor \frac{n+1}{4} \rfloor$ to 2.

3.2 *Ex-ante* Optimal Coalition Structure Generation

In contrast to *ex-post* optimal coalition structure generation, *ex-ante* optimal coalition structure generation assumes that coalition values are known. Since these coalitions values are not dependent upon co-existing coalitions, many *ex-ante* algorithms exploit this feature to generate an optimal structure without having to analyze all of them. To this end, in this section, an overview of the main *ex-ante* algorithms are provided. Each algorithm is assessed against the following criteria:

Worst case complexity: The largest number of operations required to generate an optimal solution;

Memory requirements: The amount of data that must be stored in memory;

Robustness against technical failure: The ability of an algorithm to output a solution should it encounter technical failure; and,

Algorithm running times: The time it takes an algorithm to output an optimal solution.

From a computational perspective, because the number of coalition structures grows exponentially in the number of agents, it is desirable that any algorithm generates an optimal coalition structure using as little memory as possible, with low complexity and in as quick as possible running time. Also, it is desirable that, should the algorithm encounter technical failure, it is able to output a structure that is 'nearly' optimal (as with the anytime algorithms considered in the previous section). These criteria were chosen as they have been used in the literature thus far [84, 59, 55, 58, 57].

3.2.1 A Dynamic Programming Algorithm

Essentially, algorithms that employ dynamic programming techniques are used to solve problems which satisfy both *optimal substructure* and *overlapping subproblems* properties.

Definition 3.1 *(Taken from [55]) A problem is said to exhibit* optimal substructure *if it can be broken into subproblems that can be recursively solved so that the solutions can be combined to answer the original problem. Conversely, a problem is said to exhibit* overlapping subproblems *if these subproblems are not independent.*

Clearly, the problem of generating an optimal coalition structure exhibits both *optimal substructure* and *overlapping subproblems* since it can be broken into the following subproblem which can be solved for every coalition $C \subseteq Ag$.

> *DP subproblem.*
> For every coalition $C \subseteq Ag$, let $\mathcal{S}_C$ denote the set of all pairs (C', C'') such that $C' \cap C'' = \emptyset$ and $C' \cup C'' = C$. For every $(C', C'') \in \mathcal{S}_C$, $v(C)$ is compared with $v(C') + v(C'')$. If it is the case that, for every $(C', C'') \in \mathcal{S}_C$, $v(C) > v(C') + v(C'')$ then output $v(C)$. Else, if there exists $(C', C'') \in \mathcal{S}_C$ such that $v(C) < v(C') + v(C'')$ then, for all such (C', C''), output (any of) the pair(s) (C', C'') that have maximal combined value.

A so-called *dynamic programming (DP)* algorithm was developed in [84] that recursively solves this subproblem for every coalition $C \subseteq Ag$ and then combines the results to generate an optimal coalition structure. In this algorithm;

(i) The space of all coalitions is organized so that all coalitions of the same size are grouped together, and

(ii) For every coalition $C \subseteq Ag$, the algorithm employs two functions: $f_1(C)$ and $f_2(C)$ which output and store the solution and value to the DP subproblem for every coalition $C \subseteq Ag$, respectively.

Input: $\mathcal{N}_t = \langle Ag, v \rangle$, $\forall C \subseteq Ag$, $v(C)$.

1. Firstly, for $i = 1, \ldots, n$, set $f_1(\{a_i\}) = \{a_i\}$ and $f_2(\{a_i\}) = v(\{a_i\})$.
2. Beginning with all coalitions of size 2, *i.e.*, all $C \subseteq Ag$ such that $|C| = 2$ solve the DP subproblem.
3. Sequentially repeat 2. for all coalitions of size $3, \ldots, n$
4. Set $\pi^* = \{Ag\}$
5. For every coalition $C \in \pi^*$, If $f_1(C) = C$ then output π^*. Else if $f_1(C) \neq C$, then:
   (a) Set $\pi^* := \pi^* \setminus \{C\} \cup \{f_1(C)\}$; and,
   (b) Repeat 4 for new π^*.

Output: π^*.

Algorithm 3.2.1: A dynamic programming optimal coalition structure generation algorithm

The DP algorithm is presented in Algorithm 3.2.1.

Example 3.1 *Consider a three agent system* $Ag = \{a_1, a_2, a_3\}$ *where;*

- $v(\{a_i\}) = 2$*, for* $i = 1, 2, 3$*;*
- $v(\{a_1, a_2\}) = v(\{a_1, a_3\}) = v(\{a_2, a_3\}) = 3$*; and,*
- $v(\{a_1, a_2, a_3\}) = 6$*.*

The DP algorithm for this setting is as follows.

Step 1: In the first step of the DP algorithm, for $i = 1, \ldots, 3$*,* $f_1(\{a_i\})$ *is set as* $\{a_i\}$ *and* $f_2(\{a_i\})$ *is set as* $v(\{a_i\}) = 2$*.*

Step 2: In this step;

- $v(\{a_1, a_2\})$ *is compared to* $v(\{a_1\})$ *and* $v(\{a_2\})$*,*
- $v(\{a_1, a_3\})$ *is compared to* $v(\{a_1\})$ *and* $v(\{a_3\})$ *and,*
- $v(\{a_2, a_3\})$ *is compared to* $v(\{a_2\})$ *and* $v(\{a_3\})$*.*

Observe that,

1. $v(\{a_1, a_2\}) > v(\{a_1\}) + v(\{a_2\})$*;*
2. $v(\{a_1, a_3\}) > v(\{a_1\}) + v(\{a_3\})$*; and,*
3. $v(\{a_2, a_3\}) > v(\{a_2\}) + v(\{a_3\})$*.*

Consequently,

1. $f_1(\{a_1, a_2\}) = \{\{a_1\}, \{a_2\}\}$ *and* $f_2(\{a_1, a_2\}) = 4$*;*
2. $f_1(\{a_1, a_3\}) = \{\{a_1\}, \{a_3\}\}$ *and* $f_2(\{a_1, a_3\}) = 4$*; and,*
3. $f_1(\{a_2, a_3\}) = \{\{a_2\}, \{a_3\}\}$ *and* $f_2(\{a_2, a_3\}) = 4$*.*

Step 3: Now, for the grand coalition, observe that,

$$\mathcal{S}_{Ag} = \{\{a_1, a_2\}\{a_3\}, \{a_1, a_3\}\{a_2\}, \{a_2, a_3\}\{a_1\}\},$$

and since,

$$v(\{a_1, a_2, a_3\}) > v(\{a_1, a_2\}) + v(\{a_3\}) = v(\{a_1, a_3\}) + v(\{a_2\}) = v(\{a_2, a_3\}) + v(\{a_1\}),$$

$f_1(Ag) = Ag$ *and* $f_2(Ag) = 6$.

Step 4: Set π^* *to be* Ag.

Step 5: Since $f_1(Ag) = Ag$ *then* $\pi^* = \{Ag\}$ *is output.*

The worst case complexity of the DP algorithm is presented in Lemma 3.4.

Lemma 3.4 *(Proven in [84]) The DP algorithm runs in* $O(3^n)$ *time.*

with regards to worst case complexity, Lemma 3.4 shows that, because $3^n < n^n$ for $n > 3$, the DP algorithm generates an optimal coalition structure with much lower time complexity than the *ex-post* algorithms. However, the memory requirements for this algorithm are exponentially large since, for every non-empty coalition $C \subseteq Ag$, all of $v(C)$, $f_1(C)$ and $f_2(C)$ must be stored in memory. Also, this algorithm is not anytime, meaning should the algorithm have to stop midway, *e.g.*, if there is a system failure, then the algorithm cannot output a currently optimal structure. Thus, it is not robust against technical failure.

3.2.2 An Improved Dynamic Programming Algorithm

To reduce the significant memory requirements of the DP algorithm, as well as its worst case complexity, an *Improved Dynamic Programming* (IDP) algorithm was developed in [58]. To explain how the IDP algorithm works, recall the graphical representation of the space of all coalition structures presented in Figure 3.1. In this graph, for $i = n, n-1, \ldots, 2$, every node in Level i represents every coalition structure of size i. Furthermore, edges connecting a coalition structure π of size i to every coalition structure π' of size $i-1$ represents the formation of π' *via* the partitioning of a coalition in π into exactly two coalitions.

Given this graph, for every coalition $C \subseteq Ag$, the DP algorithm computes the DP subproblem for every coalition C by analyzing every possible edge that represents a partitioning C into each $(C', C'') \in \mathcal{S}_C$. Then, starting from the node representing the grand coalition, the algorithm traverses a series of connected nodes (referred to as a path from now onward) until an the node representing the optimal coalition structure is reached.

To be precise, for every coalition $C \subseteq Ag$, every partition of C that has size two is analyzed and the partition with maximal value (denoted (C', C'')) is compared with $v(C)$ using $f_2(C)$. Now, if,

$$v(C) > v(C') + v(C''),$$

then (C', C'') is stored in $f_1(C)$. This indicates that, whenever a node is reached that represents a structure π, such that $(C; \pi) \in \mathcal{E}$, then the best path (out of all the ones that represent a partitioning of C) is the one that leads to $\pi \setminus \{C\} \cup \{C', C''\}$. On the other hand, if,

$$v(C) \leq v(C') + v(C''),$$

then C is stored in $f_1(C)$. This indicates that, whenever a node is reached that represents a structure π, such that $(C; \pi) \in \mathcal{E}$, then it is not beneficial to make any movement that involves partitioning C. Intuitively, due to the manner in which $f_2(C)$ is calculated (it takes into account $f_2(C')$ for all $C' \subset C$) the choice in

analyzing a particular path is done by taking into account the subsequent paths that will follow this one.

Observe that, for a number of nodes, there may exist more than one path leading from the node representing the grand coalition to the node representing an optimal coalition structure. Given this observation, the authors in [58] prove that if there is a path from the grand coalition to an optimal node then the DP algorithm will find it. Thus, to reduce the number of partitioning evaluations, an algorithm was proposed that computes the DP subproblem through analyzing only a subset of all possible partitions. Specifically, this is achieved by removing appropriate edges from the graph whilst still ensuring that there exists a path from the node in Level n to every other node in Levels $n-1, \ldots, 1$.

In more detail, let $E^{i',i''}$ denotes the set of all the edges that involve partitioning a coalition of size $i'+i''$ into exactly two coalitions of size i' and i'', respectively (where $i' \leq i$). The dynamic programming algorithm will analyze all of the edges in the integer partition graph (denoted E). Against this notation, let $E^* \subseteq E$ denote a subset of these edges that is defined as follows:

$$E^* = \bigcup_{i',i'':i''\leq n-(i''+i')} E^{i',i''} \cup \bigcup_{i',i'':i'+i''=n} E^{i',i''}.$$

It was proven that the the edges in E^* are sufficient to ensure that every node in the graphical representation of Π must have a path leading to it from the node representing the grand coalition. Based on this, IDP only analyzes the edges in E^* and, in so doing, performs considerably fewer operations than DP.

The DP algorithm will require analyzing the following number of partitions (from [58]):

$$t = \sum_{i=1}^{n} (\binom{n}{i} \times \sum_{i'' \in \{\lceil \frac{i}{2} \rceil, \ldots, i-1\}} N^{n-i'',i''}),$$

whereas, IDP algorithm will analyze the following number of partitions (from [58]):

$$d = \sum_{i=1}^{n} (\binom{n}{i} \times \sum_{i'' \in \{\lceil \frac{i}{2} \rceil, \ldots, i-1\}, i'' > n-i} N^{n-i'',i''}),$$

where,

$$N^{n-i'',i''} = \begin{cases} \sum_{i''=\lceil \frac{i}{2} \rceil}^{i-1} \frac{\binom{n}{i}}{2} & \text{if } n-i''=i'' \\ \sum_{i''=\lceil \frac{i}{2} \rceil}^{i-1} \binom{n}{i} & \text{otherwise.} \end{cases}$$

Note that $t < d$ and so the IDP algorithm analyzes less partitions than the DP algorithm. For instance, for $n = 25$, it was proven that the IDP algorithm requires analyzing only 38.7% of the partitions that are analyzed by the DP algorithm.

Additionally, by not storing in memory all partitions of the coalitions but, instead, re-analyzing them as the paths are traversed, the memory requirements can be reduced. As there are no more than n nodes in any path, the computation involved in this re-analysis is negligible relative to the number of coalitions. By not using $f_1(C)$ for every coalition $C \subseteq Ag$, it is shown that the IDP algorithm requires between 33.3% and 66.6% of the memory that is required for the DP algorithm. In this context, the IDP algorithm improves on the DP algorithm against both the worst case complexity and running time criteria.

3.2.3 An Integer Partition (IP) Optimal Coalition Structure Generation Algorithm

In contrast to the algorithms that employ dynamic programming techniques, the *Integer Programming (IP)* algorithm of Rahwan *et al.* employs pre-processing techniques whilst retaining anytime properties [59]. Fundamental to this algorithm are the manner in which the space of all coalition structures are represented and searched to find an optimal coalition structure.

Representing the space of all coalition structures

The authors of [59] present a novel representation of the space of all coalition structures based on the size of the coalitions which belong to them. Specifically, this representation is based on the integer partitions of the number of agents n. For example, the integer partitions of $n = 5$ are: $\{5\}$, $\{1, 4\}$, $\{2, 3\}$, $\{1, 1, 3\}$, $\{1, 2, 2\}$, $\{1, 1, 1, 2\}$ and $\{1, 1, 1, 1, 1\}$. Here, each integer k in every partition can be interpreted as a coalition of size k. For example, $\{1, 4\}$ represents the space of all coalition structures that consist of exactly one coalition of size 1 and exactly one coalition of size 4. If $\overline{G}$ denotes the space of integer partitions of n then the space of all coalition structures is partitioned into subspaces using the mapping $F : \Pi \rightarrow \overline{G}$.

Computing maximal and minimal bounds on all coalition structure values

From the input of all coalition values, it is possible to compute the maximal and minimal values of all the structures in every subspace $g \in \overline{G}$. This is achieved by computing basic statistical information on all coalitions of the same size. Specifically, for $i = 1, \ldots, n$, the maximum, minimum and average values of all coalitions of size i are computed. Let these be denoted as max_i, min_i and av_i hereafter. From this data, for each subspace $\overline{g} \in \overline{G}$:

- The maximal coalition structure value, referred to as the *upper bound* of $\overline{g}$, (denoted $UB_{\overline{g}}$) is set as $\sum_{\forall i \in \overline{g}} max_i$;
- The minimal coalition structure value, referred to as the *lower bound* of $\overline{g}$, (denoted $LB_{\overline{g}}$) is set as $\sum_{\forall i \in \overline{g}} min_i$; and,
- The average coalition structure value, referred to as the *average* bound of $\overline{g}$, (denoted $Av_{\overline{g}}$).

The maximal and minimal coalition structure values in each subspace $\overline{g} \in \overline{G}$ can be easily computed from all the coalition values. However, less easily, the average values in each subspace $\overline{g} \in \overline{G}$ is computed as in Lemma 3.5.

Lemma 3.5 *(Proven in [59]) The average value of any subspace $\overline{g} \in \overline{G}$ ($Av_{\overline{g}}$) can be computed using the formula:*

$$Av_{\overline{g}} = \sum_{\forall i \in \overline{g}} av_i.$$

For example, if $n = 5$ then, for $\overline{g} = \{1, 1, 3\}$:

- $UB_{\overline{g}} = max_1 + max_1 + max_3$;
- $AV_{\overline{g}} = av_1 + av_1 + av_3$; and,
- $LB_{\overline{g}} = min_1 + min_1 + min_3$.

It is clear from this example that these bounds may not be tight and are therefore not exact values. However, although the lower and upper bounds are not exactly equal to $LB_{\overline{g}}$ and $UB_{\overline{g}}$, respectively, it is certain that the maximal structure value in $\overline{g}$ is no more than $\overline{g} \in \overline{G}$ and the minimal value in $\overline{g}$ is no less than $LB_{\overline{g}}$. In this context, such bounds are therefore appropriate.

Determining the best subspaces of structures to search

Now, $\forall \overline{g} \in \overline{G}$, given $UB_{\overline{g}}$, $LB_{\overline{g}}$ and $AV_{\overline{g}}$, the IP algorithm computes both the upper and lower values for $\overline{G}$ (denoted $UB_{\overline{G}}$ and $LB_{\overline{G}}$, respectively) as follows:

- $UB_{\overline{G}}$ = arg max$_{\overline{g} \in \overline{G}}$ $UB_{\overline{g}}$; and,
- $LB_{\overline{G}}$ = arg max$_{\overline{g} \in \overline{G}}$ $AV_{\overline{g}}$.[2]

Once these bounds have been computed, the first pre-processing technique employed by the algorithm is to prune away all subspaces of structures $\overline{g} \in \overline{G}$ such that:

$$UB_{\overline{g}} < LB_{\overline{G}}.$$

Obviously, if $UB_{\overline{g}} < LB_{\overline{G}}$ then the value of every structure in $\overline{g}$ cannot be optimal and so every coalition structure in $\overline{g}$ is not analyzed. After pruning all $\overline{g}$ from $\overline{G}$, the values of all coalition structures in the subspace $\overline{g}^*$, where $UB_{\overline{G}} = UB_{\overline{g}^*}$, are then computed.

Computing coalition structure values in a given subspace

While computing the values of the coalition structures that belong to a particular subspace, methods are undertaken to ensure that the values of only valid (*i.e.*, not overlapping) coalition structures are computed. Similarly, methods are used to avoid redundant computations, such as computing the value of same structure multiple times (*e.g.*, computing $v(\pi_\alpha = \{\{a_1, a_2\}\{a_3, a_4\}\})$ and $v(\pi_\beta = \{\{a_3, a_4\}\{a_1, a_2\}\})$).

Specifically, suppose that $\overline{g}^* = \{\overline{g}_{i_1}, \ldots, \overline{g}_{i_k}\}$ is the subspace in $\overline{G}$ such that $UB_{\overline{g}^*} = UB_{\overline{G}}$ and, for $j = 1, \ldots, k$, let:

- $\underline{A}_j$ be the set of all coalitions of agents of size j, all ordered non-decreasingly with respect to the indices of the agents who belong to them; and,
- $\mathbf{M}_{i_j} : |\mathbf{M}_{i_j}| = \overline{g}_{i_j}$ be a temporary array that can be used to cycle through all possible coalitions which could belong to $\overline{g}_{i_j}$.

At the start of the subspace search, $\mathbf{M}_{i_1}$ is assigned to a coalition of size $\overline{g}_{i_1}$ in $\underline{A}_{i_1}$. Given this coalition, $\mathbf{M}_{i_2}$ then cycles through all coalitions of size $\overline{g}_{i_2}$ in $\underline{A}_{i_2}$ until a coalition that does not overlap with the coalition in $\mathbf{M}_{i_1}$ is found. After that, $\mathbf{M}_{i_3}$ is then used to cycle through all coalitions of size $\overline{g}_{i_3}$ in in $\underline{A}_{i_3}$ until a coalition that does not overlap with both of those in $\mathbf{M}_{i_1}$ and $\mathbf{M}_{i_2}$ is found. This is repeated until all of $\mathbf{M}_{i_1}$ to $\mathbf{M}_{i_k}$ are assigned disjoint coalitions. At this point, the value of this coalition structure is then calculated and compared with the maximum value found so far. After that, the coalitions in $\mathbf{M}_{i_1}$, ..., $\mathbf{M}_{i_k}$ are updated so as to compute the value of a different structure in $\overline{g}^*$.

Additionally, whilst computing the structure values in $\overline{g}^*$, pre-processing is also applied in order to reduce the number of coalition structure values which are computed.

Lemma 3.6 *Suppose $\overline{g}^* = \{\overline{g}_{i_1}, \ldots, \overline{g}_{i_k}\}$ and, suppose that, for $j \in [1, \ldots, k)$, $\boldsymbol{M}_{i_1}$ to $\boldsymbol{M}_{i_j}$ have been assigned coalitions $C_{i_1} \ldots C_{i_j}$, respectively. If,*

$$v(C_{i_1}) + \ldots + v(C_{i_j}) + UB_{\overline{g}_{i_{j+1}}} + \ldots + UB_{\overline{g}_{i_k}} < LB_{\overline{G}},$$

then all structures $\pi \in \overline{g}^*$ such that $C_{i_1} \ldots C_{i_j} \subseteq \pi$ cannot be optimal.

[2]It is worth noting that some algorithms which compute coalition values also compute the values of a number of structure as well. Thus, in some cases, $LB_{\overline{G}}$ = arg max$_{\overline{g} \in \overline{G}, \pi \in \Pi}$ $(Av_{\overline{g}}, v(\pi_n^*))$, where $v(\pi_n^*)$ is the biggest coalition structure value computed during the coalition value calculation stage.

This branch and bound rule employs pre-processing to identify groups of disjoint coalitions which cannot belong to an optimal structure in $\overline{g}^*$. By avoiding them, this pre-processing ensures that the values of structures which cannot be optimal are not computed.

Whilst analyzing the coalition structures in $\overline{g}^*$, if a structure $\pi^* \in \overline{g}^*$ is found such that $v(\pi^*) = UB_{\overline{G}}$ then this is output as the optimal coalition structure. However, as $UB_{\overline{G}}$ is not a tight bound, this coalition structure may not exist. Consequently, if no such structure is found, if $v(\pi_n^*)$ denotes the optimal value after computing all the coalition values in $\overline{g}^*$ then the system is updated as follows:

- $LB_{\overline{g}}$ is updated so that $LB_{\overline{G}}$ = arg $\max_{\overline{g} \in \overline{G}, \pi \in \Pi} (Av_{\overline{g}}, v(\pi_n^*))$;
- All $\overline{g} \in \overline{G}$ such that $UB_{\overline{g}} < LB_{\overline{G}}$ are pruned away; and,
- $UB_{\overline{G}}$ is updated so that it is the maximal of $v(\pi_n^*)$ or the maximal $UB_{\overline{g}'}$ $\forall \overline{g}' \in \overline{G} \setminus \{\overline{g}^*\}$.

The coalition structures in the next most promising subspace are then analyzed in the same way and, after doing this, the system is subsequently updated. This continues until either a structure is found with value equal to the system upper bound or all of the space has been searched.

Example 3.2 *Consider a four agent multi-agent system* $Ag = \{a_1, a_2, a_3, a_4\}$ *with coalition values;*

L_1	L_2	L_3	L_4
$v(\{a_4\})$ = *2*	$v(\{a_3, a_4\})$= *5*	$v(\{a_2, a_3, a_4\})$= *7*	$v(\{a_1, a_2, a_3, a_4\}) =$ 8
$v(\{a_3\})$ = *2*	$v(\{a_2, a_4\})$= *2*	$v(\{a_1, a_3, a_4\})$= *7*	$-$
$v(\{a_2\})$ = *3*	$v(\{a_1, a_4\})$= *3*	$v(\{a_1, a_2, a_4\})$= *7*	$-$
$v(\{a_1\})$ = *1*	$v(\{a_2, a_3\})$= *5*	$v(\{a_1, a_2, a_3\})$= *7*	$-$
$-$	$v(\{a_1, a_3\})$= *5*	$-$	$-$
$-$	$v(\{a_1, a_2\})$= *4*	$-$	$-$

From these values, observe that:

i	max_i	min_i	av_i
1	*3*	*1*	*2*
2	*5*	*2*	*4*
3	*7*	*7*	*7*
4	*8*	*8*	*8*

Now, let $\overline{G}$ = $\{\overline{g}_1, \overline{g}_2, \overline{g}_3, \overline{g}_4, \overline{g}_5\}$, *where* $\overline{g}_1 = \{4\}$, $\overline{g}_2 = \{2, 2\}$, $\overline{g}_3 = \{1, 3\}$, $\overline{g}_4 = \{1, 1, 2\}$ *and* $\overline{g}_5 = \{1, 1, 1, 1\}$.

$\overline{g}_i$	$UB_{\overline{g}_i}$	$LB_{\overline{g}_i}$	$AV_{\overline{g}_i}$
$\overline{g}_1$	*8*	*8*	*8*
$\overline{g}_2$	*10*	*4*	*8*
$\overline{g}_3$	*10*	*8*	*7*
$\overline{g}_4$	*14*	*8*	*11*
$\overline{g}_5$	*12*	*4*	*8*

From this data, $UB_{\overline{G}} = 14$ and $LB_{\overline{G}} = 11$. Since $UB_{\overline{g}_1}$, $UB_{\overline{g}_2}$, $UB_{\overline{g}_3} < LB_{\overline{G}}$, the values of all of the structures in these subspaces are not computed. Specifically, $\overline{g}_1$, $\overline{g}_2$ and $\overline{g}_3$ are pruned from $\overline{G}$.

The subspace with the highest upper bound, $\overline{g}_4$, is searched first and all coalition structure values in this subspace are computed. Observe that the structure in this subspace with the highest value is $\{\{a_2\}, \{a_4\}, \{a_1, a_3\}\}$. Thus, $\pi^*_n = \{\{a_2\}, \{a_4\}, \{a_1, a_3\}\}$ and $v(\pi^*_n) = 10$ are stored in memory. Since $v(\pi^*_n) < UB_{\overline{G}}$ and $v(\pi^*_n) < LB_{\overline{G}}$, both $UB_{\overline{G}}$ and $LB_{\overline{G}}$ are kept at their current values.

As $v(\pi^*_n) \neq UB_{\overline{G}}$, $\overline{g}_5$, the subspace with the next biggest upper bound value, is then searched and all coalition structure values in this subspace are computed. There is exactly one structure in this subspace, $\{\{a_1\}, \{a_2\}, \{a_3\}, \{a_4\}\}$, and it has value 8. Since $v(\{\{a_1\}, \{a_2, \}\{a_3\}, \{a_4\}\}) < v(\pi^*_n)$ and since all of $\overline{G}$ has been searched, π^*_n is output and the algorithm terminates.

Note that, the IP algorithm may have to analyze every coalition structure in order to output an optimal, meaning its worst case complexity is $O(n^n)$. However, as only coalition values are stored in memory, this memory cost is less than that of the DP algorithm. Furthermore, as this algorithm is anytime, it may be able to output a nearly optimal structure if technical failure is encountered. Also, experiments have shown that this algorithm has relatively fast running times. In this context, the IP algorithm is positive against all of memory, running time and robustness against technical failure criteria.

3.2.4 A Hybrid Algorithm

Given the IP and IDP algorithms, a *hybrid algorithm* combining facets from both of these algorithms was developed in [57]. The hybrid algorithm represents the space of all coalition structures as a *integer partition graph* which is similar to the IP algorithm's representation of Π, except that some $g \in G$ are connected by edges. To be precise, an edge exists between two subspaces $\overline{g}, \overline{g}' \in \overline{G}$ if there exists $i_1, i_2 \in \overline{g}$ and $i_1 + i_2 \in \overline{g}'$ such that $\overline{g} \setminus \{i_1, i_2\} = \overline{g}' \setminus \{i_1 + i_2\}$. Figure 3.2.4 displays an integer partition graph for a system of six agents with $Ag = \{a_1, \ldots, a_6\}$.

Firstly, the hybrid algorithm employs the IDP algorithm to analyze the partitioning of coalitions of size $\{1, 2, \ldots, m-1, m\}$, where $m < n$ is arbitrarily chosen. For coalitions $C \subseteq Ag$ of size greater than m, their value is simply set as $f_2(C)$. Observe that, for a particular m value, it may be that the IDP algorithm does not analyze the coalition structures in a number of subspaces in the integer partition graph. For example, consider the system represented by integer graph in Figure 3.2.4. If $m = 2$ then, because the subspaces $\{2, 2, 2\}$ and $\{1, 2, 3\}$ are formed through partition coalitions of size greater than 2, the structures in these subspaces would not be analyzed by the IDP algorithm. Thus, were the optimal coalition structure in either of these subspaces, the IDP algorithm would not generate it. To this end, once the IDP algorithm has searched the other subspaces, the remaining subspaces are then searched as in the IP algorithm and an optimal coalition structure is generated.

Empirical results show that this approach is much faster than both of the individual algorithms. For example, given 25 agents, for $m \in [2, \ldots, 16]$, the hybrid algorithm requires no more 28% of the time required by the IP algorithm, whereas it requires no more than 0.3% of the time required by the IDP algorithm. In this context, with respect to the running time criterion, this hybrid algorithm improves on both the IDP and IP algorithms.

3.3 Summary

In this chapter, the state-of-the-art developments with respect to optimal coalition structure generation algorithms was presented. Generally, these algorithms can be divided into two types: those that consider *ex ante* information assumptions and those that consider *ex post* information assumptions.

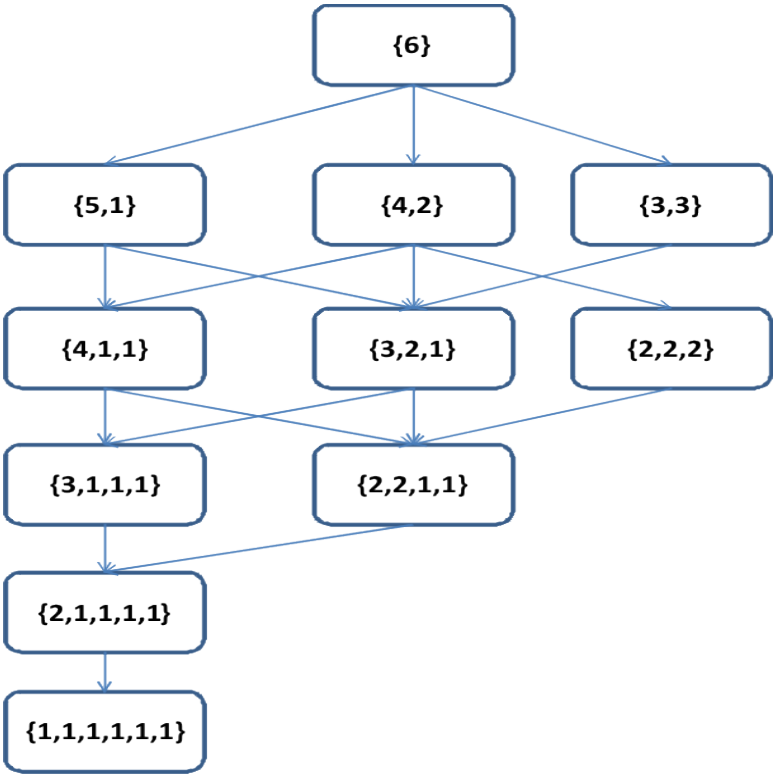


Figure 3.2.4: An integer partition graph for a six agent setting

In the former case, since it is assumed that no coalition values are stored in memory, the challenge is to develop anytime algorithms that can guarantee that the quality of solution monotonically increases with the running time of the algorithm. In contrast, in the latter case, since coalition values are stored in memory, the challenge is to develop algorithms that strike a useful balance between all of the worst case complexity, memory, running time and robustness against technical failure criteria.

Now, following on from the previous three chapters, each of which has described the existing research into coalition formation in multi-agent systems, in the subsequent three chapters the original state-of-the art contributions of this thesis are presented. In particular, in:

- Chapter 4, an optimal coalition structure generation algorithm which considers both coalition value calculation and optimal coalition structure generation processes is developed;
- Chapter 5, an optimal coalition structure generation algorithm which can efficiently generate an optimal coalition structure in partition function games is developed (the first algorithm to do this); and.
- Chapter 6, a novel cooperative representation that models coalition formation between self-interested agents is presented. It is shown that for certain, natural instances of this representation, problems concerning stability can be answered with polynomial time complexity.

Part II

Research Contributions

Chapter 4

Towards a Distributed Optimal Coalition Structure Generation Algorithm

As can be seen from the algorithms presented in Section 3.2, *ex ante* optimal coalition structure generation algorithms take, as input, all coalition values and, from this input, generate an optimal coalition structure. All of these algorithms assume that all coalition values have been computed and do not consider the computational process that are involved in computing all of them. This is surprising because, even for moderate numbers of agents, there are an exponential number of coalitions and the process of computing all coalition values is not trivial.

The state-of-the-art coalition value calculation algorithm is the so-called *distributed coalition value calculation* (DCVC) algorithm [56]. In this algorithm, the process of computing all coalition values is equally distributed among all of the agents in the system. To achieve this, the space of all coalitions is divided into $|Ag| = n$ lists $\mathcal{L}_1, \ldots, \mathcal{L}_n$ where list $\mathcal{L}_s$ contains all coalitions of size s. In particular, this algorithm satisfies the following criteria that are desirable for any distributed algorithm:

- Low communication complexity;
- No redundant computations;
- An equally balanced computational load among the agents; and,
- Minimal resource usage.

Since the amount of data transmitted among the agents, the number of calculations performed by the agents and the amount of resource used by the agents can affect the running time of the DCVC algorithm, minimizing both of these can improve the efficiency in which all coalition values are computed. Furthermore, as all agents are assumed to have equal computational abilities, the third criterion ensures that the computational load is balanced in a way that also improves the efficiency in which all coalition values are computed. Thus, these four criteria ensure that the running time of the DCVC algorithm is as quick as possible.

Now, recall from Section 3.2.2 that, for $s = 1, \ldots, n$, one of the first steps of the IP optimal coalition structure generation algorithm is to compute the maximum, minimum and average values of all coalitions in $\mathcal{L}_s$. In this context, the output from the DCVC algorithm is an ideal input to the IP algorithm. However, since the computational processes in the DCVC algorithm are distributed among all of the agents in the system whereas in the IP algorithm they are coordinated by a single entity, connecting the two algorithms is not trivial. For instance, once all values have been computed, every individual agent must transfer all the coalition values they have computed to a single entity. Consequently, as each agent computes approximately $\lfloor \frac{2^n-1}{n} \rfloor$ coalition values in the DCVC stage, transmitting all these values may result in high communication complexity which, following the above discussions, is undesirable.

In this chapter, an optimal coalition structure generation algorithm is developed which consists of a two stage process. In the first stage, coalition values are calculated and, in the second stage, an optimal coalition structure is generated. This algorithm is based on the sequential application of the DCVC and IP algorithms and, for this algorithm, pre-processing techniques are developed that can be incorporated into both the coalition value calculation and optimal coalition structure generation stages. These techniques are represented as *filter rules* that identify all coalitions that cannot belong to an optimal structure. Upon doing this, an appropriate action is then performed. Typically, this involves filtering coalition values from the input or avoiding all structures in which these coalitions are embedded.

