Decentralised Coalition Formation Methods for Multi-Agent Systems

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

Luke Riley

May 2015

Contents

Illustrations

List of Figures

List of Tables

Notations

The following notations and abbreviations are found throughout this thesis:

Chapter 2

$\mathcal{G} = \langle N, v \rangle$	A characteristic function game
N	The set of agents
v	The characteristic function
C	A coalition
$v(C)$	The value of the coalition C
CS	A coalition structure
CS^*	The optimal coalition structure
$v(CS)$	The value of the coalition structure CS
x	A payoff vector
x_i	The payoff for agent i
$x(C)$	The payoff for the coalition C
$Imp(N, v)$	The full set of imputations for the characteristic function game $\mathcal{G} = \langle N, v \rangle$
$\langle CS, x \rangle$	The outcome of a characteristic function game
$\epsilon^s(C, x)$	The strong excess of coalition C given the payoff vector x
$\epsilon^w(C, x)$	The weak excess of coalition C given the payoff vector x
ϵ^s	The maximum strong excess in the characteristic function game
ϵ^w	The maximum weak excess in the characteristic function game
$\delta(x)$	The deficit vector of the payoff vector x
d_i	The percentage-based demand of agent i to join a coalition
d	The percentage-based demand of all agents to join a coalition
L_s	A list of all coalitions of size s
M_i	A list of all coalitions whose smallest agent ID is i
W	The coalitions of the synergy coalitional group representation
$f_1[C]$	The most valuable partition of C
$f_2[C]$	The numeric value of the most valuable partition of C
AF	An argumentation framework
$R(a_x, a_y)$	An attack by argument a_x onto argument a_y
VAF	A value-based argumentation framework

VAF_i	An audience specific value-based argumentation framework for agent i

Chapter 3

$C\text{-}inq$	A coalition inquiry dialogue
$C\text{-}pAct$	A coalition persuasion to action dialogue
Q^i	The finite set of states recognised by agent i
q_0^i	The designated initial state of agent i
Ac^i	The finite set of single actions recognised by agent i
ac	A single action
μ	A joint action comprising single actions
Av^i	The finite set of social-values recognised by agent i
$\rho^i(\mu)$	The set of states agent i believes μ may be executed from
$\tau^i(q_x, \mu)$	The state that agent i believes would result from the performance of joint action μ from state q_x
Φ^i	The set of propositions that agent i uses to represent the world
$\pi(q_i)$	The propositions true in the state q_i
$\delta^i(q_x, q_y, v)$	The status (promoted ($+$), demoted ($-$), or neutral ($=$)) of a social-value v ascribed by agent i to the transition between states q_x and q_y
$\zeta^i(j, ac)$	Whether agent i believes if agent j will perform the given action ac or not
ξ	A tuple matching agents to single actions
$\mathcal{C} = \langle q_x, \mu, \xi, q_y, p, v, s \rangle$	A $\mathcal{C}$-argument
$\mathcal{A}$	A $\mathcal{C}$-argument or a critical question
θ	The dialogue type; either $C\text{-}inq$ or $C\text{-}pAct$
γ	A proposition representing the dialogue goal (if $\theta = C\text{-}pAct$) or a set of propositions that is being inquired over ($\theta = C\text{-}inq$)
Υ	Either a set of $\mathcal{C}$-arguments and critical questions *or* a set of $\mathcal{B}$-arguments, defeasible facts and defeasible rules
$\mathcal{D}_r^t$	A dialogue beginning at time-point r and currently at time-point t
m_t	The dialogue move at time-point t
Ω	The collection of all on-going dialogues
$CoSt_i^t$	The commitment store of agent i at time-point t
$CoSt^t$	The combined commitment store of all the agents at time-point t
ϕ	A defeasible fact
λ	A defeasible rule
$\mathcal{B} = \langle \Phi, \phi \rangle$	A $\mathcal{B}$-argument
Σ^i	Agent i's belief base

Chapter 4

$\underline{t}$	An increment array
$L_{s,\underline{t}}$	A two dimensional list of coalitions of size s that can be generated with the increment array $\underline{t}$
I	An integer partition
$\mathcal{I}(n-s)$	All the integer partitions that total $(n-s)$
$\pi(\underline{t})$	The length of the repeating period within $\underline{t}$
$C(i,\underline{t})$	The coalition generated by agent i from the increment array $\underline{t}$
r	The number of agents to use each increment array

Chapter 5

v^*	The superadditive cover of the characteristic function v
CV_i	The coalition value calculation share for agent i
Γ^i	The information communicated by agent i
$totalposn(\underline{C},\underline{n},s)$	Returns the index of coalition $\underline{C}$ of size s, given the full set of agents $\underline{n}$
$totalposn^{-1}(m,\underline{n})$	Returns the coalition that corresponds to the index m for the full set of agents $\underline{n}$
op	The operation vector for the DDP algorithm
op^*	The operation vector for the DDP* algorithm

Chapter 6

G^v	A valuation disagreement coalitional game
G^b	A coalitional game with beliefs
p_i	The expected payoff of agent i in a coalitional game with beliefs
w^i	The coalition valuation function for agent i in valuation disagreement coalitional games
$w^i(C)$	The valuation of coalition C by agent i in the valuation disagreement coalitional game model
$w(C)$	All the valuations of coalition C by the members of C in the valuation disagreement coalitional game model
w	The vector of all agents' coalition valuation functions in valuation disagreement coalitional games
$u^i(C)$	The valuation of coalition C in the coalitional game with beliefs model
$\pi(C,i)$	The set of agents who value the coalition C less than or equal to agent i
e	The estimated payoff vector
e_i	The estimated payoff of agent i

$e(C)$	The estimated payoff for the agents of C
$\epsilon^v(C, e)$	The estimated excess of the coalition C given the estimated payoff vector e
ϵ^v	The maximum estimated excess of the uncertainty coalitional game
κ	A contract function
$\kappa(\langle C, e(C), w(C), v(C)\rangle)$	Returns the payoff vector for the agents of C, given the estimated payoff vector $e(C)$, the agent valuations $w(C)$ and the true value of the coalition $v(C)$

Preface

This thesis is primarily my own work. The sources of other materials are identified.

Abstract

Coalition formation is a process whereby agents recognise that cooperation with others can occur in a mutually beneficial manner and therefore the agents can choose appropriate temporary groups (named coalitions) to form. The benefit of each coalition can be measured by: the goals it achieves; the tasks it completes; or the utility it gains. Determining the set of coalitions that should form is difficult even in centralised cooperative circumstances due to: (a) the exponential number of different possible coalitions; (b) the "super exponential" number of possible sets of coalitions; and (c) the many ways in which the agents of a coalition can agree to distribute its gains between its members (if this gain can be transfered between the agents). The inherent distributed and potentially self-interested nature of multi-agent systems further complicates the coalition formation process.

How to design decentralised coalition formation methods for multi-agent systems is a significant challenge and is the topic of this thesis. The desirable characteristics for these methods to have are (among others): (i) a balanced computational load between the agents; (ii) an optimal solution found with distributed knowledge; (iii) bounded communication costs; and (iv) to allow coalitions to form even when the agents disagree on their values.

The coalition formation methods presented in this thesis implement one or more of these desirable characteristics. The contribution of this thesis begins with a decentralised dialogue game that utilise argumentation to allow agents to reason over and come to a conclusion on what are the best coalitions to form, when the coalitions are valued qualitatively.

Next, the thesis details two decentralised algorithms that allow the agents to complete the coalition formation process in a specific coalition formation model, named characteristic function games. The first algorithm allows the coalition value calculations to be distributed between the agents of the system in an approximately equal manner using no communication, where each agent assigned to calculate the value of a coalition is included in that coalition as a member. The second algorithm allows the agents to find one of the most stable coalition formation solutions, even though each agent has only partial knowledge of the system.

The final contribution of this thesis is a new coalition formation model, which allows the agents to find the expected payoff maximising coalitions to form, when each agent may disagree on the quantitative value of each coalition. This new model introduces more risk to agents valuing a coalition higher than the other agents, and so encourages pessimistic valuations.

Acknowledgements

I would like to start by thanking my family, and especially my parents Marion and Joesph Riley, for all the help and support they have given me over the long *long* time I have been studying! Even though it is still an open mystery regarding where I inherited my love for technology from, I know I could not have done it without you both; therefore this thesis is as much mine as it is yours. One day I hope to implement my new found knowledge for pressing family matters, such as making sure dad is always connected *on-the-line* to his cricket scores, or making sure mum can take, send and edit as many digital photographs as she could ever imagine. I would also like to specifically thank the Declas family (Caterina, Jacques, Amy, Jasmine, Mattis and Solene) for your general encouragement and excitability about life. There is always so much going on with you all, long may it continue!

Moving on to my academic life, I would like to thank all my supervisors over the years (this list is surprising long). Firstly Katie Atkinson, I very much appreciate the time and effort you put into guiding me through my studies and providing valuable feedback whenever possible, especially considering you were juggling pregnancy and a significant number of other PhD students! Luckily for me your guidance will now continue into my KTP associate job, I look forward to continuing our productive working relationship! Secondly Terry Payne, I am grateful for you stepping in as my second supervisor early on and actually turning out to be my longest serving supervisor! When you get going analysing my work, your attention to detail is always very beneficial. The different perspective to my research that you brought has definitely been valuable. Thirdly Trevor Bench-Capon, thanks so much for covering my supervision *maternity year*, it was interesting to listen and learn from your extensive experience. May you have a long and enjoyable retirement! Finally Peter McBurney, I thank you for kickstarting my thesis journey and for helping me to acclimatise the academic environment! I hope we continue to bump into each other at various events to catch up.

I would also like to thank all the support I have received from members of the Computer Science Department at the University of Liverpool. A big mention here must go to Paul Dunne, who significantly helped me to cleanly convert my ideas and experimental implementation of the algorithm of Chapter 4 into theory form. Also Paul's work on an indexing scheme also became useful in Chapter 5 to help minimise communication costs. Thanks a lot Paul! Additionally I would like to acknowledge the effort of my thesis examiners Simon Parsons and Sarvapali Ramchurn whose comments definitely improved the quality of the final version of my thesis, thanks!

Furthermore I would like to acknowledge the funding support I received from the Engineering and Physical Sciences Research Council (EPSRC). Without their funding I would not have been able to undertake my PhD study, so this support was very much appreciated.

Finally I would like to thank all my friends over the PhD study years who have supported me directly, through general or specific advice, or indirectly by helping to take my mind off work. Big mentions must go to David Willmer and Dave Taymouri (aka the *tripod*) - we had some good parties *back in the day*! Back to the department, and the day-to-day *long hard slog* was made all the better having such nice (i.e. *sounnnd*) office mates over the years such as Anton Minnion, Martyn Lloyd-Kelly, Adam Wyner, Muhammad Khan, Jodi Schneider, Maya Wardeh, Daan Bloembergen and anyone else who stopped by. Its been a pleasure *serving this time* with you ladies and gentlemen. Then there is every member of the computer science *renegade* football team past and present, its been a good experience to help the team grow and evolve. Hopefully we will win more trophies in the future! Into my last year now, and a *shout out* needs to go to the *Bethelona* lads: John Farrell, Si Crockett and Matt Morris. You lads have always been supportive, even when I'm strolling home at 11pm ready for the *lateshift* to begin. Last but not least, a big *big* thank you to Matoula Kotsialou, who has supported me very much over the last year (how did I manage beforehand?!). Meeting you has definitely improved my life, I even listen to people who talk about theory now!

P.S. If you're reading this and thinking I've forgotten you, don't worry I haven't, I am grateful to you as well! ;-)

Chapter 1

Chapter 1

Introduction

In this section the two main topics of this thesis will be outlined: agents and multi-agent systems in Section 1.1 followed by coalition formation in Section 1.2. Next the research question and research objectives will be detailed in Section 1.3. After that, the thesis outline and contributions will be provided in Section 1.4. Finally the published work (to date) related to this thesis will be detailed in Section 1.5.

1.1 Agents and Multi-Agent Systems

This thesis concerns the branch of artificial intelligence known as *multi-agent systems*. Multi-agent systems contain many single *agents*, defined in [134, 136] as:

Definition 1: *An* ***agent*** *is a computer system that is* situated *in some* environment*, and is capable of* autonomous action *in this environment in order to meet its design objectives.*

The autonomy in this definition is on a *spectrum*. As detailed in [136], an agent that *always* seeks assistance from its user (the lower bound of the autonomy spectrum) will be unhelpful, while an agent that *never* seeks assistance (the upper bound of the autonomy spectrum) will probably also be useless, because there is no opportunity for its user to update the agent's objectives. Finding the correct level of autonomy can be a difficult task, it may even be advantageous to have *adjustable autonomy* [107].

A special class of agents are *intelligent agents*. To be classified as intelligent, the following capabilities detailed in [134, 136] should be satisfied:

- **Reactivity**: Intelligent agents should respond in a timely fashion to changes in the environment in order to satisfy their design objectives.
- **Proactiveness**: Intelligent agents should exhibit goal-directed behaviour by taking the initiative to satisfy their design objectives.
- **Social ability**: Intelligent agents should be capable of interacting with other agents in order to satisfy their design objectives.

There can be different levels of intelligent behaviour in agents. The field of *game theory* [83] studies the most rational strategy an agent should perform to give itself a maximum expected utility-value, and so game theory gives one definition of a very specific type of *optimal intelligence*. Additionally, the field of *argumentation* [87] studies how agents, given possibly conflicting information, should find the most rational conclusion, and so provides another definition of a type of *optimal intelligence*. In this thesis, agents use methods from both fields to guide their decision making.

Finding the most rational strategy or conclusion is not a simple task, especially when: (a) agents may have partial knowledge; (b) the environment may be uncertain; and (c) agents may have different preferences. How agents deal with these issues, while still displaying a form of optimal intelligence, is a central theme of this thesis.

In a multi-agent system the agents may be *cooperative* or *self-interested*. Cooperative agents aim to work together to maximise the social welfare of the entire system (i.e. maximise the global benefit), with little (or perhaps no) regard to the benefit they themselves gain. Self-interested agents on the other hand aim to work (possibly in groups) to maximise their own personal benefit, with little (or perhaps no) regard to the benefit of the whole society.

Additionally it is possible that self-interested agents may act strategically (e.g. by misreporting information). Mechanism design [80] can be used to incentivise agents to act in a certain manner (e.g. to truthfully reveal private information), usually by having a mediator promise to give each agent a certain monetary transfer that is maximised only when the agent acts as the mediator requires.

It is typical to consider cooperative agents to be developed by a single designer and self-interested agents to be developed by multiple different agent designers with competing interests. In both single and multiple agent designer cases, it may be desirable for the agents to act to achieve their design objectives in a decentralised manner, without a centralised body dictating every decision. Without a centralised dictating authority there can be (as stated in [73]): no centralised bottleneck; increased robustness against failure; and reduced overall computation time. Decentralised methods for multi-agent systems is investigated within this thesis for the specific process of *coalition formation*, introduced in the next section.

1.2 Coalition Formation

In a multi-agent system, the agents can interact with each other to fulfill their objectives and improve their performance. These interactions will typically involve a form of cooperation, coordination and/or negotiation in an *organisational paradigm*, such as the following (as stated in [63]):

- Coalitions: Where the agents are arranged into short lived temporary groups that are goal-directed and have a flat organisational structure.
- Congregations: Where the agents are arranged into long term groups with no specific goal in mind. They also have a flat organisational structure.

Model	**Key Characteristic**	**Benefits**	**Weaknesses**
Coalitions	Dynamic, goal-directed	Exploit strength in numbers	Short term benefits may not outweigh organisation costs
Congregations	Long-lived	Facilitates agent discovery	The groups may be overly restrictive
Federations	Middle-agents	Matchmaking, brokering, translation services; facilitates dynamic agent pool	Intermediaries become bottlenecks
Hierarchy	Decomposition	Maps to many common domains; handles scale well	Potentially brittle; can lead to bottlenecks or delays
Holarchy	Decomposition with autonomy	Exploit autonomy of functional units	Must organise holons to begin with; lack of predictable performance
Markets	Competition through pricing	Good at allocation; increased utility through centralization; increased fairness through bidding	Potential for collusion and malicious behavior; allocation decision complexity can be high
Martrix	Multiple managers	Resource sharing; multiple agents can be influenced	Potential for conflicts; need for increased agent sophistication
Societies	Open system	Public services; well defined conventions	Potentially complex, agents may require additional society-related capabilities

TABLE 1.1: Comparing different organisational paradigms.

- Federations: Where the agents are arranged into groups and have ceded some amount of autonomy to a single delegate that represents each group.
- Hierarchies: Where the agents are arranged in a tree-like structure, with each agent (apart from the root agent), having a single parent named the manager (this assumption is relaxed in the Matrix paradigm).
- Holarchies: Where the agents are arranged in multi-leveled, grouped hierarchies.
- Markets: Where the agents are arranged into a producer-consumer system, with some agents buying products, some agents selling products and some agents doing both.
- Matrix: Where the agents are arranged into some form of a grid, where each agent may have to report to multiple managers.
- Societies (including markets): Where the agents are arranged into an inherently open system with a defined set of laws, where different agents may enter and leave as they please but the society continues to exist.

Each of these paradigms has its own strength and weaknesses, as summarised in Table 1.1 (taken from [63]). The coalition organisational paradigm is one of the most popular in multi-agent systems and is becoming increasingly important due to its natural fit to many scenarios

where there may be no central authority, especially when economies of scale can be exploited [63, 88].

In more detail, *Coalition formation* is a process whereby agents recognise that cooperation with others can occur in a mutually beneficial manner and therefore the agents have the capability to choose appropriate temporary groups (named *coalitions*) to form. The benefit of each coalition can be measured by: the goals it achieves; the tasks it completes; and/or the utility it gains. The reason that a *self-interested* agent joins a coalition, is that it will benefit more by working with the other agents in the coalition compared to staying alone by: achieving more goals; completing more tasks; or gaining more utility. A *cooperative* agent will join a coalition, if the coalition increases the social welfare of the multi-agent system. Throughout this thesis, it will generally be assumed that coalition formation takes place within *coalitional games*, which is an abstract model for the complicated interactions and agreements required during the process of coalition formation.

Coalition formation in multi-agent systems has a wide range of potential applications. According to [32, 36, 88], current or proposed applications include:

- Communication networks: Where coalitions of nodes can form in a network to reduce its power consumption, reduce transmission costs or even increase network security.
- Distributed vehicle routing: Where coalitions of delivery companies can be formed to reduce transportation costs by sharing deliveries.
- Electronic auctions/market places: Where coalitions of buyers can form to buy in bulk and take advantage of price discounts.
- Grid computing: Where coalitions of agents can form to receive optimal resource allocations from the grid.
- Multi-agent planning: Where coalitions of agents can form to achieve mutually beneficial goals.
- Sensor networks: Where coalitions of sensors can form to track targets.
- Smart grid: Where coalitions of distributed energy resources (electronic storage devices or energy generators) can form: to meet power generation targets (in the case of coalitions consisting of renewable energy providers); balance power supply and demand (for coalitions of storage devices and energy provides); or to exploit economies of scales (for coalitions of storage devices).