These filter rules are important for two reasons. Firstly, they can reduce the number of coalition values an individual agent needs to transfer after completing their computations and, secondly, they can reduce the number of coalition structures that need be analyzed by the IP algorithm. Secondly, following previous discussions, these filter rules may be useful foundations from which a distributed optimal coalition structure generation algorithm can be developed for a system of fully cooperative agents.

The rest of this chapter goes as follows:

- In Section 4.1, the *distributed coalition value calculation* algorithm is formally presented;
- In Section 4.3, the filter rules, as well as the intuition and theory behind them, are formally presented;
- In Section 4.4, an optimal coalition structure generation algorithm is presented which describes the sequential application of the DCVC and IP algorithms, particularly focusing on how the filter rules are incorporated into both these algorithms; and,
- In Section 4.5, the effectiveness of the filter rules is empirically tested for normally and uniformly distributed coalition values. Empirical results show that the filter rules can greatly reduce the communication load between the DCVC and IP stages for both of these distributions. Furthermore, these results indicate that the filter rules can also greatly reduce the overall running time of sequential application of the DCVC and IP algorithms, especially for normally distributed values, where filter rules can offer an exponential improvement in running time.

4.1 The Distributed Coalition Value Calculation Algorithm

The *distributed coalition value calculation* algorithm (developed by the authors of [56]) efficiently computes all coalition values by distributing the computational processes among all of the agents. Fundamental to this algorithm are the way in which the space of all coalitions is represented and the way in which all computational processes are distributed among the agents.

4.1.1 Representing the Space of all Coalitions

As mentioned in the introduction, the space of all coalitions is divided into n lists $\mathcal{L}_1, \ldots, \mathcal{L}_n$ where list $\mathcal{L}_s$ contains all coalitions of size s. Within each coalition, the agents are ordered non-decreasingly with respect to the value of their indices whereas the coalitions are ordered in each $\mathcal{L}_s$ so that:

- The first coalition in the list is: $\{a_{n-s+1}, \ldots, a_{n-1}, a_n\}$;
- The last coalition in the list is: $\{a_1, \ldots, a_{s-1}, a_s\}$; and,
- The coalition occupying the $(j-1)^{th}$ place in list $\mathcal{L}_s$ can be generated from the coalition occupying the j^{th} place.

$\mathcal{L}_1$	$\mathcal{L}_2$	$\mathcal{L}_3$	$\mathcal{L}_4$	$\mathcal{L}_5$	$\mathcal{L}_6$
$\{a_6\}$	$\{a_5, a_6\}$	$\{a_4, a_5, a_6\}$	$\{a_3, a_4, a_5, a_6\}$	$\{a_2, a_3, a_4, a_5, a_6\}$	$\{a_1, a_2, a_3, a_4, a_5, a_6\}$
$\{a_5\}$	$\{a_4, a_6\}$	$\{a_3, a_5, a_6\}$	$\{a_2, a_4, a_5, a_6\}$	$\{a_1, a_3, a_4, a_5, a_6\}$	
$\{a_4\}$	$\{a_4, a_5\}$	$\{a_3, a_4, a_6\}$	$\{a_2, a_3, a_5, a_6\}$	$\{a_1, a_2, a_4, a_5, a_6\}$	
$\{a_3\}$	$\{a_3, a_6\}$	$\{a_3, a_4, a_5\}$	$\{a_2, a_3, a_4, a_6\}$	$\{a_1, a_2, a_3, a_5, a_6\}$	
$\{a_2\}$	$\{a_3, a_5\}$	$\{a_2, a_5, a_6\}$	$\{a_2, a_3, a_4, a_5\}$	$\{a_1, a_2, a_3, a_4, a_5\}$	
$\{a_1\}$	$\{a_3, a_4\}$	$\{a_2, a_4, a_6\}$	$\{a_1, a_4, a_5, a_6\}$	$\{a_2, a_3, a_4, a_5, a_6\}$	
	$\{a_2, a_6\}$	$\{a_2, a_4, a_5\}$	$\{a_1, a_3, a_5, a_6\}$		
	$\{a_2, a_5\}$	$\{a_2, a_3, a_6\}$	$\{a_1, a_3, a_4, a_6\}$		
	$\{a_2, a_4\}$	$\{a_2, a_3, a_5\}$	$\{a_1, a_3, a_4, a_5\}$		
	$\{a_2, a_3\}$	$\{a_2, a_3, a_4\}$	$\{a_1, a_2, a_5, a_6\}$		
	$\{a_1, a_6\}$	$\{a_1, a_5, a_6\}$	$\{a_1, a_2, a_4, a_6\}$		
	$\{a_1, a_5\}$	$\{a_1, a_4, a_6\}$	$\{a_1, a_2, a_4, a_5\}$		
	$\{a_1, a_4\}$	$\{a_1, a_4, a_5\}$	$\{a_1, a_2, a_3, a_6\}$		
	$\{a_1, a_3\}$	$\{a_1, a_3, a_6\}$	$\{a_1, a_2, a_3, a_5\}$		
	$\{a_1, a_2\}$	$\{a_1, a_3, a_5\}$	$\{a_1, a_2, a_3, a_4\}$		
		$\{a_1, a_3, a_4\}$			
		$\{a_1, a_2, a_6\}$			
		$\{a_1, a_2, a_5\}$			
		$\{a_1, a_2, a_4\}$			
		$\{a_1, a_2, a_3\}$			

Figure 4.1.1: Space of coalitions for a system of six agents $Ag = \{a_1, \ldots, a_6\}$

In more detail, suppose coalition $C_j = \{a_{l_1}, \ldots, a_{l_s}\}$ occupies the j^{th} place in $\mathcal{L}_s$. The agents can generate the coalition which occupies the $(j-1)^{th}$ place in $\mathcal{L}_s$ (C_{j-1}) by checking the indices of the agents in C_j. Specifically, for $x = s, s-1, \ldots, 2, 1$, the indices of the agents in C_j are sequentially analyzed until an index value l_x is found that is less than the index value of the x^{th} agent in the first coalition (C_1) in $\mathcal{L}_s$. When this index value l_x is found then C_{j-1} is generated from C_j as follows:

1 For $k \in [1, x)$, the index value of the k^{th} agent in C_{j-1} is set to equal the index value of the k^{th} agent in C_j;

2 For $k = x$, the index value of the of the k^{th} agent in C_{j-1} is set to $l_x + 1$, *i.e.*, it is set to the index value of the k^{th} agent in C_j plus 1; and,

3 For $k \in (x, s]$, the index value of the of the k^{th} agent in C_{j-1} is set to equal the value of the index value of the $k - 1^{th}$ agent in C_{j-1} plus 1.

For example, consider $\mathcal{L}_3$ in a system of six agents $Ag = \{a_1, \ldots, a_6\}$. In this list, the first coalition is $C_1 = \{a_4, a_5, a_6\}$, whereas the last coalition is $C_{|\mathcal{L}_3|=20} = \{a_1, a_2, a_3\}$. Coalition C_{19} can be generated from C_{20} as follows.

Observe that the index value of the index of the third agent in C_{20} is less than the index value of the third agent in C_1. Consequently, the index value of the first two agents in C_{19} are set to equal the index value of the first two agents in C_{20}. Also, the index value of the third agent in C_{19} is set to equal the index value of the third agent in C_{20} plus one. Therefore, $C_{19} = \{a_1, a_2, a_4\}$. In this way, given only the first and last coalitions, each agent can incrementally construct every $\mathcal{L}_s$. Figure 4.1.1 displays the space of all coalitions for a system of six agents $Ag = \{a_1, \ldots, a_6\}$.

Ordering the coalitions in this manner means that a coalition can be determined from the place it occupies in $\mathcal{L}_s$. Thus, each agent need only maintain coalition values and not both the coalition and its value in memory.

4.1.2 Computing The Coalition Values (basic approach)

Given the space of all coalitions, observe that in $\mathcal{L}_1$ there are exactly n coalitions. This means that there is exactly one coalition value for each agent to compute. Therefore, in the first step of this algorithm, every agent $a_i \in Ag$ computes the value of the i^{th} coalition $C_i \in \mathcal{L}_1$.

In the subsequent steps of this algorithm, every agent $a_i \in Ag$ sequentially computes the values of the coalitions in their share of the coalitions in the lists $\mathcal{L}_2, \ldots, \mathcal{L}_{n-1}$ (denoted $\mathcal{L}_{s,i}$). To achieve this, for $s = 2, \ldots, n-1$, every agent a_i begins by first computing,

$$|\mathcal{L}_{s,i}| = \lfloor \frac{|\mathcal{L}_s|}{n} \rfloor.$$

Upon doing this, they then compute the location of the last coalition which belongs to $\mathcal{L}_{s,i}$. This is done by computing $\text{index}_{s,i}$, where,

$$\text{index}_{s,i} = i \times |\mathcal{L}_{s,i}|.$$

Using the procedure described in the previous section, every agent a_i then sequentially generates the coalition located at $\text{index}_{s,i}$ in list $\mathcal{L}_s$, as well as the $|\mathcal{L}_{s,i}| - 1$ coalitions ordered above this coalition. After generating each coalition, its value is then computed. In more detail, a_i starts by setting M to be the last coalition in $\mathcal{L}_{s,i}$ (*i.e.*, to the coalition located at $\text{index}_{s,i}$) and calculates its value. After that, a_i then sets M to the coalition before it (*i.e.*, to the coalition located at $\text{index}_{s,i} - 1$) and calculates its value. This is repeated until the value of every coalition in $\mathcal{L}_{s,i}$ has been calculated.

Of course, it may be that the number of coalitions in $\mathcal{L}_s$ is not exactly divisible by n. Thus, there may be a number of additional value in $\mathcal{L}_s$ that have to be computed. To calculate these values, every agent computes the number of left over values as follows,

$$|\mathcal{L}'_s| = |\mathcal{L}_s| - (n \times |\mathcal{L}_{s,i}|).$$

Thus far, every agent has calculated the same number of values in $\mathcal{L}_s$. Therefore, to equally balance the computational processes among the agents, each of these left over values should be calculated by a different agent. To achieve this, a sequence of $|\mathcal{L}'_s|$ agents A' calculate the coalition values in $\mathcal{L}'_s$ and the set A' is updated after the values in every $\mathcal{L}'_s$ have been computed. Specifically, this updating is performed by maintaining a value α, initially set to 1, such that, for any list $\mathcal{L}_s$, if $|\mathcal{L}'_s| > 0$ then A' would contain $|\mathcal{L}'_s|$ agents beginning with agent a_α. In more detail, α is updated, such that, if,

$$\alpha + |\mathcal{L}'_s| < n,$$

then,

$$\alpha = \alpha + |\mathcal{L}'_s|.$$

Otherwise, if $\alpha + |\mathcal{L}'_s| \geq n$ then,

$$\alpha = \alpha + |\mathcal{L}'_s| - n.$$

Given this update, A' is constructed, such that, if,

$$\alpha + |\mathcal{L}'_s| - 1 < n$$

then,

$$A' = \{a_\alpha, a_{\alpha+1}, \ldots, a_{\alpha+|\mathcal{L}'_s|-1}\}.$$

- Set $\alpha := 1$;
- Sequentially, for $s = 1, \ldots, n-1$:
  - Compute $|\mathcal{L}_{s,i}|$;
  - Compute index$_{s,i}$;
  - Generate the remaining coalitions in $\mathcal{L}_{s,i}$ and compute their values;
  - Compute $|\mathcal{L}'_s|$;
  - If $|\mathcal{L}'_s| > 0$ then generate A' and if $a_i \in A'$ then generate and compute the value of the coalition allocated to them in $\mathcal{L}'_s$; and,
  - Update α.
- Upon executing the above, for $s = n$, agent indexed a_α computes $v(Ag)$.

Algorithm 4.1.2: Actions for every $a_i \in Ag$ in the 'basic' DCVC algorithm

Otherwise, if $\alpha + |\mathcal{L}'_s| - 1 \geq n$ then,

$$A' = \{a_\alpha, a_{\alpha+1}, \ldots, a_n, a_1, \ldots, a_{\alpha+|\mathcal{L}'_s|-n}\}.$$

For notation, if a_i is allocated a coalition in $\mathcal{L}'_s$ then their share of this distribution is denoted by $\mathcal{L}'_{s,i}$ from now onward.

Finally, when the coalition values in lists $\mathcal{L}_1, \ldots, \mathcal{L}_{n-1}$ have been computed, the agent indexed by α computes the value of the grand coalition. In this way, efforts are undertaken to ensure that the computational processes are equally distributed among the agents.

4.1.3 Refinements to The Basic Approach

It should be noted that the refinements presented in this section were developed by the authors of [56] and are not an original contribution of this thesis. Observe that the distribution process specified in Section 4.1.2 does not take into consideration the time required for every agent to set M from one coalition to another. Specifically, after an agent calculates the value of a coalition, it needs to set M to the coalition ordered above it. Given coalition C_j in $\mathcal{L}_s$, generating C_{j-1} from C_j will require no more than:

a. s agent index comparisons with the agents in C_j; and,

b. s additions to the index values in C_j.

Consequently, in total, no more than $2s$ operations are involved in updating M. Now (ignoring the left over coalitions), in this distribution, for every list $\mathcal{L}_s$, the agents with low index values compute the values of those coalitions that are located high up $\mathcal{L}_s$. Due to the ordering of the coalitions in each list, as well as the agents in each coalition, this means that these agents may execute more operations updating M than the agents with higher index values. Furthermore, this difference grows as the number of agents grows and, although every agent computes the same number of values, it may be that they all finish at different times.

To circumvent this problem and ensure that the agents complete their calculations at approximately the same time, the authors in [56] refine the distribution of coalitions so that, for $s = 1, \ldots, n$, each agent $a_i \in Ag$ is allocated exactly two sub lists: $\mathcal{L}^1_{s,i}$ and $\mathcal{L}^2_{s,i}$, where each sublist is located at different positions within $\mathcal{L}_s$. These sublists are generated as in the basic approach. First, the size of each sublist is computed as follows:

- $|\mathcal{L}^1_{s,i}| = \lfloor |\mathcal{L}_{s,i}| \times 0.4 \rfloor$; and,
- $|\mathcal{L}^2_{s,i}| = \lceil |\mathcal{L}_{s,i}| \times 0.6 \rceil$.

It should be noted that the fractions 0.4 and 0.6 were found to be the best ratios *via* experiments undertaken by the authors in [56]. Upon computing the sizes of the two sublists, the index of the last coalition in each sublist is computed as follows:

- $\text{index}^1_{s,i} = i \times |\mathcal{L}^1_{s,i}|$; and,
- $\text{index}^2_{s,i} = |\mathcal{L}_s| - |\mathcal{L}'_s| - ((i-1) \times |\mathcal{L}^2_{s,i}|)$.

The remaining 'left over' coalitions in these lists are then allocated to the agents as in the 'basic' algorithm. Figure 4.1.3 displays this refined allocation for a system of six agents.

Now, the following assumptions are inherent to the algorithm presented thus far:

(i) All agents have equal computational abilities; and,

(ii) The system does not dynamically change.

When these assumptions do not hold, the basic approach was refined in [56] so that all coalition values can be computed when these assumptions are not valid. However, in this chapter, it is assumed that all agents have equal computational abilities and that the system does not dynamically change and so, for this reason, these refinements are not considered.

Once all coalition values have been calculated, they can be used to generate an optimal coalition structure. To this end, in the next section, filter rules are presented which can be incorporated into both the coalition value calculation and optimal coalition structure generation stages to identify coalitions that cannot belong to an optimal coalition structure. Removing these coalitions can potentially reduce the number of coalition structures that can be analyzed by the IP algorithm.

4.2 The Integer Partition (IP) Algorithm

To refresh the mind of the reader, the main points of the IP algorithm are presented. Recall from Section 3.2.3 in Chapter 3 that the space of all coalition structures is represented by $\overline{G}$ which denoted the set of all integer partitions of n (the number of agents). For example, the integer partitions of $n = 5$ are: $\{5\}$, $\{1,4\}$, $\{2,3\}$, $\{1,1,3\}$, $\{1,2,2\}$, $\{1,1,1,2\}$ and $\{1,1,1,1,1\}$. Here, each integer k in every partition can be interpreted as a coalition of size k. For example, $\{1,4\}$ represents the space of all coalition structures that consist of exactly one coalition of size 1 and exactly one coalition of size 4. If $\overline{G}$ denotes the space of integer partitions of n then the space of all coalition structures is partitioned into subspaces using the mapping $F : \Pi \rightarrow \overline{G}$.

From the input of all coalition values, it is possible to compute the maximal and minimal values of all the structures in every subspace $g \in \overline{G}$. This is achieved by computing basic statistical information on all coalitions of the same size. Specifically, for $i = 1, \ldots, n$, the maximum, minimum and average values of all coalitions of size i are computed. Let these be denoted as max_i, min_i and av_i hereafter. From this data, for each subspace $\overline{g} \in \overline{G}$:

- The maximal coalition structure value, referred to as the *upper bound* of $\overline{g}$, (denoted $UB_{\overline{g}}$) is set as $\sum_{\forall i \in \overline{g}} max_i$;
- The minimal coalition structure value, referred to as the *lower bound* of $\overline{g}$, (denoted $LB_{\overline{g}}$) is set as $\sum_{\forall i \in \overline{g}} min_i$; and,

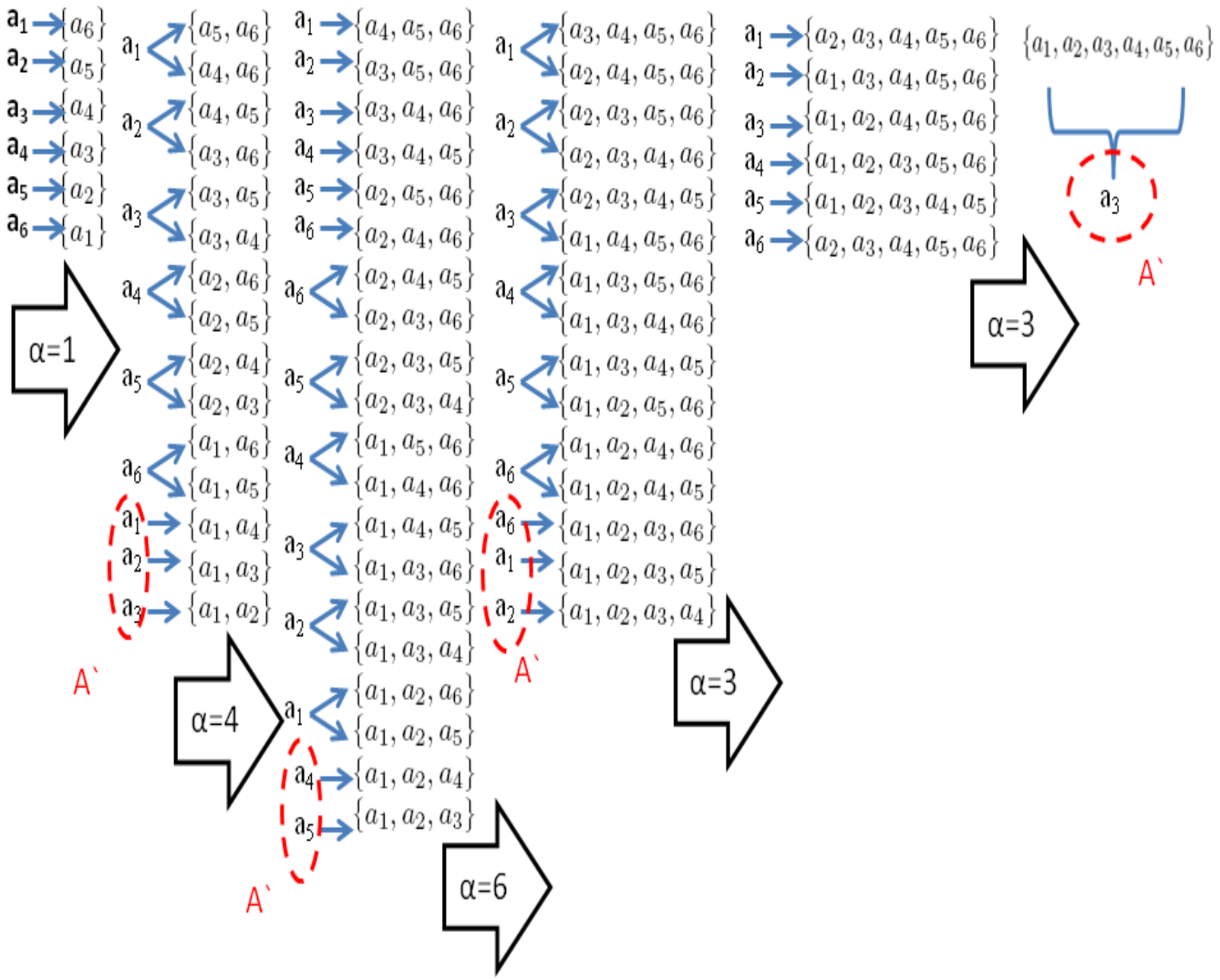


Figure 4.1.3: Refined DCVC allocation

- The average coalition structure value, referred to as the *average* bound of $\overline{g}$, (denoted $Av_{\overline{g}}$).

From these values, the maximal,minimal and average coalition structure values in each subspace $\overline{g} \in \overline{G}$ can be computed through summing the maximal, minimal and average coalition values in $\overline{g}$.

(

Now, $\forall \overline{g} \in \overline{G}$, given $UB_{\overline{g}}$, $LB_{\overline{g}}$ and $AV_{\overline{g}}$, the IP algorithm computes both the upper and lower values for $\overline{G}$ (denoted $UB_{\overline{G}}$ and $LB_{\overline{G}}$, respectively) as follows:

- $UB_{\overline{G}}$ = arg max$_{\overline{g} \in \overline{G}}$ $UB_{\overline{g}}$; and,
- $LB_{\overline{G}}$ = arg max$_{\overline{g} \in \overline{G}}$ $AV_{\overline{g}}$.

Once these bounds have been computed, the first pre-processing technique employed by the algorithm is to prune away all subspaces of structures $\overline{g} \in \overline{G}$ such that,

$$UB_{\overline{g}} < LB_{\overline{G}}.$$

Obviously, if $UB_{\overline{g}} < LB_{\overline{G}}$ then the value of every structure in $\overline{g}$ cannot be optimal and so every coalition structure in $\overline{g}$ is not analyzed. After pruning all $\overline{g}$ from $\overline{G}$, the values of all coalition structures in the subspace $\overline{g}^*$, where $UB_{\overline{G}} = UB_{\overline{g}^*}$, are then computed.

While computing the values of the coalition structures that belong to a particular subspace, methods are undertaken to ensure that the values of only valid (*i.e.*, not overlapping) coalition structures are computed. Similarly, methods are used to avoid redundant computations, such as computing the value of same structure multiple times (*e.g.*, computing $v(\pi_\alpha = \{\{a_1, a_2\}\{a_3, a_4\}\})$ and $v(\pi_\beta = \{\{a_3, a_4\}\{a_1, a_2\}\})$).

Specifically, suppose that $\overline{g}^* = \{\overline{g}_{i_1}, \ldots, \overline{g}_{i_k}\}$ is the subspace in $\overline{G}$ such that $UB_{\overline{g}^*} = UB_{\overline{G}}$ and, for $j = 1, \ldots, k$, let:

- $\underline{A}_j$ be the set of all coalitions of agents of size j, all ordered non-decreasingly with respect to the indices of the agents who belong to them; and,
- $\mathbf{M}_{i_j} : |\mathbf{M}_{i_j}| = \overline{g}_{i_j}$ be a temporary array that can be used to cycle through all possible coalitions which could belong to $\overline{g}_{i_j}$.

At the start of the subspace search, $\mathbf{M}_{i_1}$ is assigned to a coalition of size $\overline{g}_{i_1}$ in $\underline{A}_{i_1}$. Given this coalition, $\mathbf{M}_{i_2}$ then cycles through all coalitions of size $\overline{g}_{i_2}$ in $\underline{A}_{i_2}$ until a coalition that does not overlap with the coalition in $\mathbf{M}_{i_1}$ is found. After that, $\mathbf{M}_{i_3}$ is then used to cycle through all coalitions of size $\overline{g}_{i_3}$ in in $\underline{A}_{i_3}$ until a coalition that does not overlap with both of those in $\mathbf{M}_{i_1}$ and $\mathbf{M}_{i_2}$ is found. This is repeated until all of $\mathbf{M}_{i_1}$ to $\mathbf{M}_{i_k}$ are assigned disjoint coalitions. At this point, the value of this coalition structure is then calculated and compared with the maximum value found so far. After that, the coalitions in $\mathbf{M}_{i_1}$, ..., $\mathbf{M}_{i_k}$ are updated so as to compute the value of a different structure in $\overline{g}^*$.

Additionally, whilst computing the structure values in $\overline{g}^*$, pre-processing is also applied in order to reduce the number of coalition structure values which are computed.

Lemma 4.1 *Suppose $\overline{g}^* = \{\overline{g}_{i_1}, \ldots, \overline{g}_{i_k}\}$ and, suppose that, for $j \in [1, \ldots, k)$, $\boldsymbol{M}_{i_1}$ to $\boldsymbol{M}_{i_j}$ have been assigned coalitions $C_{i_1} \ldots C_{i_j}$, respectively. If,*

$$v(C_{i_1}) + \ldots + v(C_{i_j}) + UB_{\overline{g}_{i_{j+1}}} + \ldots + UB_{\overline{g}_{i_k}} < LB_{\overline{G}},$$

then all structures $\pi \in \overline{g}^*$ such that $C_{i_1} \ldots C_{i_j} \subseteq \pi$ cannot be optimal.

This branch and bound rule employs pre-processing to identify groups of disjoint coalitions which cannot belong to an optimal structure in $\overline{g}^*$. By avoiding them, this pre-processing ensures that the values of structures which cannot be optimal are not computed.

Whilst analyzing the coalition structures in $\overline{g}^*$, if a structure $\pi^* \in \overline{g}^*$ is found such that $v(\pi^*) = UB_{\overline{G}}$ then this is output as the optimal coalition structure. However, as $UB_{\overline{G}}$ is not a tight bound, this coalition structure may not exist. Consequently, if no such structure is found, if $v(\pi^*_n)$ denotes the optimal value after computing all the coalition values in $\overline{g}^*$ then the system is updated as follows:

- $LB_{\overline{g}}$ is updated so that $LB_{\overline{G}}$ = arg $\max_{\overline{g} \in \overline{G}, \pi \in \Pi} (Av_{\overline{g}}, v(\pi^*_n))$;
- All $\overline{g} \in \overline{G}$ such that $UB_{\overline{g}} < LB_{\overline{G}}$ are pruned away; and,
- $UB_{\overline{G}}$ is updated so that it is the maximal of $v(\pi^*_n)$ or the maximal $UB_{\overline{g}'}$ $\forall \overline{g}' \in \overline{G} \setminus \{\overline{g}^*\}$.

The coalition structures in the next most promising subspace are then analyzed in the same way and, after doing this, the system is subsequently updated. This continues until either a structure is found with value equal to the system upper bound or all of the space has been searched.

4.3 Filter Rules

Following on from the existing DCVC and IP algorithms, in the remaining sections of this chapter, a novel optimal coalition structure generation algorithm, which considers both coalition value calculation and optimal coalition structure generation processes. Inherent to this algorithm are filter rules which identify coalitions that cannot belong to an optimal coalition structure and, in this section, these filter rules are presented.

For notation, the space of all coalitions will be denoted by $\mathcal{F}$. Those coalitions that meet the requirements of the filter rules and, therefore, definitely do not belong to an optimal structure will be denoted by $\mathcal{F}_{np} \subseteq \mathcal{F}$. On the other hand, those coalitions which do not meet the requirements of any of the filter rules and, therefore, may belong to an optimal structure will be denoted by $\mathcal{F}_p \subseteq \mathcal{F}$. Obviously, $\mathcal{F}_p \cap \mathcal{F}_{np} = \emptyset$. Initially, before the filter rules are applied, it is assumed that all coalitions may belong to an optimal structure, *i.e.*, $\mathcal{F}_p = \mathcal{F}$.

Firstly, consider the following theorem.

Theorem 4.1 *Consider any coalition $C \subseteq Ag$ where the agents in C can be partitioned into k coalitions $C_1, \ldots, C_k$. If,*

$$v(C_1) + \ldots + v(C_k) > v(C)$$

then,

$$\forall \pi : (C; \pi) \in \mathcal{E}, \pi \text{ is not optimal.}$$

Proof: Consider any coalition $C \subseteq Ag$. If the agents in C can be partitioned into k coalitions $C_1, \ldots, C_k$ such that $v(C_1) + \ldots + v(C_k) > v(C)$ then, clearly, for every structure $\pi \in \Pi$ such that $C \in \pi$, the corresponding structure $\pi' = \pi \setminus \{C\} \cup \{C_1 \cup \ldots \cup C_k\}$ has value greater than the value of π. Therefore, all $\pi \in \Pi$ such that $(C; \pi) \in \mathcal{E}$ cannot be optimal. ∎

Recall from Section 3.2.1 that the dynamic programming algorithm determines if a given coalition $C \subseteq Ag$ can belong to an optimal structure by comparing the value of C with the combined value of all partitions $C_1, \ldots, C_k$ such that $k = 2$. However, in the DCVC algorithm, since every agent only knows a fraction of all coalition values, to do this for every coalition, it may be that every agent has to transfer all values they

have computed to the other agents in the system and, against **D1**, this may not minimize the communication complexity. To this end, a more natural approach is to compare $v(C)$ with the value:

$$\sum_{a_i \in C} v(\{a_i\}).$$

This would only require each agent to transfer the single value they computed in $\mathcal{L}_1$ and, consequently, the following filter rule is introduced.

Definition 4.1 (FR1) *If the value of a coalition C is smaller than the combined value of single agents who belong to C then this coalition is said to be unpromising,* i.e.*,*

if,

$$\sum_{a_i \in C} v(\{a_i\}) > v(C),$$

then,

$$F_p = F_p \setminus \{C\} \text{ and } F_{np} = F_{np} \cup \{C\}.$$

Theorem 4.1 can be extended so that it can be applied to collections of coalitions which have been grouped together with respect to some criteria. Due to the way in which the space of all coalitions are represented, a natural criterion is coalition size.

Theorem 4.2 *Consider any coalition $C \in \mathcal{L}_s$, as well as any integer partition $p = s_{i_1}, \ldots, s_{i_k}$ of the value s. If the sum of the smallest coalition values in $\mathcal{L}_{s_{i_1}}, \ldots, \mathcal{L}_{s_{i_k}}$ (denoted $d_s(p)$) is strictly greater than $v(C)$ then C cannot belong to an optimal coalition structure. More formally, if,*

$$d_s(p) := \sum_{j=1}^{k} arg\ min_{C' \in \mathcal{L}_{s_{i_j}}} v(C') > v(C),$$

then,

all $\pi \in \Pi : (C; \pi) \in \mathcal{E}$ cannot be optimal.

Proof: Consider any coalition structure π which contains coalition C. If the sum of the smallest values in $\mathcal{L}_{s_{i_1}}, \ldots, \mathcal{L}_{s_{i_k}}$ is greater than $v(C)$ then, clearly, for all disjoint coalitions $C_{s_{i_l}}, \ldots, C_{s_{i_k}}$ in $\mathcal{L}_{i_l}, \ldots, \mathcal{L}_{i_k}$ such that $\cup_{j=1}^{k} C_{s_{i_j}} = C$, it must be that,

$$v(C_{s_{i_1}}) + \ldots + v(C_{s_{i_k}}) > v(C).$$

Therefore, against Theorem 4.1, all π such that $(C; \pi) \in \mathcal{E}$ cannot optimal. ∎

Let $P(s)$ denote the set containing the value $d_s(p)$ for all partitions p of the value s. Following Theorem 4.2, it is possible to compare every coalition C of size s with every $d_s(p)$ value and immediately disregard those for which $v(C) < d_s(p)$. However, with regard to reducing the number of redundant computation, this is not desirable. Instead, there is no need to apply Theorem 4.2 to all partitions in $P(s)$ but only to $d_s(p)_{max}$ which is maximal in $P(s)$.

Obviously, if,

$$v(C) > d_s(p)_{max}$$

then,

$$v(C) > d_s(p'), \forall d_s(p') \in P(s) \setminus \{d_s(p)_{max}\}.$$

On the other hand, if,

$$\exists d_s(p') \in P(s) \setminus \{d_s(p)_{max}\} \text{ such that } v(C) < d_s(p'),$$

then,

$$v(C) < d_s(p)_{max}.$$

In this context, comparing $v(C)$ with the single $d_s(p')_{max}$ value can determine whether C belongs to an optimal structure whilst also minimizing the computational resource that is used. This maximal value will be referred to as the *domination value* of all coalitions of size s and is formally defined as follows.

Definition 4.2 *For any set containing coalitions of size s and, for every integer partition p of the value s, the* domination value $\tilde{d}_s$ *is value of the partition with the biggest $d_s(p)$ value in $P(s)$,* i.e.,

$$\tilde{d}_s = arg\ max_{d_s(p) \in P(s)} d_s(p).$$

Definition 4.3 (FR2) *If the value of any coalition C in list $\mathcal{L}_s$ is smaller than $\tilde{d}_s$ then this coalition is said to be unpromising,* i.e.,

if,

$$\tilde{d}_s > v(C)$$

then,

$$F_p = F_p \setminus \{C\} \text{ and } F_{np} = F_{np} \cup \{C\}.$$

Recall that, with respect to filtering, the IP algorithm already employs a branch and bound filter rule when computing the values of coalition structures in a particular subspace in $\overline{G}$. Specifically, to decide whether any partial structure $\{C_{m_1}, ..., C_{m_l}\}$ can all belong to an optimal structure in a promising subspace $\overline{g}^*$, a branch and bound (**B & B**) filter rule is employed which is based on the following proposition.

Proposition 4.1 *Given a subspace $\overline{g}^* = \{s_1, \ldots, s_k\}$, as well as the coalitions $C_{m_1} \in \mathcal{L}_{s_1}, \ldots, C_{m_l} \in \mathcal{L}_{s_l}$, where $l < k$, if,*

$$\sum_{j=1}^{l} v(C_{m_j}) + \sum_{j=l+1}^{k} arg\ max_{C \in \mathcal{L}_{s_j}} v(C) \leq v(\pi^*_N)$$

then,

all $\pi \in \overline{g}^*$ such that $\{C_{m_1}, ..., C_{m_l}\} \subseteq \pi$ cannot be optimal.

Intuitively, for any $\{C_{m_1}, ..., C_{m_l}\}$, if the combined value of these coalitions ($\sum_{i=1}^{l} v(C_{m_i})$) plus the value of the sum of the maximum coalition values in the remaining sets $\mathcal{L}_{s_{l+1}}, \ldots, \mathcal{L}_{s_k}$ is less than the current optimal value $v(\pi^*_N)$ then no structures to which contain all of $C_{m_1}, ..., C_{m_l}$ can be optimal. Thus, this filter rule highlights all such coalitions and does not analyze any structure which contains all of them in the optimal coalition structure generation process. A restricted version of this filter rule, which can be incorporated into the DCVC stage of this algorithm, is presented in Definition 4.4.

Definition 4.4 ***[FR3]*** *For a given subspace $\overline{g} = \{s_1, \ldots, s_k\}$ and for $j \in [1, \ldots, k]$, a coalition $C \in \mathcal{L}_{s_j}$ is unpromising if the value of this coalition plus the maximum values of the coalitions in the remaining lists is less than the value of the current optimal,* i.e.,
if $\exists C \in \mathcal{L}_{s_j}$, such that,

$$v(C) + \sum_{q=1}^{j-1} arg\ max_{C' \in \mathcal{L}_q} v(C') + \sum_{q=j+1}^{k} arg\ max_{C'' \in \mathcal{L}_q} v(C'') \leq v(\pi^*_N),$$

then,

$$F_p = F_p \setminus \{C\} \text{ and } F_{np} = F_{np} \cup \{C\}.$$

Intuitively, all coalition values which satisfy **FR1**, **FR2** and **FR3** do not have to be transferred from the DCVC to the IP stage. To this end, in the next section, a novel optimal coalition structure algorithm, which considers both coalition value calculation and optimal coalition structure generation processes, is presented. This consists of the sequential application of the DCVC and IP algorithms, combined with the filter rules.

4.4 An Optimal Coalition Structure Generation Algorithm

In this section, the application of the filter rules in a sequential execution of the DCVC and IP algorithms is considered. As is consistent with the optimal coalition structure generation algorithms presented in Chapter 3, it is assumed that the system is closed and that the number of agents and coalition values do not dynamically change. Furthermore, it is also assumed that every agent has equal computational abilities and so, with respect to the DCVC algorithm, it is assumed that the values are calculated as in the 'basic' approach but that the coalitions are allocated as in Figure 4.1.3, *i.e.*, (excluding 'left over' coalitions) each agent a_i receives up to two subsets of values to calculate within each list.

4.4.1 Assumptions About Data Transmission Among Agents

In this procedure, before optimal coalition structure generation, agents will have to transmit data among themselves. For the purposes of this procedure, it is assumed that the agents transmit the data by recording it in a common data structure, to which every agent has unrestricted access. To this end, when referring to the agents transmitting data among themselves, it is in the context of agents recording data in the common structure. Consequently, with respect to transferring data, the following assumptions hold for this algorithm:

- It is faster to transmit coalition values than to calculate them;
- Every agent is able to transmit data simultaneously to all the other agents in the system; and,
- Every agent is able to receive and transmit data at the same time.

A number of transmission protocols are robust against various real world factors that can affect the transfer of data, such as data corruption or data loss. However, since the focus of this work is concerned with how the filter rules can offer computational improvements with respect to optimal coalition structure generation, the processes in which the agents transfer data are not considered in this thesis and are left for future work.

4.4.2 Application of Filter Rules in DCVC Stage

The agents begin this algorithm as they begin the DCVC algorithm, *i.e.*, every agent a_i calculates the value of the i^{th} coalition in list $\mathcal{L}_1$. However, this time, upon calculating this value, every agent then transmits it to all of the other agents in the system. By doing so, every agent is then able to:

A. Execute **FR1** immediately after computing the value of every coalition allocated to them in $\mathcal{L}_2, \ldots, \mathcal{L}_n$; and,

B. Immediately after executing **FR1**, for every coalition $C \notin \digamma_{np}$ allocated to them, compute the value of all structures $\{C, \cup_{a_j \notin C}\{a_j\}\}$.

The agents then proceed to compute the remaining coalition values in $\mathcal{L}_2, \ldots, \mathcal{L}_n$ as they would in the DCVC algorithm, executing **A** and **B** after every value has been computed. In this context, **B** ensures that part of the optimal coalition structure generation process is distributed among the agents. The integer partition graph in Figure 4.4.2 displays the space of all coalition structures whose values are computed in the coalition value calculation stage for a system of six agents.

As well as executing **A** and **B**, after computing the coalition values in each of the two subsets of coalitions assigned to them in each sublist, the agents record the maximum, minimum and average values of all

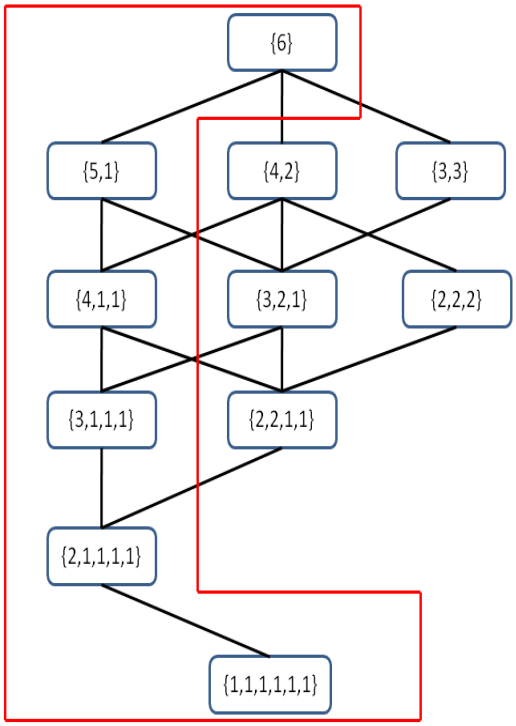


Figure 4.4.2: An integer partition graph representing the space of all structures whose values are computed before optimal coalition structure generation in the algorithm presented in this chapter

coalition values that they computed for every sublist. To be precise, the maximum and minimum values in each sublist are computed by storing the first coalition value they computed in each list as the maximum and minimum value. Then, for every other value computed in each sublist, if a value is computed which is greater than or less than the maximum or minimum, respectively, these values are updated. In addition, once all values have been computed, the average is found by summing these values and dividing them by the number in each list.

As well as computing this statistical data, the agents also store in memory the biggest structure value they compute. This is achieved by first setting the value of the coalition structure consisting exclusively of coalitions of size one as optimal. Then, the first structure value computed by each agent which is greater than this value is stored as the current optimal value for this agent. In this way, every time a coalition structure value is computed which has value greater than the current optimal, this is set as the new optimal value computed by the agent.

By computing this additional data, this means that, upon completing their calculations (at approximately the same time), every agent $a_i \in Ag$ can transmit the following data among themselves:

1. The maximum, minimum and average values in both $\mathcal{L}^1_{s,i}$ and $\mathcal{L}^2_{s,i}$;
2. Any coalition values they computed in $\mathcal{L}'_{s,i}$; and,
3. The maximum structure value they have computed ($v(\pi^*_n(a_i))$).

After this information has been exchanged, every agent will be able to compute:

1. The maximum, minimum and average values in $\mathcal{L}_2, \ldots, \mathcal{L}_n$;

2. The domination value of every list $\mathcal{L}_2, \ldots, \mathcal{L}_n$; and,

3. The current optimal of the system.

Clearly, both the maximum and minimum values in every $\mathcal{L}_2, \ldots, \mathcal{L}_n$, as well as both the domination and currently optimal structure value can be easily computed from the transmitted data. However, from this transmitted data, calculating the average value in every $\mathcal{L}_2, \ldots, \mathcal{L}_n$ is less trivial. To do this, the agents calculate the sum of all the values in every $\mathcal{L}_s$. For a given $\mathcal{L}_s$, this is achieved by first computing the combined utility of all the coalitions in $\mathcal{L}_s$. This is done by first computing:

$$\sum_{C \in \mathcal{L}_s} v(C) = X + Y,$$

where,

$$X = \sum_{i=1}^{n}((\text{arg average}_{C \in \mathcal{L}^1_{s,i}} v(C) \times |\mathcal{L}^1_{s,i}|) + (\text{arg average}_{C \in \mathcal{L}^2_{s,i}} v(C) \times |\mathcal{L}^2_{s,i}|)),$$

and,

$$Y = \sum_{C \in \mathcal{L}'_{s,i}} v(C).$$

From this, the average value in a given $\mathcal{L}_s$ is computed as follows,

$$\text{arg average}_{C \in \mathcal{L}_s} v(C) = \frac{X + Y}{|\mathcal{L}_s|}.$$

Note that, by transmitting these values, no agent will transmit more than $4n + 1$ values. Since every agent is assigned approximately $\lfloor \frac{2^n - 1}{n} \rfloor$ coalition values to compute, this is clearly much more desirable than transmitting all values.

Now, observe that, after this transfer, every agent can determine which subspaces in $\overline{G}$ are promising. In particular, they can compute the upper and lower bounds of each subspace as follows:

- The upper bound of $\overline{g}$, (denoted $UB_{\overline{g}}$) is set as $\sum_{\forall i \in \overline{g}} max_i$;
- The lower bound of $\overline{g}$, (denoted $LB_{\overline{g}}$) is set as $\sum_{\forall i \in \overline{g}} min_i$; and,
- The average bound of $\overline{g}$, (denoted $Av_{\overline{g}}$).

From this data, the following system bounds can then be computed:

- $UB_{\overline{G}}$ = arg max$_{\overline{g} \in \overline{G}}$ $UB_{\overline{g}}$; and,
- $LB_{\overline{G}}$ = arg max$_{\overline{g} \in \overline{G}, \pi \in \Pi}$ $(Av_{\overline{g}}, v(\pi^*_n))$.

Once these bounds have been computed, the first pre-processing technique employed by the algorithm is to prune away all subspaces of structures $\overline{g} \in \overline{G}$ such that:

$$UB_{\overline{g}} < LB_{\overline{G}}.$$

Of course, after doing this, it may be that $UB_{\overline{G}} = LB_{\overline{G}}$, at which point an optimal coalition structure has already been computed by the agents during the coalition value calculation stages. At this stage, the algorithm can terminate and output π^*_n as the optimal coalition structure.

Otherwise, suppose $\overline{g}^* = \{s_{i_1}, \ldots, s_{i_m}\}$ is the most promising subspace. By transferring $\overline{g}^*$, $UB_{\overline{G}}$, $LB_{\overline{G}}$, $Av_{\overline{G}}$, as well as the non-filtered coalition values they have computed in lists $\mathcal{L}_{s_{i_1}}, \ldots, \mathcal{L}_{s_{i_m}}$ to the entity who is executing the IP algorithm, this entity can then compute the values all the structures in $\overline{g}^*$ that consist of all non-filtered coalitions and transmits this value to the agents. If an optimal is found, *i.e.*, a structure π is generated with value equal to $UB_{\overline{G}}$ then this procedure terminates.

Otherwise, if an optimal is not found then the entity transmits the updated optimal value to the agents who then update the system bounds. The agents can then repeat the above transmissions for the next most promising subspace (only transmitting those non-filtered coalition values that were not previously transmitted) and continue this procedure until all of $\overline{G}$ has been searched or an optimal coalition structure is found.

In this way, by removing this procedure from the optimal coalition structure generation stage to the coalition value calculation stage, certain lists of coalition values may not be transferred. For instance, if none of the promising subspaces in $\overline{G}$ contain any of the coalitions in list $\mathcal{L}_s$ then this list of values will not be transferred. Therefore, this could further minimize the amount of data that is transferred. Also, agents only execute filter rules over the coalition values in these lists, meaning this could also reduce the computational resource used by the agents.

Of course, with knowledge of the current optimal structure value, as well as the maximum, minimum, average and domination values in each of $\mathcal{L}_1, \ldots, \mathcal{L}_n$, before the lists of non-filtered coalition values are transfered to the entity who is to execute the IP algorithm, the agents can reduce the transfer load further by employing further filter rules. Specifically, before this transfer, agents can apply **FR2** and **FR3** to segments $\mathcal{L}_s^{1/2}$ where $\mathcal{L}_s \in \overline{g}^*$. Formally, these filter rules are defined as follows:

FR2a For all coalitions $C \in \mathcal{L}_{s,i}^{1/2}$ if,

$$\tilde{d}_s > \arg max_{C \in \mathcal{L}_{s,i}^{1/2}} v(C)$$

then,

$$\forall C \in \mathcal{L}_{s,i}^{1/2}, F_p = F_p \setminus \{C\} \text{ and } F_{np} = F_{np} \cup \{C\}.$$

FR3a Given $\overline{g} = \{s_1, \ldots, s_k\}$, for $j = 1, \ldots, k$, if,

$$\arg max_{C \in \mathcal{L}_{s_j,i}^{1/2}} v(C) + \sum_{q=1}^{j-1} \arg max_{C \in \mathcal{L}_{s_q}} v(C) + \sum_{q=j+1}^{k} \arg max_{C \in \mathcal{L}_{s_q}} v(C) < v(\pi_N^*)$$

then,

$$\forall C \in \mathcal{L}_{s_j,i}^{1/2}, F_p = F_p \setminus \{C\} \text{ and } F_{np} = F_{np} \cup \{C\}.$$

Sequential execution of the DCVC and IP Algorithms (without filter rule FR3c).

Input: $\langle Ag, v\rangle$

Step 1: After the agents have computed the value of the coalition assigned to them in $\mathcal{L}_1$, they then exchange this value among themselves. Whilst calculating the value of any coalition $C \in \mathcal{L}_2, \ldots, \mathcal{L}_n$, every agent $a_i \in Ag$ will:

- **(1.1)** Apply **FR1**;
- **(1.2)** Compute the value of all structures containing coalition C, as well as, the singleton coalitions consisting of the agents who do not belong to C, recording the biggest structure value they compute (denoted $v(\pi^*_n(a_i))$); and,
- **(1.3)** For $s = 2, \ldots, n$, compute and store the maximum, minimum and average coalition values in their segments $\mathcal{L}^{1/2}_{s,i}$;

After calculating the values in all of their segments, agents exchange the information in 1.2 and 1.3 among themselves.

Step 2: The agents then determine the most promising subspace $\overline{g}^*$ and compute the domination value $\tilde{d}_s$ for every list $\mathcal{L}_s : s \in \overline{g}^*$ that needs to be transmitted;

Step 3: Each agent executes both **FR2a** and **FR3a** over segments $\mathcal{L}_s$, where $s \in \overline{g}^*$. If a segment is not filtered out then **FR2b** and **FR3b** are applied again to individual coalitions within this segment that have not been filtered out by **FR1**.

Step 4: Agents transmit the promising values they have computed in the most promising subspace, as well as, the upper lower and average bounds of the system, to the entity who is to execute the IP algorithm. The agent executing the IP algorithm then computes the values of all coalition structures in this subspace, executing the **B&B** filter rule. The entity then transmits the optimal coalition structure and its value to the other agents. These agents then update the system bounds and compute the next most promising subspace and then repeat *Step 3* for those lists that have yet to be transmitted.

Output: π^*

Algorithm 4.4.2: Sequential execution of the DCVC and IP algorithms with filter rules

If a sublist is not filtered out then **FR2** and **FR3** can be applied to individual coalitions within this segment that have not been filtered out by **FR1**. Formally, these filter rules are as follows:

FR2b If $\exists C \in \mathcal{L}^{1/2}_{s,i}$ such that

$$\tilde{d}_s > v(C)$$

then,

$$\digamma_p = \digamma_p \setminus \{C\} \text{ and } \digamma_{np} = \digamma_{np} \cup \{C\}.$$

FR3b Given $\overline{g} = \{s_1, \ldots, s_k\}$, for $j = 1, \ldots, k$, if $\exists C \in \mathcal{L}^{1/2}_{s_j}$ such that,

$$v(C) + \sum_{q=1}^{j-1} \arg\, max_{C \in \mathcal{L}_{s_q}} v(C) + \sum_{q=j+1}^{k} \arg\, max_{C \in \mathcal{L}_{s_q}} v(C) < v(\pi^*_N)$$

then,

$$\digamma_p = \digamma_p \setminus \{C\} \text{ and } \digamma_{np} = \digamma_{np} \cup \{C\}.$$

Although it is not explicitly stated, if $\mathcal{L}'_{s,i} \neq \emptyset$ then every agent $a_i \in Ag$ will also execute both **FR2a** and **FR3a** over the coalitions allocated to them in $\mathcal{L}'_{s,i}$. In this context, by executing the filter rules over the coalition values in the lists that belong to the promising subspaces only, not only is it possible that the transfer load between the DCVC and IP stages is greatly reduced but the computational resource is also minimized. This sequential protocol is formally presented in Algorithm 4.4.2.

4.4.3 Transmitting Values From DCVC To The IP Stage

As all information computed by the agents thus far has been stored in a common data structure, this data is transferred to the entity who is executing the IP algorithm by enabling them unrestricted access to all the information in this data structure. Thus, agents transmit system bounds and promising subspaces to the entity who is executing the IP algorithm by recording them in this data structure. Also, agents transmit non-filtered values to the entity by executing filter rules **FR2a-b** and **FR3a-b** on the values they have computed, removing those which meet the requirements of the filter rules from the data structure. Similarly, the entity executing the IP algorithm transmits the optimal coalition structure in the subspace by recording it in the common data structure.

Recall, the ordering of the coalitions in each of $\mathcal{L}_1, \ldots, \mathcal{L}_n$ ensures that only values and not both coalitions and values need be stored in memory. Thus, if it is desirable to maintain this feature then transmitting the promising and not promising coalition values from the DCVC stage to the IP stage is not trivial since the order of the coalitions, as presented in the DCVC algorithm, must be maintained.

As this chapter is primarily concerned with the computational improvements the filter rules can offer, and provides only a foundation from which a distributed optimal coalition structure algorithm can be developed, this is left as future work. However, as an initial solution, the ordering could be maintained by transmitting a characteristic bit vector with the coalition values. This vector can indicate the position of each promising coalition value and maintain the list structure. In such a vector '1' indicates that the value is of a promising coalition, whereas '0' indicates a not promising one. In this way, both the ordering of the coalition values and the gains from filtering are maintained (with respect to searching through Π) even though the transfer load (*i.e.*, the number of values stored in the common data structure) is not reduced.

Of course, if this feature is not desirable then each agent can transmit both every promising coalition and its value to the entity that will execute the IP algorithm. In this way, although two items of data are transmitted for each promising coalition, unpromising values are not transmitted.

4.4.4 Application of Filter Rules in IP Stage and improved Search

Recall that, given both a subspace and the non-filtered values of the coalitions that can belong to the structures in this subspace, the IP algorithm cycles through all coalition structures that contain these coalitions, computing the coalition structure values in the process. Whilst doing this, efforts are undertaken to ensure that overlapping coalition structures are avoided (that is, coalition structures which do not contain disjoint coalitions) and that the value of the same structure is not computed multiple times. Additionally, this algorithm also employs the **B & B** filter rule to identify *a priori* if certain coalition structures in the subspace cannot be optimal. The values of these structures then need not be computed.

In many systems, the filtering from the branch and bound filter rule may be sufficient for the IP stage. However, for those systems where it is not, by ensuring that the agents store additional data about the values they have been computed in the DCVC stage, this filter rule can be extended so that further structures are avoided. Of course, when deciding if to incorporate this rule, the potential gains should be weighed up against the extra computational resource that is used.

To this end, let $\overrightarrow{Z}_s(j)$ denote a segment of list L_s such that a_j is the first agent in every coalition in $\overrightarrow{Z}_s(j) \subseteq L_s$.[1] For example, referring to Figure 4.1.1,

$$\overrightarrow{Z}_3(2) = \{\{a_2, a_5, a_6\}, \{a_2, a_4, a_6\}, \{a_2, a_4, a_5\}, \{a_2, a_3, a_6\}, \{a_2, a_3, a_5\}, \{a_2, a_3, a_4\}\}.$$

Now, assume that, while calculating the coalition values assigned to them, every agent $a_i \in Ag$ also records the maximum coalition value in $\overrightarrow{Z}_s(j)$ for every $a_j \in Ag$. Additionally, as well as the values of the promising coalitions they have computed, suppose every agent also transmits the maximum value in every set $\overrightarrow{Z}_s(j)$ to the IP stage. With this additional information, in the spirit of the branch and bound filter rule, the following filter rule can also be employed.

Definition 4.5 (FR3c) *Given $\overline{g}^* = \{s_{m_1}, \ldots, s_{m_k}\}$, as well as the partial structure $\{C_{m_1}, ..., C_{m_l}\}$, where:*

- *$l < k$; and,*
- *For $i = 1, \ldots, l$, $C_{m_i} \in s_{m_i}$.*

For any $a_j \in Ag \setminus \{\bigcup_{q=1}^{l} C_{m_q}\}$, if,

$$\sum_{i=1}^{l} v(C_{m_i}) + \max \overrightarrow{Z}_{l+1}(j) + \sum_{i=l+2}^{k} \mathcal{L}_{m_i} < v(CS_N^*),$$

then,

$$\forall \pi \in \overline{g}^* : \{C_{m_1}, ..., C_{m_l}\} \subset \pi, v(\pi) \text{ is not computed.}$$

In many systems where, for example, it is time consuming to both compute and transmit values, it may not be worthwhile to employ this filter rule as the potential gains may be reduced. In contrast, when there is no such time consideration, the potential gains may be so great that it is worthwhile executing these additional computations. To this end, the decision to incorporate this filter rule into the IP stage is left to the discretion of the system designer.

Now, in the IP algorithm, in each subspace, coalition structures are constructed *via* branch and bound between coalitions of ascending size. For instance, in a system of six agents, given $\overline{g} = \{1, 2, 3\}$, coalition structures are generated through employing a branch and bound rule between the coalitions in $\mathcal{L}_1$ and the coalitions in $\mathcal{L}_2$, leaving a coalition of size three from the remaining agents. Clearly, from a computational perspective, it is advantageous if the branch and bound filter rule stops constructing a structure as early as possible because less work is done. Therefore, rather than construct structures in the manner described above, in this algorithm, the branch and bound rule constructs coalition structures from the subspaces which are ordered non-decreasingly with respect to the number of promising coalitions they contain. This can improve the effectiveness of the branch and bound filter rule (and, therefore, the efficiency in which an optimal coalition structure is generated) because it can prune away the biggest part of the subspace at a very early stage.[2] To this end, it is assumed that every subspace is analyzed as in the IP algorithm only the structures are constructed from the list that are ordered non-decreasingly with respect to the number of promising coalitions they contain. The entity who executes the IP algorithm can easily deduce this number from the values that are transmitted to them.

[1] Recall that, due to the ordering of coalitions, all coalitions beginning which each agent will naturally be grouped together.

[2] In an extension to this work, Tomasz Michalak *et al.* prove that this method of search is guaranteed to be more effective than the existing method. However, as this paper has yet to be published, no citation can be given at the current time.

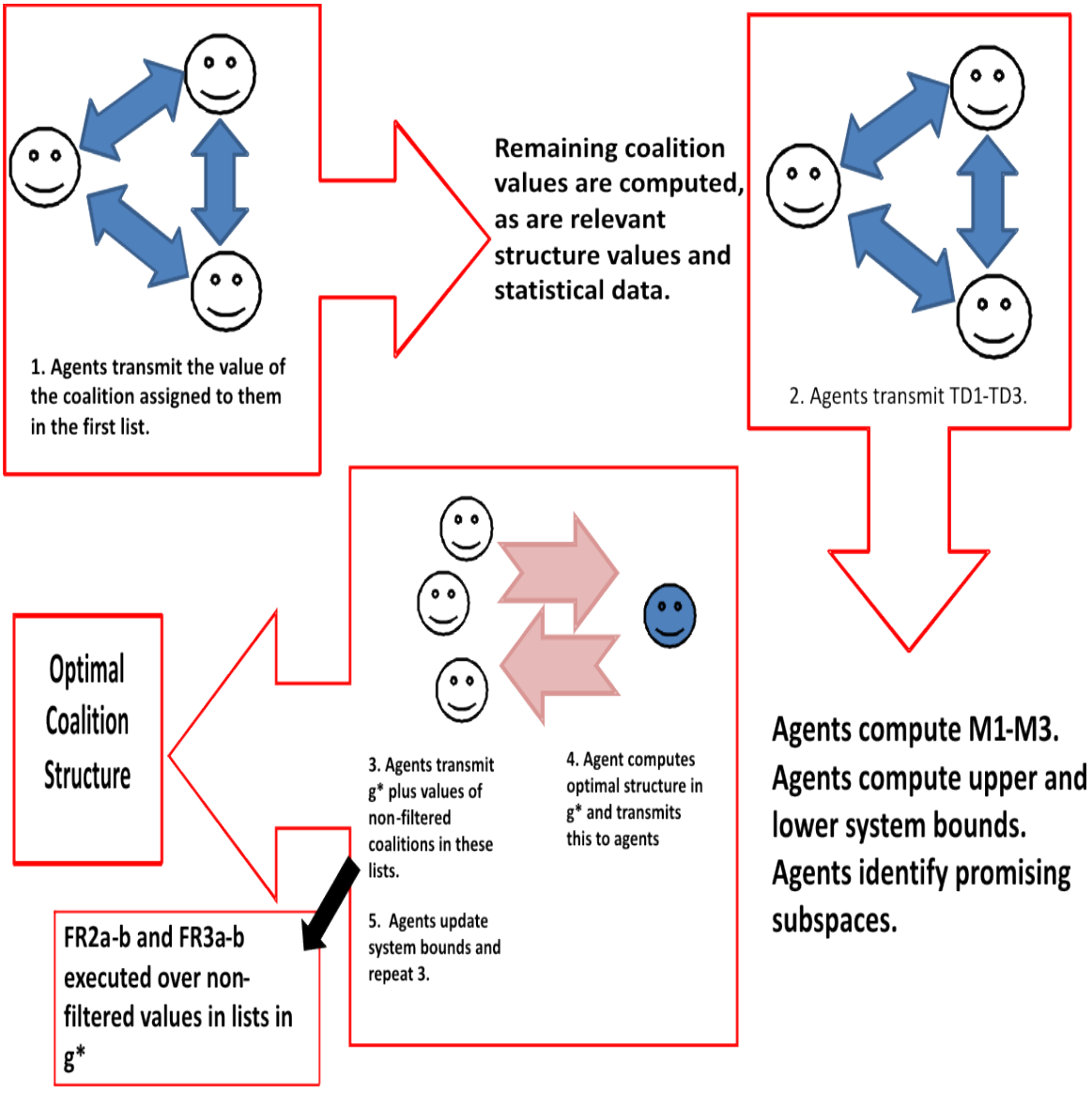


Figure 4.4.4: The sequential application of the DCVC and IP algorithms with filter rules

4.5 Assessing The Effectiveness of The Filter Rules

In this section, the effectiveness of the filter rules are assessed with respect to the percentage of coalition values they filter out. This is achieved by comparing the sequential execution of DCVC and IP algorithms both with and without filter rules. To do this, all real world factors which could affect the transmission of data or the ability of the agent to compute values (such as technical failure, data corruption, data loss in transmission *e.t.c*) are ignored. To this end, it is assumed that:

1. The calculation of both coalition and coalition structure values takes no time;
2. The agents are fully cooperative and have equal computational abilities; and,
3. Any data transfers are instantaneous.

Note that, against these assumptions, it is advantageous to employ **FR3c**. The effectiveness of the filter rules are assessed with respect to the percentage of $\digamma$ which does not have to be transmitted from the DCVC to the IP stages as a consequence of executing them. This can reduce the number of coalition structures that are analyzed by the IP algorithm, as well as the number of coalition values that are transmitted to the entity who is to execute the IP algorithm. Clearly, from a computational perspective, this is desirable. Additionally, these filter rules have the power to reduce the communication complexity, transmission load and the computational load of the agents, meaning these filter rules can provide a useful foundation from which a distributed optimal coalition structure generation algorithm can be developed.

For a system of $n = 11, \ldots, 20$ agents, the sequential application of DCVC and IP algorithms, with and without filter rules, was executed 25 times where the coalition values were distributed as follows,

Normal: $v(C) = max(0, |C| \times p)$, where $p \in N(\mu = 1, \sigma = 0.1)$; and,

Uniform: $v(C) = max(0, |C| \times p)$, where $p \in U(a, b)$ and $a = 0, b = 1$.

These distributions were chosen since they were also used in both [66] and [59]. The sub- and super-additive cases were omitted as, following Theorem 2.2, their solution is trivial. The algorithms were implemented in MATLAB and the results were reported within a a 95% confidence interval. The results are presented in Table 4.5a and Table 4.5b. In these tables:

- Column 1 shows the number of agents (n);
- Column 2 shows the number of coalition values that need be computed before the DCVC stage commences ($|\digamma|$);
- Column 3 shows the percentage of $\digamma$ that needs to be transmitted from the coalition value calculation stage to the optimal coalition structure generation stage when there are no filter rules;
- Column 4 shows the percentage of $\digamma$ transmitted from the coalition value calculation stage to the optimal coalition structure generation stage when filter rules are applied;
- Columns 5 - 9 the percentage of coalitions filtered by **FR1**, **FR2a**, **FR2b**, **FR3a** and **FR3b**, respectively (expressed as a percentage of the number which are filtered); and,
- The last column shows compares the running times of the sequential execution of DCVC and IP algorithms with and without filter rules.

Specifically, the figure in the last column is computed in the following manner,

$$R := \frac{\text{Running time of IP algorithm with unfiltered input and without } \textbf{FR3c}}{\text{Running time of IP algorithm with filtered input and } \textbf{FR3c}}.$$

n	$\|F\|$	% of $\|F\|$ transmitted without filter rules	% of $\|F\|$ filtered by filter rules	% of % of $\|F\|$ filtered by **FR1**	% of % of $\|F\|$ filtered by **FR2a**	% of % of $\|F\|$ filtered by **FR2b**	% of % of $\|F\|$ filtered by **FR3a**	% of % of $\|F\|$ filtered by **FR3b**	R
11	2,047	91.00 ± 4.5	8.68 ± 1.69	53.12	0	0.01	21.61	25.26	2.75 ± 0.81
12	4,095	91.00 ± 2.4	8.43 ± 1.51	55.71	0	0.01	18.13	26.15	4.17 ± 1.55
13	8,191	90.00 ± 5.5	5.29 ± 1.37	54.91	0	0	16.69	28.40	6.61 ± 2.03
14	16,383	91.00 ± 5.3	6.05 ± 1.61	54.99	0	0	16.70	28.31	16.51 ± 5.00
15	32,767	93.00 ± 8.4	5.04 ± 1.21	53.11	0	0	14.70	32.19	17.88 ± 6.04
16	65,535	91.00 ± 5.1	4.48 ± 1.14	49.79	0	0	14.04	36.17	23.21 ± 7.40
17	131,071	92.00 ± 4.5	4.29 ± 1.01	53.00	0	0	11.19	35.81	112.20 ± 44.00
18	262,143	93.00 ± 6.3	3.69 ± 0.89	52.47	0	0	9.01	38.52	169.00 ± 54.00
19	534,287	94.00 ± 6.1	3.12 ± 0.81	51.48	0	0	7.80	40.72	197 ± 67.00
20	1,048,575	91.00 ± 5.3	2.49 ± 0.67	50.03	0	0	5.99	43.98	380.00 ± 8.00

Table 4.5a: Assessment of filter rules for a normal distribution of coalition values

Clearly, if $R > 1$ then this means that the running time of the algorithm without filter rules is greater than the running time of the algorithm with filter rules. Thus, if $R > 1$ then this means that the filter rules can improve the speed in which an optimal coalition structure is generated and, in this way, offer computational improvement.

From the obtained data, the following observation and trends are common to both the uniformly and normally distributed coalition values:

Observation 1 Filter rule **FR1** is the most effective with respect to filtering coalition values;

Observation 2 Filter rules **FR2a** and **FR2b** are generally ineffective with respect to filtering coalition values;

Trend 1 As n increases, the number of coalition values filtered by **FR3a** decreases; and,

Trend 2 As n increases, the number of coalition values filtered by **FR3b** increases.