Regardless of the application of coalition formation, the main points of interest tend to centre around the following two issues:

1. *What set of coalitions should form?*
2. *How should the benefit of each coalition be divided to its members?*

Yet investigating these issues is a difficult task, due to: (i) the exponential number of different possible coalitions; (ii) the "super exponential" number of possible sets of coalitions; (iii) the possible self-interested behaviour of the agents; and (iv) the many ways in which the agents' of a coalition can agree to distribute the benefit of each coalition around the coalition's members (if this benefit can be transferable).

The two previously highlighted issues are not just important in the multi-agent systems domain, since the process of coalition formation can be seen in economic, political and social settings [96], such as: cartel formation; political party/government formation; customs unions; and public goods provisions. Therefore coalition formation has also been studied extensively by economists and sociologists.

The formal theory of coalitional games for more than two people was first proposed within the economic game theory field in John von Neumann and Oskar Morgenstern's book *The Theory of Games and Economic Behaviour* [79] (1944). Prior to this point, modern game theory research had focused on two player games [47].

Even though the formal theory of coalitional games are a modern creation, economic game-theoretic insights can be traced back to ancient times. The stand out coalitional game example being the Talmud's[1] recommendation that a recently deceased man's estate be divided to its creditors according to the coalitional game solution concept named the *nucleolus* [16], even though the nucleolus was formally described only in 1969 [108].

Modern work in the multi-agent systems field on coalition formation differs from work in other fields, because it focuses on the *computational* and *communication* aspects of coalition formation. Simplistically, the different fields' approaches to coalition formation can be viewed as the manner in which they investigate the two previously highlighted issues: economists would detail *exactly what* solutions satisfy the issues; sociologists would attempt to describe *why* these solutions satisfy the issues; while computer scientists would detail *how* the agents would find these satisfying solutions. Of course this is a simplistic overview, as researchers from the different fields can and do overlap in their approaches depending on the circumstances of their research.

As the multi-agent system approach to coalition formation is generally different than other field's approaches, new interesting research questions are appearing, as described in the next section.

1.3 Research Question

In multi-agent systems, some of the traditional assumptions of game-theoretic coalition formation may not be appropriate. This generally relates to the distributed nature of multi-agent systems, where there are distributed computational resources and the individual agents may have: (a) partial knowledge of the system; (b) uncertainty over certain sections of the system; and (c) communication costs when forming a coalition.

[1] The Talmud serves as a basis of Jewish law and is a collection of ancient law and tradition from the first five centuries A.D.

In a recent book on the computational aspects of coalition formation in multi-agent systems [36], it was highlighted that an interesting current research question is *"how can decentralised coalition formation protocols with desirable characteristics for multi-agent systems be designed?"*. This potential research question requires a definition of *desirable characteristics*. Some desirable characteristics for multi-agent systems coalition formation protocols are:

(i) **Balanced computational load**: How can the required computation of the coalition formation protocol be distributed approximately evenly between the agents?

(ii) **Distributed knowledge will not affect results**: How can optimal results be found when each agent has only partial knowledge on the values of the potential coalitions?

(iii) **Bounded communication costs**: What are the communication costs when forming coalitions and can they be bounded or eradicated?

(iv) **Valuation disagreements can be handled**: How can the agents form coalitions when they disagree on the value of some or all of the coalitions?

Given these preliminaries, the research question of this thesis can be introduced:

How can decentralised coalition formation methods be designed that: (i) balance the computational load; (ii) do not allow distributed knowledge to affect results; (iii) bound the communication costs; and (iv) handle valuation disagreements of a quantitative or qualitative manner?

Not all of the desirable characteristics are fulfilled by every coalition formation method introduced in this thesis. This is because (in general) the complexity level of the design increases, when more desirable criteria are added. Additionally, given the environment the agents are in, all of the given desirable criteria may not need to be satisfied. For example, in an environment where certain values on each coalition is guaranteed, then valuation disagreements do not have to be considered.

Regardless, the integration of more desirable criteria into a single coalition formation method, compared to what has been managed within this thesis, is left as an open challenge.

1.4 Thesis Outline and Contributions

The process of coalition formation studied in this thesis consists of three stages [104]:

1. Finding the value of each possible coalition;
2. Agreeing on the set of coalitions to form;
3. Agreeing on how the benefit of each coalition should be distributed to the individual agents (if there are different possible ways that it can be distributed).

For the first stage, the agents may value coalitions in a quantitative or qualitative manner. In Chapter 3, the qualitative coalition valuation style is investigated. Chapter 3 assumes that the different agents' qualitative valuations of the coalitions comes from the agents using (possibly different) instantiations of the Value-based Alternating Transition System environmental model [135]. Allowing different instantiations of the environmental model naturally allows for the possibility of conflicting information to occur. In Chapter 3, a well known argumentation scheme for practical reasoning, named AS1 and given in [13], is extended to allow for logical reasoning over conflicting information on what coalitions should form. To use this new argumentation scheme to find acceptable coalitions to form, critical questions are also presented that are modifications of the critical questions in [13]. Instantiations of these argumentation scheme are then communicated between the agents through an inquiry and persuasion argumentation-based dialogue game that is a modified version of the dialogue games of [22] and [21]. Finally, new definitions of argument acceptability were defined for Value-based Argumentation Frameworks [19], so that the agents can be informed on the strength of the different coalition formation arguments.

Additionally in the first stage of coalition formation, the agents may value the coalitions in a quantitative manner. If the quantitative value of a coalition is defined by a formula that is the same for any agent, then the process of calculating the value of every possible coalition can be shared between all the agents. Sharing the burdening of the coalition value calculations is useful because the number of possible coalitions grows at a rate of $2^n - 1$ (where n is the number of agents). In Chapter 4, a new algorithm, named the *distributed coalition generation* (DCG) algorithm, for distributing the coalition value calculations around the agents is presented that: (a) requires no communication; (b) gives each agent an approximately equal share of coalition value calculations; (c) allows every coalition's value to be calculated; (d) allows no coalition's value to be calculated more than once; (e) requires any two agents who have equal value calculation shares to have an equal number of operations to perform; and (f) assigns each agent a coalition value to calculate *only* if that agent is a member of the coalition. It is in fact (e) and (f) which are the unique elements of the DCG algorithm, as (a to d) are achieved in [90].

The DCG algorithm is faster than an agent calculating the value of every coalition that includes itself (named in this thesis as the AgentInAll algorithm), if the complexity of the value calculation function is greater than linear according to the size of the coalition. For instance, for 25 agents and a coalition value calculation function of $O(s^2)$ complexity (where s is the coalition's size), an agent using the AgentInAll algorithm took approximately 3.5 seconds to calculate all of its assigned coalitions, while an agent using the DCG algorithm took approximately 1 second[2]. The performance of DCG against AgentInAll gets better the higher the complexity of the coalition value calculation function. For instance, for 25 agents and a coalition value calculation function of $O(2^s)$ complexity (where s is the coalition's size), an agent using the AgentInAll algorithm took approximately $10\frac{1}{2}$ minutes, while an agent using the DCG algorithm took approximately 4 seconds.

[2]The PC on which the simulations were run had a processor: Intel(R) Core(TM) i7-4810MQ 2.80 GHz, with 8GB of RAM.

Once the agents have calculated each coalition's value, then they can use this information to find: the utility maximising set of coalitions to form (referred to as an optimal solution); and a manner in which to divide the benefit of each coalition between the agents so that no single agent can object to this division (known as a stable solution). In Chapter 5, a distributed algorithm named the *decentralised dynamic programming* (DDP) algorithm, takes distributed coalition value calculations as input and guarantees that an optimal and most stable solution will be found that is equivalent to the full knowledge solution (under the specific weak least core$^+$ definition of stability, defined in Chapter 5). This guarantee is provided by: (i) an extension to a theorem from [41] to proof that the synergy coalitional group (SCG) representation of [41] always includes the necessary coalitions for a weak least core$^+$ solution to be found; and (ii) a modification to the DP algorithm of [138] so that all the SCG coalitions are definitely found. Just using the SCG representation to find a stable outcome can be beneficial, because the SCG representation can be a lot smaller than the full exponential representation for certain standard coalition value distributions. For example, for 12 agents and a normal coalition value distribution, the experiments of Chapter 5 found that approximately 6% of the coalitions in the exponential representation were present in the SCG representation.

Finally even when the agents value the coalitions quantitatively, there can still be valuation disagreements. This issue was investigated in Chapter 6, that introduced the *valuation disagreement coalitional game* (VCG) model. The associated stability solution concepts of this model allow the agents to be more satisfied with their expected payoff than in the stability solution concepts of models that solves the valuation disagreement issue with a single percentage based agreement on the division of the gains of the coalition, such as [34][3]. For instance, for 7 agents using a uniform coalition value distribution, the valuation disagreement model offered the agents approximately 18% more expected payoff compared to using a single percentage based model (given the same reported valuations for each agent in both models). Additionally the VCG model allows coalitions to cooperate that models using the single percentage based agreement method do not. For instance for 7 agents using a uniform coalition value distribution, in 90% of the experiments conducted, coalitions were formed using the VCG model that the single percentage based model did not allow to cooperate. Finally the solution concepts of VCG encourage pessimistic coalition valuations. Pessimistic valuations mean the agent's expected payoffs will be less likely to be above their true payoff and so the agents are more likely to be in profit over their expected payoffs once the true values of the coalitions are revealed.

To summarise, this thesis is structured into seven chapters outlined as follows:

Chapter 2 presents a literature review on the topics related to coalition formation in multi-agent systems and the various coalition formation methods detailed within this thesis.

Chapter 3 presents two decentralised dialogue game protocols, named the *coalition inquiry* ($C\text{-}inq$) and the *coalition persuasion to action* ($C\text{-}pAct$) dialogue games. These dialogue

[3] An example of a single percentage based agreement is the following: if a coalition of agent i and agent j is to form, then agent i will take 60% of the coalition's gain, while agent j will take 40%

games can be used together to allow agents to reason over the best coalitions to form, using an argumentation-based method. The agents using these two protocols may have conflicting preferences, views of the world and coalition valuations, all described in a qualitative manner.

Chapter 4 presents a new decentralised algorithm, named the *distributed coalition generation* (DCG) algorithm. The DCG algorithm distributes the coalition value calculations to the agents of the system in a manner that gives no redundant coalition value calculations and every coalition assigned to an agent, includes that agent as a member.

Chapter 5 presents two new decentralised algorithms, the *decentralised dynamic programming* (DDP) algorithm and the DDP* algorithm (which is a slight modification of DDP). These algorithms, designed for characteristic function games, allow the agents to find the optimal coalition structure and a stable payoff vector (i.e. complete the coalition formation process), given distributed coalition value calculations.

Chapter 6 presents a new coalitional game model, named the *valuation disagreement coalitional game* model, with associated solution concepts, for situations where the agents can value their coalitions differently in a quantitative manner. This model increases the total expected payoff of the agents compared to using a model that divides the gains of each coalition via a single percentage demand vector.

Chapter 7 presents the thesis conclusions and identifies possible future research paths.

Additionally an appendix has been included. **Appendix A** is a proof, developed with Professor Paul Dunne, to show that the DCG algorithm of Chapter 4 achieves the properties claimed.

1.5 Published Work

The research presented in this thesis has been developed under the supervision of Dr. Katie Atkinson, Dr. Terry Payne and Professor Trevor Bench-Capon. A summary of the publications (to date) relating to the work presented in this thesis is as follows:

- The contributions of the $C\text{-}inq$ protocol, $C\text{-}pAct$ protocol and critical questions of Chapter 3, builds on work undertaken with Katie Atkinson, Terry Payne and Elizabeth Black and was published as "An Implemented Dialogue System for Inquiry and Persuasion" in the first International Workshop on the Theory and Applications of Formal Argumentation (TAFA), in Barcelona, Spain, 2011, [101].

- The contributions of the argumentation scheme of Chapter 3 and the application of the critical questions of [101] to coalition formation, builds on work published as "A Persuasive Dialogue Game for Coalition Formation" in the inaugural Imperial College Student Workshop (ICCSW), in London, UK, 2011, [98].

- The contribution of the DCG algorithm for the distribution of coalition value calculations of Chapter 4, initially builds on work undertaken with Terry Payne, Trevor Bench-Capon and Katie Atkinson and was published as an extended abstract "Distributing Coalition Value Calculations to Self-Interested Agents" in the thirteenth International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS), in Paris, France, 2014, [102].

- The contribution of the DCG algorithm for the distribution of coalition value calculations of Chapter 4 additionally builds on work undertaken with Katie Atkinson, Paul Dunne and Terry Payne and was published as "Distributing Coalition Value Calculations to Coalition Members" in the twenty-ninth Conference on Artifical Intelligence (AAAI), in Austin, Texas, USA, 2015, [99].

Chapter 2

Chapter 2

Literature Review

In this literature review, the topics of *coalitional games*, *cooperative game theory*, *coalition formation in multi-agent systems*, *agent communication* and *argumentation* are discussed. Discussing these topics allows the background of coalition formation to be detailed and the limitations of current multi-agent system coalition formation methods to be described, thus showing where the motivation for this thesis's research question occurs from.

The literature review commences by detailing the formal foundations of coalitional games in Sections 2.1, 2.2 and 2.3. Section 2.4 goes on to describe, with examples, how multi-agent systems operate within these formal foundations to form coalitions. Later in this thesis, Chapters 4 and 5 offer new methods for agents to form coalitions within these predefined quantitative coalitional game foundations, while Chapter 6 gives a new formal quantitative coalitional game definition.

The literature review concludes with Sections 2.5 and 2.6 that discuss models of agent communication and argumentation, including how agents can communicate effectively and reason logically over possibly conflicting qualitative information. Later in this thesis, these models are built on in Chapter 3 to give a new method for agents to form coalitions using these predefined qualitative foundations.

2.1 Characteristic Function Games

In the seminal work of [79], von Neumann and Morgenstern constructed the theory of n-*person cooperative games*, where groups of agents can join together to form coalitions. They detailed n-person cooperative games in characteristic function form, where each coalition has an associated real numeric *utility* value that it can achieve:

Definition 2: *A* **characteristic function game** *is denoted* $\mathcal{G} = \langle N, v \rangle$*, where* N *is the set of agents in the game and* v *is the characteristic function that maps every potential coalition* $C \subseteq N$ *to a real numeric value (i.e.* $v(2^N) \rightarrow \mathbb{R}$*). By default an empty coalition receives no utility payoff (i.e.* $v(\emptyset) = 0$*).*

In the literature, some assumptions on characteristic function games are made (see Section 2.3 for discussion on alternatives to these assumptions):

- The characteristic function formula is not detailed in the formalism.
- The characteristic function formula is agreed on by all the agents beforehand.
- All the agents have perfect information to use the characteristic function formula to get an accurate utility value for each coalition.
- The value $v(C)$ returned by the characteristic function is the largest value that coalition C could achieve.
- Each coalition's utility value is independent of what non-members do.
- The value of each coalition does *not* change over time.
- The value of each coalition can be distributed to its members in any manner. Games with this property are known as *transferable utility games*.

Additionally, it is traditionally assumed that in characteristic function games the coalition of all agents (known as the *grand coalition*) forms.

Definition 3: *A coalition C is the* **grand coalition** *for a characteristic function game $\mathcal{G} = \langle N, v \rangle$ iff $C = N$.*

It is rational to assume that the grand coalition will form in superadditive games [106]:

Definition 4: *A characteristic function game $\mathcal{G} = \langle N, v \rangle$ is* **superadditive** *if for any two disjoint coalitions D and E, where $D, E \subset N$ and $D \cap E = \emptyset$, $v(D \cup E) > v(D) + v(E)$.*

Traditionally the grand coalition is assumed to form in any characteristic function game because it is argued that at worst the agents in the larger coalition can behave as if they were in smaller coalitions [104]. Yet [104] details the reasons that this classical assumption should not always be assumed in multi-agent systems: (i) there maybe a coordination overhead for forming the grand coalition, e.g. communication costs or anti-trust penalties; (ii) finding the optimal manner the grand coalition can work together maybe more costly than finding out how a group of smaller coalitions can optimally work together; and (iii) if time is limited, the agents may not be able to carry out the communication and computation required to coordinate effectively within the larger coalition. Therefore in multi-agent systems it may be beneficial to form a set of coalitions, known as a coalition structure [104], defined as:

Definition 5: *A* coalition structure *(CS) is the set of coalitions in a system, denoted: $CS = \{C^1, ..., C^k\}$. In a characteristic function game $\mathcal{G} = \langle N, v \rangle$, the coalition structure has the following properties:*

$$\bigcup_{j=1}^{k} C^j = N \text{ and} \tag{2.1}$$

$$C^i \cap C^j = \emptyset \text{ for any } i, j \in \{1, ..., k\} \text{ where } i \neq j. \tag{2.2}$$

The first condition states that the union of all the coalitions must equal the total set of agents of the game. The second condition states that each agent should only be a member of a single coalition. Overlapping coalitional games (e.g. [35, 42]) relax the second condition, yet they are not addressed here as they are outside the scope of this thesis. Throughout the thesis $v(CS)$ is written to denote the total value of a coalition structure CS, i.e. $v(CS) = \sum_{C \in CS} v(C)$. A special type of coalition structure, measured by it's *social welfare* (i.e. the sum of the values of all the coalitions in the coalition structure), is as follows:

Definition 6: *An* optimal coalition structure *(CS^*) is a coalition structure that maximises social welfare, i.e.: $\neg\exists CS'$ where $v(CS') \geq v(CS^*)$.*

Finding an *optimal coalition structure* (CS^*) that maximises social welfare is known as the *coalition structure generation* problem [104], discussed in Section 2.4.2.

Example 1: *Consider the characteristic function game $\mathcal{G} = \langle N, v \rangle$ where $N = \{1, 2, 3\}$ and the characteristic function v gives the following valuations: $v(\{1\}) = 2$ (i.e. the utility value, of agent 1 by itself, is 2), $v(\{2\}) = 1$, $v(\{3\}) = 2$, $v(\{1, 2\}) = 10$ (i.e. the utility value of the coalition, containing agents 1 and 2, is 10), $v(\{1, 3\}) = 8$, $v(\{2, 3\}) = 9$ and $v(\{1, 2, 3\}) = 10$ (i.e. the utility value, of the grand coalition, is 10). Then $CS^* = \{\{1, 2\}, \{3\}\}$ because $v(CS^*) = v(\{1, 2\}) + v(\{3\}) = 10 + 2 = 12$, and no other valid coalition structure can give a higher total value for the game $\mathcal{G}$.*

Finding an acceptable coalition structure is only half the requirement in characteristic function games. Agents also need to be motivated to join a coalition. In characteristic function games, this motivation comes from the transferable utility that each agent in the coalition receives. A *payoff vector* x is used to distribute the total utility value of the coalition structure to the individual agents [79].