Consider **Observation 1** and **Observation 2** first. Intuitively, **FR1** is effective if the values of all coalitions of size one are large relative to the values of all coalitions of size greater than one. On the other hand, the effectiveness of both **FR2a** and **FR2b** are dependent upon the size of the domination value relative to the coalition values. In this context, if the values of all coalitions of size one are large relative to the values of all coalitions of size greater than one and the values of all coalitions of size greater than one are within a small range of one another then this may explain these observations. Thus, if the converse was true then perhaps **FR1** would not perform as well and both **FR2a** and **FR2b** would be more effective at filtering coalition values.

n	$\|F\|$	% of $\|F\|$ transmitted without filter rules	% of $\|F\|$ filtered by filter rules	% of % of $\|F\|$ filtered by **FR1**	% of % of $\|F\|$ filtered by **FR2a**	% of % of $\|F\|$ filtered by **FR2b**	% of % of $\|F\|$ filtered by **FR3a**	% of % of $\|F\|$ filtered by **FR3b**	R
11	2,047	51.31 ± 16.51	3.32 ± 1.79	49.24	0.07	1.17	35.21	14.31	1.12 ± 0.14
12	4,095	39.99 ± 18.01	2.78 ± 1.41	51.30	0.06	1.00	28.01	19.63	1.22 ± 0.09
13	8,191	49.68 ± 17.52	2.71 ± 1.02	49.43	0.12	0.68	27.83	21.94	1.34 ± 0.18
14	16,383	41.13 ± 17.52	2.42 ± 0.68	50.32	0.04	0.32	23.15	26.17	1.14 ± 0.12
15	32,767	48.52 ± 21.00	2.35 ± 0.48	51.12	0.00	0.11	18.51	30.26	1.23 ± 0.16
16	65,535	52.98 ± 19.40	1.26 ± 0.33	50.51	0.00	0.04	16.54	32.91	1.14 ± 0.09
17	131,071	50.41 ± 18.97	1.15 ± 0.29	51.81	0.00	0.02	11.19	36.98	1.01 ± 0.03
18	262,143	46.41 ± 19.70	1.13 ± 0.28	50.18	0.00	0.00	10.91	38.91	1.03 ± 0.02
19	534,287	41.71 ± 18.42	1.05 ± 0.13	50.02	0.00	0.00	8.01	41.97	1.04 ± 0.04
20	1,048,575	32.80 ± 19.70	1.04 ± 0.10	50.25	0.00	0.00	7.46	42.29	1.19 ± 0.11

Table 4.5b: Assessment of filter rules for a uniform distribution of coalition values

Now, consider both **Trend 1** and **Trend 2**. Interestingly, as the value of n increases, the number of coalition values filtered by **FR3a** decreases whereas the number of coalition values filtered by **FR3b** increases. With regards to **FR3a**, the higher the value of n, the bigger $|\mathcal{L}_s|$ becomes for $s \in [2, \ldots, n-1]$, meaning there is greater probability that the randomly drawn extremal values in these segments are similar to each other. As this filter rule is based on maximum value in each segment, this may explain this particular trend in the results obtained. In contrast, as **FR3b** focuses on individual coalition values, for $s \in [2, \ldots, n-1]$, since $|\mathcal{L}_s|$ increases as n increases, there is a greater chance that this rule may become increasingly effective.

Now, comparing the results obtained for uniformly and normally distributed coalition values, the following observations and trends can be noticed:

Observation 3 With uniformly distributed coalition values, exponentially less values are input to the optimal coalition structure generation stage in the absence of filter rules than with normally distributed coalition values;

Observation 4 Filter rules reduce a greater percentage of the input to the optimal coalition structure generation stage in the normal case than in the uniform case; and,

Trend 3 As n increases, the filter rules offer exponentially greater improvements in the running time of the procedure for normally distributed coalition values compared to only linear improvements for uniformly distributed values.

One interpretation of **Observation 3** is that, for uniform values, an optimal coalition structure is found during the coalition value calculation stage or during the search of only a few promising subspaces. Alternatively, it maybe that the promising subspaces all contain a common set of lists meaning only these lists need be transmitted exactly once to the entity who is executing the IP algorithm.

Given this explanation, if exponentially more coalition values are input to the coalition structure generation stage in the normal case than in the uniform case then this means that, in the former case, there are exponentially more coalition values over which the filter rules are applied. This, in turn, means that, in the normal case, there are exponentially more coalitions to filter out than in the uniform case. Thus, this reasoning could explain **Observation 4**.

Finally, **Trend 3** can be explained as a consequence of **Observation 4**.

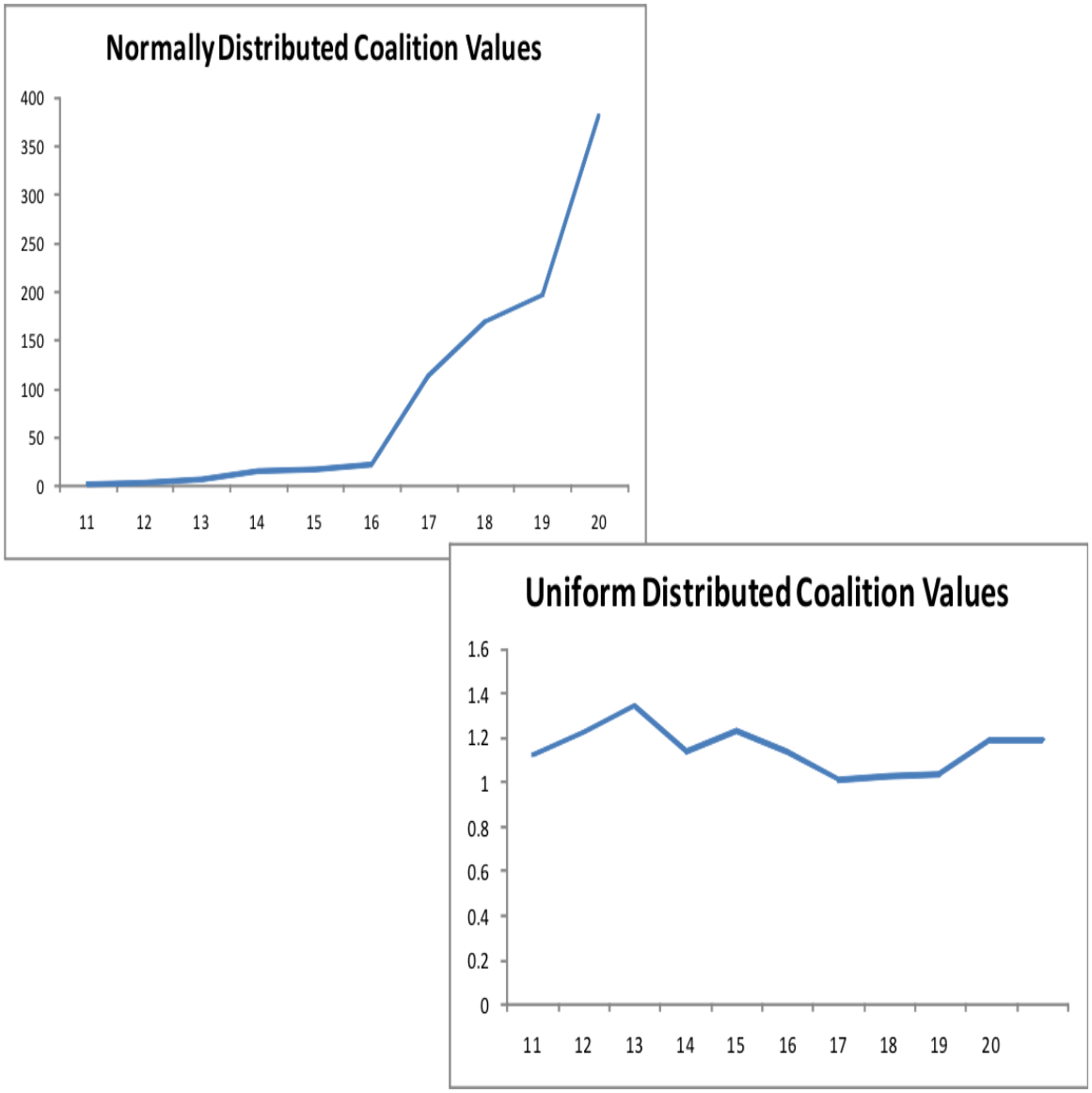
In conclusion, given the predefined assumptions, these results suggest that, for uniformly and normally distributed values, the combination of the filtered input and the application of **FR3c** results in a much faster performance of this algorithm. This can be attributed to both the filter rules and the order in which the subspaces are searched. With regards to the latter case, the new search method ensures that more coalitions are filtered during the IP stage and, consequently, the number of coalition structure values that have to be computed is already significantly less. The results also suggest that, if the coalition values are known to be either uniformly or normally distributed then it may only be necessary to employ filter rules **FR1** and **FR3b** in the DCVC stage as these are the most effective.

4.6 Summary

In this chapter, pre-processing techniques, represented as filter rules that can be incorporated into the sequential application of the DCVC and IP algorithms, were presented. These rules identified coalitions that could not belong to an optimal coalition structure and, when these coalitions were found, appropriate action was taken. These actions involve filtering coalition values from the input or avoiding all structures containing these coalitions in Π.

From a computational perspective, these filter rules can reduce the number of coalition structure values that need be computed by the IP algorithm. Also, they can reduce the number of coalition values an individual agent needs to transfer after completing their computations. In this context, these filter rules may be useful foundations from which a distributed optimal coalition structure generation algorithm can be developed for a system of fully cooperative agents. To this end, a sequential process was developed in which both the algorithms and filter rules were employed.

To reduce the transfer load, as well as the number of values over which a number of the filter rules are computed, the processes of identifying promising subspaces was switched from the IP stage and put in the DCVC stage instead. Additionally, the way in which the values of the structures in each subspace were computed was refined so as to increase the effectiveness of filtering. The effectiveness of these filter rules was tested for normally and uniformly distributed coalition values. Empirical results showed that the filter rules can greatly reduce the transmission load between the stages for both of these distributions. Furthermore, these results also showed that the filter rules can greatly reduce the overall running time of sequential application of both algorithms, especially in the normal case, where filter rules can offer an exponential improvement. In conclusion, empirical results seem to suggest that the filter rules can provide an important foundation from which to build a distributed optimal coalition structure generation algorithm for a system of fully cooperative agents.



Graph 4.5: The relative running times of the algorithms for uniform and normally distributed coalition values. The x axis represents the number of agents in the system (n), whereas the y axis represents R for the corresponding n value

Chapter 5

Optimal Coalition Structure Generation in Partition Function Games

In characteristic function games, when all coalition values are given as input, the integer partition (IP) algorithm (presented in Section 3.2.3) can efficiently generate an optimal coalition structure through determining *a priori* if certain groups of coalition structures cannot be optimal. This is achieved, in part, by bounding the values of all coalition structures. Then, only the values of the 'promising structures' in Π are computed and those which cannot be optimal are avoided.

For many multi-agent systems, characteristic function game representations are sufficient to model coalition formation as the coalitions either do not interact with each other while pursuing their own goals or because such interactions are insignificant enough to be neglected. However, in a number of multi-agent system environments, there may exist non-negligible externalities from coalition formation where the utility obtained from forming coalitions may be affected by the formation of other distinct coalitions. In particular, following previous discussions, real world examples include coalition formation between agents representing different companies.

When there exist externalities from coalition formation, the value of a coalition $C \subseteq Ag$ can be dependent upon the structure in which it is embedded. Thus, partition function games $\mathcal{P} = \langle Ag, P \rangle$ are more appropriate than characteristic function games in order to model coalition formation when there are externalties from coalition formation. Recall that, in this representation, if $\mathcal{E}$ represents the space of all embedded coalitions then, for any $(C; \pi) \in \mathcal{E}$, function $P(c; \pi)$ represents the value obtained from forming coalition C in structure π. However, to directly execute the IP algorithm in partition function games, the partition function values of all embedded coalitions $(C; \pi) \in \mathcal{E}$ must be given. Since $|\mathcal{E}| \geq \mathcal{B}_n$,[1] $\forall n \in \mathbb{N}$, this is infeasible, even for relatively small values of n. Therefore, in practice, it is not possible to input all $P(C; \pi)$ values when dealing with optimal coalition structure generation in partition function games. This, in turn, means that in partition function games, when given only the partition function, it is not possible to pre-determine the value of a coalition C which is embedded in a structure π without actually computing $P(C; \pi)$. Consequently, $P(C; \pi)$ must be computed for all $(C; \pi) \in \mathcal{E}$ to guarantee that an optimal coalition structure is generated. This clearly presents a major computational challenge.

In this chapter, for partition function games where certain features regarding the nature of the externality or the nature of the partition function are known then an algorithm is developed which can exploit this additional information so that only a fraction of both Π and $\mathcal{E}$ need be analyzed to guarantee an optimal structure. Specifically:

- In Section 5.1, the notions of positive and negative externalities, as well as the notions of super-additivity and sub-additivity in partition function games are formally defined. Based on these notions,

[1] Recall that $\mathcal{B}_n$ is the Bell number for n and represents the number of coalition structures that can be formed from n agents.

four natural classes of partition function games are then presented;

- In Section 5.2, for these classes of games, it is proven that, by computing the values of a number of embedded coalitions, the values of all coalitions can be bounded;
- In Section 5.3, an algorithm is developed which, for these classes of games, can generate an optimal coalition structure through computing the values of only a fraction of Π; and,
- In Section 5.4, the effectiveness of this algorithm is evaluated. As this is the first algorithm that considers coalition structure generation in partition function games, this algorithm is assessed with respect to the number of coalition structure values that are computed by algorithm. It is shown that, for a system of 10 agents, this algorithm can, in some cases, generate an optimal coalition structure by analyzing only 4% of Π.

5.1 Natural Classes of Partition Function Games

The optimal coalition structure concept formulated in Definition 2.12 can be easily extended to partition function games as follows.

Definition 5.1 *Given any $\mathcal{P} = \langle Ag, P \rangle$, an optimal structure is a structure $\pi^* \in \Pi$ such that:*

$$\pi^* = \textit{arg max}_{\pi \in \Pi} \sum_{C \in \pi} P(C; \pi).$$

Intuitively, if nothing is known about the function P or the nature of the externalities then the value of coalition C in every structure to which it is embedded cannot be known *a priori*. Therefore, for every $(C; \pi) \in \mathcal{E}$, $P(C; \pi)$ must be computed exactly once in order to generate π^* and no algorithm can do this through analyzing fewer embedded coalitions. This, in turn, implies that all of Π must be analysed in order to generate an optimal coalition structure.

However, if the nature of the partition function and the externalities are known then it is possible to circumvent the above reasoning and bound the values of the coalitions and, therefore, the values of the structures in which they are embedded. In particular, based upon both the nature of the partition function and the externalities, it is possible to identify natural classes of partition function games.

Definition 5.2 *Given a PFG representation $\mathcal{P} = \langle Ag, P \rangle$, $\mathcal{P}$ is said to exhibit* strict positive externalities *if for:*

- *All disjoint coalitions $C, S, T \subseteq Ag$;*
- *All partitions π' of the agents in $Ag \setminus S \cup T \cup C$; and,*
- *All structures*
  - $\pi = \{C, S, T, \pi'\}$;
  - $\pi_\alpha = \{C, S \cup T, \pi'\}$;
  - $\pi_\beta = \{S, C \cup T, \pi'\}$; *and,*
  - $\pi_\gamma = \{T, C \cup S, \pi'\}$;

then,

1. $P(C; \pi_\alpha) > P(C; \pi)$;
2. $P(S; \pi_\beta) > P(S; \pi)$; *and,*

3. $P(T; \pi_\gamma) > P(T; \pi)$.

In contrast, if the '$>$' sign is interchanged with the '$<$' sign then $\mathcal{P}$ is said to exhibit strict negative externalities.

Against this definition, if either the '$>$' or '$<$' signs are interchanged with the '$\geq$' or '$\leq$' signs then $\mathcal{P}$ is said to exhibit *weak positive* and *weak negative externalities*, respectively.

In words, $\mathcal{P}$ is said to exhibit strict positive externalities from coalition formation if the creation of every coalition increases the value of all co-existing coalitions. On the other hand, if the creation of every coalition decreases the value of all the other coalitions in the structure then $\mathcal{P}$ is said to exhibit strict negative externalities from coalition formation. It is apparent from this definition that the characteristic function game representation is a special case of the partition function game representation where all externalities from coalition formation have value zero.

In addition to the notions of negative and positive externalities, the notion of sub- and super- additivity can also be formulated for partition function games as follows (taken from [27]).

Definition 5.3 *Given a partition function game representation $\mathcal{P} = \langle Ag, P \rangle$, $\mathcal{P}$ is* super-additive *if for:*

- *All $C, S \subseteq Ag$ such that C and S are disjoint;*
- *All partitions π' of the agents in $Ag \setminus C \cup S$; and,*
- *All structures $\pi = \{C, S, \pi'\}$, $\pi_1 = \{C \cup S, \pi'\}$;*

then,

$$P(C \cup S; \pi_1) \geq P(C; \pi) + P(S; \pi).$$

Conversely, $\mathcal{P}$ is sub-additive *if the converse is true,* i.e.*,*

$$P(C \cup S; \pi_1) \leq P(C; \pi) + P(S; \pi).$$

Since the notions of externalities and additivity are independent of one another, this gives rise to four natural classes of partition function game:

1. Super-additive games with positive externalities ($\mathcal{P}^+_{sup}$);
2. Super-additive games with negative externalities ($\mathcal{P}^-_{sup}$);
3. Sub-additive games with positive externalities ($\mathcal{P}^+_{sub}$); and,
4. Sub-additive games with negative externalities ($\mathcal{P}^-_{sub}$).

Example 5.1 *[Taken from [27]]. Consider $\mathcal{P} = \langle Ag, P \rangle$ where $Ag = \{a_1, a_2, a_3\}$ and Π exclusively consists of the following structures:*

$$\begin{aligned}
\pi_\alpha &= \{\{a_1\}, \{a_2\}, \{a_3\}\}; \\
\pi_\beta &= \{\{a_1, a_2\}, \{a_3\}\}; \\
\pi_\gamma &= \{\{a_1, a_3\}, \{a_2\}\}; \\
\pi_\delta &= \{\{a_2, a_3\}, \{a_1\}\}; and, \\
\pi_\epsilon &= \{\{a_1, a_2, a_3\}\}.
\end{aligned}$$

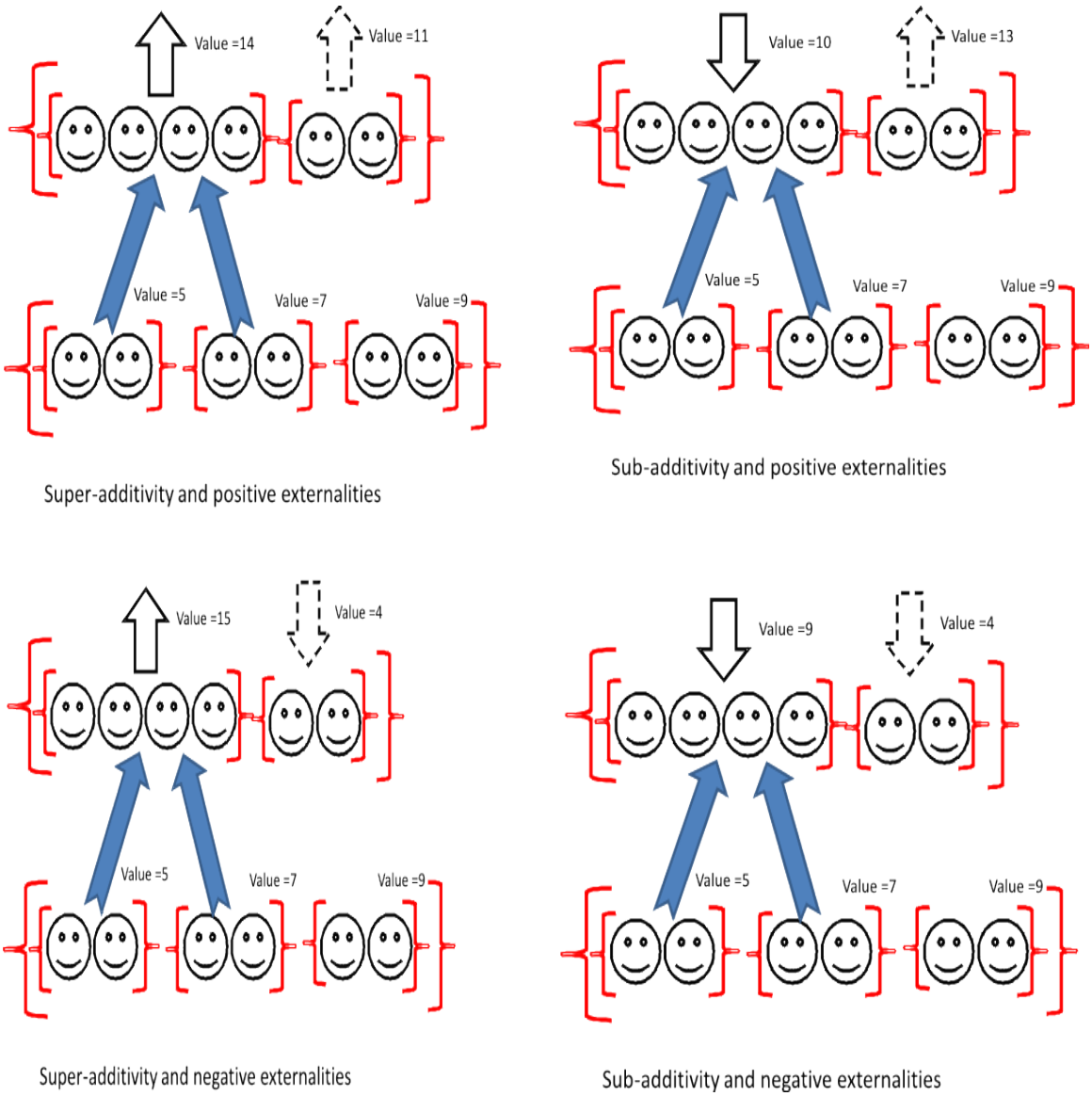


Figure 5.1: The four natural classes of partition function games studied in this chapter

For this game, $\mathcal{E}$ consists of the following embedded coalitions and partition function values:

$$P(\{a_1\};\pi_\alpha) = P(\{a_2\};\pi_\alpha) = P(\{a_3\};\pi_\alpha) = 4;$$
$$P(\{a_1,a_2\};\pi_\beta) = P(\{a_1,a_3\};\pi_\gamma) = P(\{a_2,a_3\};\pi_\delta) = 9;$$
$$P(\{a_1\};\pi_\delta) = P(\{a_2\};\pi_\gamma) = P(\{a_3\};\pi_\beta) = 1; and,$$
$$P(Ag;\pi_\epsilon) = 11.$$

In this example:

- *The structures π_β, π_γ and π_δ are formed from two agents in structure π_α merging to form a coalition of size two; and,*
- *The structure π_ϵ is formed from a merger of the two coalitions which make up the structures π_β, π_γ and π_δ.*

Observe that in structures π_β, π_γ and π_δ, the formation of the coalitions of size two induce a negative externality of $1-4=-3$ upon the agent who does not cooperate with them. Also, observe that the value of the formed coalition in any structure is greater than the combined values of the coalitions from which it was formed. For instance, the value of the coalitions of size 2 in π_β, π_γ and π_δ are greater than the combined value of the two agents who make them up and the grand coalition formed in π_ϵ is greater than the value of the structures π_β, π_γ and π_δ. In this way, the partition function in this example induces negative externalities and is super-additive.

Intuitively, both sub-additivity and positive externalities are displayed if P is changed so that:

$$P(\{a_1\};\pi_\alpha) = P(\{a_2\};\pi_\alpha) = P(\{a_3\};\pi_\alpha) = 3;$$
$$P(\{a_1,a_2\};\pi_\beta) = P(\{a_1,a_3\};\pi_\gamma) = P(\{a_2,a_3\};\pi_\delta) = 2;$$
$$P(\{a_1\};\pi_\delta) = P(\{a_2\};\pi_\gamma) = P(\{a_3\};\pi_\beta) = 8; and,$$
$$P(Ag;\pi_\epsilon) = 4.$$

Recall Theorem 2.2. This theorem states that if the characteristic function is super- or sub- additive then $\pi^* = \{Ag\}$ or $\pi^* = \{\{a_1\},\ldots,\{a_n\}\}$, respectively. Clearly, this theorem also holds for the partition function in both $\mathcal{P}^+_{sup}$ and $\mathcal{P}^-_{sub}$. However, it can be shown through example that, for classes $\mathcal{P}^-_{sup}$ and $\mathcal{P}^+_{sub}$, Theorem 2.2 may not hold.

Example 5.2 *Recall Example 5.1. Consider first the situation where function P exhibits both super-additivity and negative externalities. Observe that, despite the super-additivity constraint, $\pi_\epsilon = \{\{a_1,a_2,a_3\}\}$ is not an optimal structure. Instead, the structure $\pi_\alpha = \{\{a_1\},\{a_2\},\{a_3\}\}$ is optimal.*

Now, consider the situation where function P exhibits both sub-additivity and positive externalities. Observe that, despite the sub-additivity constraint, $\pi_\alpha = \{\{a_1\},\{a_2\},\{a_3\}\}$ is not optimal but, instead, the structures $\pi_\beta = \{\{a_1,a_2\},\{a_3\}\}$, $\pi_\gamma = \{\{a_1,a_3\},\{a_2\}\}$ and $\pi_\delta = \{\{a_2,a_3\},\{a_1\}\}$ are optimal.

Example 5.2 shows that if the function P is subject to both super-additivity and negative externalities then it may be that $\pi^* \neq \{Ag\}$, whereas if P is subject to both sub-additivity and positive externalities then it may be that $\pi^* \neq \{\{a_1\},\ldots,\{a_n\}\}$. This implies that, in the worst case, the values of all coalition structures $\pi \in \Pi$ must be computed to generate an optimal coalition structure in either $\mathcal{P}^+_{sub}$ and $\mathcal{P}^-_{sup}$. However, in the next section, it is proven that, for both of these games, the values of all coalitions (and, therefore, the structures in which they are embedded) can be bounded by first computing the values of the coalitions which are embedded in structures that belong to only a fraction of Π. In this context, the values of the remaining coalition structures can be computed in the spirit of the IP algorithm, avoiding those that cannot be optimal.

5.2 Bounding Coalition Structure Values

Before describing how the values of the coalitions are bounded, first consider an integer partition graph representation of the space of all coalition structures. In contrast to the representation presented in Figure 3.2.4, for $i = 1, \ldots, n-1$, the edges are directed from coalition structures of size i to coalition structures of size $i+1$. In this representation, each node represents a subspace of all coalition structures denoted that contain coalitions of size represented by the integer values (that is, following IP algorithm notation, each node represents each $\overline{g} \in \overline{G}$). For example, $\{4, 2\}$ denotes the subspace of all coalition structures that consist of exactly one coalition of size 2 and one coalition of size 4. This time, as opposed to the integer partition graph representation used for the hybrid algorithm (presented in Section 3.2.4), for $i = 1, \ldots, n-1$, an edge between structures of size i and $i+1$ represents the formation of the structure of size i from a merge between two coalitions in the structure of size $i+1$.

Recall that, in partition function games, $P(C; \pi)$ represents the utility obtained from forming a coalition $C \subseteq Ag$ given that the coalitions in $\pi \setminus \{C\}$ have formed. To this end, this section commences with the following insight.

Theorem 5.1 *Consider a PFG $\mathcal{P} = \langle Ag, P \rangle$ where $|Ag| = n$. For every coalition $C \subseteq Ag$, consider the following structures:*

- $\pi_\alpha = \{C, \{a_1\}, \ldots, \{a_{n-|C|}\}\}$ *where* $\{a_1, \ldots, a_{n-|C|}\} = Ag \setminus C$*; and,*
- $\pi_\beta = \{C, C'\}$ *where* $C' = Ag \setminus C$.

If $\mathcal{P} = \mathcal{P}^-_{sup}$ then, for every $\pi_\gamma \in \Pi \setminus \{\pi_\alpha, \pi_\beta\}$ such that $(C; \pi_\gamma) \in \mathcal{E}$, the following is true:

1. $P(C; \pi_\beta) \leq P(C; \pi_\alpha)$*;*
2. $P(C; \pi_\beta) \leq P(C; \pi_\gamma)$*; and,*
3. $P(C; \pi_\alpha) \geq P(C; \pi_\gamma)$.

Proof: Inherent to this proof is the following insight.

> In any integer partition graph, for every coalition $C \subseteq Ag$, there exist paths from π_α to π_β such that every structure π_γ, where $(C; \pi_\gamma) \in \mathcal{E}$, is represented by exactly one node in every one of these paths.

Assume $\mathcal{P} = \mathcal{P}^-_{sup}$ and consider the value of any coalition C in π_α (*i.e.*, $\mathcal{P}(C; \pi_\alpha)$). Observe that no other coalitions have been formed in the structure π_α. Therefore, there are no negative externalities induced on the coalition $C \in \pi_\alpha$. However, as every path from π_α to π_β is traversed, coalitions are formed that induce consecutive negative externalities upon coalition C. Consequently:

- The value of coalition $C \in \pi_\alpha$ can be no less than the value of coalition C in both π_β and π_γ; and,
- The value of coalition C in every π_γ can be no less than the value of coalition C in π_β.

Clearly, against this reasoning, all of 1-3 hold.

∎

The intuition behind Theorem 5.1 can be extended to the $\mathcal{P}^+_{sub}$ class of games.

Theorem 5.2 *For any $\mathcal{P}^+_{sub}$ then, for every $\pi_\gamma \in \Pi$ such that $C \in \pi_\gamma$, the following is true:*

1. $P(C; \pi_\beta) \geq P(C; \pi_\alpha)$*;*
2. $P(C; \pi_\beta) \geq P(C; \pi_\gamma)$*; and,*

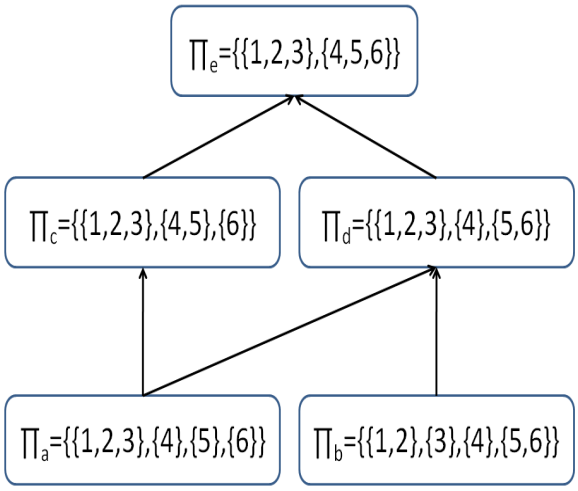


Figure 5.2: An extract from the representation of Π for six agents. Here, numbers represent the indices of the agents in the coalitions

3. $P(C;\pi_\alpha) \leq P(C;\pi_\gamma)$.

Proof: Assume $\mathcal{P} = \mathcal{P}^{+}_{sub}$. This time, as every path from π_α to π_β is traversed, coalitions are formed that induce consecutive positive externalities upon every coalition. Consequently, for every coalition $C \subseteq Ag$:

- The value of coalition $C \in \pi_\alpha$ can be no greater than the value of coalition C in both π_β and π_γ; and,
- The value of coalition C in every π_γ can be no greater than the value of coalition C in π_β.

Therefore, 1-3 hold.

∎

To provide further intuition regarding both Theorem 5.1 and Theorem 5.2, consider Figure 5.2. Observe that the coalition $\{a_1, a_2, a_3\}$ belongs to every structure in this figure. Theorem 5.1 says that, under $\mathcal{P}^{-}_{sup}$ assumptions, for $i = b, \ldots, e$, $P(\{a_1, a_2, a_3\}; \pi_a) \geq P(\{a_1, a_2, a_3\}; \pi_i)$. Initially, it may seem possible that, $P(\{a_1, a_2, a_3\}; \pi_a) < P(\{a_1, a_2, a_3\}; \pi_d)$ because π_d emerged after agent a_3 joined coalition $\{a_1, a_2\}$ in π_b and, due to super-additivity, $\mathcal{P}(\{a_1, a_2, a_3\}; \pi_d)$ may be greater than $\mathcal{P}(\{a_1, a_2, a_3\}; \pi_a)$. However, this super-additivity is offset by the negative externalities induced by the formation of the coalition $\{a_5, a_6\}$, meaning the value of $\{a_1, a_2, a_3\}$ in π_d can be no greater than the value of $\{a_1, a_2, a_3\}$ in π_a. Similar reasoning can be used to provide intuition with respect to the claims made for $\mathcal{P}^{+}_{sub}$ assumptions in Theorem 5.1.

The significance of both Theorem 5.1 and Theorem 5.2 is as follows. Given $\mathcal{P}^{-}_{sup}$, for every coalition $C \subseteq Ag$, $P(C;\pi_\beta)$ represents the smallest value of C whereas $P(C;\pi_\alpha)$ represents the biggest value of C in the system. In contrast, given $\mathcal{P}^{+}_{sub}$, for every coalition $C \subseteq Ag$, $P(C;\pi_\beta)$ represents the biggest value of C, whereas, $P(C;\pi_\alpha)$ represents the smallest value of C in the system. In this way, by computing these values, the maximum and minimum values of all coalition can be computed and, in the spirit of the IP algorithm, (non-tight) upper and lower bounds on the values of the remaining coalition structures can be determined. Figure 5.3 displays the subspaces of coalition structures that should be analyzed in order to bound the coalition values in a six agent setting.

Against both Theorem 5.1 and Theorem 5.2, in the next section, an algorithm is developed for both $\mathcal{P}^{+}_{sub}$ and $\mathcal{P}^{-}_{sup}$ in which, upon bounding the values of all coalitions, bounds the values of the remaining structures

whose values have not been computed thus far. In this way, the structures that cannot be optimal in the remaining space will be avoided and so, for these classes of partition function games, an optimal coalition structure may be generated without having to analyze all of Π.

5.3 An Optimal Coalition Structure Generation Algorithm

An algorithm to generate an optimal coalition structure in either $\mathcal{P}^{+}_{sub}$ or $\mathcal{P}^{-}_{sup}$ is presented in Algorithm 5.3.2. In this algorithm, the space of all coalition structures is represented as an integer partition graph, as described in the previous section. Recall that in this graph, in terms of the IP algorithm notation, each node represents a subsapce $\overline{g} \in \overline{G}$. Also, for any two $\overline{g} = \{s_{i_1}, \ldots, s_{i_m}\}, \overline{g}' = \{s_{i_1}, \ldots, s_{i_{m-1}}\} \in \overline{G}$, an edge connects $\overline{g}$ to $\overline{g}'$ if and only if $\exists s_{i_k}, s_{i_l} \in \overline{g}$ and $\exists s_{i_j} \in \overline{g}'$ such that:

1. $\overline{g} \setminus \{s_{i_k}, s_{i_l}\} = \overline{g}' \setminus \{s_{i_j}\}$; and,
2. $s_{i_k} + s_{i_l} = s_{i_j}$.

Given this representation, the following processes are fundamental to the algorithm:

1. The manner in which the coalition values are bounded; and,
2. The way in which the remaining space of coalition structures are searched.

5.3.1 Bounding Coalition Values

In Step 1 of this algorithm, the maximum and the minimum values of each coalition C are computed. Following Theorem 5.1, for both $\mathcal{P}^{-}_{sup}$ and $\mathcal{P}^{+}_{sub}$ and for all coalitions $C \subseteq Ag$, this is achieved by computing $P(C; \pi_\alpha)$ and $P(C; \pi_\beta)$ in $\mathcal{E}$. As well as this, the value of all $\pi_\alpha, \pi_\beta \in \Pi$, where π_α, π_β are as described in Theorem 5.1 and Theorem 5.2, are also computed.

Both the maximum and minimum values of every coalition are then stored in memory. For notation, let $v^{max}(C)$ and $v^{min}(C)$ denote the maximum and minimum values of every coalition $C \subseteq Ag$, respectively. Specifically, in the spirit of the *distributed coalition value calculation* algorithm presented in [56], these values are stored so that all coalitions of the same size are grouped together and ordered with respect to the indices of the agents who make them up. In this context, for every coalition $C \subseteq Ag$, only $v^{max}(C)$ and $v^{min}(C)$ need be stored in memory and not the coalitions as well. This is because the coalitions can be determined from the place these values occupy in the group. Also, this input representation is an ideal representation from which maximum, minimum and average coalition values can be computed.

Additionally, whilst computing these embedded coalition values, the values of the structures in which these coalitions are embedded can also be computed, storing both the structure with biggest value (π^*_n) and its value $v(\pi^*_n)$ in memory.

For a system of six agents, Figure 5.3 displays the integer partition graph representation of a system of six agents, as well as, the subspaces that are searched in order to bound the coalition values.

5.3.2 Computing the Remaining Coalition Structure Values

For $s = 1, \ldots, n$, let $v^{max_s}(C)$ and $v^{min_s}(C)$ denote the maximum and minimum values of all coalitions of size s, respectively. When all maximum and minimum coalition values have been computed, all subspaces containing the structures that were analyzed to compute these values are pruned and the upper and lower bounds for the remaining configurations are computed.

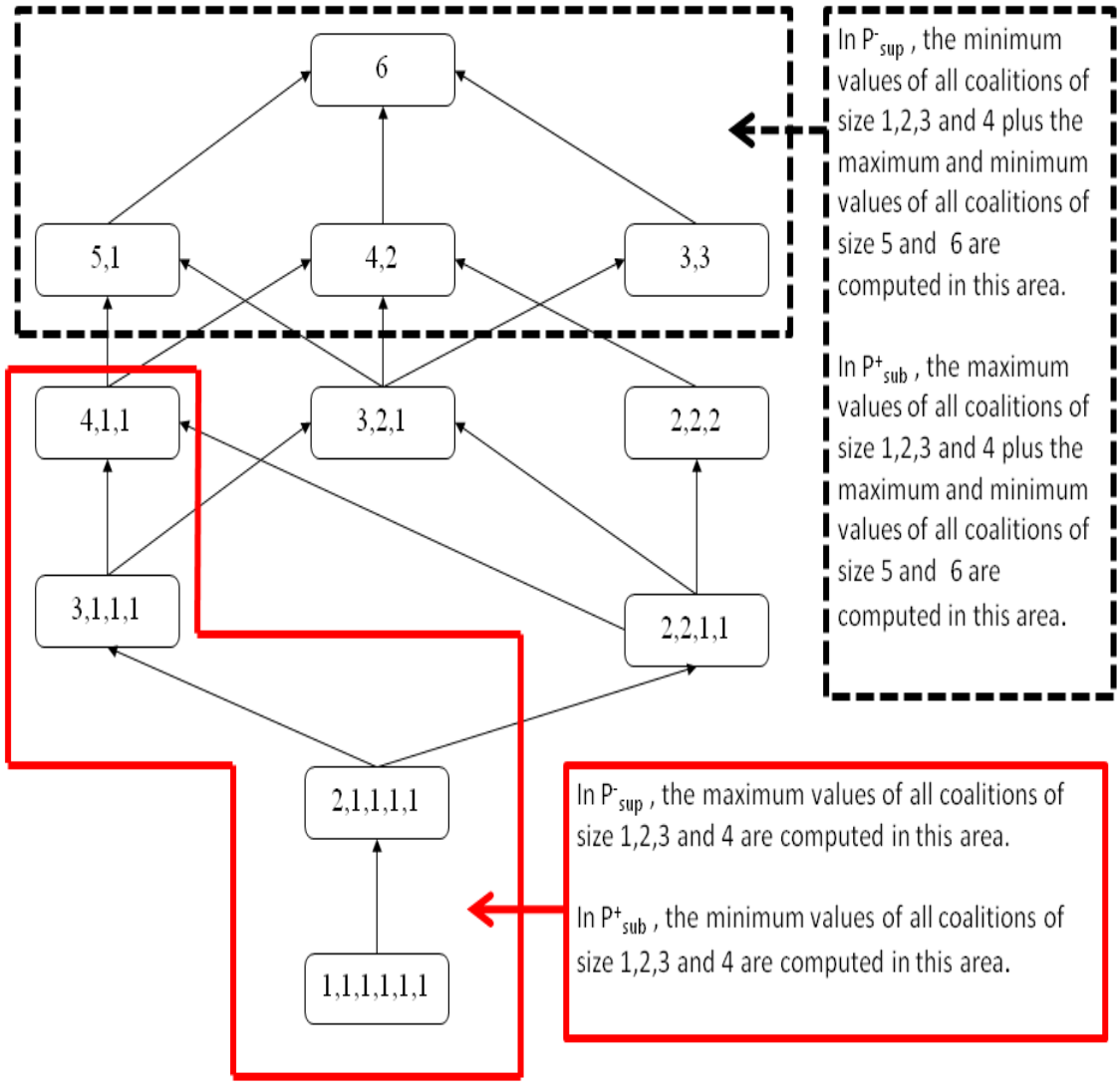


Figure 5.3: Configurations searched to bound coalition values in a six agent setting

For every configuration $\overline{g} = \{s_{i_1}, \ldots, s_{i_m}\}$, the upper bound ($UB_{\overline{g}}$) and lower bound ($LB_{\overline{g}}$) of $\overline{g}$ are computed as follows:

$$UB_{\overline{g}} = \sum_{j=1}^{m} v^{max_{s_j}}(C),$$

and,

$$LB_{\overline{g}} = \sum_{j=1}^{m} v^{min_{s_j}}(C).$$

Once these bounds have been computed for the remaining configurations, the upper bound ($UB_{\overline{G}}$) and lower bound ($LB_{\overline{G}}$) for the entire system $\overline{G}$ is computed as follows:

$$UB_{\overline{G}} = \text{ arg max}_{\overline{g} \in \overline{G}} UB_{\overline{g}},$$

and,

$$LB_{\overline{G}} = \text{ arg max}_{\overline{g} \in \overline{G}, \pi \in \Pi} \{ \sum_{C \in \pi_n^*} P(C; \pi_n^*), Av_{\overline{g}} \},$$

where, for each $\overline{g} = \{s_{i_1}, \ldots, s_{i_m}\} \in \overline{G}$,

$$Av_{\overline{g}} = \sum_{j=1}^{m} \text{ arg } av_{C \in s_{i_j}} \; v^{min_{s_{i_j}}}(C),$$

and arg $av_{C \in s_{i_j}} \; v^{min_{s_{i_j}}}(C)$ denotes the mean average of all the minimal values of all coalitions of size s_{i_j}.

Upon doing this, the remaining configurations whose upper bound is less than the lower bound for the entire system are pruned. The most promising configuration $\overline{g}^*$, *i.e.*, the configuration with highest upper bound, is then searched as in the IP algorithm presented in Section 3.2.3.

As with the IP algorithm, once the values of all coalition structures in this subspace have been computed, if a structure is found with value equal to $UB_{\overline{G}}$ then this is output as optimal. Otherwise, if the value of the optimal structure in $\overline{g}^*$ is less than $UB_{\overline{G}}$ then the upper and lower bounds of the system are updated as follows:

$$UB_{\overline{G}} = \text{ arg max}_{\overline{g} \in \overline{G}, \pi \in \Pi} \{ \sum_{C \in \pi_n^* \in \overline{g}^*} P(C; \pi_n^*), UB_{\overline{g}'} \},$$

and,

$$LB_{\overline{G}} = \text{ arg max}_{\overline{g} \in \overline{G}, \pi \in \Pi} \{ \sum_{C \in \pi_n^* \in \overline{g}^*} P(C; \pi_n^*), Av_{\overline{g}} \},$$

where:

1. $\sum_{C \in \pi_n^* \in g^*} P(C; \pi_n^*)$ is the biggest structure value in $\overline{g}^*$; and,
2. $UB_{\overline{g}'}$ is the upper bound of the next most promising subspace $\overline{g}' \in G \setminus \{\overline{g}^*\}$.

All configurations whose upper bound is less than the new lower bound are pruned and the next promising configuration is searched. This is repeated until all configurations have been searched or an optimal is found.

Suppose that $\overline{g}^* = \{s_{i_1}, \ldots, s_{i_m}\}$ is the configuration with the highest upper bound that has yet to be searched. Recall that, in characteristic function games, the IP algorithm employs a branch and bound rule which avoids evaluating coalition structure in $\overline{g}^*$ that cannot have value greater than $LB_{\overline{G}}$. Against this background, instead of exact coalition values, the maximum values, as computed in Step 1, can be used and incorporated into this rule for both $\mathcal{P}^-_{sup}$ and $\mathcal{P}^+_{sub}$ settings. In more detail, for $k = 1, \ldots, m-1$, given $C_1 \in L_{s_{i_1}}$,$C_2 \in L_{s_{i_2}}$, ..., $C_k \in L_{s_{j_k}}$, if,

$$\sum_{j=1}^{k} v^{max}(C_j) + UB_{s_{i_{j+1}}} + \ldots + UB_{s_{i_m}} < LB_{\overline{G}},$$

then any structure in g^* which contains all of C',C'', ..., C'^k cannot be optimal. The values of these coalition structures are, therefore, not computed. Of course, with only maximum values, such a branch and bound rule is likely to be less effective than in the characteristic function game settings where exact coalition values are used.

Optimal Coalition Structure Generation Algorithm For Partition Function Games.

Input: $P = \langle Ag, \mathcal{P} \rangle$

Step 1. Compute the value of the grand coalition. For every coalition C of size $s \in [1, \ldots, n]$, compute its value in the structures where;

(i) All the other agents not in C form coalition $C' = Ag \setminus C$; and,

(ii) Every other agent not in C acts alone.

These values, as well as the structure with highest value (π^*_n) plus its value ($v(\pi^*_n)$) , are stored in memory.

Step 2. Prune those configurations which were searched in Step 1.

Step 3. Compute the upper and lower bounds of every remaining configuration G, as well as the upper and lower bounds of the all the remaining configurations using the maximum and minimum coalition values from Step 1 plus the current optimal value.

Step 4. Prune away those subspaces which cannot deliver a coalition structure greater than $LB_{\overline{G}}$, *i.e.*, $UB_{\overline{g}} < LB_{\overline{G}}$;

Step 5. Search the configuration with highest upper bound using a refined version of the branch and bound rule employed by the IP algorithm.

Step 6. Once the search of the configuration in Step 5 is completed, check whether the value of the optimal structure in the most promising configuration is equal to $UB_{\overline{G}}$. If this is the case then output the structure as optimal. If this not the case but the value of this structure is greater than $v(\pi^*_n)$ then;

1. Update $v(\pi^*_n)$ to be the value of this structure and set this coalition structure to be the new optimal of the system;
2. Update the upper and lower bounds of the system; and,
3. Go to Step 4.

Output: π^*_n.

Algorithm 5.3.2: Refined IP Algorithm for Partition Function Games

5.4 Assessment of Algorithm

Following the work presented in the previous section, inherent to the effectiveness of the algorithm presented in this chapter is the manner in which the coalition values are bounded and the way in which the remaining coalition structure values are computed. Against this reasoning, as no existing optimal coalition structure generation algorithms have been developed for partition function games, the algorithm is assessed through answering the following questions:

Q1 How many coalition structure values must be analyzed in order to bound the coalition values? and,

Q2 Once all coalition values have been bounded, how many coalition structure values must be computed to generate an optimal coalition structure?

These two questions are addressed in the following two sub-sections.

5.4.1 Complexity of Bounding Coalition Values

Consider **Q1** first. This question is answered in the following theorem.

Theorem 5.3 *In either $\mathcal{P}^{+}_{sup}$ or $\mathcal{P}^{-}_{sub}$, exactly,*

$$2^n + 2^{n-1} - 2n - 1,$$

structures will be analyzed in Π in order to compute the maximum and minimum values of all coalitions $C \subseteq Ag$.

Proof: To prove this theorem, consider the following subsets of Π,

- $\Pi' \subset \Pi$ which contains all structures of form $\{C, \{a_1\}, \ldots, \{a_{n-|C|}\}\}$ where $\{a_1, \ldots, a_{n-|C|}\} = Ag \setminus C$ (excluding all structures where $C = Ag$); and,
- $\Pi'' \subset \Pi$ which contains all structures of form $\{C, C'\}$ where $C' = Ag \setminus C$ and both C and C' have size greater than two (including the structure where $C = Ag$ and excluding the structures where $|C| = n - 1$).

The values of all structures in $\Pi' \cup \Pi''$ will be computed exactly once in order to determine the maximum and minimum coalition values. Observe that, every coalition of size $2, \ldots, n-1$ appears in exactly one structure in Π'. Thus, in Π', all $C \subset Ag$ such that $|C| \in [2, \ldots, n-1]$ belong to exactly

$$(2^n - 1) - (n + 1),$$

structures in Π'. Since there is also one structure in Π' that exclusively contains coalitions of size one, it follows that,

$$|\Pi'| = (2^n - 1) - (n + 1) + 1 = (2^n - 1) - n.$$

Now, consider all coalition structures in Π''. From [66], there are 2^{n-1} such structures. However, all structures where $|C| = n - 1$ and $|C'| = 1$ can be ignored since they also belong to Π'. Since there are $\binom{n}{n-1} = n$ such structures, it follows that,

$$|\Pi''| = 2^{n-1} - n.$$

Therefore, in order to bound all coalition values,

$$((2^n - 1) - n) + (2^{n-1} - n) = 2^n + 2^{n-1} - 2n - 1,$$

n	$\lvert\Pi\rvert$	Number of structure values computed to bound coalition values	Column 3 expressed as a percentage of $\lvert\Pi\rvert$
5	52	37	71.15
6	203	83	40.89
7	877	177	20.18
8	4,140	367	8.86
9	21,147	749	3.54
10	115, 975	1,515	1.31
11	678.570	3,049	0.45
12	4,213,597	6,119	0.15
13	27,644,437	12,261	0.04
14	190,899,322	24,547	0.01
15	1,382,958,545	49,121	0.004
16	10,480,142,147	98,271	0.0009
17	82,864,869,804	196,573	0.0002

Table 5.4.1: Comparing the number of coalition structures analyzed in order to bound coalition values relative to the number of all coalition structures

coalition structure values will be computed in $\Pi' \cup \Pi''$. This completes the proof. ∎

Theorem 5.3 proves that in either $\mathcal{P}^{-}_{sup}$ or $\mathcal{P}^{+}_{sub}$ settings, the maximum and minimum values of all coalitions can be determined through computing the values of exactly,

$$2^n + 2^{n-1} - 2n - 1,$$

coalition structures.

Table 5.4.1 compares the number of coalition structures analyzed in order to bound all coalition values relative to the total number of coalition structures that can be formed. This table shows that as n linearly increases, the percentage of Π that is analyzed in order to bound the coalition values exponentially decreases. This means that, even for moderate values of n, once all coalition values have been bounded, there are still exponentially many structure values that may have to be computed.

Table 5.4.1 also provides an indication of the best case performance of this algorithm as no fewer coalition structure values can be computed in order to bound the coalition values. Since the process of bounding the value of all coalitions is fundamental to the algorithm, in the best case, this algorithm will generate an optimal coalition structure during this stage.

5.4.2 Number of Coalition Structure Values Computed

Against Theorem 5.3, exactly $2^n + 2^{n-1} - 2n - 1$ coalition structure values must be computed in order to bound the values of all coalitions. After this, the number of coalition structure values computed in the remainder of Π is dependent upon the percentage of Π that is pruned, as well as the effectiveness of the branch and bound filter rule.

Intuitively, it may be that no configurations are pruned away and that every structure within each configuration is analyzed. Consequently, in the worst case, all of Π will be searched in order to generate π^*. However, as this is a worst case, it may be that, in the general case, only the values of those coalition structures in a

fraction of Π are computed.

The algorithm is assessed in $\mathcal{P}^{-}_{sup}$ settings with 10 agents. Here, there are two factors which influence the value of a coalition in a given structure, namely:

- the effect of the super-additivity; and
- the effect of the externality.

These simulations were implements on MATLAB and repeated 25 times for a number of values of a and b. In these simulations, when a new coalition is formed, the 'gain' from super-additivity is accounted for by adding a factor $\frac{\alpha}{a}$ to its value. Conversely, the 'loss' from the externality on the other coalitions in the structure is accounted for by multiplying their values by factors $\frac{b-\beta}{b}$, where $\alpha, \beta \in [0, 1)$ are randomly-generated uniform variables and $a, b \geq 1$ are constants.

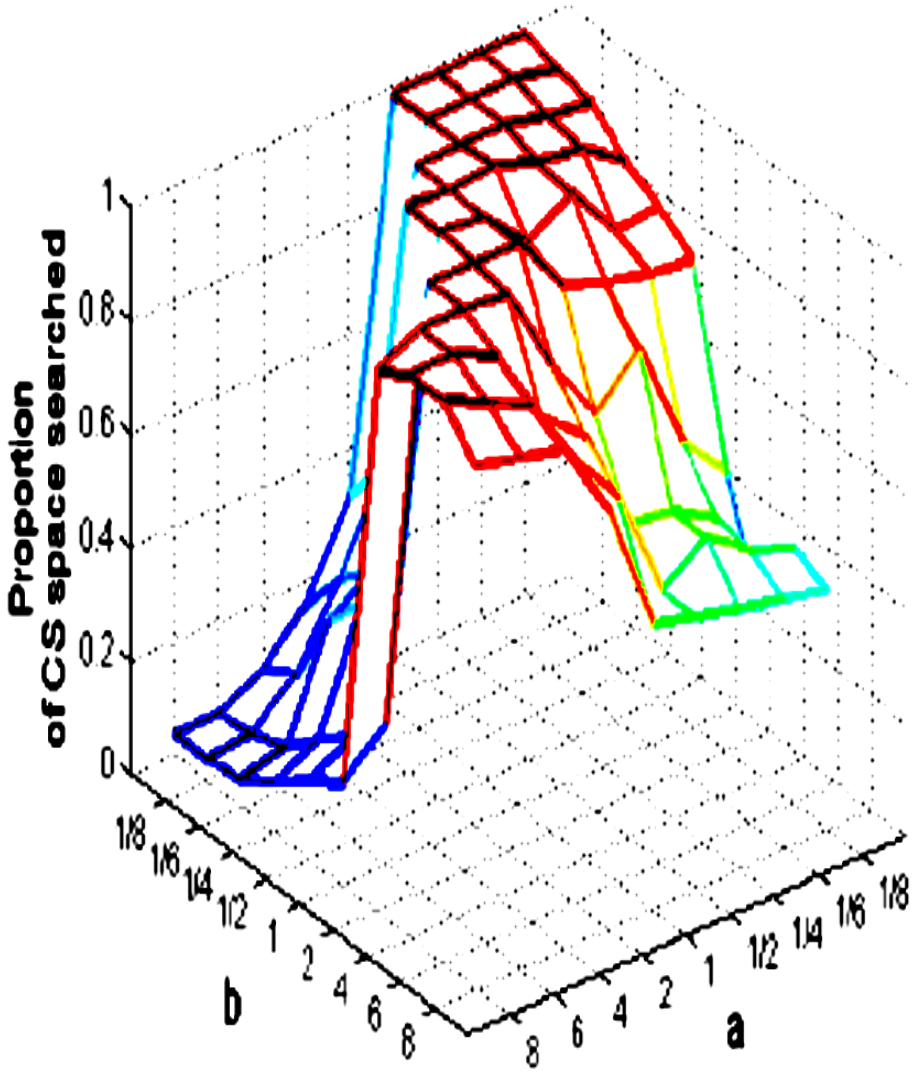
The results of the simulations are presented in Graph 5.4.2. The vertical axis in Graph 5.4.2 represents the proportion of Π searched, whereas a and b are indicated on the x and z axes, respectively. As the values of a and b increase, the 'gain' from super-additivity and the 'loss' from externalities decreases. The surface shown in Graph 5.4.2 is the average proportion of space searched by the algorithm.

Observe that when the 'gain' from super-additivity is high and the 'loss' from the negative externality is low, only a minimal proportion (under 4 $\%$) of the space need be searched in order to compute the optimal structure. In fact, in such cases, the grand coalition or a coalition structure containing only two or three coalitions is usually optimal. Consequently, it would seem that the smaller the externality and the higher the super-additivity, the more the $\mathcal{P}^{-}_{sup}$ setting becomes like a super-additive characteristic function game, thus explaining why so little of the space is searched.

Similarly, when the converse is true, *i.e.*, when the 'gain' from super-additivity is low and the 'loss' from the negative externality is high, only a fraction of the search space was searched. This time, the $\mathcal{P}^{-}_{sup}$ setting becomes more akin to the sub-additive characteristic function game setting, where the structure exclusively consisting of coalitions of size one or a structure with a relatively small number of cooperating agents is optimal. However, as is apparent from the gradient of the graph, a greater proportion of Π is searched when a is low and b is high than when a is high and b is low. This is because, in the latter case, an optimal coalition structure usually did not belong to the group of structures evaluated during the process of bounding the coalition values. Thus, in the latter case, the representation did not become totally akin to a sub-additive characteristic function game which implies that, for all values a and b, the 'gain' that super-additivity contributes to the coalition value is significantly greater than the 'loss' the negative externality takes away from the coalition value.

Now, in situations where the 'loss' from the externality and the 'gain' from the super-additivity are both either high or low, it seems that pruning is ineffective since nearly all of the search space has to be searched in order to guarantee an optimal outcome (more than 98% in many cases). This could be due to the inherent characteristic of the $\mathcal{P}^{-}_{sup}$ setting, namely, the values of the structures in each subspace are dependent on the value of the structures in the subspace from which it was formed. This implies that when the 'gain' from the super-additivity and the loss from externalities are of a similar magnitude, the extreme values of coalition structures in different subspaces are more likely to be akin, making pruning techniques less effective.

Finally, it is worth noting that, as Π is searched in the manner of the IP algorithm, the algorithm presented in this chapter does not lose any of the anytime properties which are inherent to the IP algorithm. Thus, this algorithm is robust against technical failure. Additionally, with regards to memory, for every coalition $C \subseteq Ag$, exactly two values- $v^{max}(C)$ and $v^{min}(C)$- as well as the optimal coalition structure and its value must be stored in memory. Therefore, there will be $2^{n+1} + 1$ values stored in memory plus exactly one coalition structure. Relative to characteristic function game representations, this is more than double the memory space.



Graph 5.4.2: Simulation results for PF^{-}_{sup} setting

5.5 Summary

In this chapter, optimal coalition structure generation was considered in partition function games. In particular, for a general partition function game, it was shown that when nothing is known about the nature of the externalities or the partition function, it is not possible to determine *a priori* the value of a coalition in every structure to which it is embedded meaning, to guarantee an optimal coalition structure, all of Π must be searched.

Against this reasoning, an optimal coalition structure generation algorithm was developed which, for two natural classes of partition function games, was able to bound all coalition values after searching only a fraction of Π. When these values were known, the remaining structures in Π could be analyzed in the spirit of the IP algorithm, avoiding all structures which could not be optimal. Specifically, this algorithm was developed for partition function games which displayed both super-additivity and negative externalities ($\mathcal{P}^{-}_{sup}$) as well as, partition function games which displayed both sub-additivity and positive externalities ($\mathcal{P}^{+}_{sub}$). The minimum and maximum number of structures searched by this algorithm was computed.

It was argued that the performance of this algorithm was affected by the size of the externality, as well as the size of the 'gain' or 'loss' from super- or sub-additivity. For $\mathcal{P}^{-}_{sup}$, when the 'gain' from super-additivity is low and the 'loss' from the negative externality is high or, for $\mathcal{P}^{+}_{sub}$, the 'loss' from sub-additivity is low but the 'gain' from super-additivity high then only a fraction of Π was searched. It was reasoned that this was because, in these instances, the partition function game representation is similar to either a sub-additive or super-additive characteristic function game and an optimal coalition structure can be found. Simulations seem to support this reasoning.

Chapter 6

Coalition Structure Generation in Hedonic Qualitative Coalitional Games

In this chapter, *hedonic qualitative coalitional games* (HQCG) - a novel class of cooperative games that represent coalition formation between self-interested agents - are introduced. In the spirit of both hedonic and qualitative coalitional games, these representations can be used to model coalition formation in domains where self-interested agents, each with preferences over the other agents, can cooperate to accomplish any of their individual goals.

HQCGs can capture coalition formation in a number of multi-agent systems. For example, as discussed in the introduction, consider an electronic market populated with automated agents which represent different enterprises who buy and sell [34]. Here, buyers and sellers have different goals they wish to accomplish, *i.e.*, the goals of the seller may revolve around improving their profit whereas the goals of the buyer may revolve on minimizing the amount of money they spend. Now, a number of electronic markets employ reputation systems where buyer agents rate seller agents based on past experiences of cooperating with them (such as *Amazon* or *e-bay*). In such systems, agents can form preferences over other agents based on this reputation, meaning agents can make decisions regarding coalition formation based on both their preferences and whether there exists a mutually beneficial set of goals the coalition is able to accomplish. Thus, HQCGs can capture coalition formation in these systems.

Also, consider coalition formation between self-interested agents operating in a multi-agent system that uses the *Regret* structure (as presented in [64]). Typically, such a system contains buyer and seller agents who use social networking analysis techniques to identify relations between agents. Using this feedback, agents can then formulate preferences over other agents in the system. Following reasoning presented in the paragraph above, in those systems where changes in reputation over time rarely occur, HQCGs can model coalition formation in these systems.

Against previous reasoning, it cannot be guaranteed that the agents in a system represented by a HQCG will cooperate so that the welfare of the system is maximized. However, it can be guaranteed that agents will form coalitions that are stable. Thus, in this chapter, the core, Nash, individual and contractual individual stability concepts presented in Section 2.4.5 are formalized for these representations.

Specifically, this chapter studies only those HQCGs which are naturally concise and in which assumptions regarding the preferences of the agents hold. Against these assumptions, there is always guaranteed to exist at least one contractual individual stable structure. However, this guarantee cannot be extended to either Nash, individual or core stability (recall, this is in contrast to hedonic coalitional games which guarantee at least one contractual individual and at least one individual stable structure). Nevertheless, in the games studied in this chapter, it is shown that all core stable structures are also Nash, individual and contractual individual stable (recall, this is not always true in hedonic coalitional games) and, even if the representation

is concise, no algorithm can solve decision problems concerning non-emptiness of the core with time complexity that is polynomial in the size of the representation.

Core stability is especially important in understanding coalition formation in HQCGs since if a structure is core stable then the agents can do no better than partition themselves into this structure. Therefore, in representations where there are no core stable structures it is not immediately apparent which coalitions will be formed by the agents. To this end, a sequential coalition formation protocol is presented for which an equilibrium strategy can be computed *via* backward induction. It is shown that, if the agents play according to the equilibrium strategy then the formed structure is core stable if and only if there exists a core stable structure in the HQCG. However, even if the HQCG representation is concise in both the number of goals and agents, for large numbers of agents and goals, this strategy can be exponentially complex to compute.

Given this insight, motivated from real world examples, a natural class of HQCGs are studied. It is shown that, in this class of games, if the aforementioned assumptions still hold then there always exists a core (and also Nash, individual and contractual individual) stable structure and this structure is always formed if the agents play the equilibrium strategy in the protocol. It is also shown that the equilibrium strategy can be computed without executing backward induction and an algorithm is developed which can compute this strategy with time complexity that is polynomial in the size of the representation.

Against this discussion, the rest of this chapter goes as follows:

- In Section 6.1, the hedonic qualitative coalitional game framework is formally defined and stability concepts for this framework are formalized. Also, assumptions regarding the size of the representation are formally presented. These assumptions can enable conciseness and only concise representations are considered;
- In Section 6.2, the stability concepts are formally defined for the HQCG represntation. Even with the preference and conciseness assumptions, it is shown that core and Nash stability cannot be guaranteed and that computing non-emptiness of the core cannot be done with time complexity that is polynomial in the size of the representation;
- In Section 6.3, the sequential protocol is formally presented. It is shown that every agent is motivated to participate in the protocol since an equilibrium strategy is guaranteed to exist. However, computing this strategy can require time complexity that is exponential in the number of agents and goals; and,
- In Section 6.4, motivated from real world examples, a natural class of hedonic qualitative coalitional games are defined. For these class of games, it is shown that there always exists a Nash and core stable structure. Furthermore, if the agents participate in the sequential protocol and play the equilibrium strategy then the formed structure is always core and Nash stable. Also, this equilibrium strategy can be computed with time complexity that is polynomial in the size of the representation.

Before the rest of the chapter is considered, to refresh the memory of the reader, the qualitiative coalitional game and hedonic coalitional game representations considered in this chapter (as formally defined in Definition 2.26 and Definition 2.19, respectively) are defined once again:

Definition 6.1 *A QCG is a* $(n+3)$*-tuple* $\Gamma = \langle G, Ag, G_1, \ldots, G_n, v \rangle$*, where:*

- $G = \{g_1, \ldots, g_m\}$ *is a set of possible goals;*
- $Ag = \{a_1, .., a_n\}$ *is a set of agents;*
- $G_i \subseteq G$ *is a set of goals for each agent* $a_i \in Ag$*; and,*
- $v : 2^{Ag} \rightarrow 2^{2^G}$ *is a function which takes, as input, a coalition and outputs a set of subsets of the goals in* G*.*

Definition 6.2 *A hedonic game is a tuple* $\mathcal{H} = \langle Ag, \{\succ_i\}_{\forall i \in Ag} \rangle$ *where:*

- $Ag = \{a_1, \ldots, a_n\}$ *are the set of agents; and,*
- *Every* $\succ_i$ *is a rational preference relation over coalitions* $C \subseteq Ag$ *such that* $a_i \in C$.

6.1 Hedonic Qualitative Coalitional Games (HQCGs)

Given both Defintion 6.1 and 6.2, hedonic qualitative coalitional game representations are defined as follows.

Definition 6.3 *A hedonic qualitative coalitional game (HQCG) is a* $(2n + 3)$*-tuple,*

$$\Gamma_{\mathcal{H}} = \langle G, Ag, G_1, \ldots, G_n, v, \succeq_1, \ldots, \succeq_n \rangle$$

where:

- $G = \{g_1, \ldots, g_m\}$ *is a set of possible goals;*
- $Ag = \{a_1, \ldots, a_n\}$ *is a set of agents;*
- *For* $i = 1, \ldots, n$, $G_i \subseteq G$ *is a set of goals for each agent* $a_i \in Ag$*;*
- $v : 2^{Ag} \rightarrow 2^{2^G}$ *is a function which takes, as input, a coalition and outputs a set containing sets of the individual goals in* G*; and,*
- *For* $i = 1, \ldots, n$, $\succeq_i$ *is the preference ordering of agent* $a_i \in Ag$ *over all of the agents in* $Ag \setminus \{a_i\}$.

With respect to the preference orderings $\succeq_1, \ldots, \succeq_n$, given any two agents $a_j, a_k \in Ag \setminus \{a_i\}$, if $a_j \succ_i a_k$ then agent a_i *strictly prefers* a_j to a_k whereas if $a_j \sim_i a_k$ then agent a_i is *indifferent between* a_j and a_k. In this way, these preference orderings define whom the agents strictly prefer or are indifferent between.

Intuitively, $\Gamma_{\mathcal{H}}$ combines features from both the QCG framework Γ and the hedonic game representation $\mathcal{H}$. For every coalition $C \subseteq Ag$ in $\Gamma_{\mathcal{H}}$, $v(C)$ outputs a set containing sets of goals with the interpretation being that each set of goals represents a choice available to that coalition. As with qualitative coalitional games, the following assumptions are inherent to all $\Gamma_{\mathcal{H}}$:

1. Every agent $a_i \in Ag$ is *indifferent* between the goals in their own set G_i, meaning they would be *satisfied* if they accomplished *any* of these goals; and,
2. The goals are not public, meaning any agent a_i is satisfied if and only if they accomplish a non-empty set of goals that contains any of the goals in G_i.

6.1.1 Preferences in Hedonic Qualitative Coalitional Games

In $\Gamma_{\mathcal{H}}$, agents can construct preferences over the coalitions they can form using both the preference orderings $\succ_1, \ldots, \succ_n$ in $\Gamma_{\mathcal{H}}$, as well as the set of choices available to every coalition. To formalize these preferences, it is assumed that the agents cooperate in *teams*.

Definition 6.4 *A* team *is a pair* $T = (C, G')$ *consisting of a coalition* $C \subseteq Ag$ *and a set of goals* $G' \subseteq G$ *such that* $G' \in v(C)$.

As with qualitative coalitional games (see Section 2.4.6) :

- A team $T = (C, G')$ is *successful* if G' contains any of the goals of every member of C; and
- A team $T = (C, G')$ is *minimal* and *successful* if T is successful and all $C' \subseteq C$ do not belong to successful teams.

To describe how the preferences of the agents as given in $\succ_1, \ldots, \succ_n$ imply preferences over the teams they can form, let:

- $\mathcal{T}$ denote the set of all minimal and successful teams that can be formed;
- $\mathcal{T}_i \subseteq \mathcal{T}$ denote the set of all minimal and successful teams that can be formed by an agent $a_i \in Ag$;
- $T \rhd_i T'$ denote that a_i strictly prefers team T to team T';
- $T \trianglerighteq_i T'$ denote that a_i strictly prefers or is indifferent between T and T'; and,
- $T \sim_i T'$ denote that a_i is indifferent between T and T'.