Definition 7: *A* **payoff vector** *is denoted: $x = (x_1, ...x_n) \in \mathbb{R}^n$ where $x_i \geq 0$ for all $i \in N$ and x_i corresponds to the utility payoff assigned to agent i. Throughout the thesis, notation will be abused by using $x(C)$ to denote the component of the payoff vector x that has individual payoffs only for agents of the coalition C, i.e. $x(C) = \sum_{i \in C} x_i$.*

Given a payoff vector, the worst payoff any agent can receive is zero. An example payoff vector is $x(1, 4, 5, 2)$, where the coalition $\{1, 4\}$ receives $x(\{1, 4\}) = x_1 + x_4 = 1 + 2 = 3$ total payoff. Traditionally all the payoff of a coalition can only be divided between agents of that coalition, i.e. traditionally side payments between coalitions are *not allowed* [83]. Yet there exists some research in multi-agent systems that is starting to relax the assumption of this property (e.g. [2, 66]).

A problem with some payoff vectors is that they may not satisfy the requirements of all the agents of the system. The most obvious requirement for a payoff vector is the following [83]:

Definition 8: *A payoff vector is* **individually rational** *if all agents receive at least as much utility payoff as they would receive by themselves, i.e. $\forall i \in N$, $x_i \geq v(\{i\})$.*

A payoff vector that satisfies individual rationality is known as an *imputation* and is defined as [83]:

Definition 9: *An* **imputation** *is a payoff vector that satisfies individual rationality. The full set of imputations of a characteristic function game is denoted* $Imp(N, v)$.

Two different imputations may not be of the same quality, which motivates the next definition [83]:

Definition 10: *Given two payoff vector imputations* x^p *and* x^q, x^p *is said to* **dominate** x^q *if there is a non-empty coalition* S *such that* $\forall i \in S$, $x_i^p > x_i^q$ *and* $x^p(S) \leq v(S)$.

The notion of dominance is used by the *core* and *stable sets* definitions (described in Section 2.2), which detail the acceptable payoff vectors for self-interested agents.

Example 2: *Consider the characteristic function game* $\mathcal{G} = \langle N, v \rangle$ *where* $N = \{1, 2, 3\}$ *and the characteristic function* v *gives the following valuations* $v(\{1\}) = 1$, $v(\{2\}) = 2$, $v(\{3\}) = 2$, $v(\{1, 2\}) = 4$, $v(\{1, 3\}) = 6$, $v(\{2, 3\}) = 5$ *and* $v(\{1, 2, 3\}) = 9$. *The optimal coalition structure of this game is* $CS^* = \{\{1, 2, 3\}\}$. *The following payoff vectors, dividing* $v(CS^*) = 9$ *between the agents, are imputations:* $x^1(3, 3, 3)$*; and* $x^2(2, 5, 2)$ *(where* x^1 *dominates* x^2 *via coalition* $\{1, 3\}$ *because* $x_1^1 > x_1^2$, $x_3^1 > x_3^2$ *and* $x^1(\{1, 3\}) \leq v(\{1, 3\})$*). The following payoff vectors are not imputations:* $x^4(0, 4, 5)$ *because* $x_1^4 < v(\{1\})$*;* $x^5(4, 1, 4)$ *because* $x_2^5 < v(\{2\})$*; and* $x^6(0, 9, 0)$ *because* $x_1^6 < v(\{1\})$ *and* $x_3^6 < v(\{2\})$.

Given the above information, the outcome of a characteristic function game can now be detailed [104]:

Definition 11: *An* **outcome** *of a characteristic function game* $\mathcal{G} = \langle N, v \rangle$ *for* n *agents is a coalition structure and payoff vector pair, denoted:* $\langle CS, x \rangle$ *where* CS *is a set of* k *coalitions, denoted* $\{C^1, ..., C^k\}$ *and* x *is the payoff vector, denoted* $x(x_1, ..., x_n)$.

Example 3: *Consider the characteristic function game* $G = \langle N, v \rangle$ *where* $N = \{1, 2, 3\}$ *and the characteristic function* v *gives the following valuations:* $v(\{1\}) = 1$, $v(\{2\}) = 2$, $v(\{3\}) = 2$, $v(\{1, 2\}) = 4$, $v(\{1, 3\}) = 6$, $v(\{2, 3\}) = 5$ *and* $v(\{1, 2, 3\}) = 9$. *Example outcomes to this game are:* $\langle \{\{1, 2, 3\}\}, x(3, 3, 3) \rangle$*;* $\langle \{\{1, 2\}, \{3\}\}, x(2, 2, 2) \rangle$*; and* $\langle \{\{1, 3\}, \{2\}\}, x(2, 2, 4) \rangle$.

Some outcomes are favoured more than others for reasons of social welfare or individual self-interest. The next section looks at methods to reason over what are the 'best' characteristic function game outcomes.

In this thesis, characteristic function games are used in Chapters 4 and 5. Chapter 4 details an algorithm to distribute the calculation of the coalition values around the agents of the system. Chapter 5 describes a distributed algorithm, that finds characteristic function game outcomes that are 'best', given a specific definition on what a 'best' solution is.

2.2 Cooperative Game Theory Solution Concepts

Game theory [83] studies the theory of optimal decision making in environments populated by self-interested agents and has two strands: *cooperative game theory*, where the agents are allowed to form temporary binding contracts for cooperative behaviour (thereby creating a coalition); and *non-cooperative game theory*, where the agents cannot create these contracts.

In cooperative game theory, the cooperative element of the game does not imply that each agent will definitely do what the other agents want. Instead the cooperative element refers to

the agents cooperating in temporary coalitions towards a common goal, if this is in each agent's *best interest*. Therefore, similar to non-cooperative game theory, the agents of a characteristic function game are motivated by utility maximisation.

Cooperative game theory has many solution concepts that detail what outcomes of the coalitional game are desirable, based on the properties of *fairness* or *stability*. Generally, the process of finding an outcome in a solution concept is not detailed in its definition. When a solution concept focuses on the property of stability, the solution concept relies on the idea that a subset of agents might "object" to an outcome and so become a *blocking coalition* for this outcome. An outcome has a blocking coalition if there is an alternative outcome for that coalition that improves the payoff of at least one of its members and *does not* worsen the payoff of the other members. Alternatively, when a solution concept focuses on the property of fairness, the solution concept aims to give each agent a payoff that reflects its full potential contribution to the game as a whole.

The concern of this thesis is stability solution concepts due to the possible self-interest of the agents using the decentralised methods presented within the subsequent chapters. This reasoning is expanded on at the end of Section 2.2.1, after the formal introduction of various stability solution concepts.

2.2.1 Classical Core Based Solution Concepts

In this section, the classical definitions of stability solution concepts for characteristic function games are detailed, where the grand coalition is assumed to form. In Section 2.2.3, these classical definitions are modified for the possible formation of different coalition structures within characteristic function games. In Section 2.3, some solution concepts for other coalitional game types are discussed.

This section begins with the following general definition of stability, taken from [136]:

Definition 12: *A coalitional game is* **stable** *if all the agents have* no valid objection *to the coalitional game outcome. Stability is a necessary but not sufficient condition for a payoff vector (or coalition structure) to be agreed, because there could be multiple possible stable outcomes.*

The idea of a valid objection changes depending on which stability concept is used. Perhaps the most intuitive stability based solution concept is known as the **core**. The core was introduced in [58] (defined below) and is the set of stable outcomes where no subset of agents have an incentive to deviate, i.e., there is no payoff distribution that would make at least one member of a deviating coalition better off, without negatively affecting the other members of the deviating coalition.

Definition 13: *The* **core***:- For a set of agents N, a payoff vector $x = (x_1, ..., x_n)$ for the grand coalition is in the core iff:*

$$\sum_{i \in N} x_i = v(N)$$
$$\sum_{i \in C} x_i \geq v(C), \forall C \subset N$$

The payoff vector must be *feasible* and *efficient*. The payoff vector must be feasible in the sense that the total value of the payoff vector should not be above the value of the grand coalition and the payoff vector must be efficient in the sense that the grand coalition payoff should be totally shared between the agents of the coalitional game (i.e. $x(N) = v(N)$). Additionally there must be no *blocking coalition*, meaning there should be no coalition who could deviate from the grand coalition and get more payoff by dividing $v(C)$ between themselves (i.e. $\sum_{i \in C} x_i \geq v(C), \forall C \subset N$)[1]. For the core solution concept with a coalition structure see Section 2.2.3.

Example 4: *Consider a characteristic function game $\mathcal{G} = \langle N, v \rangle$ where $N = \{1, 2, 3, 4\}$ and the characteristic function v gives the following valuations: $v(\{1, 2\}) = 12$, $v(\{3, 4\}) = 10$, $v(\{2, 3\}) = 8$, $v(\{1, 2, 3, 4\}) = 24$ and $v(C) = 0$ for every other possible coalition C. In the traditional definition of the core, the grand coalition $\{1, 2, 3, 4\}$ forms. A payoff vector in the core has to give every potential deviating coalition C' a payoff $x(C')$ greater than or equal to its value $v(C')$. Example payoff vectors that satisfy these restrictions are: (i) $x(6, 6, 6, 6)$; (ii) $x(7, 7, 5, 5)$; and (iii) $x(7, 6, 2, 9)$. Example payoff vectors that violate these restrictions are: (iv) $x(5, 5, 7, 7)$ because $x(\{1, 2\}) < v(\{1, 2\})$; (v) $x(8, 8, 4, 4)$ because $x(\{3, 4\}) < v(\{3, 4\})$; and (vi) $x(10, 4, 2, 8)$ because $x(\{2, 3\}) < v(\{2, 3\})$.*

A problem with the core is that it can sometimes be empty. To tackle this issue, the concept of the ϵ-*core* is introduced, where the ϵ-core is a more general case of the core. Two different definitions of the ϵ-*core* were introduced in [114], named the *strong* ϵ-*core* and the *weak* ϵ-*core*. Both concepts rely on the idea of coalitional excess:

Definition 14: *The* **strong excess** *of a coalition C for a payoff vector x is denoted $\epsilon^s(C, x)$ and calculated by:*

$$\epsilon^s(C, x) = v(C) - \sum_{i \in C} x_i$$

Definition 15: *The* **weak excess** *of a coalition C for a payoff vector x is denoted $\epsilon^w(C, x)$ and calculated by:*

$$\epsilon^w(C, x) = \frac{v(C) - \sum_{i \in C} x_i}{|C|}$$

Positive excess for a potential coalition C means that C has a higher utility value (i.e. $v(C)$) than the combined payoff the members of C currently receive (i.e. $\sum_{i \in C} x_i$). In this case, the members of C will benefit if C forms (providing there is no penalty on forming a new coalition). Negative excess for a potential coalition C, means that C has a lower utility value (i.e. $v(C)$) than the combined payoff the members of C currently receive (i.e. $\sum_{i \in C} x_i$). In this case, the

[1] Remember that sometimes in the thesis, notation will be abused by using $x(C)$ to denote $\sum_{i \in C} x_i$.

members of C will not benefit if C is formed. The maximum strong or weak excess, denoted ϵ^s and ϵ^w respectively, of all the possible coalitions $C \subset N$, is formalised as [114]:

Definition 16: *The* **maximum strong excess** ϵ^s *of a characteristic function game is the following:*

$$\epsilon^s = \max_{C \subset N} \left(v(C) - \sum_{i \in C} x_i \right)$$

Definition 17: *The* **maximum weak excess** ϵ^w *of a characteristic function game is the following:*

$$\epsilon^w = \max_{C \subset N} \left(\frac{v(C) - \sum_{i \in C} x_i}{|C|} \right)$$

The ϵ-cores are useful as they can relax the strict conditions of the core solution concept. The ϵ-cores are the set of stable outcomes where no subset of agents has an incentive to deviate *when* the subset has to pay a penalty for deviation [114]:

Definition 18: *The* **strong ϵ-core***:- For a characteristic function game $\mathcal{G} = \langle N, v \rangle$ and a value ϵ, a payoff vector $x = (x_1, ..., x_n)$ for the grand coalition is in the strong ϵ-core iff:*

$$\sum_{i \in N} x_i = v(N) \tag{2.3}$$

$$\sum_{i \in C} x_i \geq v(C) - \epsilon, \ \forall C \subset N \tag{2.4}$$

Definition 19: *The* **weak ϵ-core***:- For a characteristic function game $\mathcal{G} = \langle N, v \rangle$ and a value ϵ, a payoff vector $x = (x_1, ..., x_n)$ for the grand coalition is in the weak ϵ-core iff:*

$$\sum_{i \in N} x_i = v(N) \tag{2.5}$$

$$\sum_{i \in C} x_i \geq v(C) - |C|\epsilon, \ \forall C \subset N \tag{2.6}$$

The difference between the strong and weak ϵ-cores is that under the weak definition, the penalty of forming a new coalition is dependent on the size of that coalition, while the penalty for forming a new coalition under the strong definition is a fixed amount for any coalition.

When $\epsilon = 0$, the ϵ-core definitions are the same as the core. If ϵ is positive then the ϵ-cores are a wider version of the core whereas if ϵ is negative then they are tighter versions of the core. Again like the core, an ϵ-core payoff vector must be *feasible* and *efficient* (i.e. 2.3 and 2.5). Additionally there must be no *blocking coalition*, which means there should be no coalition who could deviate from the grand coalition, *pay the deviation penalty* and get more payoff by dividing the remaining payoff between themselves (i.e. 2.4 and 2.6). Like core solutions, ϵ-core solutions can only be defined over the grand coalition, for the ϵ-core solution concept with a coalition structure see Section 2.2.3.

Example 5: *Consider a characteristic function game* $\mathcal{G} = \langle N, v \rangle$ *where* $N = \{1, 2, 3, 4\}$ *and the characteristic function* v *gives the following valuations:* $v(\{1, 2\}) = 16$, $v(\{3, 4\}) = 14$, $v(\{2, 3\}) = 8$, $v(\{1, 2, 3, 4\}) = 24$ *and* $v(C) = 0$ *for every other possible coalition* C. *In the traditional definition of the* ϵ*-cores, the grand coalition* $\{1, 2, 3, 4\}$ *forms. A payoff vector in the* ϵ*-core has to give every potential deviating coalition* C' *a payoff* $x(C')$ *greater than or equal to its value* $v(C')$ *minus the deviation penalty. Example payoff vectors (that are indexed via their superscript) in different* ϵ*-cores are: (i)* $x^1(10, 0, 1, 13)$ *where* $\epsilon^s = 7$ *because* $x^1(\{2, 3\}) + 7 = v(\{2, 3\})$, *while* $\epsilon^w = 3.5$ *because* $\frac{v(\{2,3\}) - x^1(\{2,3\})}{|\{2,3\}|} = 3.5$*; (ii)* $x^2(8, 8, 4, 4)$ *where* $\epsilon^s = 6$ *because* $x^2(\{3, 4\}) + 6 = v(\{3, 4\})$, *while* $\epsilon^w = 3$ *because* $\frac{v(\{3,4\}) - x^2(\{3,4\})}{|\{3,4\}|} = 3$*; and (iii)* $x^3(8, 4, 4, 8)$ *where* $\epsilon = 4$ *because* $x^3(\{1, 2\}) + 4 = v(\{1, 2\})$, *while* $\epsilon^w = 2$ *because* $\frac{v(\{1,2\}) - x^3(\{1,2\})}{|\{1,2\}|} = 2$.

There are an infinite number of possible ϵ-cores. The *least core* is the smallest, non-empty ϵ-core, and so again there are two different definitions [114]:

Definition 20: *The* **strong least core***:- For a characteristic function game* $\mathcal{G} = \langle N, v \rangle$*, a payoff vector* $x = (x_1, ..., x_n)$ *is in the strong least core iff:*

x *is in the strong* ϵ*-core*
$\forall \epsilon' < \epsilon$*, the strong* ϵ'*-core is empty*

Definition 21: **weak least core***:- For a characteristic function game* $\mathcal{G} = \langle N, v \rangle$*, a payoff vector* $x = (x_1, ..., x_n)$ *is in the weak least core iff:*

x *is in the weak* ϵ*-core*
$\forall \epsilon' < \epsilon$*, the weak* ϵ'*-core is empty*

A payoff vector in the least cores minimises the maximum dissatisfaction of any coalition. The least cores therefore contain the payoff vectors that are *'least objectionable'*, under the strong or weak ϵ-core definition of objectionable. If the core of a characteristic function game is empty, the value of the least cores can be viewed as the minimum deviation penalty that is needed to stop a potential coalition from deviating from the grand coalition under the strong or weak ϵ core definition [109]. For the least core with coalition structure, see Section 2.2.3.

Example 6: *Consider a characteristic function game* $\mathcal{G} = \langle N, v \rangle$ *where* $N = \{1, 2, 3, 4\}$ *and the characteristic function* v *gives the following valuations:* $v(\{1, 2\}) = 12$, $v(\{3, 4\}) = 10$, $v(\{2, 3\}) = 8$, $v(\{1, 2, 3, 4\}) = 20$ *and* $v(C) = 0$ *for every other possible coalition* C. *In the traditional definition of the least-cores, the grand coalition* $\{1, 2, 3, 4\}$ *forms. A payoff vector in the least cores minimises the maximum strong or weak excess every potential deviating coalition* C' *has. Example payoff vectors that satisfy these restrictions are: (i)* $x(6, 5, 5, 4)$*; (ii)* $x(4, 7, 2, 7)$*; and (iii)* $x(5.5, 5.5, 4.5, 4.5)$*, all of which give the minimal* $\epsilon^s = 1$ *or* $\epsilon^w = 0.5$ *value, which are the least core values of this game. These are the least core values of the game*

because: (a) $\epsilon^s(\{1,2\}) = \epsilon^s(\{3,4\}) = 1$ *and* $\epsilon^w(\{1,2\}) = \epsilon^w(\{3,4\}) = 0.5$*; (b) any transfer of payoff from an agent of coalition* $\{1,2\}$ *to an agent of coalition* $\{3,4\}$ *(or vice versa) will result in the excess value of one potential coalition increasing; and (c) any transfer of payoff between the agents of the coalition* $\{1,2\}$ *(or between the agents of the coalition* $\{3,4\}$*) will not result in the excess value of either coalition decreasing. Example payoff vectors that are not in the least core are: (iv)* $x(2,5,7,6)$ *because* $\epsilon^s(\{1,2\}) = 12-(2+5) = 5 > 1$ *and* $\epsilon^w(\{1,2\}) = \frac{12-(2+5)}{|\{1,2\}|} = 2.5 > 0.5$*; (v)* $x(6,6,4,4)$ *because* $\epsilon^s(\{3,4\}) = 10-(4+4) = 2 > 1$ *and* $\epsilon^w(\{3,4\}) = \frac{10-(4+4)}{|\{3,4\}|} = 1 > 0.5$*; and (vi)* $x(10,0,0,10)$ *because* $\epsilon^s(\{2,3\}) = 8-(0+0) = 8 > 1$ *and* $\epsilon^w(\{2,3\}) = \frac{8-(0+0)}{|\{2,3\}|} = 4 > 0.5$.