For any two teams (C, G') and (C', G'') where $a_i \in C \cap C'$, the preferences of agent a_i over these teams can be formulated as follows. If,

$$\exists a_k \in C \setminus (C \cap C') \text{ such that } \forall a_m \in C' \setminus (C \cap C'), a_k \succ_i a_m,$$

then,

$$(C, G') \rhd_i (C', G'').$$

Following discussions presented in Section 2.4.5, the preferences of the agents in $\succ_i$ imply preferences over the teams each agent can form in the spirit of $\mathcal{B}$-preferences. Of course, if $C = C'$ or the most preferred agent of a_i in C is also a_i's most preferred agent in C' then,

$$(C, G') \sim_i (C', G'').$$

Now, with regards to the set of choices available available to the coalitions, in all of the HQCGs studied in this chapter, it is also assumed that the agents will adhere to the following criteria when constructing their preferences over the teams they can form:

P1 Every agent will prefer all successful teams to all unsuccessful teams; and,

P2 With respect to the size of the coalition within them, agents will prefer smaller teams to larger teams.

Recall, a successful team is one where all of the agents within it cooperate to accomplish a set of goals that contain any of the goals from every team member's individual set. Obviously, no rational agent will form a team in which they are not satisfied as none of their individual goals will be accomplished. Therefore, rational agents will not form unsuccessful teams and, against this reasoning, it is assumed that **P1** holds.

The motivation for assumption **P2** stems from the fact that, in many settings, smaller teams are typically more reliable than larger ones from a technical perspective, *e.g.*, there is lower probability of technical failure and the enforcement of cooperation is easier. For instance, in the context of political science, systems containing agents representing countries in the European Monetary Union (EMU), as considered in Chapter 5 of [53], make this assumption. This assumption is particularly valid in HQCGs since, with smaller coalitions, there is less chance of an agent defecting to cooperate with an agent it prefers more.

Given **P2**, every $a_i \in Ag$ will prefer minimal and successful teams to successful but not minimal teams in $\mathcal{T}_i$. This means that successful but not minimal teams will not be formed since, for all such teams (C, G'), there is a subset $C' \subset C$ and a set of goals $G'' \in v(C')$ such that every agent $a_i \in C'$ prefers $(C'G'')$ to (C, G'). In this way, both **P1** and **P2** imply that *Riker's size principle* is true, *i.e.*, agents will only form minimal and successful teams [61].[1] In this way, if both **P1** and **P2** hold then the agents will only formulate

[1]Formally, this principle states that, in the context of political science, only minimal and winning coalitions will be formed by the voting agents. However, it is argued in [48], that this principle is true in a number of games which are not in the context of political science. For more details regarding this principle, see [61].

have preference over the minimal and successful teams they can form.

In addition to these assumptions, in the HQCGs studied in this chapter, it is also assumed that, for all $a_i \in Ag$, $v(\{a_i\}) = \emptyset$. This is because, against **P1** and **P2**, if an individual agent can successfully accomplish any of their goals then the agents will always form this coalition. Consequently, the coalition formation problem is trivial for these agents. Therefore, this assumption implies that the agents can only accomplish their goals if they cooperate with others.

Now, it may be that, during the coalition formation process, as all of the other agents form teams, an agent $a_i \in Ag$ may not be able to join any of their minimal and successful teams in $\mathcal{T}_i$. When this is the case, to provide a_i with a reasonable course of action, consider Definition 6.5.

Definition 6.5 *For any agent $a_i \in Ag$, the* null team *is the one where a_i, as a team consisting of itself alone, accomplishes an empty set of goals* $(\{a_i\}, \emptyset)$.

Against Definition 6.5, it is understood that agents would prefer to form null teams than cooperate in teams that do not satisfy them. This has interpretation that agents would rather do nothing than cooperate to accomplish a set of goals that do not contain any of their own. It is clear from the above discussions that if the above assumptions hold then, when given Γ_H, the agents are able to construct preferences over the minimal and successful teams they can form.

6.1.2 Conciseness Assumptions

Clearly, agents must construct their preferences over the the teams they can form through analyzing every set of goals every coalition can accomplish. From a conciseness perspective, for every coalition $C \subseteq Ag$, $v(C)$ may contain a number of sets of goals that is exponential in the number of individual goals. This means that, in a number of instances of $\Gamma_{\mathcal{H}}$, agents may have to construct their preferences by analyzing a number of coalitions and sets of goals that are exponential in both the number of individual agents and goals. Thus, for large numbers of agents and goals, it may be infeasible for computationally bounded agents to explicitly define their preferences.

To this end, in this chapter, coalition structure generation is only considered in those frameworks which are naturally concise in both the number of agents and goals. In particular, the HQCGs studied in this chapter are those where the following assumptions hold:

HQCG1 The number of coalitions C such that $v(C) \neq \emptyset$ is polynomial in the number of agents; and,

HQCG2 For every coalition $C \subseteq Ag$, the number of sets of goals in every $v(C)$ is bounded so that $|v(C)|$ is polynomial in the number of goals.

Clearly, both **HQCG1** and **HQCG2** ensure that $\Gamma_{\mathcal{H}}$ will have size that is polynomial in both the number of agents and goals. Consequently, although it is not always explicitly stated, when referring to the framework $\Gamma_{\mathcal{H}}$, it is assumed that **HQCG1** and **HQCG2** hold and that both the number and size of all non-empty $v(C)$ are such that the agents can efficiently construct their preferences.

6.2 Stability Concepts

In the spirit of hedonic coalitional games, stability concepts for HQCGs are formulated in terms of partitions of agents. To this end, in $\Gamma_{\mathcal{H}}$, agent partitions are formulated in terms of the *team structure* concept which is defined as follows.

Definition 6.6 *A* team structure $\pi = \{T_1, \ldots, T_k\} = \{(C_1, G'_1), \ldots, (C_k, G'_k)\}$ *is a collection of teams such that the coalitions of agents within these teams form a partition of* Ag.

To define solution concepts for $\Gamma_{\mathcal{H}}$, the stability concepts defined for hedonic representations are reformulated so that they are applicable to teams. To achieve this, in addition to the previous notation, let:

- Π_T denote the set of all team structures that can be formed in $\Gamma_{\mathcal{H}}$;
- $T_j(\pi)$ denote the team in π to which each agent $a_j \in Ag$ belongs, $\forall a_j \in Ag$; and,
- $a_j \in T$ denote the fact that a_j belongs to the coalition in team T.

Now, consider the following definition.

Definition 6.7 *Any team* $T = (C, G') \in \mathcal{T}$ blocks π *if and only if,*

$$T \rhd_j T_j(\pi), \forall a_j \in T$$

Intuitively, if a team T blocks a structure $\pi \in \Pi_T$ then π cannot be stable as all $a_j \in T$ will prefer to form T as opposed to $\mathcal{T}_j(\pi)$. Conversely, if no $T \in \mathcal{T}$ blocks a structure π then this implies stability in π.

Definition 6.8 *A team structure* π *is* Core stable *if there does not exist* $T \in \mathcal{T}$ *such that* T *blocks* π.

In addition to core stability, team structures may be individually rational.

Definition 6.9 *A team structure* π *is* individually rational *if* $\forall a_i \in Ag$, $\forall G' \in v(\{a_i\})$,

$$T_i(\pi) \trianglerighteq_i (\{a_i\}, G').$$

Trivially, core stability implies individual rationality. As well as both core stability and individual rationality, the Nash, individual and contractually individual stability concepts are defined as follows.

Definition 6.10 *A team structure* π *is* Nash stable *if* $\forall a_i \in Ag$, $\forall (C', G') \in \pi \setminus \{T_i(\pi)\}$,

$$\nexists G'' \in v(C' \cup \{a_i\}) \text{ such that } (C' \cup \{a_i\}, G'') \rhd_i T_i(\pi).$$

In words, π is Nash stable if every agent $a_i \in Ag$ joins any of the coalitions in the other teams in π then there is no set of goals this coalition can accomplish such that a_i prefers the resulting team to $T_i(\pi)$.

Definition 6.11 *A team structure* π *is* individually stable *if* $\forall a_i \in Ag$, $\forall (C', G') \in \pi \setminus \{T_i(\pi)\}$,

$$\nexists G'' \in v(C' \cup \{a_i\}) \text{ such that } (C' \cup \{a_i\}, G'') \rhd_i T_i(\pi),$$

and,

$$(C' \cup \{a_i\}, G'') \rhd_j (C', G'), \forall a_j \in C.$$

In this context, π is *individually stable* if every agent $a_i \in Ag$ joins any of the coalitions in the other teams in π then there is no set of goals this coalition can accomplish such that a_i and every agent a_j in this coalition prefers the resulting team to $T_i(\pi)$ and $T_j(\pi)$, respectively.

Definition 6.12 *For every* $a_i \in Ag$, *let* C_i *denote the coalition in* $T_i(\pi)$. *A team structure* π *is* contractually individually stable *if* $\forall a_i \in Ag$, $\forall (C', G') \in \pi \setminus \{T_i(\pi)\}$,

$$\nexists G'' \in v(C' \cup \{a_i\}) \text{ such that } (C' \cup \{a_i\}, G'') \rhd_i T_i(\pi),$$

and,

$$(C' \cup \{a_i\}, G'') \rhd_j (C, G'), \forall a_j \in C,$$

and,

$$\nexists G''' \in v(C_i \setminus \{a_i\}) \text{ such that } (C_i \setminus \{a_i\}, G''') \rhd_k T_i(\pi), \forall a_k \in C_i \setminus \{a_i\}.$$

Intuitively, π is *contractually individually stable* if every agent $a_i \in Ag$ joins any of the coalitions in the other teams in π then there is no set of goals this coalition, or the coalition in $T_i(\pi)$ can accomplish so that a_i, every agent a_j in this coalition and every $a_k \in T_i(\pi) \setminus \{a_i\}$ prefer the resulting teams to $T_i(\pi)$ and $T_j(\pi)$.

As with hedonic coalitional games, Nash stability implies individual stability and, in turn, individual stability implies contractual individual stability.

6.2.1 Membership and Non-emptiness of Stability Concepts

Trivially, since $v(\{a_i\}) = \emptyset \; \forall a_i \in Ag$, all $\pi \in \Pi_T$ are individually rational. However, with regards to the other solution concepts presented in Section 6.2, even with assumptions **HQCG1**, **HQCG1**, **P1** and **P2**, there is no guarantee that the set of core and Nash stable team structures will be non-empty. For instance, consider the following example.

Example 6.1 *Consider $\Gamma_{\mathcal{H}}$ where:*

- $Ag = \{a_1, a_2, a_3\}$*;*
- $G = \{g_1, g_2, g_3\}$*;*
- *For* $i = 1, \ldots, 3$*,* $G_i = \{g_i\}$*;*
- $\succ_1$: $a_2 \succ a_3$*;*
- $\succ_2$: $a_3 \succ a_1$*;*
- $\succ_3$: $a_1 \succ a_2$*; and*
- $v(C) = \{\{g_1, g_2, g_3\}\}$*,* $\forall C \subseteq Ag$ *such that* $|C| = 2$ *and* $v(C') = \emptyset$ *for all other coalitions* C'*.*

*Clearly, all teams containing a coalition of size two and a set of goals $\{g_1, g_2, g_3\}$ are minimal and successful. Therefore, given this framework, against both **P1** and **P2**, the agents will construct the following preferences over the teams they can form:*

1. $\mathcal{T}_1 = (\{a_1, a_2\}, \{g_1, g_2, g_3\}) \rhd_1 (\{a_1, a_3\}, \{g_1, g_2, g_3\}) \rhd_1 (\{a_1\}, \emptyset)$ *;*
2. $\mathcal{T}_2 = (\{a_2, a_3\}, \{g_1, g_2, g_3\}) \rhd_2 (\{a_1, a_2\}, \{g_1, g_2, g_3\}) \rhd_2 (\{a_2\}, \emptyset)$ *; and,*
3. $\mathcal{T}_3 = (\{a_1, a_3\}, \{g_1, g_2, g_3\}) \rhd_3 (\{a_2, a_3\}, \{g_1, g_2, g_3\}) \rhd_3 (\{a_3\}, \emptyset)$.

Against **P1** and **P2**, observe that:

$$\Pi_T = \left\{ \begin{array}{c} \{\{(\{a_1, a_2\}, \{g_1, g_2, g_3\})\}, \{(\{a_3\}, \emptyset)\}\}; \\ \{\{(\{a_1, a_3\}, \{g_1, g_2, g_3\})\}, \{(\{a_2\}, \emptyset)\}\}; \\ \{\{(\{a_2, a_3\}, \{g_1, g_2, g_3\})\}, \{(\{a_1\}, \emptyset)\}\}; \text{ and,} \\ \{\{(\{a_1\}, \emptyset)\}, \{(\{a_2\}, \emptyset)\}, \{(\{a_3\}, \emptyset)\}. \end{array} \right\}$$

Given Π_T, also observe that:

1. $\{\{(\{a_1, a_2\}, \{g_1, g_2, g_3\})\}, \{(\{a_3\}, \emptyset)\}\}$ is blocked by the team $(\{a_2, a_3\}, \{g_1, g_2, g_3\})$;
2. $\{\{(\{a_1, a_3\}, \{g_1, g_2, g_3\})\}, \{(\{a_2\}, \emptyset)\}\}$ is blocked by the team $(\{a_1, a_2\}, \{g_1, g_2, g_3\})$;
3. $\{\{(\{a_2, a_3\}, \{g_1, g_2, g_3\})\}, \{(\{a_1\}, \emptyset)\}\}$ is blocked by the team $(\{a_1, a_3\}, \{g_1, g_2, g_3\})$; and,
4. $\{\{(\{a_1\}, \emptyset)\}, \{(\{a_2\}, \emptyset)\}, \{(\{a_3\}, \emptyset)\}\}$ is blocked by all non-null teams.

Thus, there are no core stable team structures in this example. As well as core stability, emptiness of the Nash stable structure can be witnessed as follows:

1. $\{\{(\{a_1\}, \emptyset)\}, \{(\{a_2\}, \emptyset)\}, \{(\{a_3\}, \emptyset)\}\}$ cannot be Nash stable since any agent transfer leads to a team that every agent prefers to its null team;
2. $\{\{(\{a_1, a_2\}, \{g_1, g_2, g_3\})\}, \{(\{a_3\}, \emptyset)\}\}$ cannot be Nash stable since a_2 can transfer from $(\{a_1, a_2\}, \{g_1, g_2, g_3\})$ to $(\{a_3\}, \emptyset)$ and $(\{a_2, a_3\}, \{g_1, g_2, g_3\}) \rhd_2 (\{a_1, a_2\}, \{g_1, g_2, g_3\})$;

3. $\{\{(\{a_1, a_3\}, \{g_1, g_2, g_3\})\}, \{(\{a_2\}, \emptyset)\}\}$ cannot be Nash stable since a_1 can transfer from $(\{a_1, a_3\}, \{g_1, g_2, g_3\})$ to $(\{a_2\}, \emptyset)$ and $(\{a_1, a_2\}, \{g_1, g_2, g_3\}) \rhd_1 (\{a_1, a_3\}, \{g_1, g_2, g_3\})$; and,

4. $\{\{(\{a_2, a_3\}, \{g_1, g_2, g_3\})\}, \{(\{a_1\}, \emptyset)\}\}$ cannot be Nash stable since a_3 can transfer from $(\{a_2, a_3\}, \{g_1, g_2, g_3\})$ to $(\{a_1\}, \emptyset)$ and $(\{a_1, a_3\}, \{g_1, g_2, g_3\}) \rhd_3 (\{a_2, a_32\}, \{g_1, g_2, g_3\})$;

Finally, emptiness of the individual stability concept is evidenced by the fact that, in any structure, it is possible for an agent to defect from the team it belongs to in this structure and form the team that blocks it so that both of the agents in this new team are better off as a consequence.

In this way, Assumptions **P1** and **P2** do not guarantee core, Nash or individual stable structures in any $\Gamma_{\mathcal{H}}$. However, these concepts are related by the following theorem.

Theorem 6.1 *Given **P1** and **P2**, if a structure $\pi \in \Gamma_H$ is core stable then it is Nash stable.*

Proof: Assume π is core stable and consider the following agent transfers within π:

1. Agent transfers from one minimal and successful team to another minimal and successful team;

2. Agent transfers from one minimal and successful team to a null team;

3. Agent transfers from a null team to a minimal and successful team; and,

4. Agent transfers between null teams.

First, consider all transfers that are described in both 1 and 3. Against the aforementioned preference assumptions of the agents, if an agent joins a coalition in a minimal and successful team then any team containing this new coalition cannot be minimal. Thus, even if this team is successful, because Riker's size principle is true, it will not be formed and, consequently, no agent will gain by transferring to minimal and successful teams in π.

Now, consider those transfers described in both 2 and 4. If an agent transfers to a null team from either a minimal and successful or another null team in π then it is possible for this agent to prefer this newly formed team to the team from which it transferred. However, if this is the case then this newly formed team must be minimal and successful, meaning the agent in the null team must also prefer this newly formed team to the null team they belong to in π. This, in turn, means that this new team blocks π which is impossible since π is core stable. Therefore, if π is core stable then none of the agent transfers within π, as described in 1 - 4, can benefit the agent who transfers. Thus, if a structure π is core stable then it is Nash stable. ∎

Since Nash stability implies individual stability, all core stable structures are also individually stable and, therefore, contractually individual stable as well. In this way, to understand coalition formation, core stability is especially important since, if there exists a core stable team structure then none of the agents can do better than form the teams in this structure. Consequently, the following decision problems can be answered:

Core membership
Input: $\Gamma_{\mathcal{H}}, \mathcal{T}_1, \ldots, \mathcal{T}_n, \pi$.
Question: Is π core stable?

Core non-emptiness
Input: $\Gamma_{\mathcal{H}}, \mathcal{T}_1, \ldots, \mathcal{T}_n$.
Question: Does there exist $\pi \in \Pi_T$ such that π is core stable?

The *Core membership* problem can be computed as follows. For agent $a_1 \in Ag$, for every team T in $\mathcal{T}_1$ that a_1 prefers to $T_i(\pi)$, check if every other agent a_j in T also prefers this team to $T_j(\pi)$. By repeating this for $a_2, \ldots, a_n$, this problem can be solved. This can be computed with time complexity that is polynomial in the input size (which also has size that is polynomial in the number of agents and goals due to the previously

mentioned assumptions).

Against this insight, the *Core non-emptiness* problem can be solved non-deterministically by guessing π and checking, in polynomial time, if π is core stable.

Theorem 6.2 Core non-emptiness *is NP-complete*

Proof: For any team structure, computing if it is core stable can be done in polynomial time. Thus membership to the class NP is trivial. To prove completeness, an instance of the following problem is reduced to *Core non-emptiness*.

> *The stable roommates problem* (from [36])
> *Input:* $Ag' = \{a_1, \ldots, a_m\}$. $P_1, \ldots, P_m$ - where, for $i = 1, \ldots, m$, P_i is the preferences of the agent a_i over the other agents in Ag
> *Output:* Does there exist a partition of the agents π' such that π' is a stable matching?

In the stable roommates problem, each agent constructs preferences over the other agents as represented in their preference list. A matching is now a partition of the set of agents into disjoint partners. A matching is unstable if there are two persons who prefer each other to their partners in the matching. Such persons are said to block the matching. Otherwise, if no blocking pair exists then the matching is stable. An instance is solvable if it admits a stable matching, otherwise it is unsolvable.

Generally, the stable roomates problem can be efficiently solved by the Gale-Shapley algorithm [30]. However, if there are indifferences between the agents, as well as unacceptable partners (*i.e.*, certain matchings cannot occur) then this problem is NP-complete [62]. To this end, an NP-complete instance $I' = \langle Ag', P_1, \ldots, P_m \rangle$ of *stable roommates problem* is reduced to an instance $I'' = \langle \Gamma_{\mathcal{H}}, \mathcal{T}_1, \ldots, \mathcal{T}_n \rangle$ of *Core non-emptiness* as follows:

1. $Ag = Ag'$;
2. $\succeq_i = P_i, \forall a_i \in Ag$;
3. $G_i = \{g_i\}, \forall a_i \in Ag'$; and
4. For all acceptable partners (a_i, a_j), $v(\{a_i, a_j\}) = \{\{g_i, g_j\}\}$ and $v(C) = \emptyset$ for all other coalitions $C \subseteq Ag$.

Clearly, as only the teams containing the coalitions that represent acceptable pairs and the set of goals each coalition can accomplish are successful, these teams are also minimal and successful. For every agent $a_i \in Ag$, $\mathcal{T}_i$ (which, against the conciseness assumptions, has size that is polynomial in the number of agents and goals) can be easily constructed from $\Gamma_{\mathcal{H}}$. Also, since the concept of stability defined for the *stable roommates problem* is equivalent to core stability for HQCGs, it follows that there is a core stable partition in I'' if and only if there is a stable matching in I'. ∎

The above theorem shows that no algorithm is guaranteed to compute if there exists a core stable structure with time complexity that is polynomial in the input. Nevertheless, in contrast to the core, it can be shown that there always exists a contractual individual stable structure in $\Gamma_{\mathcal{H}}$.

Theorem 6.3 *In any $\Gamma_{\mathcal{H}}$, there exists $\pi \in \Pi_T$ such that π is contractual individual stable.*

Proof: Consider any structure in $\pi \in \Pi_T$ such that all other structures $\pi' \in \Pi_T \setminus \{\pi\}$ do not contain more agents who belong to minimal and successful teams (trivially, such a team is guaranteed to exist). To prove this structure is always contractually individually stable, consider the four agent transfers within π that were identified in Theorem 6.1:

1. Agent transfers from one minimal and successful team to another minimal and successful team;
2. Agent transfers from one minimal and successful team to a null team;
3. Agent transfers from a null team to a minimal and successful team; and,
4. Agent transfers between null teams.

First, consider the transfers that are described in both 1 and 2. Against **P2**, if any agent transfers from a minimal and successful team in π then the resulting team is unsuccessful and so the agents in the minimal and successful team from which a_i transferred cannot gain from this transfer. Now, consider the transfers described in both 3 and 4. Clearly, none of the agents in a minimal and successful team can gain from an agent transferring to this team from a null team in π due to Riker's principle. Also, none of the agents in the null teams in π can gain from agents transferring to another null team as there cannot be a minimal and successful team containing a coalition consisting of these two agents. This is because no structure other than π can contain more agents who belong to minimal and successful teams. Therefore, in all of the HQCGs considered in this chapter, there always exists a contractually individually stable.

∎

Following Theorem 6.3, there will always exist a contractually individually stable structure in $\Gamma_{\mathcal{H}}$.

If there is no core stable structure then this is problematic for the agents as it is not immediately clear which teams they should form. To this end, in the next section, the problem of core emptiness is circumvented by developing a coalition formation protocol in which every agent is motivated to participate in even if there is no guarantee of a non-empty core solution. For this protocol, in addition to the assumptions already specified, it is also assumed that the agents have *perfect recall, i.e.,* they are able to remember and observe all actions that occur at every stage of the protocol.

6.3 A Sequential Coalition Formation Protocol

In this section, the sequential coalition formation protocol presented in [5] is refined for HQCG domains.

Before negotiations, there is a period during which the agents can construct their preferences over the teams they can form. Computing if a set of goals $G' \in v(C)$ satisfies all of the agents in C can be easily verified from $\Gamma_{\mathcal{H}}$: Simply check if $G' \cap G_i \neq \emptyset, \forall a_i \in C$. Also, by analyzing the sets of goals in every set $v(C)$ in increasing order of coalition size, *i.e.*, from smallest to biggest, the preferences of the agents over the teams they can form can be constructed.

In the negotiation phase of the protocol, when it is the turn of an agent to propose a team to be formed or respond to a on-going proposal then this is described as a *stage* in the protocol. The actions taken by an agent when it is their turn in the protocol are represented by a *strategy* which consist of either proposing a team to be formed or accepting or rejecting an on-going proposal. In the negotiation phase, the agents take it in turns to play strategies and the strategy played by an agent is dependent upon the *history* of the protocol.

Definition 6.13 *At stage t ($t > 0$), the* history *of the protocol (h^t) is the set of all proposals, rejections and counter-proposal that occurred up to stage t.*

Agent a_i is called *active* at history h^t if it is their turn in the protocol at stage t. During negotiations, the following rules must be adhered to (these rules are from the protocol described in [5]):

SP1 The order in which the agents propose teams to be formed or respond to a given proposal is given by a rule of order $\mathcal{R}$;

SP2 Agents can propose a team to be formed or respond to a proposed team if and only if they are a member of that team;

SP3 A proposed team is formed if and only if all the agents who belong to the team agree to forming it;

SP4 If an agent does not agree to forming a particular team then they must propose another team instead; and,

SP5 When a team is formed, those agents that make up the team withdraw from the protocol.

In addition to these rules, the following rules are added for the protocol considered in this chapter:

Ra A rule is added so that if a team is rejected then it cannot be proposed again in the protocol;[2] and,

Rb A time period (δ) is introduced during which the agents can either respond to the proposal or propose themselves. Failure for an agent to propose or respond in this time period will result in this agent being excluded from the procedure and, consequently, forced to withdraw (*i.e.*, they are forced to form their null team).

These rules ensure that, in the worst case, the structure consisting exclusively of the agent's null teams will form, meaning:

(i) The protocol always terminates;

(ii) A team structure is always formed; and,

(iii) Agents do not stall or refuse to propose.

In this context, the negotiation phase can be divided into z stages $1, \ldots, z$, where z is the stage after which all agents have withdrawn and the protocol has terminated. Given these rules, at any point in the protocol, history h^t determines:

H1 A set $Ag^{(-)}$ of agents who have already formed teams;

H2 A (possibly partial) team structure $\pi_{Ag^{(-)}}$ formed by the agents in $Ag^{(-)}$;

H3 An ongoing proposal (if any) $\hat{T} = (\hat{C}, G')$;

H4 A set of agents Ag^A who have already accepted the proposal (including the initiator);

H5 A list of rejected teams $\mathcal{L}_{reject}$; and,

H6 An *active* agent $a_i \in Ag$ whose turn is at stage t, as determined by $\mathcal{R}$.

The set of histories at which agent a_i is active is denoted H_i and, for each $h^t \in H_i$, a strategy is formally defined as follows.

Definition 6.14 *For every $a_i \in Ag$, a strategy λ_i is a mapping from H_i to their set of actions. Specifically, for all $a_i \in Ag$ such that $a_i \notin Ag^A$, if it is the turn of a_i at stage t and:*

1. $\hat{C} = G' = \emptyset$ (i.e., *there is no on-going proposal) then*

$$\lambda_i(h^t) = \text{Propose}(C, G'),$$

where:

(a) $(C, G') \in \mathcal{T}_i$;

[2] This is in the spirit of [53]

(b) $(C, G') \notin \mathcal{L}_{reject}$*; and,*

(c) $\forall a_k \in C$, $a_k \notin Ag^{(-)}$.

2. $\hat{C} \neq \emptyset$ *and* $G' \neq \emptyset$ *such that* $a_i \in \hat{C}$ *then*

$$\lambda_i(h^t) = \text{Accept}(\hat{C}, G'),$$

or,

$$\lambda_i(h^t) = \text{Reject and Propose}(C', G''),$$

where:

(a) $(C', G'') \in \mathcal{T}_i$*;*

(b) $(C', G'') \notin \mathcal{L}_{reject}$*; and,*

(c) $\forall a_k \in C'$, $a_k \notin Ag^{(-)}$.

The protocol is formally presented in Figure 6.3. For notation, let $\mathcal{P}^{seq}$ denote the sequential coalition formation protocol described in this section. Due to the autonomous and self-interested nature of the agents in HQCGs, when given $\mathcal{P}^{seq}$, the agents will compute the strategy that is best for themselves. As core stability cannot be guaranteed, no strategies played by the agents will definitely result in core stable coalitions being formed. Thus, it is not immediately apparent whether the agents will be motivated to participate in the protocol.

To this end, let $\lambda = \langle \lambda_1, \ldots, \lambda_n \rangle$ denote a profile of strategies played by the agents, where $\pi(\lambda)$ is the resulting structure that is formed. A natural question arises,

What makes the agents motivated to play the strategies in profile $\lambda = \langle \lambda_1, \ldots, \lambda_n \rangle$?

To answer this question, consider the following definition (taken from [19]).

Definition 6.15 *A strategy profile* $\lambda^* = \langle \lambda_1^*, \ldots, \lambda_n^* \rangle$ *is a* sub-game perfect equilibrium strategy *if each agent* $a_i \in Ag$ *does not have a different strategy* λ_i *yielding an outcome that it prefers to that generated when it chooses* λ_i^**, given that the other agents follow their profile strategies,* i.e.*,* $\forall a_i \in Ag$, $\forall h^t \in H_i$, $\forall \lambda_i$,

$$T_i(\pi(\lambda^*)) \trianglerighteq_i T_i(\pi(\lambda^* \setminus \lambda_i^* \cup \lambda_i).$$

In words, a profile is a sub-game perfect equilibrium if, at any step of the negotiation process, no matter what the history is, no agent is motivated to deviate and use another strategy other than that defined in the profile. In this context, self-interested agents will be motivated to play this strategy and, therefore, should such a strategy exist, it can be reasoned that they will be motivated to participate in the protocol. Against this reasoning, consider Corollary 6.1.

Corollary 6.1 *(proven in [49]) Given* $\mathcal{R}$*, the strategy profile computed using* backward induction *is a sub-game perfect equilibrium strategy.*

Intuitively, backward induction involves computing the a strategy profile through analyzing all strategies that can be played in all histories leading to the formation of every possible structure in Π_T. The fact that a rejected team cannot be proposed and the perfect information assumption ensures that backward induction algorithms can always be used to compute the equilibrium strategy.

Input: $\Gamma_{\mathcal{H}}$, $\mathcal{R} = \langle a_1 \rightarrow \ldots \rightarrow a_n \rangle$, δ.

Preference Construction Period.
Stage 0: All $a_i \in Ag$ construct $\mathcal{T}_i$.

Negotiation Period.
Stage 1: a_1 is active and plays $\lambda_1 =$ Propose(C, G'), where $(C, G') \in T_1$. At this point, the history is updated as follows: (i) $Ag^{(-)} = \emptyset$; (ii) $\pi_{Ag^{(-)}} = \emptyset$; (iii) $Ag^A = \{a_1\}$; (iv) $\mathcal{L}_{reject} = \emptyset$; and, (v) $\hat{T} = (C, G')$.

Stage t>1: While $Ag \neq Ag^-$, if agent $a_j \in Ag$ is active at stage t and there is no on-going proposal then a_j plays $\lambda_j =$ Propose(C, G'), where: (a) $(C, G') \in \mathcal{T}_j$; (b) $(C, G') \notin \mathcal{L}_{reject}$; and, (c) $\forall a_k \in C$, $a_k \notin Ag^{(-)}$. If $(C, G') \neq (\{a_j\}, \emptyset)$ then the history is updated as follows: (i) $Ag^{(-)} = Ag^{(-)}$; (ii) $\pi_{Ag^{(-)}} = \pi_{Ag^{(-)}}$; (iii) $Ag^A = Ag^A \cup \{a_j\}$; (iv) $\mathcal{L}_{reject} = \mathcal{L}_{reject}$; (v) $\hat{T} = (C, G')$; and (vi) $t = t + 1$ and j is updated so that it is the turn of the next agent as given by $\mathcal{R}$. Otherwise, if $(C, G') = (\{a_j\}, \emptyset)$ then the history is updated as follows: (i) $Ag^{(-)} = Ag^{(-)} \cup \{a_j\}$; (ii) $\pi_{Ag^{(-)}} = \pi_{Ag^{(-)}} \cup \{(\{a_j\}, \emptyset)\}$; (iii) $Ag^A = \emptyset$; (iv) $\mathcal{L}_{reject} = \mathcal{L}_{reject}$; (v) $\hat{T} = (\emptyset, \emptyset)$; and (vi) $t = t + 1$ and j is updated so that it is the turn of the next agent as given by $\mathcal{R}$.

On the other hand, if agent $a_j \in Ag$ is active at stage t and there is an on-going proposal $\hat{T} = (\hat{C}, G')$ (where $a_j \in \hat{C}$) and a_j plays $\lambda_j =$ Accept$(\hat{T})$ then the history is updated as follows: (i) $Ag^{(-)} = Ag^{(-)}$; (ii) $\pi_{Ag^{(-)}} = \pi_{Ag^{(-)}}$; (iii) $Ag^A = Ag^A \cup \{a_j\}$; (iv) $\mathcal{L}_{reject} = \mathcal{L}_{reject}$; (v) $\hat{T} = \hat{T}$; and, (vi) $t = t + 1$ and it is the turn of the next agent as given by $\mathcal{R}$. If $Ag^A = \hat{C}$ then, $\forall a_i \in \hat{C}$, the history is updated as follows: (i) $Ag^{(-)} = Ag^{(-)} \cup \{a_i\}$; (ii) $\pi_{Ag^{(-)}} = \pi_{Ag^{(-)}} \cup \{\hat{T}\}$; (iii) $Ag^A = \emptyset$; (iv) $\mathcal{L}_{reject} = \mathcal{L}_{reject}$; (v) $\hat{T} = (\emptyset, \emptyset)$; and, (vi) $t = t + 1$ and j is updated so that it is the turn of the next agent as given by $\mathcal{R}$.

Else, if $\hat{T} = (\hat{C}, G')$ and a_j plays $\lambda_j =$ *Reject and Propose*(C', G'') where: (a) $(C, G') \in \mathcal{T}_j$; (b) $(C, G') \notin \mathcal{L}_{reject}$; and, (c) $\forall a_k \in C$, $a_k \notin Ag^{(-)}$ then the history is updated as follows: (i) $Ag^{(-)} = Ag^{(-)}$; (ii) $\pi_{Ag^{(-)}} = \pi_{Ag^{(-)}}$; (iii) $Ag^A = \emptyset$; (iv) $\mathcal{L}_{reject} = \mathcal{L}_{reject} \cup \{\hat{T}\}$; (v) $\hat{T} = (C', G'')$; and, (vi) $t = t + 1$ and j is updated so that it is the turn of the next agent as given by $\mathcal{R}$.

Output: $\pi_O = \pi_{Ag^{(-)}}$.

Figure 6.3: $\mathcal{P}^{seq}$ for a HQCG with n agents

6.3.1 Backward Induction

To execute backward induction, the sequential protocol $\mathcal{P}^{Seq}$ is represented in extensive form as a game tree $\mathcal{G}_{game}$ where:

- A vertex represents the agent whose turn it is in the protocol; and,
- An edge from a vertex represents a strategy played by the agent who is represented by that vertex.

To fully define this representation, consider the following terminology.

Definition 6.16 *In a game tree $\mathcal{G}_{game}$,*

- *The* root *is the single vertex at the top of the tree;*
- *The* children *of a vertex consist of all the nodes that are connected, by a directed edge, from the node;*
- *The* leaves *are the set of vertices that have no children;*
- *The* level *represents the depth of the tree, starting from the root. For instance, the root occupies level 1 and, for a vertex at level i, its children occupy level $i+1$; and,*
- *A* complete path *is a path in $\mathcal{G}_{game}$ which begins at the root and ends at a leaf.*

Given $\mathcal{G}_{game}$, the root of this tree will consist of a node representing the first agent in $\mathcal{R}$. The edges from this node will represent the strategies this agent can play. The children of this node will represent the agent's who turn it is next in the protocol, depending upon the strategy played by the first agent. More generally, given a vertex at level t in $\mathcal{G}_{game}$, the path from the root to this node will represent the history h^t. Additionally, all edges protruding from this node will represent the strategies the agent a_i can play given h^t. Clearly, at any vertex in $\mathcal{G}_{game}$, the history can be found through evaluating all the actions that led to this agent's (who is represented by the vertex) turn. A complete path represents the history of the protocol from the turn of the first agent up to the formation of the structure.

Example 6.2 *Consider Γ_H, where $Ag = \{a_1, a_2, a_3\}$, $G = \{A, B, C\}$,$G_1 = \{A\}$,$G_2 = \{B\}$, $G_3 = \{C\}$ and $v(Ag) = \{\{A, B, C\}\}$ and $v(C) = \emptyset$ for all other coalitions. Thus, $\forall a_i \in Ag, (\{a_1, a_2, a_3\}, \{A, B, C\}) \rhd_i (\{a_i\}, \emptyset)$. If $\mathcal{R} = a_1 \rightarrow a_2 \rightarrow a_3$ then the complete game tree representation of this protocol is presented in Figure 6.3.1.*

Given $\mathcal{G}_{game}$, a natural approach to compute the best strategy for each agent would be to employ backward induction. Specifically, for a game tree with z levels, backward induction involves sequentially evaluating the vertices in levels $z, z-1, \ldots, 2, 1$ and computing the best strategy for the agents represented by the vertices to play (as given by their preferences over the teams they can form). In this way, the best strategies at each vertex of the tree are computed given the best strategies at all of its descendants. Given that the best strategies for every node have been computed, each agent can then compute the best strategies form themselves. For example, consider the game tree in Figure 6.3.1. Observe that:

- Level 1 contains one vertex representing the turn of the agent a_1;
- Level 2 contains two vertices, both representing the turn of the agent a_2;
- Level 3 contains three vertices, two of which represent the turn of the agent a_3 and one represents the turn of a_1;
- Level 4 contains two vertices, one representing the turn of the agent a_1 and one representing the turn of the agent a_3; and,
- Level 5 contains one vertex representing the turn of the agent a_2.

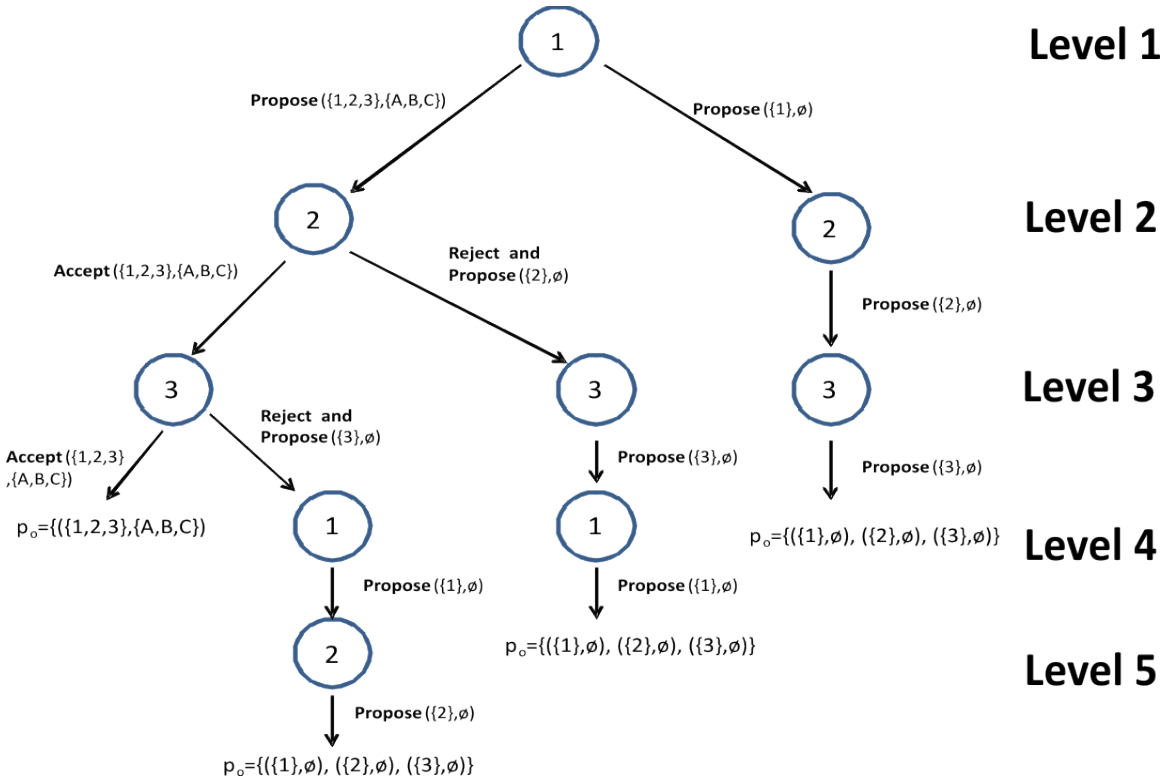


Figure 6.3.1: Complete game tree representation for the protocol in Example 6.2

Beginning with the vertex in level 5, backward induction algorithms sequentially evaluate the vertices in levels $4, \ldots, 1$, computing the best strategy for each agent to play with respect to their preferences over the teams they can form. In Level 5, the only strategy a_2 can play is *Propose*$(\{a_2\}, \emptyset)$. Conversely, in Level 4, the best strategy a_1 and a_3 can play is *Propose*$(\{a_1\}, \emptyset)$ and *Propose*$(\{a_3\}, \emptyset)$, respectively. However, in Level 3, with respect to their preferences over the team, the best strategy for a_3 is to play *Accept*$(\{a_1, a_2, a_3\}, \{A, B, C\})$. Given that they play this, in Level 2, out of the three different strategies they can play, the best strategy for a_2 to play is also *Accept*$(\{a_1, a_2, a_3\}, \{A, B, C\})$. Likewise, in Level 1 the best strategy for a_2 to play is *Propose*$(\{a_1, a_2, a_3\}, \{A, B, C\})$. From this backward induction process, the profile

$$S = \langle \textit{Propose}(\{a_1, a_2, a_3\}, \{A, B, C\}), \textit{Accept}(\{a_1, a_2, a_3\}, \{A, B, C\}),$$

$$\textit{Accept}(\{a_1, a_2, a_3\}, \{A, B, C\})\rangle,$$

is a sub-game perfect equilibrium in this example.

6.3.2 Complexity of Backward Induction

Since every vertex in every level of G_{game} is analyzed during backward induction, a bound on the number of levels in G_{game} can be used as a measure of the complexity of backward induction.

Theorem 6.4 *In any $\Gamma_{\mathcal{H}_U}$, let $\mathcal{T}$ denote the non-null teams that can be proposed by the agents. For any $\mathcal{G}_{game}$ representation of $\mathcal{P}^{Seq}$ in $\Gamma_{\mathcal{H}}$, all complete paths in will contain,*

$$O(n + \sum_{(C,G') \in \mathcal{T}} (|C| - 1)),$$

vertices.

Proof: A longest path is created as follows. Consider the path representing the formation of the structure consisting exclusively of null teams, in which all teams in $\mathcal{T}$ are proposed so that, whenever a team $(C, G') \in \mathcal{T}$ is proposed, the first $|C| - 1$ agents, as given by $\mathcal{R}$, accept the proposal but the last agent rejects it and proposes another team instead.

Obviously, since $\mathcal{G}_{game}$ represents all possible histories up to the formation of every team structure, such a path will be guaranteed to exist in $\mathcal{G}_{game}$. In this path, every time an agent proposes their null team will be represented by exactly one node, whereas every subsequent strategy involving any $(C, G') \in \mathcal{T}$ will be represented by a path containing exactly $|C| - 1$ nodes. Because a team cannot be proposed more than once, no path can contain more vertices than the path representing this scenario and, therefore, all complete paths in will contain no more than,

$$n + \sum_{(C,G') \in \mathcal{T}} (|C| - 1),$$

vertices. ∎

Backward induction evaluates the best strategy for every vertex in every level, starting from the highest level and ending at the root. Therefore, against Theorem 6.4, backward induction process will compute the best strategy for the agents at every node for no more than,

$$n + \sum_{(C,G') \in \mathcal{T}} (|C| - 1),$$

levels. This is clearly demonstrated in Figure 6.3.1 where the longest complete path is of size $3+(3-1) = 5$.

It can be observed that since $\mathcal{G}_{game}$ represents all histories defining all ways in which all $\pi \in \Pi_T$ can be formed, the number of nodes in $\mathcal{G}_{game}$ grows exponentially in the number of levels. For instance, if $BF \geq 1$ denotes the *average branching factor* of $\mathcal{G}_{game}$ (computed, say, from the average number of teams each agent can form at any moment in time) then at level $L > 0$ in $\mathcal{G}_{game}$, there are BF^L nodes. Therefore, even if $\Gamma_{\mathcal{H}}$ is concise in the number of agents and goals, $\mathcal{G}_{game}$ may contain a number of vertices that are exponential in both the number of agents and goals. Consequently, backward induction may require analyzing a number of vertices that are exponential in the number of agents and goals.

Nevertheless, in the next section, a natural class of HQCGs is presented where the equilibrium strategy can be computed without using backward induction. In this context, the complexity of computing an equilibrium strategy is circumvented for this class of games.

6.4 HQCGs with Universal Preference

Consider a class of HQCGs in which the preference orderings of the agents are universal (denoted by HQCG_U and Γ_{H_U}). For a real world example, consider a set of agents representing 'small' businesses or universities, that is, universities or business that have very limited resource. There exist leagues tables (*e.g.*, *Fortune 500* or university league tables) which rank university or business as better than others with respect to certain criteria. Intuitively, these leagues tables can be assumed to constitute a universal ordering over the entities involved in the coalition formation process and, under the assumption that the resource is small that each institution can commit to no more than one team, HQCG_U can represent collaboration between these institutions (of course, if this idealisation is relaxed then this may not always be possible).

Against both **P1** and **P2**, in HQCG_Us, every agent can compute the preferences of the other agents over the teams they can form from the sets of goals in every set $v(C)$. This, in turn, implies that agents can compute the structure that will be formed if agents participate in the sequential protocol. This insight also implies that the equilibrium stratergy can be computed without executing backward induction. To this end, consider the following algorithm for computing the strategy of the agents.

6.4.1 An Algorithm to Compute the Sub-game Perfect Equilibrium Strategy

In this subsection, a novel algorithm which generates a team structure π_O is presented. Once π_O has been generated, in the equilibrium strategy, it is assumed that every $a_i \in Ag$ will:

- *Propose* $T_i(\pi_O)$;
- *Accept* $T_i(\pi_O)$ if proposed it; and,
- If proposed any team $T' \in \mathcal{T}_i \setminus \{T_i(\pi_O)\}$, *Reject and Propose* $T_i(\pi_O)$ instead.

For notation, denote the strategy described above as $\mathcal{S}^*$. Without loss of generality, when describing this algorithm, it is assumed that the agents are indexed with respect to their position in the universal ordering, *i.e.*, a_1 is the most preferred, a_2 the second most preferred *e.t.c.* and that, for every coalition $C \subseteq Ag$, the agents are ordered non-decreasingly in C with respect to their index values.

This algorithm will incrementally construct π_O. To do this, Γ_{H_U} and all $v(C)$ such that $v(C) \neq \emptyset$ are input. The space of all coalitions is organized so that all coalitions $C \subseteq Ag$ of the same size are grouped together. For notation, let $S_i(C)$ denote the set of all coalitions of size i and let $S(C) = \cup_{i=1}^{n} S_i(C)$ denote the space of all coalitions. In the spirit of [56], for $i = 1, \ldots, n$, all coalitions $C \subseteq Ag$, as well as all appropriate $v(C)$, are ordered in $S_i(C)$ as follows,

- For $i = 1, \ldots, n$, order coalitions in each $S_i(C)$ as they are in each list $\mathcal{L}_i$ of the DCVC algorithm presented in Section 4.1.1; but,
- For each coalition which occupies the j^{th} position of $\mathcal{L}_i$, set it to occupy the $((|\mathcal{L}_i| - j) + 1)^{th}$ position of $S_i(C)$.

Table 6.4.1 shows how the input space will be represented for a system with agents $Ag = \{a_1, \ldots, a_4\}$ (where, in practice, only those coalitions $C \subseteq Ag$ such that $v(C) \neq \emptyset$ would be stated). This representation ensures that the coalitions are ordered non-decreasingly in each $S_i(C)$ with respect to the most preferred coalition of the most preferred agents (as given from the universal ordering). This means that for any coalition $C_{i,j} \in S_i(C)$ that is part of a minimal and successful team, all $a_k \in C_{i,j}$ will prefer to form the team containing this coalition to all minimal and successful teams in $\mathcal{T}_k$ which contain either:

(i) a coalition ordered below it in $S_i(C)$; or,

(ii) a coalition in any of $S_{i+1}(C), \ldots, S_n(C)$.

In the first step of the algorithm, for all $C \subseteq Ag$, pre-processing is employed to identify all $G' \in v(C)$ which satisfy all of the agents in C. To achieve this, a function F_1 is introduced such that, $\forall G' \in v(C)$,

$$F_1(G') = \begin{cases} 1 & \text{if } \forall a_i \in C, G_i \cap G' \neq \emptyset; \\ 0 & \text{otherwise.} \end{cases}$$

Sequentially, for $i = 1, \ldots, n$ and $j = 1, \ldots, |S_i(C)|$, this function identifies the sets of goals $G' \in v(C_{i,j})$ which coalition $C_{i,j}$ can successfully accomplish. Specifically, the function F_1 is sequentially computed for all sets of goals that the coalitions in $S_1(C), \ldots, S_n(C)$ can accomplish.

For the first coalition C^* encountered where $\exists G' \in v(C^*)$ such that $F_1(G') = 1$, the team (C^*, G') is a minimal and successful team as all teams containing any of the coalitions $C \subset C^*$ are not successful. Furthermore, due to the ordering of the coalitions in $S_1(C), \ldots, S_n(C)$, all agents who make up C^* will prefer to form a team containing this coalition over the teams which contain any other coalition in $S_l(C), \ldots, S_n(C)$ to which they belong. Thus, the following action can be undertaken,

$$\pi_O := \pi_O \cup \{(C^*, G')\}.$$

$S(C)$	$v(C)$	F_1
$S_1(C)$ $C_{1,1} = \{1\}$ $C_{1,2} = \{2\}$ $C_{1,3} = \{3\}$ $C_{1,4} = \{4\}$		
$S_2(C)$ $C_{2,1} = \{1,2\}$ $C_{2,2} = \{1,3\}$ $C_{2,3} = \{1,4\}$ $C_{2,4} = \{2,3\}$ $C_{2,5} = \{2,4\}$ $C_{2,6} = \{3,4\}$		
$S_3(C)$ $C_{3,1} = \{1,2,3\}$ $C_{3,2} = \{1,2,4\}$ $C_{3,3} = \{1,3,4\}$ $C_{3,4} = \{2,3,4\}$		
$S_4(C)$ $C_{4,1} = \{1,2,3,4\}$		

Table 6.4.1: Representation of space of coalitions for algorithm

Now, the remaining teams can be found through sequentially executing F_1 in $v(C)$ for the remaining coalitions (which are disjoint to π_O) input. Clearly, due to the ordering of coalitions in the input space, all of the agents in any coalition $C_{i,j}$ which belongs to a minimal and successful team will prefer to form the team consisting of this coalition over all other teams containing either:

(i) a coalition ordered below it in $S_i(C)$; or,

(ii) a successful teams containing a coalition in any of $S_{i+1}(C), \ldots, S_n(C)$.

Therefore, due to the manner in which the space of all coalitions is searched, every time a coalition which (does not contain any of the agents in the partial team structure π_O and) can successfully accomplish a set of goals is identified, the team consisting of this coalition and this set of goals will definitely be formed and, consequently, it is added to π_O. This process can be iteratively repeated for the remaining coalitions in the input space. In this manner, π_O can be incrementally constructed.

Once the input space of all coalitions and all $v(C)$ has been exhaustively searched, it may be that some of the agents do not belong to teams in π_O. Against previous reasoning, this means that the only teams they can form are the null ones. Therefore, these null teams are added to π_O and it is output. This algorithm is formally presented in Algorithm 6.4.1. To show that this strategy is an equilibrium strategy, consider the following theorem.

Theorem 6.5 *For any HQCG$_U$, if all the agents play $\mathcal{S}^*$ in $\mathcal{P}^{Seq}$, π_O will* always *be core stable.*

Proof: Suppose that the structure $\pi_O = \{T_1, \ldots, T_k\}$ is formed when every agent plays $\mathcal{S}^*$ in the sequential protocol $\mathcal{P}^{Seq}$ in any $\Gamma_{\mathcal{H}_U}$. To complete this proof, it must be shown that for every $a_i \in Ag$, if there exists a team $T \in \mathcal{T}$, such that $T \triangleright_i T_i(\pi_O)$ then $\exists a_j \in T$ such that $T_j(\pi_O) \triangleright_j T$. This is proven to be true *via reductio ad absurdum*, *i.e.*, the converse is assumed to be true and it is then proven that it cannot be. Thus, it is claimed:

$\exists T \in \mathcal{T}_i$ such that $\forall a_j \in T, T \triangleright_j T_j(\pi_O)$.

In $\mathcal{S}^*$, agents form their most preferred team given that those above them in the universal ordering form their most preferred teams. Consequently, if all the agents played $\mathcal{S}^*$ and it was the case that $T \rhd_j T_j(\pi_O)$, $\forall a_j \in T$ then every $a_j \in T$ would have,

- Proposed T instead of $T_j(\pi_O)$;
- If proposed $T_j(\pi_O)$, would have rejected it and counter-proposed T instead; and,
- If proposed T, accepted the proposal.

Therefore, if every agent played $\mathcal{S}^*$, T would have been formed instead of $T_j(\pi_O)$. Thus, since every agent played $\mathcal{S}^*$, such a team cannot exist and so the converse claim is not true. Therefore, π_O is core stable in any HQCG$_U$ where the agents play $\mathcal{S}^*$ in $\mathcal{P}^{Seq}$. ∎

Algorithm For Computing S^*

Input: $\Gamma_{\mathcal{H}_U}, S(C)_1, \ldots, S(C)_{|Ag|}, \delta, \pi_O := \emptyset$.

Step 1. For $i = 1, \ldots, n$ and $j = 1, \ldots, |S_i(C)|$, $\forall G' \in v(C_i, j)$ do $F_1(G')$. If a coalition C^* is found such that $\exists G' \in v(C^*)$ where $F_1(G') = 1$ then do $\pi_O := \pi_O \cup \{(C^*, G')\}$.

Step 2. Repeat Step 1 for all coalitions C' in $S_{|C'|}(C), \ldots, S_{|Ag \setminus C'|}(C)$ such that $C' \subseteq Ag \setminus \cup_{C \in \pi_O} C$, maintaining the sequential order in which the coalitions are searched.

Step 3. After all coalitions have been searched, if $Ag \setminus \cup_{C \in \pi_O} C = \emptyset$ then output π_O. Otherwise, if $Ag \setminus \cup_{C \in \pi_O} C \neq \emptyset$ then $\forall a_i \in Ag \setminus \cup_{C \in \pi_O} C$, set $\pi_O := \pi_O \cup \{(\{a_i\}, \emptyset)\}$ and output π_O.

Output: π_O.

Algorithm 6.4.1: An algorithm for computing a sub-game perfect equilibrium strategy for $\Gamma_{\mathcal{H}_U}$

Theorem 6.5 proves that there does not exist a team $T \in \mathcal{T}$ such that, for every agent $a_i \in T$, $T \rhd_i \mathcal{T}_i(\pi_O)$. Therefore, following this theorem, were any $a_i \in Ag$ to deviate from their strategy $\lambda_i \in \mathcal{S}^*$ and play λ_i' instead, the team proposed instead of $\mathcal{T}_i(\pi_O)$ will be rejected by at least one agent who also belongs to it, regardless of the history. Thus, if there exists a team $T \in \mathcal{T}$ such that, for every agent $a_i \in T$, $T \rhd_i \mathcal{T}_i(\pi_O)$ then, were this team proposed in strategy λ_i, it will not be formed as at least one agent will reject it. Thus, at any step of the negotiation process, no matter what the history is, no agent is motivated to deviate and use another strategy other than that which they played in $\mathcal{S}^*$. Theorem 6.6 immediately follows.

Theorem 6.6 *In the sequential protocol $\mathcal{P}^{Seq}$ of any HQCG$_U$, $\mathcal{S}^*$ is a sub-game-perfect Nash Equilibrium strategy.*

Theorem 6.5 also proves that, in contrast to the general HQCG representation, in HQCGs with universal preference, there always exists a core stable structure. Against Theorem 6.1, this also implies that there always exists a Nash stable and, therefore, an individual and contractual individual stable structure.

When analyzing the sets of goals in every set $v(C)$, since no coalition can contain more than n agents and no set of goals can contain more than m goals, verifying if a given set of goals in set $v(C)$ satisfies every agent in the coalition C requires no more than nm operations. Therefore, the time complexity of Algorithm 6.4.1 is,

$$O(\sum_{C \subseteq Ag} (|v(C)| \times (mn))),$$

where $|v(C)|$ denotes the number of sets of goals in $v(C)$. Against **HQCG1** and **HQCG2**, this implies that this algorithm can compute the equilibrium strategy in a number of operations that are polynomial in both the number of agents and goals. Furthermore, this algorithm can be computed whilst the agents are constructing their preferences, meaning backward induction is avoided.

6.5 Further Assessment of the Protocol

Having addressed the computational complexity of computing an equilibrium strategy in the sequential protocol, in this section the protocol is assessed with respect to other factors, such as:

1. The fairness of the protocol, that is, whether certain agents have an advantage over others with respect to forming their most preferred teams;
2. The stability of the formed structure; and,
3. The worst case optimal guarantees of the formed structure.

First, consider *fairness* in $\mathcal{P}^{seq}$. In particular, consider the problem of *control* [6]. The problem of control computes if it is possible for an agent to affect the formed structure through changing the order in which agents propose teams, that is, through controlling $\mathcal{R}$. This is undesirable in the sequential process, since an agent may have an advantage over others based only on the rule of order and the ordering may affect the structure which is actually formed. Thus, a dishonest entity may be able to influence the rule of order so that certain agents have an advantage over others.

In a number of instances of $\Gamma_{\mathcal{H}}$, the protocol may not be control-proof. For instance, consider Example 6.1 and suppose a_1 is the first agent as given by $\mathcal{R}$. Here, a_1 can guarantee not to form their null team. For instance, were they to play *Propose*($\{a_1, a_3\}$,$\{g_1, g_2, g_3\}$) then this team would be formed as it is a_3's most preferred team. On the other hand, if a_1 plays *Propose*($\{a_1, a_2\}, \{g_1, g_2, g_3\}$) then a_2 can either accept or reject the proposal. If they accept it then this team will be formed, however, if a_2 plays *Reject and Propose*($\{a_2, a_3\}, \{g_1, g_2, g_3\}$) instead then a_3 will play *Reject and Propose* ($\{a_1, a_3\}, \{g_1, g_2, g_3\}$) as it prefers this team. Given that a rejected team cannot be proposed again, a_1 will accept this team because otherwise they will have to form their null team. In this way, a_1 can guarantee not to form their null team. Following this intuition, were either a_2 or a_3 the first agents, as given by $\mathcal{R}$, they too can be guaranteed not to form their null teams. In this way, it can be argued that the first agent, as given by $\mathcal{R}$, has a distinct advantage over the other agents. Consequently, because different rules of order can result in different structures being formed, the protocol is not control-proof in the general HQCG representation.

Against this insight, to circumvent any issues regarding control, it is assumed that the rule of order is randomly generated in $\Gamma_{\mathcal{H}}$. Of course, in $\Gamma_{\mathcal{H}_U}$, control-proofness in $\mathcal{P}^{Seq}$ is guaranteed since a core stable coalition structure is guaranteed to exist and the strategy of the agents involves generating a core stable structure and playing a strategy that results in its formation. Therefore, no matter the rule of order, this structure will always be formed. In this way, as well as circumventing the complexity in computing equilibrium strategies, the problem of control is also circumvented in $\Gamma_{\mathcal{H}_U}$.

Now, consider stability of the formed structure. Even though there is no guarantee of a core, Nash or individual stable team structures existing in $\Gamma_{\mathcal{H}}$, stability of π_O is guaranteed because, when forming teams in $\mathcal{P}^{Seq}$, the commitment is binding between the agents. This means that, once π_O is formed, every agent is unable to defect from the teams they belong to in this structure and form another team instead. Therefore, because agent transfers between teams are prohibited, if there is no core or Nash stable structure, π_O is inherently stable, even if π_O does not necessarily satisfy the requirements of core and Nash Stability. In this context, problems concerning non-emptiness of the core are circumvented by the binding commitment assumption.

Of course, this is not a problem in $\Gamma_{\mathcal{H}_U}$ because, following Theorem 6.5, the structure formed from all the agents playing the strategies in profile $\mathcal{S}^*$ in $\mathcal{P}^{seq}$ is always core stable. Furthermore, against the intuition expressed in Theorem 6.1, π_O must also be Nash stable (and, therefore, individual and contractual individual stable as well). In this way, even without binding commitment, π_O is always stable if the agents play their strategy that belongs to the equilibrium profile in $\mathcal{P}^{seq}$.

As well as fairness and stability, in any HQCG, a notion of optimality can be captured in the following definition.

Definition 6.17 *A team structure $\pi^* \in \Pi_T$ is optimal if it maximizes the number of satisfied agents.*

Obviously, due to the self-interested nature of the agents, there is no guarantee that π_O will maximize the number of satisfied agents. Nevertheless, note that no more than n agents can be satisfied in any team structure.

For notation, let:

- Π_T denote the set of team structures that consist of null or minimal and successful teams (that is, those that would be formed by the agents with preferences according to **P1** and **P2**);
- Π'_T denote the set of all possible team structures;) ($\Pi_T \subseteq \Pi'_T$);
- π^*_F denote an optimal structure in Π'_T;
- π^*_T denote an optimal structure in Π_T; and,
- π_O the structure formed from the agents participating in the protocol.

Proposition 6.1 *In any $\Gamma_{\mathcal{H}}$,*

the number of agents satisfied in π^*_F − the number of agents satisfied in $\pi_O \leq n-1$.

Proof: Firstly, suppose there are no successful teams in $\Gamma_{\mathcal{H}}$. This, means that Π_T contains a single structure which consists exclusively of null teams and $\pi_O = \pi^*_T$. In such a case, as there are no externalities from coalition formation, in any unsuccessful team, no more than $n-1$ agents can be satisfied. For example, suppose $Ag = \{a_1, \ldots, a_n\}$ and $\forall i = 1, \ldots, n$, $G_i = \{g_i\}$. If $v(\{a_1, .., a_i, ..a_n\}) = \{g_1, .., g_{i-1}, g_{i+1}, ..g_n\}$ and $v(C) = \emptyset$ for all $C \subset Ag$ then the structure $\pi^*_T = \{(\{a_1, \ldots, a_n\}, \{g_1, .., g_{i-1}, g_{i+1}, ..g_n\})\}$ satisfies $n-1$ agents. Therefore:

the number of agents satisfied in $\pi^*_F -$ the number of agents satisfied in $\pi_O \leq n-1$.

Now, if there exists at least one successful team then at least one of the minimal and successful teams will be formed by the agents. Thus, a structure containing any of these teams will be formed, meaning at least one agent will be satisfied. If there is exactly one successful team consisting of the grand coalition then, trivially, this team is also minimal and successful, meaning $\pi_O = \pi^*_F$ and the number of agents satisfied in π^*_F is equal to the number of agents satisfied in π^*_T. However, if there exists other successful teams then, against **P2**, some of these teams will be formed in π_O. Consequently, it follows that:[3]

the number of agents satisfied in $\pi^*_F -$ the number of agents satisfied in $\pi_O \leq n-1$.

∎

Proposition 6.1 shows that either an optimal team structure will be formed or at least one agent will be satisfied in the the formed structure. In addition to this proposition, it should be noted that π_O will be optimal if and only if:

[3] Although the HQCGs studied in this chapter assume that no agent can accomplish any of their goals, this proof does not consider such assumptions. Therefore, for the games considered in this chapter, in Proposition 6.1, '$n-1$' must be changed with '$n-2$'.

(a) all successful teams contain the coalition Ag; or,

(b) all team structures in Π'_T satisfy zero agents.

If (a) is true then all teams containing the coalition $C = Ag$ will be minimal and successful. Against previous discussion, the agents will formulate their preferences over these teams and will be indifferent between these minimal and successful teams but will prefer any of them to their null team. Consequently, any of the team structures consisting of this minimal and successful team will be formed by the agents in $\mathcal{P}^{seq}$ and π_O will consist of a structure in which n agents are satisfied.

On the other hand, if (b) is true then all team structures that can be formed will satisfy zero agents. Given there are no minimal and successful teams, π_O will exclusively consist of every agent's null team and so, in this system, π_O will also satisfy zero agents.

6.6 Summary

In this chapter, hedonic qualitative coalitional games were formally defined. However, only those games where assumptions regarding the conciseness of the representation and the preferences of the agents were studied. For this class of games, stability concepts, based on those formulated for hedonic coalitional games, were formalized for HQCGs. In these games, although there is always guaranteed to be a contractually individually stable structure, there is no guarantee that there will be a core, Nash or individually stable structure. Furthermore, computing if there exists a core stable structure cannot be done in polynomial time complexity, even if the representation is concise.

To this end, a sequential coalition formation protocol was refined for HQCGs. In this protocol, there always exists an equilibrium strategy and, if all the agents play this strategy in any HQCG, then this structure is core stable if and only if there exists a core stable structure. However, computing this strategy requires backward induction which can be exponentially complex. Furthermore, this protocol is not control-proof.

Given this insight, motivated through real world examples, a natural class of HQCGs were studied. In this class of games there always exists a core and Nash stable structure. Furthermore, if all the agents play their equilibrium strategy then the formed structure is always core and Nash stable (and, therefore, individual and contractual individual as well). Additionally, it was demonstrated that this strategy can be computed with time complexity that is polynomial in the size of the representation. This is particularly significant if the representation is naturally concise. Also, if the agents play their equilibrium strategy then this protocol is control-proof in this class of games.

Chapter 7

Conclusions and Future Work

Coalition formation is fundamental to the functioning of certain multi-agent systems. In these multi-agent systems, to understand which coalitions will be formed by the agents, the system can be represented as a cooperative game and concepts from game theory can be employed. Although there are advantages to using game theory, there also exist a number of limitations. For example, a number of natural problems concerning game theory are intractable, *i.e.*, there is no guarantee that these problems can be efficiently solved. In particular, the problem concerned with generating an optimal coalition structure concept, which can be used to identify the coalitions that will be formed by fully cooperative agents, is NP-hard. The complexity of this problem stems from the fact that the number of coalition structures grows exponentially in the number of agents and, consequently, one line of work has focused on designing algorithms that can generate an optimal coalition structure as efficiently as possible. Building upon this research, this thesis makes two significant contributions.

Contribution 1.

Firstly, an optimal coalition structure generation algorithm was developed for characteristic function games. This contribution is particularly significant because it is the first optimal coalition structure generation algorithm that considers both coalition value calculation and optimal coalition structure generation processes. Furthermore, this algorithm was a heavily refined version of the sequential application of the distributed coalition value calculation algorithm and the integer partition algorithm. To be precise, the procedure was heavily refined by:

1. Introducing filter rules to both the DCVC and IP stages to identify coalitions that cannot belong to optimal coalition structures;
2. Transferring the process of identifying promising subspaces of coalition structures from the IP stage to the DCVC stage; and,
3. Changing the manner in which the coalition structure values are computed in each subspace.

Empirical results suggest that, for normally distributed of coalition values, these filter rule can exponentially improve the efficiency in which an optimal coalition structure is generated in this sequential DCVC-IP procedure. Also, the anytime property of the IP optimal coalition structure generation algorithm is retained and it is ensured that the transfer load of the agents, as well as the resource usage, is minimized.

Contribution 2.

An optimal coalition structure generation algorithm was developed for natural classes of partition function games. This was the first algorithm that considered optimal coalition structure

generation when there are non-zero externalities from coalition formation. Inherent to the effectiveness of this algorithm is the proof that all coalition values can be bounded by first computing the values of all coalitions that are embedded in those coalition structures that belong to a fraction of Π. Empirical results show that, when the effect of the externalities are large and the effect of the sub- or super-additivity is small, or *vice-versa*, the algorithm generates an optimal coalition structure through computing the values of only a number of structures. In contrast, when the effects are similar, almost every coalition structure value is computed in order to generate an optimal.