The *nucleolus*, introduced by Schmeidler [108], is a further refinement of the strong least core (even though it could easily be modified to be a refinement of the weak least core). The nucleolus is a single payoff vector and is referred to as the "most stable point" inside the core/strong least core. To find the nucleolus, as detailed in [36], all of the strong excesses of each coalition can be organised into an *deficit vector* $\delta(x) = (\epsilon^s(x, C_1), ..., \epsilon^s(x, C_{2^n}))$, where the coalitions $C_1, ..., C_{2^n}$ is the ordering of all the possible coalitions from largest excess to smallest, i.e. $\epsilon^s(x, C_i) \geq \epsilon^s(x, C_j)$ where $i < j$. The nucleolus is the payoff vector that has the lexicographically smallest deficit vector. A deficit vector $\delta(x)$ for the payoff vector x is lexicographically smaller then the deficit vector $\delta(y)$ for the payoff vector y, i.e. $\delta(x) <_{lex} \delta(y)$ if the first $i \in \{0, ..., 2^{n-1}\}$ entries of the deficit vectors are equal and then the $i+1$ entry of $\delta(x)$ is smaller than the $i+1$ entry of $\delta(y)$.

Definition 22: *The* **nucleolus***:- For a characteristic function game* $\mathcal{G} = \langle N, v \rangle$*, a payoff vector* $x = (x_1, ..., x_n)$ *for the grand coalition is the nucleolus of the game iff:*

$$x \in Imp(N, v)$$
$$\delta(x) <_{lex} \delta(y) \text{ for all } y \in Imp(N, v) \text{ where } x \neq y$$

The first condition states that the nucleolus has to be an imputation payoff vector (one that satisfies individual rationality for all agents). The second condition states that the nucleolus must be lexicographically smaller than any other possible imputation payoff vector. If the same definition is used, except with all possible payoff vectors (i.e. not just imputations) then the *pre-nucleolus* is found.

Example 7: *Consider a characteristic function game* $\mathcal{G} = \langle N, v \rangle$ *where* $N = \{1,2,3\}$ *and the characteristic function* v *gives the following valuations:* $v(\{1,2\}) = 12$*,* $v(\{1,3\}) = 10$*,* $v(\{2,3\}) = 8$*,* $v(\{1,2,3\}) = 15$ *and* $v(C) = 0$ *for every other possible coalition* C*. In the traditional definition of the nucleolus, the grand coalition* $\{1,2,3\}$ *forms. There is only one payoff vector (that is also an imputation) in the nucleolus. The nucleolus for this game is:* $x(7,5,3)$ *that gives the strong least core value* $\epsilon^s = 0$ *and the deficit vector of* $\delta(x) = (\epsilon^s(x, \{1,2\}) = 0, \epsilon^s(x, \{1,3\}) = 0, \epsilon^s(x, \{2,3\}) = 0, \epsilon^s(x, \{3\}) = -3, \epsilon^s(x, \{2\}) = -5, \epsilon^s(x, \{1\}) = -7)$*. It can be seen that* x *is in fact the nucleolus because any payoff transfer from any* x_i *to another* x_j *will raise the strong excess value of the coalition including* i *and not* j *over the current*

maximum excess value of zero.

This thesis uses solution concepts based on the core, ϵ-cores and least cores, yet does not use the solution concept of the nucleolus. The reason for this decision is that the nucleolus is significantly more computationally intensive to find due to the deficit vector containing all $C \subseteq N$ coalitions, compared to the core/ϵ-cores/least cores solution concepts that need to only consider the coalition at the start of the deficit vector. Therefore, in decentralised environments, with the possibility of agents not having full coalitional value knowledge, finding the nucleolus will require much more communication in the majority of cases compared to searching for a variation of the core or least-core.

The reasoning for not using the nucleolus, also applies for not using the most *fair* cooperative game solution concept, named the Shapley value [113]. Like the nucleolus the Shapley value is computationally intensive to find, as the Shapley value requires multiple calculations per coalition (after the coalition value has been found), to calculate the marginal contribution of each agent to each potential coalition it is a member of. Additionally the Shapley value may not even assign a payoff vector that is individually rational [49], which self-interested agents should not agree with.

2.2.2 Other Classical Solution Concepts

The solution concepts mentioned in detail so far are the main core-related solution concepts. In this section, the other main stability solution concepts for characteristic function games will be detailed, and the reasons for not using them in this thesis will be explained.

- When the core is empty, the ϵ-core or least core definition allows stability to be recovered. An alternative method to recover stability in an empty core game is through the *cost of stability* solution concept [18]. The cost of stability solution concept finds the minimum payment Δ needed to give to the agents of a characteristic function game in order for the core to be non-empty. The least core value of a characteristic function game corresponds to the upper bound on Δ [18].

  The cost of stability is not used in this thesis as it requires a benevolent external party to pay Δ and this thesis does not make the assumption that such a party exists.

- The *kernel* introduced in [45] is another stability solution concept. It measures the strength (surplus) of the agents by the maximum excess they can obtain by forming a new coalition. An agent i compares the strength of itself with the strength of another agent j in its current coalition by comparing i's surplus in coalitions that do not include j with j's surplus in coalitions that do not include i. The comparison between agents indicates whether the current payoff distribution is acceptable or not. Unlike the core, the kernel always exists.

  The kernel was not used in this thesis since it is not linked directly to an optimal coalition structure [2]. The work of this thesis is not restricted to self-interested agents and so a socially optimal coalition structure may be an important outcome. For instance, Chapter 5 explores how to get a socially optimal yet stable coalition structure. As kernel-stable

payoff vectors can be found for many coalition structures, it cannot be guaranteed that the agents will find the optimal coalition structure. Also in a fully distributed algorithm with self-interested agents, it cannot be guaranteed that agents will be motivated to find a kernel-stable payoff vector for any coalition structure, as self-interested agents may benefit from hiding information [1]. Yet the distributed algorithm detailed in Chapter 5 motivates agents to reveal new information when looking for a payoff vector in a variant of the least core.

- The *stable set* (also known as the von Neumann and Morgenstern solution) [79] is based on the idea of payoff vector dominance (as introduced in Section 2.1) and is, unlike the other solution concepts, a multi-set solution concept. If the core is non-empty, then the core is contained in all stable sets. While it was initially believed at least one stable set always existed, a complicated 10-agent counter example was produced in [69] showing a stable set *may not* always exist for every characteristic function game. Stable sets are not used in this thesis as the core set of results can be quite large as it is, and the core coincides nicely with other complementary non-empty solution concepts such as the strong and weak least core, the coalition structure-least core (introduced in Section 2.2.3), and the valuation-disagreement least core (introduced in Chapter 6).

- The *bargaining set* [15], which is always non empty, relaxes the stability requirements of the core. It is based on the idea of *objections* and *counter objections*. An agent i can object to a payoff vector x and ask for payoff from agent j. This *objection* can only occur if there is a coalition C not currently formed that includes i yet excludes j, where the proposed payoff for every agent in C is greater than or equal to what it received in x. Yet this objection is ineffective for the bargaining set if agent j can find a *counter objection*. This *counter objection* can only occur if there is an coalition C not currently formed that excludes i yet includes j, where the payoff for every agent in C is greater than or equal to what it received in x *and* agent i's objection.

  The bargaining set was not used in this thesis because this thesis relaxes the requirements of the core through its least core variants.

2.2.3 Coalition Structure Stability Solution Concepts

In multi-agent systems, the classical assumption that the grand coalition always forms can be discarded [104], because as previously stated: (i) there maybe a coordination overhead for forming the grand coalition, e.g. communication costs or anti-trust penalties; (ii) finding the optimal manner the grand coalition can work together maybe more costly than finding out how a group of smaller coalitions can optimally work together; and (iii) if time is limited, the agents may not be able to carry out the communication and computation required to coordinate effectively within the larger coalition. Therefore the classical definitions of the stability solution concepts need to be modified. In this section, stability solutions from Section 2.2.1 are modified to include coalition structures.

Firstly the *coalition structure-core* (CS-core) is presented, which is a modified version of the core [14]:

Definition 23: *The* **CS-core***:- For a set of agents N, a coalition structure CS and a payoff vector pair $\langle CS, x\rangle$ is in the CS-core iff:*

$$\sum_{i\in C} x_i = v(C), \forall C \in CS$$
$$\sum_{i\in C} x_i \geq v(C), \forall C \subseteq N$$

CS-core solutions allow any coalition structure to form and not just the grand coalition. If the core is non-empty then the CS-core is also non-empty, i.e. if x is a core stable solution then $\langle N, x\rangle$ is a CS-core stable solution [55]. But the core can be empty when the CS-core is non-empty, as [112] shows with the following example:

Example 8: *Consider a characteristic function game $G = \langle N, v\rangle$ where $N = \{1, 2\}$ and the characteristic function v gives the following valuations: $v(C) = 1$, $\forall C \subseteq N$. Then there exists no solution in the core because each agent can get the value of the grand coalition by itself. Yet the solution $\langle\{\{1\}, \{2\}\}, x(1, 1)\rangle$ (i.e. where agent 1 and 2 are both by themselves and both receive a utility value of 1) is in the CS-core because the only alternative is to form the grand coalition but forming the grand coalition will decrease the utility value that one or both of the agents receive.*

Just like the core, the CS-core can sometimes be empty. Therefore the solution concepts of the *ϵ-CS-core* and *CS-least core* were introduced, which are modifications of the strong ϵ-core and strong least core respectively [112].

Definition 24: *The* **ϵ-CS-core***:- For a set of agents N and a value ϵ, a coalition structure and payoff vector pair $\langle CS, x\rangle$ is in ϵ-CS- core iff:*

$$\sum_{i\in C} x_i = v(C), \forall C \in CS$$
$$\sum_{i\in C} x_i \geq v(C) - \epsilon, \forall C \subseteq N$$

Every coalition in an ϵ-CS-core stable coalition structure must have its full utility value divided between its members. The ϵ value can be viewed as a penalty for deviation from the given coalition structure. When $\epsilon = 0$, the ϵ-CS-core definition is the same as the CS-core.

Example 9: *Consider a characteristic function game $\mathcal{G} = \langle N, v\rangle$ where $N = \{1, 2, 3, 4\}$ and the characteristic function v gives the following valuations: $v(\{1, 2\}) = 16$, $v(\{3, 4\}) = 14$, $v(\{2, 3\}) = 8$, $v(\{1, 4\}) = 20$, $v(\{1, 2, 3, 4\}) = 24$ and $v(C) = 0$ for every other possible coalition C. An outcome in the ϵ-CS-core has to give every potential deviating coalition C' a payoff $x(C')$ greater than or equal to its value $v(C')$ minus the deviation penalty ϵ. Example outcomes in different ϵ-CS-cores are: (i) $\langle\{\{1, 2\}, \{3, 4\}\}, x(8, 8, 7, 7)\rangle$ where $\epsilon = 5$ because $x(\{1, 4\}) + 5 = v(\{1, 5\})$; (ii) $\langle\{\{1, 2, 3, 4\}\}, x(9, 1, 3, 11)\rangle$ where $\epsilon = 6$ because $x(\{1, 2\}) +$*

$6 = v(\{1,2\})$*; and (iii)* $\langle\{\{2,3\},\{1,4\}\}, x(10,4,4,10)\rangle$ *where* $\epsilon = 2$ *because* $x(\{1,2\}) + 2 = v(\{1,2\})$.

There are an infinite number of possible ϵ-CS-cores, yet the smallest non-empty ϵ-CS-core is known as the *CS-least core* [112]:

Definition 25: **CS-Least core***:- For a set of agents N, a coalition structure and payoff vector pair $\langle CS, x\rangle$ is in CS-least core iff:*

$$\langle CS, x\rangle \text{ is in the } \epsilon\text{-CS-core}$$
$$\forall \epsilon' < \epsilon, \text{ the } \epsilon'\text{-CS-core is empty}$$

Like the classical definition of the least core, if the CS-core of a coalitional game is empty, the CS-least core value can be viewed as the minimum deviation penalty that is needed to stop a potential coalition from deviating from the given coalition structure. The coalition structures in the CS-least core *may or may not* contain optimal social welfare maximising coalition structures, as the following example from [112] shows:

Example 10: *Consider a characteristic function game $\mathcal{G} = \langle N, v\rangle$ where $N = \{1,2,3\}$ and the characteristic function v gives the following valuations: $v(C) = 10$ if $|C| = 1$, $v(C) = 30$ if $|C| = 2$, and $v(\{1,2,3\}) = 39$. A solution in the CS-least core minimises the maximum excess every potential deviating coalition C' has. The optimal social welfare coalition structures in this game consist of two coalitions, of size one and two respectively, e.g. $CS^* = \{\{1\},\{2,3\}\}$. The most stable payoff vector for this coalition structure is $x(10,15,15)$ giving the ϵ value of 5 (i.e. $x(\{1,2\}) + 5 = v(\{1,2\})$). Yet the coalition structure $CS = \{\{1,2,3\}\}$ can give the payoff vector $x(13,13,13)$ which gives the CS-least core value of 4 (e.g. $x(\{1,2\}) + 4 = v(\{1,2\})$). Therefore the optimal coalition structure is not contained within the CS-least core* in this example.

In this thesis, coalition structure stability solution concepts are used within in Chapter 6.

2.3 Non-Classical Types of Coalitional Games

So far we have discussed the traditional coalitional game type: *characteristic function games*. Yet characteristic function games make three big assumptions: (i) the value of the coalition can be split up in any manner between the agents of the coalition; (ii) the value of each coalition is the same for all the agents of the coalition; and (iii) a coalition's value is not affected by the other coalitions that form. However these assumptions do not necessarily hold for certain types of coalitional games, described below. Assumption (i) is discarded in *non-transferable utility games*. Assumption (ii) is discarded in *coalitional game models for uncertainty/valuation disagreements*. Assumption (iii) is discarded in *partition function games*. A variant of a non-transferable utility game, a coalitional game model for valuation disagreements and a partition function game is considered in Chapter 3 of this thesis. While another variant of a coalitional

game model for valuation disagreements, where assumptions (i) and (ii) hold, is considered in Chapter 6 of this thesis.

2.3.1 Non-Transferable Utility Games

When the utility of a coalition is not transferable between the agents, then this coalitional game is a *non-transferable utility game*. Examples include *qualitative coalition games*, *coalition Boolean games* and *hedonic games* [26, 51, 132, 133]. Two non-transferable utility coalitional games that most relate to the work of Chapter 3 are now briefly discussed.

Firstly, qualitative coalitional games (QCGs) [132] model the payoff of a coalition in terms of goal sets that the coalition can achieve by performing a certain choice. The incentive for an agent to join a coalition in a QCG is to achieve a goal that it would not be able to achieve individually. In QCGs, agents have a set of goals they want to achieve (yet have no preference order over), and the formation of one coalition does not affect the goal sets that another coalition can achieve. A coalition C is in the core of a QCG if: (i) C can successfully achieve the goal set G; (ii) this goal set satisfies all agents of C; (iii) C is minimal (otherwise some agents could defect to form other coalitions and achieve other goals at the same time as G is achieved).

Several approaches have extended the idea of qualitative coalition games by adding preferences to the goals to be achieved [28, 40, 52]. Rephrasing the stability definition of these approaches (to incorporate coalition structures): a coalition C, that can achieve the goal set G, is in the core of a QCG (with preferences), if: there does not exist a coalition S (where $S \cap C \neq \emptyset$) that can achieve a goal set G', which all the members of S favour compared to their current coalition goal sets.

In this thesis, Chapter 3 investigates a variant of QCGs where the goals that coalitions can achieve are linked to social-values (social-values are described in Section 3.5). In Chapter 3, unlike QCGs, the formation of a coalition can affect the goals that another coalition could achieve.

Secondly, coalition resource games (CRGs) model where the choices of coalitions in QCGs come from. The choices of a coalition in a CRG depend on the resources available to the agents of that coalition, and the amount each agent has of these resources. CRGs are a modification of QCGs. CRGs can always be mapped to a QCG; whereas a QCG cannot always be mapped to a CRG [133]. In Chapter 3, the choices of each coalition are the joint-actions that the coalition can undertake, given the current capabilities of its member agents.

2.3.2 Valuation Disagreements in Coalitional Games

In many systems, it is reasonable to believe that agents may only have partial, incomplete knowledge of their environment, including knowledge of the abilities of other agents, and on the potential effects of individual and coalitional actions. Therefore these environments require a coalitional game model that allows agents to hold different opinions on each coalition's value. Finding solutions to a coalitional game in an environment with partial agent knowledge involves trying to satisfy all agents as much as possible, with respect to their own beliefs on the value of each coalition.

Various studies have introduced solution concepts that find stable coalition structures and/or payoff vectors for coalitional games with uncertain valuations or valuation disagreements, e.g: [33, 34, 65, 77, 78]. The use of mechanism design for coalition formation was introduced in [77], where the first Bayesian model of coalitional games was detailed. In the Bayesian model of [77], each agent has probabilistic beliefs regarding the capabilities of the other agents and so is uncertain regarding each coalition's true value. This coalition formation process in this Bayesian model is completed via a centralised and trusted mediator that chooses the coalitions to form and the manner in which the payoff should be distributed. In this model it is essential that the agents only communicate with the mediator, otherwise information may be leaked to other agents, who may use this information to their own strategic advantage.

The most closely related coalitional game model, compared to the *valuation disagreement coalitional game* model introduced in this thesis, is found in [34]. Chalkiadakis *et al.* [34] proposed a transferable utilty coalitional game model that allows agents to have single point beliefs on the predicted coalition values. A single point belief represents an agent's best guess on each coalition's value but can lead to possible valuation disagreements between the agents. Another method of modeling beliefs involves assigning a probability to a distribution of different possible coalition values. Yet, as stated in [34], the single point belief assumption is helpful because: probabilistic beliefs may not be available; single point beliefs are easier to form; reasoning over single point beliefs is computationally less complex; and single point beliefs are a natural assumption in many real-world situations.