Now, the complexity of problems concerning non-emptiness and membership of the core solution concept, which can be used to identify coalitions that will be formed by self-interested agents, is related to the size of the cooperative game representation. Because the number of coalitions grows exponentially in the number of agents, this means that, given a fully-expressive cooperative game representation, these problems cannot be solved with polynomial time complexity. Consequently, one line of research has focused on developing fully expressive representations of cooperative games which, for certain natural instances, enable core-related problems to be computed with time complexity that is polynomial in the number of agents. Following on from this research, the work presented in this thesis makes the following contribution.

Contribution 3.

Hedonic qualitative coaltional games were formalized. Given this novel representation, it was shown that, in those games where assumptions regarding both the preferences of the agents and the conciseness of the representation, although there is always guaranteed to exist a contractual individual stable structure, this guarantee could not be extended to core, Nash or individual stability. Nevertheless, under these assumptions, core stability implies both Nash and individual stability but, even if the representation was concise, no algorithm could be guaranteed to compute if there exists a core stable structure with time complexity that is polynomial in the size of the representation.

If there is no core stable structure in a HQCG then it is not immediately apparent how the agents should partition themselves. To this end, a sequential coalition formation protocol was developed for which there always existed a equilibrium strategy for the agents to play. In this way, the agents were motivated to participate in the protocol, even if there were no core stable structures. However, computing this strategy may require time complexity that is exponential in the size of the representation.

Finally, a natural class of hedonic qualitative coaltional games were studied. It was shown that if the previously defined assumptions preferences of the agents held, then there always exist a core stable structure. Furthermore, this structure could be computed with time complexity that was polynomial in the size of the representation.

As can be seen from the three contributions presented in this section, this thesis uses algorithm design and representation to tackle the computational complexity of using game theory to understand coalition formation in multi-agent systems. In this way, this thesis provides important contributions to the state-of-the-art understanding of distributed artificial intelligence, game theory and complexity theory.

7.1 Future Work

Following on from these contributions, a natural progression of **Contribution 1** would be to develop a fully distributed optimal coalition structure generation algorithm. The algorithm and filter rules presented in Chapter 4 would form a useful foundation from which such an algorithm could be developed.

In addition, one line of future work would involve extending the optimal coalition structure generation problem so that different metrics could be used to assess the welfare of the system. For example, consider a characteristic function game with non-transferable utility where the gain from forming a coalition is allocated among the agents who belong to it using a *fixed sharing rule*. Here, if $x_i \in v(C)$ denotes a_i's share of the gain then metrics that measure the welfare of the system using the utilities of the individual agents can be employed including:

- The *egalitarian metric*, where the value of a structure $\pi \in \Pi$ is given by:

  $$\arg\min_{x_i \in v(C)} x_i, \forall C \in \pi; \text{ and,}$$

- The *elitist metric*, where the value of a structure $\pi \in \Pi$ is given by:

  $$\arg\max_{x_i \in v(C)} x_i, \forall C \in \pi.$$

With regards to these metrics, the structure with biggest egalitarian or elitist value is egalitarian or elitist optimal, respectively. Following reasoning presented in [23], in many settings that can be represented as characteristic function games with non-transferable utility, these metrics may be more suitable than a utilitarian one. For instance, an elitist metric may be appropriate for systems where agents cooperate to support the agent with the highest utility. A typical scenario could be where fully cooperative agents, who all have a common goal, cooperate and are satisfied if at least one agent achieves that goal no matter what happens to the others. In contrast, an egalitarian metric may be more appropriate in systems where the minimal needs of a large number of agents must be satisfied.

From a computational perspective, once $v(C)$ is known for all coalitions $C \subseteq Ag$ then an elitist optimal coalition structure is any structure that contains the coalition which has the biggest individual utility value. However, generating an egalitarian optimal coalition structure is not as trivial and, in the worst case, every possible structure must be analyzed in order to guarantee an egalitarian optimal coalition structure. Thus, one line of future work could involve developing algorithms that can efficiently generate an egalitarian optimal coalition structure without having to analyze every possible structure that can be formed.

Finally, another possible line of future work could involve investigating the link between optimal coalition structure generation and winner determination in combinatorial auctions. In the multi-agent systems paradigm, auction mechanisms can be used for allocating items such as goods, services and resources among agents. The agent allocating the item(s) is referred to as the auctioneer and the agents who desire the item(s) are referred to as the bidders. Typically, the auctioneer is acting on behalf of a *seller* who is selling the goods.

For a set of atomic goods $\mathcal{Z} = \{z_1, \ldots, z_m\}$, bidder agents may bid over bundles $B_1, \ldots, B_m$ of the goods in $\mathcal{Z}$, where, for $j = 1, \ldots, m$, $B_j \subseteq \mathcal{Z}$. The aim of the auctioneer is to maximize the profit of the seller. This means that if several bids have been submitted on the same bundle B_j, the auctioneer is only concerned with the highest bid and discards the others since it can never be beneficial for the seller to accept one of these inferior bids. Consequently, any bundle B_j can be attributed a value,

$$v(B_j) = \arg\max_{i \in Ag} v_i(B_j),$$

where $v_i(B_j)$ denotes the value of the bid submitted by agent $a_i \in Ag$ for all of the goods in bundle B_j. In this context, *the winner determination problem* determines which partition(s) $\pi_z \in \Pi_z$ of $\mathcal{Z}$ is (are) maximal with respect to the combined value of all bundles that belong to it, *i.e.*,

$$\pi_z^* = \arg\max_{\pi_z \in \Pi_z} \sum_{B_i \in \pi_z} v(B_i).$$

Computing this problem draws obvious parallels with the optimal coalition structure generation problem where:

1. The set of goods represent the set of agents;
2. A bundle of goods represents a coalition of agents; and,
3. The maximal bid represents the coalition value.

As with characteristic function games, it is assumed that there are no externalities in these auctions, meaning every bundle has the same value in every partition to which it belongs. To this is end, in the spirit of *ex ante* optimal coalition structure generation algorithms, winner determination algorithms have been developed which generate π_z^* without having to analyze all possible partitions of the goods in $\mathcal{Z}$, including those that:

- Employ dynamic programming techniques [63];
- Restrict the number of combinations of goods in order to guarantee winner determination in polynomial time [68]; and,
- Exploit the sparseness in the bids in order to guarantee efficient computation of the winner determination problem [65].

In the spirit of **Contribution 2**, one line of work could involve considering winner determination in auctions where there exist externalities. For example, there may exist *identity based externalities* where the identity of the agent who submits the winning bid on a given bundle may negatively or positively influence what the other agents may bid on the remaining bundles of items [79] or *financial based externalities* where the value of the winning bid on a given bundle may negatively or positively influence what the other agents may bid on the remaining bundles of items [37]. Whereas these externalities have been considered from a mechanism design approach (see [32, 31]), the efficiency in which the winner determination problem can be solved has not been considered thus far. This line of work may involve representing the externalities in a manner that enables efficient computation of the winner determination problem or reformulating the existing algorithms so that they can efficiently solve the winner determination problem in the presence of these externalities.

Appendix A

Key Concepts from Computational Complexity Theory

Computational complexity theory is a branch of computer science that focuses on classifying problems according to their inherent difficulty. In this discipline, a *computational problem* is a general question on some mathematical object which is described by giving:

1. A general description of all of its parameters; and,
2. A statement of what properties the solution to this problem should satisfy.

The computational problems studied in this discipline are *decision problems*, that is, problems that have only 'yes' or 'no' answers. Decision problems are very important in computation because they are simple to express and many complex problems can be reduced to the solution of one or more decision problems. Given a decision problem *D*, its complementary decision problem $\overline{D}$ consists of the same object and parameters, but the complement question is asked. For instance, consider the following decision problem.

> D_1
> *INPUT:* A positive integer N.
> *QUESTION:* Does there exist a positive integer k such that $N = 4k$?

The complementary decision problem $\overline{D}_1$ is expressed as follows:

> $\overline{D}_1$
> *INPUT:* A positive integer N.
> *QUESTION:* For all positive integers k, is it the case that $N \neq 4k$?

An *instance* of a problem is obtained by specifying particular values for all of the problem's parameters. For example, $N = 12$ is an instance of problem D_1. The *length function* of a problem D (denoted *Length*) assigns a natural number (a size) to each instance of this problem. Typically, this number reflects the size of the input, that is, the number of items input. For example, in problem D_1, for instance $N = 12$, because 12 can be represented as 0011 in binary form, a natural length for this instance would be 4, *i.e.* *Length*$(N = 12) = 4$.[1]

[1] Abstractly, computation is considered *via* a Turing machine model. Typically, the problem is input to the Turing machine model as a string of symbols. Generally, numbers are input in binary form since this exponentially more concise than in unit form. For more details, see [25, 50].

A.1 The Class P

Given any decision problem, *algorithms* are general, step-by-step procedures that, when given the problem as input, output its solution. An algorithm solves an input problem if, for any instance of that problem, the algorithm gives the correct answer for that instance. For instance, if $N = 12$ then an algorithm will solve this instance of D_1 if it outputs 'yes'. To provide a measure of difficulty in solving a problem, consider the following definition.

Definition A.1 *Suppose an algorithm A is able to solve some instance I of a decision problem* D *in $s(I)$ steps. The* time complexity *of algorithm A is given by,*

$$time_A(n) = max\ arg_{Length(I)=n} s(I).$$

The *time complexity* of a problem can also be referred to as the *worst case* complexity of a problem. It is this time complexity that provides a natural foundation from which to measure the difficulty in solving problems. Firstly, consider the following definition.

Definition A.2 *An algorithm A solves an instance I of a decision problem D in polynomial time if there is a polynomial function $q : \mathbb{N} \to \mathbb{N}$ such that, for every instance I, the number of steps that A takes to solve D is bounded above by,*

$$q(Length(I)).$$

For notation, 'bound above by' can be expressed *via* 'big oh' notation 'O'. Thus, if an algorithm A solves an instance I of a decision problem D in polynomial time then the number of steps that A takes to solve I is equal to $O(q(Length(I)))$. In words, if an algorithm A solves every instance I of decision problem D in polynomial time then this means thats D can be solved with polynomial time complexity. This intuition gives rise to the following class of problems.

Definition A.3 *The complexity class P contains all decision problems that have polynomial-time complexity.*

A.2 Non-deterministic Computation and the class NP

An algorithm is said to solve problems using non-deterministic computation if it can be separated into two separate stages:

Guessing Stage Given an instance I of problem D, this stage "guesses" a potential solution (referred to as a *certificate* $C(I)$ of instance I); and,

Checking Stage Given both I and $C(I)$, this stage then computes if $C(I)$ is a yes instance to I.

Definition A.4 *The complexity class NP contains all decision problems that can be solved in polynomial time* via *non-deterministic computation.*

When analyzing the class NP, for any instance I of a decision problem D, it is not the complexity of finding a *certificate* of I $(c(I))$ that is the issue but rather the complexity of solving the problem from an input of $(I, c(I))$. For example, for the instance $N = 12$, given a certificate $k = 3$, it can be easily verified that $\frac{12}{3} = 4$. Thus, problem D_1 is in the class NP. Clearly,

$$\text{P} \subseteq \text{ NP}.$$

However, in contrast, the question of whether,

$$\text{NP} \subseteq \text{ P},$$

is one of the most famous open problems in the scientific community today. Were this to be true then it would imply that NP=P, meaning all decision problems in the class NP have polynomial-time complexity.

Although, there is no formal proof that either NP=P or NP $\neq$ P is true, there does exist plenty of evidence suggesting the latter. For example, consider the following theorem (proven in [25]).

Theorem A.1 *If a problem D belongs to the class NP then there exists a polynomial $q : \mathbb{N} \to \mathbb{N}$ such that any instance I of D can be solved by a deterministic algorithm having time complexity*

$$2^{q(Length(I))}.$$

The potential for a non-deterministic algorithm to check an exponential number of possibilities is one strong argument which supports the claim that no decision problems in NP $\setminus$ P have polynomial time complexity, *i.e.*, NP $\neq$ P. Against this intuition, consider the following definition.

Definition A.5 *A decision problem* D *is* intractable *if it does not have polynomial-time complexity.*

In other words, a decision problem is intractable if there cannot exist any algorithm that can solve every instance of this problem in a number of steps that are polynomial in the size of the input. From a computational perspective, intractable problems are undesirable since, even for moderately large inputs, it may be impossible to solve them in reasonable time, even with the help of very powerful or non-deterministic computers. This is since the rate of growth of their complexity is exponential in the input size ($Length(I)$). Clearly, all problems in the class P $\cap$ NP are tractable. However, this intuition is not applicable to all problems in the class NP $\setminus$ P, particularly if NP $\neq$ P. In particular, consider the following class of problems.

A.3 NP-Complete Problems

To define this class of problems, a notion of equivalence among decision problems must first be defined.

Definition A.6 *For any two decision problems D and D', a* polynomial transformation *between D and D' is a function F that maps all instances of D to all instances of D' subject to satisfying the following two conditions:*

1. *F is computable by a polynomial time algorithm; and,*
2. *For every instance I of D, $F(I)$ is a 'yes' instance of D' if and only if I is a yes instance of D.*

If there exists a polynomial transformation of D to D' then this is denoted as $D \propto D'$. Decision problems D and D' are said to be polynomial equivalent whenever both,

$$D \propto D' \text{ and } D' \propto D,$$

are true.

From this definition, it is possible to define the concept of NP-completeness as follows.

Definition A.7 *A decision problem D is said to be NP-complete if and only if:*

1. *$D \in$ NP; and,*
2. *For all other decision problems $D' \in$ NP, $D' \propto D$.*

Against Definition A.6:

- If any NP-complete problem can be solved in polynomial time then every NP-complete problem can be solved in polynomial time; and,
- If any NP-complete problem is intractable then every NP-complete problem is also intractable.

Therefore, under the assumption that P $\neq$ NP, all NP-complete problems are intractable.

A.4 The Class CoNP and the 'Difference' Class

As well as the classes P and NP, two other natural complexity classes include:

coNP - A decision problem D belongs to the class coNP if its complement $\overline{D}$ belongs to the class NP; and,

DP - A decision problem D belongs to the difference class DP if it belongs to both the class NP and the class coNP, *i.e.*, it is contained in the class NP $\cap$ coNP.

Intuitively, the complement to every NP-complete decision problem is complete for the class coNP. As with NP and P, it is generally accepted that NP $\neq$ coNP, although there is no formal proof to either support or contradict this. Of course, if P=NP then NP=coNP since P is closed under complement, however, conceivably, it could be that P $\neq$ NP and NP=coNP. Generally, it is assumed that NP $\neq$ coNP.

The difference class DP was formally introduced in [51]. In words, the difference class is the class of decision problems that can be solved *via* both NP and coNP computation. As proven in [51], DP $\neq$ NP $\cap$ coNP. Conversely, like NP and P, there exist problems that are complete for the class DP. To highlight the difference between the complexity classes considered thus far, observe the following problems.

> *SAT**
> *Input:* A set of boolean variables $X = \{x_1, \ldots, x_n\}$, a boolean expressions Φ over these variables and a truth assignment $Z \subseteq X$ (meaning that all variables in X belonging to Z are assigned $\top$ and all else $\bot$).
> *Question:* Does $\Phi[Z] = \top$?
>
> *SAT*
> *Input:* A set of boolean variables $X = \{x_1, \ldots, x_n\}$ and a boolean expressions Φ over these variables.
> *Question:* Does there exist a truth assignment $Z \subseteq X$ such that $\Phi[Z] = \top$?
>
> *UNSAT*
> *Input:* A set of boolean variables $X = \{x_1, \ldots, x_n\}$ and a boolean expressions Φ over these variables.
> *Question:* Is it the case that for every truth assignment of these variables $Z \subseteq X$ Φ is never satisfied (that is, $\Phi[Z] = \bot$ for every truth assignment Z)?
>
> *SAT-UNSAT*
> *Input:* Two sets of boolean variables $X_1 = \{x_1, \ldots, x_n\}$ and $X_2 = \{x'_1, \ldots, x'_n\}$, as well as two boolean expressions Φ and Φ' over these sets of variables.
> *Question:* Does there exist a truth assignment $Z_1 \subseteq X_1$ such that Φ is satisfiable (that is, $\Phi[Z_1] = \top$) and is it the case that for all truth assignments $Z_2 \subseteq X_2$ Φ' is unsatisfied (that is, $\Phi'[Z_2] = \bot$ for every truth assignment Z_2)?

Clearly, SAT* belongs to the class P (and therefore NP since P $\subseteq$ NP) since this problem can be solved in $|\Phi|$ steps[2] and, therefore, has polynomial time complexity. However, SAT belongs to the class NP since it can be solved non-deterministically by guessing an assignment $Z \subseteq X$ and verifying if $\Phi[Z] = \top$. Membership to NP follows from the fact that the latter verification can be done in polynomial time. In fact, this problem was proven to be NP-complete [25]. Since *UNSAT* is the complement to SAT then it is coNP-complete and, by definition, belongs to the class coNP. Finally, *SAT-UNSAT* belongs to the class DP since computing if Φ is satisfiable is NP-complete whereas, computing if Φ' is unsatisfiable is coNP-complete.

In this context, although all of these four problems are intuitively similar, they all belong to four different complexity classes. It is from these classes that a *polynomial hierarchy* can be formalized.

[2]Here, $|\Phi|$ denotes the number of variables in Φ.

A.5 The Polynomial Hierarchy

The polynomial hierarchy is a hierarchy of complexity classes that generalizes the relationship between the classes P and NP. For notation, if $\mathcal{C}$ and $\mathcal{C}'$ are complexity classes then $\mathcal{C}^{\mathcal{C}'}$ denotes the class of problems that are in $\mathcal{C}$ assuming the availability of an *oracle*[3] for problems in $\mathcal{C}'$ [50]. For example, NP^{NP} denotes the class of problems that may be solved in nondeterministic polynomial time, assuming the presence of an oracle that can solve problems in NP.

In this context, given the classes P and NP, a polynomial hierarchy is defined with respect to these classes. Formally, beginning with,

$$\Delta_0^P = \Sigma_0^P = \Pi_0^P = P,$$

a hierarchy of tiers containing problems of increasing inherent complexity are defined as follows,

$$\Delta_{U+1}^P = P^{\Sigma_U^P},$$

$$\Sigma_{U+1}^P = NP^{\Sigma_U^P},$$

$$\pi_{U+1}^P = co\Sigma_{U+1}^P.$$

In words, Δ_0^P, Σ_0^P and Π_0^P denote the class of problems that can be solved in deterministic polynomial time. The first level of the hierarchy is occupied by the complexity classes Δ_1^P = P, Σ_1^P = NP and Π_1^P=coNP. The second level contains the classes Δ_2^P = P^{NP}, Σ_2^P = NP^{NP} and Π_2^P=coNP^{NP}. As with the first level, although there are no formal proofs in support or against, it is assumed that all three classes are distinct. The same holds for the third level and so on. At level i, the SAT problem is expressed as follows.

> *Q_iSAT*
> *Input:* A set of boolean variables partitioned into i sets $X_1, \ldots, X_i$ and a boolean expressions Φ over these variables.
> *Question:* $\exists Z_1 \subseteq X_1$,$\forall Z_2 \subseteq X_2$,$\exists Z_3 \subseteq X_3$, ..., $QZ_i \subseteq X_i$ is Φ satisfiable, where quantifier Q is $\exists$ if i is odd and $\forall$ if i is even?

In contrast, the complement problem is expressed as follows.

> $\overline{Q_iSAT}$
> *Input:* A set of boolean variables partitioned into i sets $X_1, \ldots, X_i$ and a boolean expressions Φ over these variables.
> *Question:* $\forall Z_1 \subseteq X_1$,$\exists Z_2 \subseteq X_2$,$\forall Z_3 \subseteq X_3$, ..., $QZ_i \subseteq X_i$ is Φ unsatisfied, where the quantifier Q is $\forall$ if i is odd and $\exists$ if i is even?

It is worth noting that each class at each level includes all classes at previous levels. Therefore, if it were shown that any two classes in the hierarchy were equal, the hierarchy would collapse. As this is unlikely, it is therefore assumed that this hierarchy is unbounded.

[3]Conceptually, an *oracle* is a machine that can instantaneously output the answer to any problem without any computation whatsoever.

Bibliography

[1] T. Ågotnes, W. van der Hoek, and M. Wooldridge. Reasoning about coalitional games. *Artificial Intelligence*, 173(1):45–79, 2009.

[2] C. Ballester. NP-completeness in hedonic games. *Games and Economic Behavior*, 49(1):1–30, 2004.

[3] K. Banerjee, H. Konishi, and T. Sonmez. The core in a simple coalitional formaiton game. *Social Choice and Welfare*, 18(1):135–153, 2001.

[4] K. Binmore. *Game theory and the social contract volume 2: just playing*. Cambridge: MIT Press, 1998.

[5] F. Bloch. Sequential formation of coalitions in games with externalities and fixed payoff division. *Games and Economic Behavior*, 14:90–123, 1996.

[6] F. Bloch and S. Rottier. Agenda control in coalition formation. *Social Choice and Welfare*, 19:769 – 788, 2002.

[7] A. Bogomolnaia and M. Jackson. The stability of hedonic coalition structures. *Games and Economic Behaviour*, 38(2):201–230, 2002.

[8] E. Bolger. A set of axioms for a value for partition function games. *International Journal of Game Theory*, 18(1):37–46, 1989.

[9] O. Bondareva. Some applications of linear programming methods to the theory of cooperative games (in russian). *Problemy Kybernetiki*, 10:119 – 139, 1963.

[10] E. Catilina and R.Feinberg. Market power and incentives to form research consortia. *Review of Industrial Organization*, 28(2):129–144, 2006.

[11] Y. Chevaleyre, P. Dunne, U. Endriss, J. Lang, M.l Lemaître, N. Maudet, J. Padget, S. Phelps, J. Rodríguez-Aguilar, and P. Sousa. Issues in multiagent resource allocation. *Informatica*, 30:3–31, 2006.

[12] V. Conitzer and T. Sandholm. Computing shapley values, manipulating value division schemes, and checking core membership in multi-issue domains. In *Proceedings of the 19th National Conference on Artificial Intelligence (AAAI)*, San Jose, California, USA, 2004.

[13] V. Conitzer and T. Sandholm. Complexity of constructing solutions in the core based on synergies among coalitions. *Artificial Intelligence*, 170(6–7), 2006.

[14] V. Dang, R. Dash, A. Rogers, and N. Jennings. Overlapping coalition formation for efficient data fusion in multi-sensor networks. In *The 21st National Conference on Artificial Intelligence (AAAI)*, pages 635–640, 2006.

[15] V. Dang and N. Jennings. Generating coalition structures with finite bound from the optimal guarantees. In *Proceedings of the 3rd International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, New York, USA, 2004.

[16] G. de Clippel and R. Serrano. Marginal contributions and externalities in the value. *Economics Working Papers (reference number we057339)*, 2008.

[17] X Deng and C Papadimitriou. On the complexity of cooperative solution concepts. *Mathematical Oprerational Research*, 19(2):257–266, 1994.

[18] K. Do and H. Norde. The shapley value for partition function form games. *Discussion Paper, Tilburg University, Center for Economic Research*, 2002.

[19] P. Dunne, S. Kraus, W van der Hoek, and M. Wooldridge. Cooperative boolean games. In *Proceedings of the 7th International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, Estoril, Portugal, 2008.

[20] E. Elkind, L. Goldberg, P. Goldberg, and M. Wooldridge. Computational complexity of weighted threshold games. In *Proceedings of the 22nd Conference on Artificial Intelligence (AAAI)*, Vancouver, BC, Canada, 2007.

[21] E. Elkind, L. Goldberg, P. Goldberg, and M. Wooldridge. A tractable and expressive class of marginal contribution nets and its applications. In *Proceedings of the 7th International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, Estoril, Portugal, 2008.

[22] E. Elkind and M. Wooldridge. Hedonic coalition nets. In *Proceedings of the 8th International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, 2009.

[23] U. Endriss and N. Maudet. Welfare enginering in multiagent systems. In *Engineering Societies in the Agents World IV*, pages 93–106, 2004.

[24] M. Finus and B Rundshagen. Endogenous coalition formation in global pollution control. *EEM Working Paper No. 43.2001. Available at SSRN: http://ssrn.com/abstract=278511 or DOI: 10.2139/ssrn.278511*, 2001.

[25] M. Garey and D. Johnson. *Computers and intractability - a guide to the theory of NP-completeness*. WH Freeman, New York, USA, 1979.

[26] D. Gilles. *Some theorems on n-Person games*. PhD thesis, Department of Mathematics, Princeton University, Princeton, 1953.

[27] I. Hafalir. Efficiency in coalition games with externalities. *Games and Economic Behaviour*, 61(2):209–238, 2007.

[28] S. Hart and M.Kurz. Endogenous formation of coalitions. *Econometrica*, (51):1047–1064, 1983.

[29] S. Ieong and Y. Shoham. Marginal contribution nets: a compact representation scheme for coalitional games. In *Proceedings ofthe 6th ACM Conference on Electronic Commerce*, 2005.

[30] R. Irving. An effcient algorithm for the stable roommates problem. *Algorithms*, 6:577–595, 1985.

[31] R. Jehiel and B. Moldovanu. Allocative and informational externalities in auctions and related mechanisms. *Working paper: centre for economic policy research*, 2006.

[32] R. Jehiel, B. Moldovanu, and E. Sacchetti. Multi-dimensional mechanism design for auctions with externalities. *Journal of Economic Theory*, 1999.

[33] L. Kóczy. A recursive core for partition function form games. *Kluwer Academic Publishers-Plenum Publishers*, 2007.

[34] S. Kraus. Negotiation and cooperation in multi-agent environments. *Artificial Intelligence*, 94:79–97, 1997.

[35] S. Kraus. *Strategic negotiation in multiagent environments*. The MIT Press: Cambridge, MA, 2001.

[36] E. Kujansuu and E. Makinen. The stable roommates problem and chess tournament pairings. *Divulgaciones Matematicas*, 1(7):19 – 28, 1991.

[37] J. Lu. Unifying identity-specific and financial externalities in auction design. *Working paper: Munich Personal RePEc Archive*, 2006.

[38] W. Lucas and R. Thrall. n-person games in partition function form. *Naval Research Logististics Quarter X*, pages 281–298, 1963.

[39] M. Luck, P. McBurney, O. Shehory, and S. Willmott. *Agent technology: computing as interaction*. Agentlink III, 2004.

[40] I. Macho-Stadler, D. Perez-Castrillo, and D. Wettstein. Sharing the surplus: an extension of the shapley value for environments with externalities. *Journal of Economic Theory*, 135:339–356, 2007.

[41] T. Matsui and Y. Matsui. A survey of algorithms for calculating power in weighted majority games. *The Journal of the Operations Research Society of Japan*, 43(186), 2000.

[42] T. Michalak. *Developing a framework to model interactions between fiscal monetry policies in the context of a monetry union*. PhD Thesis. University of Antwerp, Antwerp.

[43] T. Michalak, A. Dowell, P. McBurney, and M. Wooldridge. Optimal coalition structure generation in partition function games. In *Proceedings of the 18th European Conference on Artificial Intelligence*, Patras, Greece, 2008.

[44] T. Michalak, A. Dowell, P. McBurney, and M. Wooldridge. Pre-processing techniques for anytime coalition structure generation. In *Knowledge Representation for Agents and Multi-Agent Systems (Proceedings of KRAMAS 2008)*, Sydney, Australia, 2009.

[45] T. Michalak, J. Engwerda, B. van Aarle, and G. Di Bartolomeo. Models of endogenous coalition formation between fiscal and monetary authorities in the presence of a monetary union. *Quantitative Economic Policy*, 20:103–136, 2008.

[46] T. Michalak, J. Tyrowicz, P. McBurney, and M. Wooldridge. Exogenous coalition formation in the e-marketplace based on geographical proximity. *In Electronic Commerce Research and Applications*, page to appear, 2009.

[47] R. Myerson. Values for games in partition function form. *International Journal of Game Theory*, 6:23–21, 1977.

[48] D. Newman. Coalition formation in the APB and the FASB: some evidence on the size principle. *The Accounting Review*, 56(4):897–909, 1981.

[49] M. Osborne and A. Rubinstein. *A course in game theory*. The MIT Press, Cambridge, MA, USA, 1994.

[50] C. Papadimitriou. *Computational complexity*. Addison, Wesley, Longman, California, USA, 1995.

[51] C. Papadimitriou and M. Yannakakis. The complexity of facets (and some facets on complexity). In *Proceedings of the 24th International Annual Conference Symposium on the Theory of Computing*, pages 244 – 359, 1982.

[52] P. Pintasssilgo. A coalition approach to the management of high seas fisheries in the presence of externalities. *Natural Resource Modeling*, 16(2):175–197, 2003.

[53] J. Plasmans, J. Engwerda, B. van Aarle, G. Di Bartolomeo, and T. Michalak. *Dynamic modelling of monetary and fiscal cooperation among nations*. Springer, New York USA, 2006.

[54] K. Prasad and J. Kelly. NP-completeness of some problems concerning voting games. *International Journal of Game Theory*, 19(1):1–9, 1990.

[55] T. Rahwan. *Algorithms for coalition formation in multi-agent systems*. PhD Thesis. University of Southampton, Southampton, 2007.

[56] T. Rahwan and N. Jennings. An algorithm for distributing coalitional value calculations among cooperating agents. *Artificial Intelligence*, 171(8–9):535–567, 2007.

[57] T. Rahwan and N. Jennings. Coalition structure generation: dynamic programming meets anytime optimization. In *Proceedings of the 23rd conference on artificial intelligence (AAAI)*, Chicago, USA, 2008.

[58] T. Rahwan and N. Jennings. An improved dynamic programming algorithm for coalition structure generation. In *Proceedings of the 7th international conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, Estoril, Portugal, 2008.

[59] T. Rahwan, S. Ramchurn, V. Dang, A. Giovannucci, and N. Jennings. Anytime optimal coalition structure generation. In *Proceedings of the 22nd conference on Artificial Intelligence (AAAI)*, pages 1184–1190, Vancouver, Canada, 2007.

[60] A. De Ridder, A. Rusinowska, E. Saiz, and E. Hendrix. Coalition formation: the role of procedure and policy flexibility. *Working paper:Groupe d'Analyse et de Thorie Economique (GATE), Centre national de la recherche scientifique (CNRS)*, 2008.

[61] W. Riker. *The theory of political coalitions*. Yale University Press, New Haven, 1966.

[62] E. Ronn. *On the complexity of stable matchings with and without ties*. PhD thesis, Yale, 1986.

[63] M. Rothkopf, A. Pekec, and R. Harstad. Computationally managable combinatorial auctions. *Management Science*, 44(8):1131–1147, 1998.

[64] J. Sabater and C. Sierra. Reputation and social network analysis in multi-agent systems. In *Proceedings of the 1st International Joint Conference on Autonomous Agents and Multiagent Systems (IJCAI)*, Bologna, Italy, 2002.

[65] T. Sandholm. An algorithm for optimal winner determination in combinatorial auctions. *Artificial Intelligence*, 135:542–547, 1999.

[66] T. Sandholm, K. Larson, M. Andersson, O. Shehory, and F. Tohme. Coalition structure generation with worst case guarantees. *Artificial Intelligence*, 111(1-2):209–238, 1999.

[67] T. Sandholm and V. Lesser. Coalitions among computationally bounded agents. *Artificial Intelligence*, 94:99137, 1997.

[68] T. Sandholm and S. Suri. Improved algorithms for optimal winner determination in combinatorial auctions and generalizations. In *Proceedings of the 21st Conference on Artificial Intelligence (AAAI)*, pages 90–97, 2000.

[69] S. Sen and E. Durfee. A formal study of distributed meeting scheduling. In *Group Decision and Negotiation Support Systems*, 1997.

[70] L. Shapley. Notes on the n-person game III: some variants of the von-neumann-morgenstern definition of solution. *Rand Corporation research memorandum*, 1952.

[71] L. Shapley. A value for n-person games. *Contributions to the Theory of Games*, 28(2):307–317, 1953.

[72] L. Shapley and M. Shubik. Quasi-cores in a monetary economy with non-convex preferences. *Econometra*, 34:805–827, 1966.

[73] O. Shehory, K. Sycara, and S. Jha. Multi-agent coordination through coalition formation. pages 143–154. Springer, 1998.

[74] Y. Shoham and K. Leyton-Brown. *Multiagent systems: algorithmic, game-theoretic, and logical foundations*. Cambridge University Press, 2009.

[75] S. Sung and D. Dimitrov. On core membership testing for hedonic coalition formation games. *Operation Research Letters*, 35:115 – 158, 2007.

[76] A. Taylor and W. Zwicker. *Simple games : desirability relations, trading, pseudoweightings*. Princeton University Press, Princeton, 1999.

[77] P. Tosic and G. Agha. Maximal clique based distributed group formation for autonomous agent coalitions. In *Proceedings of the 3rd International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 2004.

[78] M. van de Vijser. *Increasing realism in coalition formation for multi-agent systems*. PhD Thesis. The University of Manitoba, Winnipeg, Manitoba, 2005.

[79] G. Das Varma. Standard auctions with identity dependent externalities. *RAND Journal of Economics*, 2002.

[80] J. von Neumann and O. Morgenstern. *Theory of games and economic behavior*. Princeton University Press, Princeton, 1944.

[81] M. Wooldridge. *An introduction to mutli-agent systems*. John Wiley & sons, Chichester, England, 2009.

[82] M. Wooldridge and P. Dunne. On the computational complexity of qualitative coalitional games. *Artificial Intelligence*, 2004.

[83] M. Wooldridge and N. Jennings. Intelligent agents: theory and practice. *The knowledge engineering review*, 10(2):115–152, 1995.

[84] D. Yun Yeh. A dynamic programming approach to the complete partition problem. *BIT*, 26(4):467–474, 1986.