In the model of [34], *agent types* are used, where each agent i has a set of possible types, denoted Λ_i, where $\overrightarrow{\Lambda}$ denotes $(\Lambda_1, ..., \Lambda_n)$. An agent's type encapsulates all the information possessed by the agent that is not common knowledge, e.g. the agent's knowledge of its own type, its beliefs of its own payoff and its beliefs over other agents' payoffs [119]. Each agent i has point beliefs on every other agent's type, denoted $b_i = (b_i(1), ..., b_i(n))$ where $\overrightarrow{b}$ denotes $(b_1, ..., b_n)$. The utility function is denoted u. The utility value of a coalition C, using agent i's beliefs on the types of C, denoted $b_i(C)$, can be found by $u(b_i(C))$ (which is written in this thesis simply as $u_i(C)$). Given these preliminaries, the model presented in [34] can be formalised as:

Definition 26: *A* **coalitional game with beliefs** *is* $G^b = (N; \overrightarrow{\Lambda}; u; \overrightarrow{b})$ *where* N *is the set of agents,* $\overrightarrow{\Lambda}$ *is the collection of possible agent types,* u *is the utility function, and* $\overrightarrow{b}$ *is the collection of agent beliefs.*

To find an acceptable outcome to a coalitional game with beliefs, fixed percentage-based demand vectors are used [33, 37, 122]. In this model, to make a demand from a coalition C, an agent first finds the numeric value x_i it requires, then finds what percentage x_i is of its valuation of the coalition (i.e. $\frac{x_i}{u_i(C)}$). This percentage value becomes the demand.

Definition 27: *A* **demand vector**, *denoted* $d = \langle d_1, ..., d_n \rangle$, *for a coalition structure* $CS = \{C^1, ..., C^k\}$ *in a game* G^b, *satisfies* $d_i \geq 0$ *for all* $i \in N$ *and* $\sum_{i \in C} d_i = 1$ *for each* $C \in CS$. *Using* d, *the expected payoff for an agent* $i \in C \in CS$ *is given by* $d_i \times u_i(C)$. *The demand vector of a coalition* C *is denoted* $d(C)$. *The expected payoff for* n *agents using* d *is denoted* $p = (p_1, ..., p_n)$.

Example 11: *Consider a coalitional game with beliefs where* $N = \{1, 2, 3\}$ *and the utility function* u *gives the following valuations:* $u_3(\{3\}) = 2$, $u_1(\{1, 2\}) = 12$ *and* $u_2(\{1, 2\}) = 24$. *In this example the beliefs of an agent* i *is encapsulated within* i*'s utility function* u_i. *If the coalition structure* $\{\{1, 2\}, \{3\}\}$ *formed, then a valid demand vector would be* $d^1(0.25, 0.75, 1)$ *because for all* $i \in N$ *then* $d_i > 0$, *while* $d_1^1 + d_2^1 = 1$ *and* $d_3^1 = 1$. *This demand vector gives the expected payoff for the agents of* $p(3, 18, 2)$ *because* $u_1(\{1, 2\}) \times d_1^1 = 12 \times 0.25 = 3$, $u_2(\{1, 2\}) \times d_2^1 = 24 \times 0.75 = 18$ *and* $u_3(\{3\}) \times 1 = 2$. *An invalid demand vector would be* $d^2(0.1, 0.4, 1)$ *because* $d_1^2 + d_2^2 \neq 1$.

The solution concept proposed by Chalkiadakis *et al.* [34] to find stable coalition structures is the *CS-core for Beliefs* (CSB):

Definition 28: *The* **CS-core for Beliefs** *for a game* G^b *consists of all pairs* $\langle CS, d\rangle$ *such that for any coalition* $S \notin CS$ *and an associated demand vector* d', *there exists an* $i \in S$ *such that* $d'_i \times u_i(S) \leq p_i$ *(where* u_i *is the utility function of agent* i *and* p_i *is the expected payoff of agent* i *given* d*).*

Therefore a pair $\langle CS, d\rangle$ is in the CSB if no coalition S not in CS can provide a new demand vector d' that gives a higher expected payoff compared to d for all agents of S.

Example 12: *Consider a coalitional game with beliefs where* $N = \{1, 2, 3\}$ *and the utility function* u *gives the following valuations:* $u_1(\{1\}) = u_2(\{2\}) = u_3(\{3\}) = 2$, $u_1(\{1, 2\}) = 12$, $u_2(\{1, 2\}) = 24$, $u_1(\{1, 3\}) = 8$, $u_3(\{1, 3\}) = 20$ *and* $u_i(C) = 0$ *for any other coalition* C *and agent* $i \in \{1, 2, 3\}$. *In this example the beliefs of an agent* i *is encapsulated within* i*'s utility function* u_i. *Then* $\langle\{\{1, 2\}, \{3\}\}, d(0.75, 0.25, 1)\rangle$, *giving the expected payoff* $p(9, 6, 2)$, *would be in the CS-core for beliefs because: (i) agent 1 believes that it could earn no more than 9 in another coalition, as it believes 100% of the payoff of coalitions* $\{1\}$ *and* $\{1, 3\}$ *would only give 2 and 8 respectively; (ii) agent 2 believes that it could not earn more than 6 in another coalition, as its only other option is* $\{2\}$ *which gives a payoff of 2; and (iii) due to parts (i) and (ii), agent 3 cannot tempt any other agent into a coalition with it.*

On the other hand, the solution $\langle\{\{1, 2\}, \{3\}\}, d(0.25, 0.75, 1)\rangle$, *giving the expected payoff* $p(3, 18, 2)$, *would* not *be in the CS-core for beliefs because: the coalition* $\{1, 3\}$ *can improve the expected payoff of both agents through, for example,* $d_1 = 0.75$ *and* $d_3 = 0.25$, *giving the expected payoffs of* $p_1 = 6$ *and* $p_3 = 5$.

The CSB solution concept restricts the agents to one percentage demand per coalition (this is also the case in [33, 37, 122]). In Chapter 6, this thesis investigates the advantages of allowing different percentage demands for different potential values of the same coalition.

2.4 The Three Stages of Coalition Formation

In multi-agent systems, coalition formation can be broken down into the following three stages [88, 104]:

1. **Coalition value calculations** - This involves calculating the value of each possible coalition (usually each subset of n agents, which is an exponential number of possible coalitions).

2. **Coalition structure generation** - This involves partitioning the agents into a set of coalitions. This coalition structure should preferably be either: (i) the optimal social welfare maximising coalition structure; or (ii) a stable coalition structure (where no agent wants to defect to another coalition). Sometimes, a coalition structure may be found that satisfies both (i) and (ii), yet this is not always the case [27, 112].

3. **Determining the payoff distribution** - This involves deciding how to divide the coalitions' overall payoff between the participating agents of the coalitional game (if the given coalitional game has transferable utility). In self-interested agent domains, it is preferable that the payoff is distributed between the agents in a manner that is stable, i.e. no agent should be able to rationally object to the distribution, where a rational objection is defined by a stability solution concept.

All of the three stages above have high computation costs due to the exponential number of possible coalitions and the theoretically infinite manner with which to divide a transferable utility payoff between the agents of the game. Yet multi-agent systems have the capability to spread the costs of computation of these three stages between the agents of the system.

In this section, various approaches for multi-agent systems to complete each of these three stages are surveyed. For simplicity, each stage will be discussed from the transferable utility characteristic function game model perspective.

2.4.1 Coalition Value Calculations

The first stage of coalition formation requires the agents to calculate the utility value of each coalition. If all possible coalitions from N agents can form, then the calculation of all of the coalitions' values becomes an exponentially intense task due to the $2^n - 1$ possible coalitions [43, 90, 104]. Additionally, the complexity of calculating an individual coalition's value can vary, and potentially be exponential itself [106]. If each agent i calculates the value of all coalitions it is a member of, such as in the models of [23, 118], there will be a significant overlap of calculations and thus significant unnecessary computation costs for the system.

Instead of each agent calculating each coalition value in which it is a member, Shehory and Kraus introduced a method (named SK), in the series of papers [115–117], where agents negotiated over which coalition values to calculate. Yet as noted in [88, 90], SK suffers from the following disadvantages:

- SK requires many messages to be sent between the agents, some of which are exponentially large.
- SK does *not* guarantee that every coalition value is calculated *once and only once*. Yet SK *does* have the weaker guarantee that every coalition value is guaranteed to be calculated *at least once*.
- SK provides *no* guarantees on the quality of its distribution; i.e. SK does *not* guarantee that the agents' coalition value calculation shares are approximately equal.

- SK has a memory requirement that grows exponentially with the number of agents.

Therefore SK utilitises the resources of the system *inefficiently*. This inefficiency motivated Rahwan and Jennings to introduce the distributed coalition value calculation algorithm (that they name DCVC) [89, 90]. The DCVC algorithm groups coalitions into lists, and then uses a decentralised method to divide the lists into *shares*, one for each agent. The DCVC algorithm has the following properties:

- DCVC requires no communication between any agents.
- DCVC gives the agents' coalition value calculation shares that are exhaustive and disjoint; i.e. each coalition's value is calculated *once and only once*.
- DCVC gives the agents' coalition value calculation shares that are approximately equal. The shares are exactly equal if the number of coalitions is exactly divisible by the number of agents, else the difference in share sizes is a maximum of one coalition.
- DCVC has minimal memory requirements.
- DCVC is fully decentralised, requiring no centralised arbitrator for any part.

The DCVC algorithm achieves these properties by firstly representing all the feasible coalitions in structured lists $L_{s \in N}$, where each L_s contains all coalitions of size s ordered in reverse lexicographical order, i.e. the first coalition in the list L_s is $\{n-s+1, ..., n\}$ and the last coalition in the list L_s is $\{1, ..., s\}$. This means that the agents know how L_s is ordered although they do not actually maintain L_s. Each agent is then assigned a share of $\lfloor \frac{|L_s|}{n} \rfloor$ coalition values to calculate from each list, where the first agent has the first share, the second agent has the second share, etc. The index of the last coalition of agent i's coalition value calculation share is found by: $index_{s,i} = i \times \lfloor \frac{|L_s|}{n} \rfloor$. Therefore an agent i calculates the value of the coalition at $index_{s,i}$ and all coalitions previous to it in L_s, until $\lfloor \frac{|L_s|}{n} \rfloor$ coalition values are calculated.

For lists that contain a number of coalitions that are not an integer multiple of the number of agents, each agent must maintain a value α indicating the agent that should calculate the next *additional coalition* above $n \times \lfloor \frac{|L_s|}{n} \rfloor$. The value α is incremented for each further additional coalition (if α goes above n, it will be reset to 1). The same α is used for all lists so that the maximum difference between the number of coalitions computed by the agents will never be more than one. The distribution of the lists begins at list L_1, so that all agents recognise the correct agents to calculate the α assigned coalitions in each list. See Table 2.1 for a DCVC example.

Finally, DCVC uses a function, that will be referred to here as *GenerateCoalition*, that takes an index value and returns a coalition. Using *GenerateCoalition*, an agent can find the last coalition C in its main share[2]. Coalition C can then be used to find the other coalitions in its main share, because each agent knows that it now needs to generate the next $\lfloor \frac{|L_s|}{n} \rfloor - 1$ coalitions in lexicographical order (from C) to complete its main share. This *GenerateCoalition* function

[2]Where 'main share' refers to the coalitions not assigned by the α pointer.

L_1	L_2	L_3	L_4	L_5	L_6
1 [6	**1** [5,6	**1** [4,5,6	**1** [3,4,5,6	**1** [2,3,4,5,6	**3** [1,2,3,4,5,6
2 [5	**1** [4,6	**1** [3,5,6	**1** [2,4,5,6	**2** [1,3,4,5,6	
3 [4	**2** [4,5	**1** [3,4,6	**2** [2,3,5,6	**3** [1,2,4,5,6	
4 [3	**2** [3,6	**2** [3,4,5	**2** [2,3,4,6	**4** [1,2,3,5,6	
5 [2	**3** [3,5	**2** [2,5,6	**3** [2,3,4,5	**5** [1,2,3,4,6	
6 [1	**3** [3,4	**2** [2,4,6	**3** [1,4,5,6	**6** [1,2,3,4,5	
	4 [2,6	**3** [2,4,5	**4** [1,3,5,6		
	4 [2,5	**3** [2,3,6	**4** [1,3,4,6		
	5 [2,4	**3** [2,3,5	**5** [1,3,4,5		
	5 [2,3	**4** [2,3,4	**5** [1,2,5,6		
	6 [1,6	**4** [1,5,6	**6** [1,2,4,6		
	6 [1,5	**4** [1,4,6	**6** [1,2,4,5		
	1 [1,4	**5** [1,4,5	**6** [1,2,3,6		
	2 [1,3	**5** [1,3,6	**1** [1,2,3,5		
	3 [1,2	**5** [1,3,5	**2** [1,2,3,4		
		6 [1,3,4			
		6 [1,2,6			
		6 [1,2,5			
		4 [1,2,4			
		5 [1,2,3			
$\alpha = 1$	$\alpha = 4$	$\alpha = 6$	$\alpha = 3$	$\alpha = 3$	$\alpha = 4$

TABLE 2.1: A DCVC list of all possible coalitions for 6 agents in the form: ***agent to calculate value*** *[coalition*. The value α holds the value of the agent that should calculate the next additional coalition value above $n \times \lfloor \frac{|L_s|}{n} \rfloor$. The α pointer is set $\alpha = 1$ before any list is divided. The value of the α pointer at the end of each list, is recorded at the end of that list's column. The α assigned coalitions are separated from the other coalitions by a horizontal line.

can also be used to find any additional coalitions assigned to an agent via the α pointer. The basic version of the DCVC algorithm is detailed in Algorithm 1.

Although the basic DCVC algorithm (henceforth referred to as DCVC1) distributes the coalitions evenly among the agents of the system, DCVC1 does not evenly distribute the individual operations (of comparisons and additions) to generate all the coalitions in each agent's share. This is because the number of operations to find the next coalition in each agent's share may fluctuate. The range of operations performed by the agents using DCVC1 grows as more agents are used [90]. A modified version of DCVC1 was introduced in [90] to minimise (but not totally eradicate) this issue, henceforth named DCVC2. The difference of DCVC2 over DCVC1 is that DCVC2 assigned 60% of an agent's main coalition value calculation share to the beginning of the list and 40% of the main share to the end of the list (prior to the assignment of the additional set of coalitions).

In a follow-up study to [90], Michalak *et al.* [73], looked at how another DCVC variant (henceforth named DCVC3) could be used as input to a decentralised algorithm to efficiently find the optimal coalition structure. As DCVC3 assigns the agents' shares in difference percentages and at different points in the list to DCVC2, it does not minimise the difference between the number of individual operations for each agent.

Algorithm 1: The basic DCVC algorithm to be run by every agent.

function DCVC (N)
Input: $\langle N \rangle$; where N is the set of agent IDs.
begin;
Set $\alpha = 1$;
for For every $s \in S$ **do**
  if $|L_s| \geq n$ **then**
    Find the size of your share: $\lfloor \frac{|L_s|}{n} \rfloor$;
    Find the index of the last coalition in your share: $index_{s,i} = i \times \lfloor \frac{|L_s|}{n} \rfloor$;
    Calculate the coalition corresponding to $index_{s,i}$;
    Calculate every coalition previous to $index_{s,i}$ until your share is complete;
    Find the number of extra coalitions to calculate: $c = |L_s| - (n \times \lfloor \frac{|L_s|}{n} \rfloor)$;
  else
    Find the number of extra coalitions to calculate: $c = |L_s|$;
  end if
  if $c > 0$ **then**
    Find the sequence of agents A' to calculate an additional value, by:
    if $\alpha + c - 1 \leq n$ **then**
      $A' = (\alpha, \alpha + 1, ..., \alpha + c - 1)$;
    else
      $A' = (\alpha, \alpha + 1, ..., n, 1, ..., \alpha + c - 1 - n)$;
    end if
    Every agent A'_i calculates the i'th additional coalition's value.
    if $\alpha + c \leq n$ **then**
      $\alpha = \alpha + c$;
    else
      $\alpha = \alpha + c - n$;
    end if
  end if
end for
end;

Another variant of DCVC was the D-SlyCE algorithm [127]. D-SlyCE is an algorithm to distribute coalition value calculations when the agents are represented as nodes on a graph, where an edge between them represents some synergistic link. When the graph is fully connected (i.e. the graph resembles a characteristic function game), then D-SlyCE mimics the operations of DCVC [127].

The only paper that subsequently gave a new algorithm for distributing coalition value calculations to agents in an exhaustive and disjoint manner (that the author is aware of) is Vinyals *et al.* [126] (where this new algorithm is henceforth named VBFR). The VBFR algorithm distributes the calculation of coalition values when agents are connected in a network and a coalition has to include member agents that are connected together in a graph. In Figure 2.1, an example is given where the agents are fully connected to each other so that comparisons can be made between VBFR and DCVC. The VBFR algorithm casts the problem of generating all possible coalitions on a graph as the problem of enumerating all possible subgraphs. VBFR uses an ordering among the agents to partition the set of possible coalitions into disjoint (leading) sets M_i, where every

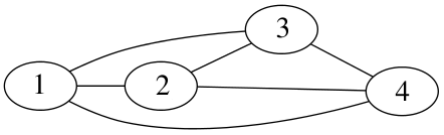

FIGURE 2.1: Example of a 4 agent, fully connected graph, used to form coalitions.

M_1	M_2	M_3	M_4
1,2,3,4	2,3,4	3,4	4
1,3,4	2,4	3	
1,2,4	2,3		
1,2,3	2		
1,4			
1,3			
1,2			
1			

FIGURE 2.2: A VBFR list of all possible coalitions that can be formed from the fully connected graph of Figure 2.1

coalition in M_i has i as the smallest agent ID of that coalition, i.e. $\forall C \in M_i, \neg\exists j \in C$ where $j < i$. Then agent i is assigned the coalition value calculation share of M_i. See Figure 2.2 for an example of how the coalition value calculations would be shared around the agents of the system using VBFR.

As can be seen in Figure 2.2, VBFR does not fairly distribute the coalition value calculations; the higher an agent's ID is, the less calculations an agent has to perform. This is a weakness of the VBFR algorithm [126] compared to the DCVC algorithm [90]. Yet the VBFR algorithm (as with the SK algorithm before it) has an interesting property: the constraint that all coalitions distributed to an agent i's calculation share should include agent i. The VBFR algorithm was created for the smart grid domain, where it is involved in a larger algorithm to see if forming coalitions with other 'virtual electricity consumers' will provide the coalition with a discount on electricity through collective buying. Thus if an agent i was given a coalition C in its value calculation share, where $i \notin C$ (as is possible in the DCVC algorithm), it would have no incentive to calculate that value because C would not personally give i a discount on electricity price.

Compact Representation Schemes

So far, it has been assumed that all coalition values for a given size have to be calculated. This results in an exponential number of possible coalition value calculations to perform where one value calculation by itself could be exponential to compute [106].

However characteristic functions that appear in practical applications can often be represented more concisely by a set of rules [81]. Possible compact representation schemes are marginal contribution nets [64], synergy coalition groups [41, 81] and weighed voting games [55]. Here, the synergy coalition group (SCG) representation is discussed because: (i) it can fully represent a characteristic function game; and (ii) it is used as a basis to develop a decentralised algorithm to find a stable payoff vector in Chapter 5. The following is the SCG definition [41]:

Definition 29: *A* ***synergy coalition group*** W *consists of a set of pairs of the form:* $(C, v(C))$. *The singleton coalitions are assumed to always be represented within* W. *For any coalition* $C \in (C, v(C)) \in W$, *the value of* C *is* $v(C)$. *For any coalition* $C' \notin W$, *the value of* C' *is:*

$$v(C') = max\{\sum_{C_j \in P_{C'}} v(C_j)\}$$

Where $P_{C'} = \{C_1, C_2, ..., C_k\}$ *is a partition of* C' *such that:*

$$\bigcup_{i=1}^{k} C_i = C' \quad \text{(a)}$$

$$C_i \cap C_j = \emptyset \text{ for any } i, j \in \{1, ..., k\} \text{ where } i \neq j \quad \text{(b)}$$

$$(C_j, v(C_j)) \in W, \text{ for all } C_j \in \{C_1, ..., C_k\} \quad \text{(c)}$$

Therefore if a coalition C' is not explicitly represented within W, then its value comes from the highest valued set of coalitions (referred to here as a *partition*), as long as this partition has the following properties: (a) all the agents of C' are in a coalition of the partition and there are no additional agents in the partition; (b) each agent of C' occurs in only one coalition of the partition; and (c) each coalition in the partition is represented within W.

Example 2.1. *Consider the characteristic function game* $G = \langle N, v \rangle$ *where* $N = \{1, 2, 3, 4\}$ *and the SCG representation* W *gives the following valuations:* $W = \{(\{1\}, 2), (\{2\}, 1), (\{3\}, 1), (\{4\}, 2), (\{1, 2\}, 5)\}$ *(i.e. the utility value of agent 1 by itself is 2, while the utility value of the coalition, of agents 1 and 2, is 5) . Example values of coalitions not explicitly represented are:* $v(\{3, 4\}) = v(\{3\}) + v(\{4\}) = 1 + 2 = 3$*;* $v(\{\{1, 2, 3\}) = v(\{1, 2\}) + v(\{3\}) = 5 + 1 = 6$*; and* $v(\{1, 2, 3, 4\}) = v(\{1, 2\}) + v(\{3\}) + v(\{4\}) = 5 + 1 + 2 = 8$.

2.4.2 Coalition Structure Generation

The *coalition structure generation (CSG) problem* is the problem of finding an optimal social welfare maximising coalition structure [94, 104]. The coalition structure generation problem is mostly in the domain of cooperative agents because optimising social welfare may conflict with an individual agent's best interests. The required coalition structure for self-interested agents is one that is stable, where stability is influenced by the payoff distribution, as described in the next section.

In this section, the focus is on algorithms that find the optimal social welfare maximising coalition structure, without regard to strategic self-interested decision making.

These algorithms, as described in [73], can be separated into 3 main classes:

- **Dynamic programming** - This is the collection of algorithms where the main problem is solved by dividing it up into subproblems and then solving the smallest first. The answers to the smallest problems help with the working out of the larger problems [44]. Some dynamic programming CSG algorithms are given in: [91, 138].

- **Heuristics** - Heuristic algorithms are algorithms that return good solutions quickly, but have no guarantees on the quality of the solution and the optimal solution may not be returned. Some heuristic CSG algorithms are given in: [111, 117].

- **Anytime optimal algorithms** - Anytime algorithms can return a solution at anytime, where the quality of this solution gets better over time (in some well behaved manner). Thus, an anytime algorithm can be terminated prematurely whilst still returning a solution that is guaranteed to be within a bound from the optimal solution [94]. Some anytime optimal CSG algorithms are given in: [43, 73, 94, 104].

The biggest advantage of dynamic programming algorithms is that they run in $O(3^n)$ time compared to the $O(n^n)$ time that an anytime algorithm could run in (where n is the number of agents) [91, 104]. But in practice anytime optimal algorithms can run significantly faster than dynamic programming algorithms for a variety of possible coalition value distributions. This result is true for when any possible coalition could form (i.e. the full exponential characteristic function representation has been used).

Additionally it should be mentioned that great speed ups can occur in the coalition structure generation process if not all of the exponential number of coalitions are considered able to form (i.e. some compact representation method has been used). For example, if the agents are represented on a graph, and a coalition can form only if the agents are connected together on a graph, than the DyCE [127] or CFSS algorithm [20] could be used for significant speed ups.

Furthermore modeling the CSG process as a distributed constraint optimization problem (DCOP) [74] allows for an approximation algorithm that can find a coalition structure (with certain quality guarantees compared to the optimal coalition structure), to be found in polynomial time [124]. But unfortunately if the optimal coalition structure is required to be guaranteed then modeling the CSG problem as a DCOP is not very helpful as an NP-hard problem needs to be solved just to obtain the value of a single coalition, so $O(2^n)$ DCOP problem instances are required to find the value of every coalition (where n is the number of agents) [124].

The CSG algorithms generally do not consider the payoff distribution of the agents. In Chapter 5, two decentralised algorithms named DDP and DDP*, guarantee that agents locate the optimal coalition structure and one of the "least objectionable" payoff vectors. The DDP algorithms are based on the centralised Dynamic Programming (DP) algorithm of [138].

The DP algorithm takes a table of coalition values as input (where $v[C]$ indicates the value of coalition C). DP uses the input table to compute two other tables, f_1 and f_2, that have a single entry per coalition C, denoted $f_1[C]$ and $f_2[C]$ respectively.

As summarised in [91], the DP algorithm (given in Algorithm 2) computes $f_1[C]$ and $f_2[C]$ as follows. Firstly, DP evaluates all possible ways of splitting C into two partitions and compares the highest evaluation with $v[C]$ to see if it is beneficial to split C into a two set partition (lines 10 and 11). If it is beneficial then $f_1[C]$ stores this best partition and $f_2[C]$ stores the value of this best partition (line 14). If a split is not beneficial then $f_1[C]$ becomes equal to the coalition C and $f_2[C]$ becomes equal to the value of the coalition C (line 12). An evaluation of a split of a coalition $C = \{C', C''\}$ is evaluated by $f_2[C'] + f_2[C'']$. Therefore the DP algorithm must evaluate the coalitions in size order (line 8).

Algorithm 2: The DP algorithm to find an optimal coalition structure.

1: **function** `DP(`v`,` N`)`
2: **Input:** $\langle v, N \rangle$; where v is the input table of values, where $v[C]$ is coalition C's value, and N is the set of agent IDs.
3: **Output:** $\langle CS^* \rangle$; where CS^* is the optimal coalition structure.
4: **begin**;
5: **for** $i \in N$ **do**
6: *set* $f_1[\{i\}] := \{i\}$ and $f_2[\{i\}] := v[\{i\}]$;
7: **end for**
8: **for** $s := 2$ to n **do**
9: **for** each $C \subseteq N$ where $|C| = s$ **do**
10: $f_2[C] := max\{f_2[C'] + f_2[C \backslash C'] : C' \subset C \text{ and } 1 \leq |C'| \leq 1/2|C|\}$;
11: **if** $f_2[C] < v[C]$ **then**
12: *set* $f_1[C] := C$ and $f_2[C] := v[C]$;
13: **else**
14: *set* $f_1[C] := C^*$ where C^* maximises $f_2[C]$;
15: **end if**
16: **end for**
17: **end for**
18: *set* $CS^* := \{N\}$
19: **for** each $C \in CS^*$ **do**
20: **if** $f_1[C] \neq C$ **then**
21: *set* $CS^* := (CS^* \backslash \{C\}) \cup \{f_1[C]\}$;
22: *restart* this for loop;
23: **end if**
24: **end for**
25: **return** $\langle CS^* \rangle$;
26: **end**;

After f_1 and f_2 are computed, the optimal coalition structure CS^* is found recursively. This is performed by first setting the optimal coalition structure equal to the grand coalition (line 18) and then performing a loop (lines 19 to 24) where all the coalitions in the optimal coalition structure are checked to see if they can be broken down into smaller more valuable coalitions by checking the f_1 table (line 20). The loop stops when no more splittings are beneficial. An example of the data manipulated by the algorithm can be seen in Table 2.2, where the optimal coalition structure $\{\{3\}, \{1, 2, 4\}\}$ is found with a value of 3.5.

2.4.3 Payoff Distribution

Once the coalitions that will form have been decided (i.e. the coalition structure generation problem has been solved), the agents now need to decide how to divide the reward (i.e. payoff) of each coalition between themselves. The payoff of a coalition can be distributed to its members in multiple ways if the coalitional game has transferable utility. In Section 2.2, stability solution concepts for transferable utility characteristic function games were defined. These stability solution concepts describe what the stable outcomes of a transferable utility characteristic function

s	C	The evalations perform before setting f_1 and f_2		f_1	f_2
1	$\{1\}$	$v[\{1\}] = 0.5$		$\{1\}$	0.5
	$\{2\}$	$v[\{2\}] = 0.4$		$\{2\}$	0.4
	$\{3\}$	$v[\{3\}] = 0.7$		$\{3\}$	0.7
	$\{4\}$	$v[\{4\}] = 0.1$		$\{4\}$	0.1
2	$\{1,2\}$	$v[\{1,2\}] = 1.8$	$f_2[\{1\}] + f_2[\{2\}] = 0.9$	$\{1,2\}$	1.8
	$\{1,3\}$	$v[\{1,3\}] = 1.1$	$f_2[\{1\}] + f_2[\{3\}] = 1.2$	$\{1\}\{3\}$	1.2
	$\{1,4\}$	$v[\{1,4\}] = 1.9$	$f_2[\{1\}] + f_2[\{4\}] = 0.6$	$\{1,4\}$	1.9
	$\{2,3\}$	$v[\{2,3\}] = 1.0$	$f_2[\{2\}] + f_2[\{3\}] = 1.1$	$\{2\}$ $\{3\}$	1.1
	$\{2,4\}$	$v[\{2,4\}] = 1.2$	$f_2[\{2\}] + f_2[\{4\}] = 0.5$	$\{2,4\}$	1.2
	$\{3,4\}$	$v[\{3,4\}] = 1.6$	$f_2[\{3\}] + f_2[\{4\}] = 0.8$	$\{3,4\}$	1.6
3	$\{1,2,3\}$	$v[\{1,2,3\}] = 2.1$	$f_2[\{1\}] + f_2[\{2,3\}] = 1.6$	$\{3\}\{1,2\}$	2.5
		$f_2[\{2\}] + f_2[\{1,3\}] = 1.6$	$f_2[\{3\}] + f_2[\{1,2\}] = 2.5$		
	$\{1,2,4\}$	$v[\{1,2,4\}] = 2.8$	$f_2[\{1\}] + f_2[\{2,4\}] = 1.7$	$\{1,2,4\}$	2.8
		$f_2[\{2\}] + f_2[\{1,4\}] = 2.3$	$f_2[\{4\}] + f_2[\{1,2\}] = 1.9$		
	$\{1,3,4\}$	$v[\{1,3,4\}] = 2.5$	$f_2[\{1\}] + f_2[\{3,4\}] = 2.1$	$\{3\}\{1,4\}$	2.6
		$f_2[\{3\}] + f_2[\{1,4\}] = 2.6$	$f_2[\{4\}] + f_2[\{1,3\}] = 1.3$		
	$\{2,3,4\}$	$v[\{2,3,4\}] = 2.4$	$f_2[\{2\}] + f_2[\{3,4\}] = 2$	$\{2,3,4\}$	2.4
		$f_2[\{3\}] + f_2[\{2,4\}] = 1.9$	$f_2[\{4\}] + f_2[\{2,3\}] = 1.2$		
4	N	$v[\{1,2,3,4\}] = 3.4$	$f_2[\{1\}] + f_2[\{2,3,4\}] = 2.9$	$\{3\}\{1,2,4\}$	3.5
		$f_2[\{2\}] + f_2[\{1,3,4\}] = 3$	$f_2[\{3\}] + f_2[\{1,2,4\}] = 3.5$		
		$f_2[\{4\}] + f_2[\{1,2,3\}] = 2.6$	$f_2[\{1,2\}] + f_2[\{3,4\}] = 3.4$		
		$f_2[\{1,3\}] + f_2[\{2,4\}] = 2.4$	$f_2[\{1,4\}] + f_2[\{2,3\}] = 3$		

TABLE 2.2: This table details a characteristic function game for the set of agents $N = \{1,2,3,4\}$. The optimal coalition structure is found through the DP algorithm.

game are, but they do *not* describe how agents can work together to find a stable outcome. *How* a stable solution is found in a collaborative manner is discussed in this section.

Payoff distribution is usually only of interest if the agents have multiple designers and thus competing interests, although not all systems with a single agent designer should discard the consideration of the payoff distribution of its agents [38]. In the study of Chapman *et al.* [38], self-interested agents are used to provide the degree of robustness and flexibility needed for a large scale distributed application. These agents are designed to act on their local knowledge so that improvements to their own payoffs increase the global solution quality.

Algorithms for agents to collaboratively work to find a payoff distribution that no single agent can object to, can broadly be placed into two different categories:

1. **Payoff transfer schemes**: This is the collection of algorithms where an agent i takes as input the current coalitional game outcome $\langle CS', x' \rangle$ and performs some operation θ on it, to give a new coalitional game outcome $\langle CS, x \rangle$ where $CS' \neq CS$ and/or $x' \neq x$. In this category of algorithms, θ is fixed and agent i does *not* consider how the other agents will react to the new outcome, meaning these algorithms are myopic.

2. **Coalitional bargaining**: This is the collection of algorithms where an agent i takes as input the current coalitional game outcome $\langle CS', x' \rangle$ and performs some operation θ on it to give a new coalitional game outcome $\langle CS, x \rangle$ where $CS' \neq CS$ and/or $x' \neq x$. In this

category of algorithms, θ is *not* fixed and is chosen by agent i from a set Θ of possible operations to the coalitional game outcome. Therefore agent i can consider how all the other agents will react to every possible $\theta' \in \Theta$, and so pick the optimal operation $\theta \in \Theta$ for itself (providing the other agents act in the predicted manner).

Payoff Transfer Schemes

Payoff transfer schemes take a coalitional game outcome, iteratively apply an operation to it, and gradually the resulting outcome will converge to a member of a cooperative game theory solution concept. Transfer schemes have been proposed for many cooperative game solution concepts such as: the kernel [121]; the nucleolus [59]; and the core [30, 67, 137]. They reduce the cognitive demands placed on the agents, as can be seen with the following motivating example. When the core is non-empty, to guarantee locating a payoff vector of the core, the payoff vector used to find the optimal solution to the following linear program (LP) can be returned:

$$\textbf{min } \epsilon$$

$$x_i \geq 0 \text{ for each } i \in N \tag{a}$$

$$\sum_{i \in N} x_i = v(N) \tag{b}$$

$$\sum_{i \in S} x_i \geq v(S) - \epsilon \text{ for each } S \subset N \tag{c}$$

This linear program is centralised and has 2^n+n+1 constraints because there are 2^n possible coalitions $S \subset N$ in (c), $|x| = n$ in (a) and one $v(N)$ in (b). Therefore finding a solution to this problem is usually a highly complex computational task for characteristic function games (unless a compact coalitional game representation is used to reduce the complexity).

However, payoff transfer schemes for the core find core solutions without the agents actually solving the defining system of linear equalities [137]. The payoff transfer schemes of [30, 137] (that converge a payoff vector x onto a pre-selected strong ϵ-core), are detailed in Algorithm 3.

The *strong ϵ^s-core* transfer scheme of [30, 137] repeats lines 7 to 22 until it converges on a solution in the required strong ϵ^s-core. To do this, firstly a starting payoff vector is chosen (line 5), then all the coalition's strong excess values are calculated (line 9). If there exists coalitions with a higher strong excess p than the required ϵ^s, then a coalition with this maximum highest strong excess is chosen (line 14) and the payoff of every member of that coalition is incremented by an equal division of $p - \epsilon^s$ (lines 15 to 17). Finally the payoff of every non-member of the chosen coalition is decremented by an equal division of $p - \epsilon^s$ (lines 18 to 20). It is this equal increment/decrement that allows the payoff vector to converge on the pre-selected strong ϵ^s-core value with a relativity simple calculation (compared to the linear programming method). Unfortunately it is also this equal gain/loss which means that it is only possible to guarantee that the payoff vector converges to the pre-selected strong ϵ^s-core value as the number of transfers approaches infinity (yet there exist many finite cases).

Example 13: *Consider a characteristic function game $\langle N, v \rangle$ where $N = \{1, 2, 3, 4\}$ and the characteristic function v gives the following valuations: $v(\{2, 3\}) = 12$, $v(\{3, 4\}) = 14$,*

Algorithm 3: The payoff vector transfer scheme algorithm presented in [30, 137].

function `CoreTS(`v`,` N`,` ϵ^s`)`
Input: $\langle v, N, \epsilon^s \rangle$; where v is the characteristic function, N is the set of agent IDs and ϵ^s is the required strong ϵ-core value.
Output: $\langle x \rangle$; where x is an ϵ^s stable payoff vector.
begin;
Choose a starting payoff vector x that distributes fully $v(N)$;
boolean $complete := false$;
repeat
  Set $x' := x$;
  Compute all the quantities $\epsilon^s(S, x)$ for all $S \subset N$ where $S \neq \emptyset$;
  Set p equal to the largest $\epsilon^s(S, x)$ found
  if $p == \epsilon^s$ **then**
    $complete := true$
  else
    Select a coalition $C \subset N$ such that $\epsilon^s(C, x) = p$
    for $i \in C$ **do**
      $x_i = x'_i + \frac{p - \epsilon^s}{|C|}$
    end for
    for $j \notin C$ **do**
      $x_j = x'_j - \frac{p - \epsilon^s}{|N| - |C|}$
    end for
  end if
until $complete == true$
return $\langle x \rangle$;
end;

$v(\{1,2,3,4\}) = 20$ *and for any other coalition* C *then* $v(C) = 0$. *This game is known to have a non-empty core (i.e. the strong* ϵ^s*-core is stable when* $\epsilon^s = 0$*) and so* $\epsilon^s = 0$ *is given as input to the* `CoreTS` *algorithm. Given a starting payoff vector of* $x^1(5,5,5,5)$*, that fully distributes the value of the grand coalition, the coalition* $C^1 = \{3,4\}$ *has the maximum excess of* $\epsilon^s(\{3,4\}, x^1(5,5,5,5)) = 14 - (5+5) = 4$. *Therefore* $p = 4$ *and* $p - \epsilon^s = 4 - 0 = 4$ *needs to be incremented and decremented from the agents of* C^1 *and the agents of* $N \backslash C^1$ *respectively. Adding an equal split of the value* $p - \epsilon^s$ *to the payoff of the agents of* C^1 *while taking an equal split of the value* $p - \epsilon^s$ *from the agents of* $N \backslash C^1$*, gives the payoff vector* $x^2(3,3,7,7)$. *repeating the process again, the coalition* $C^2 = \{2,3\}$ *has the maximum excess of* $\epsilon^s(\{2,3\}, x^2(3,3,7,7)) = 12 - (3+7) = 2$*, meaning* $p = 2$ *and* $p - \epsilon^s = 2 - 0 = 2$. *Incrementing and decrementing* p *from the correct agents, gives the payoff vector of* $x^3(2,4,8,6)$. *Repeating the process again, and each coalition* C *now has* $\epsilon^s(C, x^3) \leq 0$. *As the required strong* ϵ^s*-core value was* $\epsilon^s = 0$ *then* $complete == true$ *and the* `CoreTS` *algorithm terminates.*

Unfortunately, these core locating transfer schemes do not fully deal with the issues that arise in multi-agent system coalition formation because: the algorithms are not decentralised (in terms of distributed coalition value calculation input *and* distributed payoff vector finding operations); the grand coalition is assumed to form, therefore the optimal coalition structure of

the agents is assumed; the algorithms presented are only guaranteed to converge on the core/ϵ^s-core should payoff vector corrections be made "infinitely often", so there is no theoretical upper bound on the running time; the ϵ^s value must be chosen before runtime (and so must be known in advance of any coalition formation calculations) or the transfer scheme could develop cycles and not converge [31]; and the least core is only guaranteed to be found if the least core value is known before calculations begin [30, 137].

A payoff transfer scheme for the kernel, initially introduced in [121], has been applied to multi-agent systems in [118]. However, the communication cost issues were not considered and full coalition value knowledge for every agent was assumed.

Another payoff transfer scheme in the multi-agent systems domain is the one presented in [59] to find the nucleolus. This transfer scheme has the advantage of leading to the single point solution in the least core, known as the most stable point in the strong least core. Yet the disadvantages of this transfer scheme are that an optimal coalition structure is already considered to have been formed, full knowledge for each agent regarding the coalition values is assumed and the communication costs are not considered.

Coalitional Bargaining

Coalitional Bargaining is a specific type of general agent bargaining game and comprises the following, as discussed in [136]:

- A *negotiation set*, representing all the possible proposals that the agents can make.
- A *protocol*, defining the legal proposals the agents can make, given the history of proposals.
- A *collection of strategies*, one for each agent, that determines what proposals the agents will make.
- An rule that determines when a deal has been stuck, where the deal is known as an *agreement deal*.

Coalitional bargaining is a research area that describes how stable coalition outcomes can occur via a process of non-cooperative bargaining and outlines equilibrium concepts that characterize the agents' behaviours in these games. The general approach in a coalitional bargaining game extends the two agent *alternating offers* bargaining model of [103]. The agents first take turns to offer proposals of a coalition and a payoff vector for that coalition, then the other agents of the coalition have a chance to accept or reject the proposal. If all agents of a coalitional bargaining game accept some proposal, then that coalition abandons the game and the other agents carry on making proposals to the remaining agents in the system. When a coalitional bargaining game is *discounted* (coalition values decrease over time by a discount factor) then agents need to place more emphasis on strategic considerations (otherwise they will lose some payoff in the future).

The coalitional bargaining papers of [39, 56, 82] detail how non-cooperative game theory solution concepts match the cooperative solution concept of the core/CS-core, giving the

core/CS-core a non-cooperative justification. But these studies do not detail what cooperative game solution concept is found when the core/CS-core is empty. This is a disadvantage for multi-agent systems, as ideally agents will want to know before runtime what cooperative game solution concept will be found.

Additionally, the majority of coalitional bargaining game studies assume each agent has complete knowledge on the coalition values [32], which can be time consuming to calculate or even impossible for every agent to determine. The coalitional bargaining papers that do not assume full knowledge for every agent introduce new solution concepts to deal with these problems, such as: the *Bayesian core* [37]; the *coarse/fine core* [131]; the *virtual utility core* [78]; and the *credible core* [54]. Yet these new solution concepts only overlap with the complete knowledge core/CS-core solution concepts under specific conditions relating to: what information is correctly known; and how different the agents' beliefs are.

2.5 Agent Communication

A key feature of multi-agent systems is the communication between each agent. Communication involves the transmission of information in an attempt to change the internal state of another agent and possibly get that other agent to perform some action. In this thesis, communication between the agents is centered on whether to form a coalition or not.

As detailed in [119], agents communicate through the transmission of information either through their 'physical' actions (i.e. *talking by doing*) or via 'communicative' actions (i.e. *doing by talking*), both of which can influence another agent. The *doing by talking* method involves an agent revealing to others, information that it hopes will allow it to achieve its goals. Whereas the *talking by doing* method refers to those cases where an agent performs an action and thus signals to the other agents some new information about itself.

As each agent is an autonomous entity, there is no guarantee that communication directed towards it will perform the desired affect. However, a rational agent may respond to communication regarding the formation of a coalition that will benefit it. Agent communication is therefore a central topic within this thesis as the formation of a coalition, and working within one, is a social process where agents look to find the best solution for themselves that the other agents of the potential coalition will accept.

In this thesis, the agents can communicate: (i) their valuations of coalitions; (ii) their proposals for the formation of a coalition; and (iii) the reasons for accepting or rejecting these coalition formation proposals. Chapter 3 shows how agents can communicate rational arguments on the possible formation of different coalitions, whereas Chapter 5 shows how communication within a decentralised algorithm can be used to find which coalitions should rationally form.

The structure of this section is as follows: firstly in Section 2.5.1 the theory of speech acts will be introduced, which dialogue games (introduced in Section 2.5.2) build upon. Finally in Section 2.5.3, examples of dialogue games being used for multi-agent system coalition formation will be discussed.

2.5.1 Speech Acts

The theory of speech acts arose from the work of Austin [17], who realised that some communication between individuals, known as *utterances*, can be viewed as actions that change the state of the world, and agents can use these utterances to further their goals. Yet speech acts change the state of the world in a more subtle way than physical actions. Physical actions involve the movement of objects which the traditional human senses of sight and (possibly) touch and hearing can detect. Speech acts on the other hand, (should) affect the mental state of the participants to which the speech act has been communicated.

There are many possible speech acts that can change the state of the world in profound ways for an individual or a community, for example: the statement "I declare war" sets in motion a sequence of events that will greatly impact on many lives; the statement "I sentence you to life imprisonment" has vast negative consequences on the person being sentenced; while the statement "I declare you husband and wife" should (hopefully) be one of the happiest memories of a couple's lives. Austin noted that speech acts can be identified by a number of *performative verbs* such as: accept, advise, assert, christen, object, order, resign, testify, etc.

Austin detailed three different parts of a speech act: (i) the *locutionary* act, being the transmission of the utterance; (ii) the *illocutionary act*, which is the intended force that the utterer wants to convey on the receiver, for example, an utterance may be viewed as a request to the receiver (e.g. "pass me the sugar") or information for the receiver (e.g. "I am a vegetarian"); and finally (iii) the *perlocution*, which brings about the effect on the receiver as a result of the utterance, for example, the receiver would pass the utterer the sugar or the receiver would not give the utterer a meal including meat.

Speech acts may or may not be successful. For a speech act to be successful it needs to pass a number of *felicity conditions*, as summarised in Allan [3]:

1. A **preparatory condition** - This condition requires the circumstances of the speech act and the participants in it (i.e. speaker and hearers) to be appropriate, for example a child cannot declare a war for the United Kingdom.

2. An **executive condition** - This condition concerns whether or not the speech act has been properly executed, for example were the marriage vows correctly recited?

3. A **sincerity condition** - This condition requires the speaker to really want the perlocutionary affect to occur, else the utterance will negatively affect the speaker.

4. A **fulfillment condition** - This condition requires the intended perlocutionary effect of the speech act to occur, for example if an speaker asks for milk, then the hearer will give the utterer milk.

Searle refined Austin's work by classifying the possible types of speech acts in the following manner [110]:

- **Assertives** that commit the speaker to the truth of the expressed proposition, for example: *asserting, announcing, confirming, informing and stating*.

- **Directives** that are an attempt to cause the hearer to take a certain action, for example: *asking, begging, ordering, permitting and requesting*.
- **Commissives** commit the speaker to some future action, for example: *agreeing to, guaranteeing that, offering to, promising and volunteering*.
- **Expressives** express the speaker's attitude and emotions, for example: *apologising, congratulating, greeting, thanking and welcoming*.
- **Declarations** immediately change the reality with relation to the propositional content of the speech act, for example: *baptising, declaring war, sacking an employee, marrying someone and sentencing someone*.

As detailed in [119], speech acts have many practical applications in multi-agent systems including intelligent dialogue systems, workflow systems, agent communication languages and rational programming. In this thesis, speech acts are used within dialogue games, which are introduced in the next section.

2.5.2 Dialogue Games

Conversations are sequences of messages between two (or more) agents who are *doing by talking*. These sequences can be restricted using a completely fixed protocol which states exactly which message should come next. Alternatively, conversations can also be seen as completely free sequences of messages, where agents can decide at any moment what will be the next message they send. Yet a completely free sequence of messages would put an extremely high computational load on each agent when deciding what message to send (especially in the coalition formation domain, due to the high number of options over which coalition to form). Dialogue games allow conversations to be structured, but allow agents to have some freedom in the execution, limiting the number of possible responses at each point, but not completely fixing the sequence of messages [46]. As dialogue games follow a certain structure, it is possible to prove that in given circumstances the dialogue results in a certain outcome [46].

Dialogue games have been studied since Aristotle [6] and typically consist of a set of communicative acts (called moves), a set of rules stating which moves are legal for any point of the dialogue (the protocol), a set of rules defining the syntax and semantics of a move, and a set of rules that determine when a dialogue terminates. When an agent makes a dialogue move, this move is referred to as an *utterance*.

When an agent uses an utterance, new information can be revealed. This information can be recorded in a publicly readable *committment store*, derived from Hamblin's study of fallacious reasoning [61]. The statements placed in the commitment store do not have to represent each agent's true individual beliefs, yet agents should defend them. If they don't, this can lead to accusations of inconsistent reasonings or an attempted manipulation of the dialogue. In this thesis, an agent can show inconsistent reasoning by not joining the coalition that gives it the best offer.

In the field of Artificial Intelligence, the main application of dialogue games to date has been to design protocols for interaction between agents. Work on agent-based dialogue systems has been greatly influenced by the dialogue typology of Walton and Krabbe [129], who categorised dialogues as falling within one of six main dialogue types (though they do not claim this list is exhaustive) [71, 129]:

1. **Information-Seeking Dialogues** - Where one participant is ignorant of some knowledge and so asks a question to another to find out this information.
2. **Inquiry Dialogues** - Where the participants of the dialogue work together to answer a single question or multiple questions, the answers to which are not known to any individual participant.
3. **Persuasion Dialogues** - Where one participant seeks to persuade another to accept a proposition that the other does not currently endorse.
4. **Negotiation Dialogues** - Where the participants bargain over the division of some scarce resource.
5. **Deliberation Dialogues** - Where the participants collaborate to decide what action or plan should be adopted in some situation.
6. **Eristic Dialogue** - Where participants quarrel verbally as a substitute for physical fighting to vent perceived grievances.

A number of proposals have been set out for dialogue systems that encompass the main dialogue categories, for example: inquiry dialogues [22]; negotiation [86]; persuasion [85]; and deliberation [70]. Nesting of dialogues can also occur, for example: a negotiation dialogue inside a persuasion dialogue (e.g. [130]); or an inquiry dialogue inside a persuasion dialogue (e.g. [21, 101]).

This thesis introduces a new dialogue game to apply within multi-agent system coalition formation, which is an inquiry dialogue inside a persuasion dialogue that allows the agents to reason qualitatively over what coalitions should form (introduced in Chapter 3);

2.5.3 Dialogue Games used for Coalition Formation

Dialogue games (or variations of dialogue games) have previously been shown to be a useful way to organise coalition formation (e.g. in [4, 24, 46, 62, 95]). These coalition formation dialogue games allow for more flexibility for the agents' decision making and communications, compared to a protocol with fixed steps. An example of a protocol with fixed steps is the Contract Net [120], which has been proposed to allocate agents tasks, and so can be used to form teams/coalitions [105].

Coalition (or team) formation via a dialogue game was first proposed by Dignum *et al.* [46]. Dignum *et al.* used the belief, desire and intentions (BDI) architecture to describe the interactions of the agents when forming coalitions. In [46] the necessary communication was

divided into two different dialogue types: (i) An *information seeking dialogue* so that agents can form beliefs on the abilities, opportunities and willingness of other agents to work together; and (ii) a *persuasion dialogue* where an agent aims to persuade others to want to join a coalition to achieve some goal G.

The information seeking dialogue of (i) consists of an agent i asking another agent j about some proposition ψ representing a possible ability, opportunity or willingness of agent j. Agent j can respond in any of the following ways:

1. It can ignore i and not reply.
2. It can state that it is not willing to disclose information on ψ.
3. It can state that it does not have enough information to have a single point belief on ψ.
4. It can assert that it believes ψ to be true or false.

Additionally, the persuasion dialogue consists of three stages: *information exchange*, *rigorous persuasion* and *completion*. Information exchange consists of the agents making clear their stance towards a coalition being formed to achieve the proposed goal G. Only when a conflict between two agents' stance is found does a rigorous persuasion dialogue start. Rigorous persuasion is adapted from [129] and involves a proponent P and an opponent O taking it in turns to challenge, question and state information relating to the conflicting data. Finally the rules for completion of the persuasion dialogue must be pre-defined.

While Dignum *et al.* developed a method for coalition formation by dialogue where each coalition is valued in a qualitative manner, Ramchurn *et al.* [95] developed a negotiation protocol (which could easily be recast as a negotiation dialogue game), for coalition formation in social networks where each coalition is valued in a quantitative manner. In this protocol the agents exploit the following four illocutionary actions: *Propose*, *Counter*, *Accept* and *Reject*, to discuss offers from an agent i to an agent j that state: the coalition to join; the actions of the coalition; the action of agent j to perform in this coalition; and the payoff x_j for agent j. Ramchurn *et al* use a notion of *offer commitment*, which is akin to the commitment store of the dialogue game literature. The full protocol given in [95] leads to the creation of a coalitional game outcome (i.e. a coalition structure and payoff vector pair), which differs from the work of Dignum *et al.* that only discusses the formation of a single coalition.

However, issues in coalition formation dialogue games still remain. For instance:

- Some dialogue games (e.g. [4, 46]) only provide a two-agent dialogue game, and do not detail how the full coalition structure can be found. The dialogue game presented in Chapter 3 are for multiple agents.
- In [46], the uncertainty only comes from the agents' possible capabilities, yet there may be other environmental factors affecting the uncertainty. This issue is discussed in Chapter 3 through the newly introduced inquiry dialogue game and in Chapter 6 through the introduction of new stability solution concepts for coalitional games.

- A more thorough handling of a persuasion dialogue game in a qualitative environmental model, compared to the one in [46], can be created as introduced in Chapter 3.

2.6 Argumentation

Argumentation is the process of attempting to reason over what to believe and what to do, when information or beliefs are contradictory [87]. Therefore argumentation provides the techniques for deciding and justifying what to believe when inconsistency arises.

In this section two abstract argumentation systems, named *argumentation frameworks* and *value-based argumentation frameworks* will be detailed in Sections 2.6.1 and 2.6.2 respectively. Both systems are concerned with the overall structure of sets of arguments and consistency within these sets, rather than the internals of each argument.

The internals of each argument take the form of an instantiated argumentation scheme, introduced in Section 2.6.3. In Section 2.6.4, Value-Based Alternating Transition Systems are introduced, which detail the environmental model used in Chapter 3, for agents to exchange practical reasoning arguments for/against coalitions to undertake a joint-action. In Section 2.6.5, an argumentation model that allows agents to reason over the current state of the environment is described. This argumentation model helps the agents reason over the correct current beliefs, so that arguments exchanged are more likely to be applicable to the current world state. Finally in Section 2.6.6, previous argumentation methods to form coalitions are reviewed.

2.6.1 Argumentation Frameworks

Abstract argumentation systems (otherwise known as *argumentation frameworks*) [50] are a means to represent and reason with different, possibly conflicting data, to come to some reasonable conclusion. In such argumentation frameworks, each *abstract argument* represents some data, propositions, premises and/or conclusions. An abstract argument a_1 may attack another abstract argument a_2 if a_1 *conflicts* in some manner with a_2. Argumentation frameworks can be represented by a directed graph of nodes and arcs, where the nodes represent abstract arguments, having no internal structure, and the arcs represent attacks between the arguments. The formal definition is as follows:

Definition 30: *An* **Argumentation Framework** *is a tuple AF = (*$Args$*, R) where* $Args$ *is a set of arguments and* R *is a binary attack relation* $R \subseteq Args \times Args$.

There are various notions relating to the status of different subsets of arguments in argumentation frameworks; here are the main ones as defined in [50]:

Definition 31: *For an AF, a set of arguments* $S \subseteq Args$ *is* **conflict free** *(i.e.* $cf(S)$ *holds true), iff* $\neg\exists a_x, a_y \in S$ *where* $R(a_x, a_y)$.

As noted in [50], the notion that a set of arguments is conflict free can be thought of as the most basic requirement for a rational position, i.e. an argument set S should be internally consistent. Therefore no two arguments in a rational argument set S should attack each other.

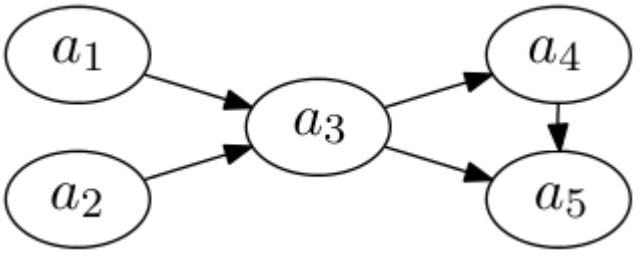

FIGURE 2.3: An argumentation framework, used in the examples of Section 2.6.1, which has one preferred extension of $\{a_1, a_2, a_4\}$.

Example 14: *Consider the example argumentation framework in Figure 2.3. The argument sets that are conflict free are:* $\emptyset$, $\{a_1\}$, $\{a_2\}$, $\{a_3\}$, $\{a_4\}$, $\{a_5\}$, $\{a_1, a_2\}$, $\{a_1, a_4\}$, $\{a_1, a_5\}$, $\{a_2, a_4\}$, $\{a_2, a_5\}$, $\{a_1, a_2, a_4\}$ *and* $\{a_1, a_2, a_5\}$. *The empty set is included because, even though it has no arguments, it satisfies the conflict free definition.*

Definition 32: *For an AF, an argument* a_x *is* **acceptable** *wrt. a set* $S \subseteq Args$ *(i.e.* $acceptable(a_x, S)$ *holds true) iff* $\forall a_y \in Args$, *where* $R(a_y, a_x)$ *then* $\exists a_z \in S$ *where* $R(a_z, a_y)$.

Therefore an argument a_x is acceptable to a set S, if every attacker a_y of a_x is attacked by an element a_z of S, i.e. a_z *defends* a_x. Note that an argument can defend itself, i.e. it is possible that $a_z = a_x$.

Example 15: *Consider the example argumentation framework in Figure 2.3. Some possible acceptable arguments are:* a_1 *for the set* $S = \{a_1, a_2, a_3, a_4, a_5\}$; a_2 *for the set* $S = \{a_1, a_2\}$; *and* a_4 *for the set* $S = \{a_2, a_4\}$. *The argument* a_3 *is not acceptable for any conflict free set because there are no arguments that attack the attackers of* a_3, *which are* a_1 *and* a_2. *Likewise, the argument* a_5 *is not acceptable for any conflict free set because, even though an argument* a_3 *exists that defends* a_5 *from the attack of* a_4, *there still exists no arguments to defend* a_3 *from attack.*

Definition 33: *For an AF, a set of arguments* $S \subseteq Args$ *is* **admissible** *(i.e.* $admissible(S)$ *holds true) iff* $cf(S)$ *holds and* $\forall a_x \in S$ *then* $acceptable(a_x, S)$.

Therefore, an admissible set is a collection of acceptable arguments.

Example 16: *Consider the example argumentation framework in Figure 2.3. The argument sets that are admissible are:* $\emptyset$, $\{a_1\}$, $\{a_2\}$, $\{a_4\}$, $\{a_1, a_2\}$, $\{a_1, a_4\}$, $\{a_2, a_4\}$ *and* $\{a_1, a_2, a_4\}$. *The empty set is again included because it again satisfies the admissible definition.*

Definition 34: *For an AF, a set of arguments* $S \subseteq Args$ *is a* **preferred extension** *(i.e.* $preferred(S)$ *holds true) iff* $admissible(S)$ *holds and* S *is a maximal (with respect to set inclusion) admissible set of arguments.*

As detailed in [50], a preferred extension S has the property that no additional argument a_x can be added to S and for the set $S \cup a_x$ to be admissible. Therefore each preferred extension is "as large as possible" wrt. admissibility.

Example 17: *Consider the example argumentation framework in Figure 2.3. There is only one preferred extension which is:* $\{a_1, a_2, a_4\}$. *No more arguments can be added to this preferred extension whilst ensuring that the resulting set is admissible. Some argumentation frameworks*

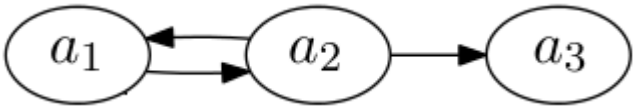

FIGURE 2.4: An argumentation framework that has two preferred extensions of $\{a_1, a_3\}$ and $\{a_2\}$, as described in example 17.

can have multiple (or zero) preferred extensions. For example the preferred extensions of Figure 2.4 are: $\{a_1, a_3\}$ *and* $\{a_2\}$.

2.6.2 Value-Based Argumentation Frameworks

Argumentation Frameworks were extended by Bench-Capon in [19] to include *social-values*[3], giving a *Value-based Argumentation Framework* (VAF). In VAFs each argument is given an associated social-value. Therefore, although Argumentation Frameworks are concerned with beliefs, VAFs enable reasoning through consideration of subjective preferences that justify different agents' conclusions. The attacks between the arguments in a VAF succeed or fail for an *audience* depending on the audience's preference ordering over these social-values. A VAF is defined in [19] as follows:

Definition 35: *A* **Value-based Argumentation Framework (VAF)** *is a 5-tuple:*
VAF $= \langle Args, R, V, val, P \rangle$ *where Args and R remains the same as in an AF,* V *is a non empty set of social-values,* val *is a function mapping from each element of* $Args$ *to an element of* V*, and* P *is the set of possible audiences (i.e. the total possible orders on* V*).*

When one possible audience considers the content of a VAF, it is known as an Audience Specific VAF (AVAF), defined as:

Definition 36: *An* **Audience specific Value-based Argumentation Framework (AVAF)***, for audience* i *is a 5-tuple:* VAF$_i$ $= \langle Args, R, V, val, ValPref_i \rangle$ *where* $Args$*,* R*,* V *and* val *remain the same as in a VAF, and* $ValPref_i \subseteq V \times V$ *reflecting the value preferences of audience* i*.* $ValPref_i$ *is the set of preferences derivable from the ordering* $i \in P$ *in the associated VAF.*

As each difference audience uses a difference preference order over the valuations V, then a notion of an attack *succeeding* or *failing* occurs. In AFs, all attacks are given the same priority. Whereas in VAFs, attacks from an argument a_x associated with a social-value $val(a_x)$, on an argument a_y associated with a social-value $val(a_y)$, will only succeed for an audience i if $val(a_x)$ is equal or higher than $val(a_y)$ in the preference order $ValPref_i$, i.e.:

Example 18: *Consider the example value-based argumentation framework in Figure 2.5, where there are three arguments* $Args = \{a_1, a_2, a_3\}$*, three values* $V = \{v_1, v_2, v_3\}$ *and the mapping of arguments to values is as follows:* $val(a_1) = v_1$*,* $val(a_2) = v_2$ *and* $val(a_3) = v_3$*. The possible audiences of these valuations are* $P = \{v_1 \succ v_2 \succ v_3, v_1 \succ v_3 \succ v_2, v_2 \succ v_1 \succ v_3, v_2 \succ v_3 \succ v_1, v_3 \succ v_1 \succ v_2, v_3 \succ v_2 \succ v_1\}$*. If the value preference used was* $ValPref_i = \{(v_1, v_2), (v_3, v_2), (v_3, v_1)\}$ *then* a_1 *would* defeat a_2 *but* a_1 *would* not defeat a_3*. Whereas if the*

[3]The term "social-values" is hyphenated in this thesis, to distinguish the qualitative interpretation of the word 'value' from a quantitative definition of 'value' used so far in this thesis to represent a coalition's utility value.

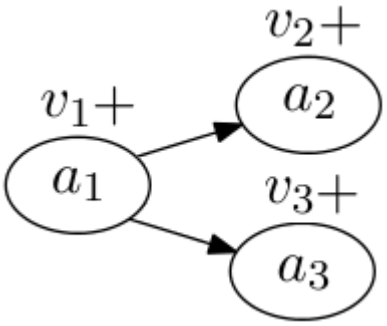

FIGURE 2.5: A example value-based argumentation framework, described in example 18. The social-value associated with the argument is present outside and adjacent to that argument. For example, argument a_1 promotes social-value v_1.

value preference used was $ValPref_j = \{(v_1, v_2), (v_1, v_3), (v_2, v_3)\}$ *then* a_1 *would* defeat *both* a_2 *and* a_3.

Definition 37: *For a* VAF_i*, an argument* $a_x \in Args$ **defeats** *an argument* $a_y \in Args$ **for an audience** i *(i.e.* $defeats_i(a_x, a_y)$ *holds true) iff* $R(a_x, a_y)$ *and* $(val(a_y), val(a_x)) \notin ValPref_i$.

As the defeat definition has been modified for VAFs compared to the one introduced for AFs in Section 2.6.1, the definitions relating to the properties of the arguments of the VAF need to be modified from AFs as well:

Definition 38: *For a* VAF_i*, a set of arguments* $S \subseteq Args$ *is* **conflict free for audience** i *(i.e.* $cf_i(S)$ *holds true), iff* $\neg\exists a_x, a_y \in S$ *where* $R(a_x, a_y)$ *and* $(val(a_y), val(a_x)) \notin ValPref_i$.

The notion that a set of arguments is conflict free in an AVAF is a modified version of the conflict free definition for an AF, such that it also includes social-values. Unlike in an AF, arguments in a conflict free set S for an AVAF can attack each other, but only if those attacks fail due to the value ordering.

Example 19: *To show the difference between AFs and VAFs for the conflict free definition, values have been added to the AF of Figure 2.3. Consider the Audience specific Value-based Argumentation Framework* VAF_i *based on the VAF of Figure 2.6, with the preference order* $ValPref_i = \{(v_1, v_2), (v_1, v_3), (v_2, v_3)\}$*. The argument sets that are conflict free are firstly the same as those produced from the AF of Figure 2.3:* $\emptyset$, $\{a_1\}$, $\{a_2\}$, $\{a_3\}$, $\{a_4\}$, $\{a_5\}$, $\{a_1, a_2\}$, $\{a_1, a_4\}$, $\{a_1, a_5\}$, $\{a_2, a_4\}$, $\{a_2, a_5\}$, $\{a_1, a_2, a_4\}$ *and* $\{a_1, a_2, a_5\}$*. However, the additional argument sets that are conflict free for audience* i *are:* $\{a_3, a_4\}$, $\{a_3, a_5\}$, $\{a_4, a_5\}$, $\{a_1, a_4, a_5\}$, $\{a_2, a_4, a_5\}$, $\{a_3, a_4, a_5\}$ *and* $\{a_1, a_2, a_4, a_5\}$*. These additional conflict free argument sets occur because even though the arguments in the additional conflict free sets attack each other, none of the attacks succeed as the social-value associated with the attacking arguments are not ranked equal or higher than the social-value associated with the arguments being attacked.*

Definition 39: *For a* VAF_i*, an argument* a_x *is* **acceptable for audience** i *wrt. a set* $S \subseteq Args$ *(i.e.* $acceptable_i(a_x, S)$ *holds true) iff* $\forall a_y \in Args$*, where* $defeats_i(a_y, a_x)$ *there exists* $a_z \in S$ *where* $defeats_i(a_z, a_y)$.

Example 20: *Consider the Value-based Argumentation Framework of Figure 2.6. Some possible acceptable arguments for different audiences for this VAF are:* a_1 *for any audience and*

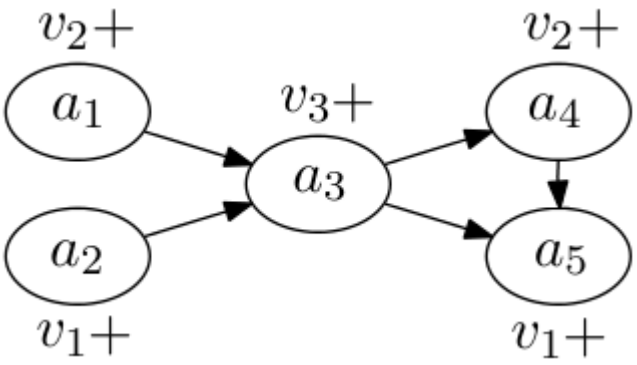

FIGURE 2.6: A value-based argumentation framework (VAF), used within the examples of Section 2.6.2. This VAF extends the argumentation framework of Figure 2.3 to include social-values. The social-value associated with the argument is present outside and adjacent to that argument. For example, argument a_1 promotes social-value v_2.

an example argument set $S = \{a_1, a_2, a_3, a_4, a_5\}$; a_2 for any audience and an example argument set $S = \{a_1, a_2\}$; a_3 for the audience i with $ValPref_i = \{(v_3, v_2), (v_3, v_1), (v_2, v_1)\}$ and the example argument set $S = \{a_1, a_2, a_3\}$; a_4 for the the audience j with $ValPref_j = \{(v_1, v_2), (v_1, v_3), (v_2, v_3)\}$ and the example argument set $S = \{a_2, a_4\}$; and a_5 for the audience k with $ValPref_k = \{(v_1, v_2), (v_1, v_3), (v_2, v_3)\}$ and the example argument set $S = \{a_4, a_5\}$. Therefore the notion of a set of arguments in a VAF being conflict free can allow more arguments to be acceptable than the same graph in AF form.

Definition 40: *For a VAF$_i$, a set of arguments $S \subseteq Args$ is* **admissible for audience** *i (i.e. $admissible_i(S)$ holds true) iff $cf_i(S)$ holds and $\forall a_x \in S$ then $acceptable_i(a_x, S)$.*

Thus similar to AFs, AVAF admissible sets are a collection of acceptable arguments (where AVAFs consider acceptability for a specific audience).

Example 21: *Consider the Value-based Argumentation Framework given in Figure 2.6. The admissible arguments for a VAF$_i$ based on Figure 2.6 with the $ValPref_i = \{(v_2, v_1), (v_2, v_3), (v_1, v_3)\}$ are: $\emptyset$, $\{a_1\}$, $\{a_2\}$, $\{a_4\}$, $\{a_1, a_2\}$, $\{a_1, a_4\}$, $\{a_2, a_4\}$ and $\{a_1, a_2, a_4\}$. The admissible arguments for a VAF$_j$ based on Figure 2.6 with the $ValPref_j = \{(v_3, v_1), (v_3, v_2), (v_2, v_1)\}$ are $\emptyset$, $\{a_1\}$, $\{a_2\}$, $\{a_3\}$, $\{a_1, a_2\}$, $\{a_1, a_3\}$, $\{a_2, a_3\}$ and $\{a_1, a_2, a_3\}$.*

Definition 41: *For a VAF$_i$, a set of arguments $S \subseteq Args$ is a* **preferred extension for audience** *i (i.e. $preferred_i(S)$ holds true) iff $admissible_i(S)$ holds and S is a maximal (with respect to set inclusion) admissible set of arguments.*

Example 22: *Consider the Value-based Argumentation Framework of Figure 2.6. The preferred extension for a VAF$_i$ based on Figure 2.6 with the $ValPref_i = \{(v_2, v_1), (v_2, v_3), (v_1, v_3)\}$ is $preferred_i(\{a_1, a_2, a_4\})$, which is the same preferred extension as the equivalent AF. The preferred extension for a VAF$_j$ based on Figure 2.6 with the $ValPref_j = \{(v_3, v_1), (v_3, v_2), (v_2, v_1)\}$ is $preferred_j(\{a_1, a_2, a_3\})$.*

Given that the preferred extensions of different audiences can vary, the following two definitions were introduced to find out what arguments are acceptable to the audiences [19]:

Definition 42: *For a* VAF $= \langle Args, R, V, val, P \rangle$*, an argument $a_x \in Args$ is* **objectively acceptable** *iff $\forall i \in P$, a_x is in every preferred extension $preferred_i$.*

Definition 43: *For a* VAF $= \langle Args, R, V, val, P \rangle$*, an argument* $a_x \in Args$ *is* **subjectively acceptable** *iff* $\exists j \in P$*, where* a_x *is in some preferred extension* $preferred_j$*.*

For an argument to be objectively acceptable requires the argument to be included by all audiences in all their possible preferred extensions of the considered VAF. Subjective acceptance is weaker instead requiring the argument to be accepted by at least one audience.

Example 23: *Consider the Value-based Argumentation Framework of Figure 2.6. If there are only two audiences* i *and* j*, where the preferred extension for a VAF*$_i$ *is* $preferred_i$ $(\{a_1, a_2, a_4\})$ *and the preferred extension for a VAF*$_j$ *is* $preferred_j(\{a_1, a_2, a_3\})$*, then arguments* a_1 *and* a_2 *are objectively acceptable, while arguments* a_3 *and* a_4 *are subjectively acceptable.*

In Chapter 3 of this thesis, value-based argumentation frameworks and preferred extensions over audience specific value-based argumentation frameworks are used to find solutions to a variant of qualitative coalitional games.

2.6.3 Argumentation Schemes

Abstract argumentation frameworks are useful for evaluating the acceptability of arguments based on the attack relations between them. However, abstract arguments themselves are not useful for representing instantiated arguments; i.e., arguments with some internal structure or content. To reason over the best coalition structure to form, *argumentation schemes* will be instantiated in Chapter 3, and the arguments yielded will be organised into a value-based argumentation framework.

An argumentation scheme is a form of inference from premises to a conclusion that represents a pattern of deductive, inductive or defeasible reasoning [97]. An instantiated argumentation scheme provides justification for the particular conclusion of the scheme [21]. Argumentation schemes allow for arguments to be represented within a particular context, and take into account the fact that the reasoning presented within them may be altered if new evidence is found [8]. Schemes are necessary for identifying arguments, finding missing premises, analysing arguments and evaluating the arguments [97]. Many computational models based on argumentation schemes exists (e.g. [60, 68, 84, 123]). Moreover there are many types of argumentation schemes have been studied (e.g. [128] lists over 30 types), some examples are:

- *Argument from Analogy*, for instance: Generally, case C_1 is similar to case C_2. A is true in case C_1. Therefore A is true in C_2.
- *Argument from Commitment*, for instance: a is committed to proposition A. Therefore in this case a should support A.
- *Argument from Evidence to Hypothesis*, for instance: If A is true (hypothesis) then B will be observed to be true (evidence). B has been observed to be true in a given instance. Therefore in this instance A is true.
- *Argument from Expert Testimony*, for instance: E is an expert in domain D. E asserts that A is known to be true. A is within D. Therefore A may plausibly be taken to be true.

- *Argument from Popularity*, for instance: If a large majority accept C to be true, then there exists a (defeasible) presumption in favour of C. A large majority accepts C as true. Therefore there exists a presumption in favour of C.

The tool used to evaluate an argumentation scheme is a set of appropriate *critical questions* [97]. An instantiation of an argumentation scheme as_x attacks another instantiation of an argumentation scheme as_y under pre-defined conditions, which attack either the premises, inference rules or conclusions of the scheme as_y. The attack by as_x to as_y can be described as a critical question to a_y, which if left unanswered will lead to the defeat of as_y.

Argumentation schemes can be used to reason over what action or joint-action to perform. Argumentation schemes can also be used to reason over what to belief (this will be discussed in more detail in Section 2.6.5). The following is a definition of practical reasoning provided by Walton in [128]:

Definition 44: Practical Reasoning *is a goal directed sequence of linked practical inferences that seeks out a prudent line of conduct for an agent in a set of particular circumstances known by the agent.*

Walton went on to describe two basic types of practical inferences related to practical reasoning (where NCS is a special type of SCS [13]):

- The necessary condition scheme (NCS): G is a goal for agent i. Doing action A is necessary for agent i to carry out goal G. Therefore agent i ought to do action A.
- The sufficient condition scheme (SCS): G is a goal for agent i. Doing action A is sufficient for agent i to carry out goal G. Therefore agent i ought to do action A.

There are four critical questions regarding these two schemes:

- CQ1: Are there alternative ways of realising goal G?
- CQ2: Is it possible to do action A?
- CQ3: Does agent i have goals other than G which should be taken into account?
- CQ4: Are there other consequences of doing action A which should be taken into account?

In [13] it was argued that the NCS/SCS argumentation schemes needed elaboration because the notion of a goal is ambiguous, potentially referring indifferently to any direct results of the action, the consequences of those results, and the reasons why those consequences are desired. Therefore in [13], the SCS scheme was expanded on to give another practical reasoning argumentation scheme (labeled AS1) with associated critical questions:

ID	Critical Question
CQ1	Are the believed circumstances true?
CQ2	Assuming the circumstances, does the action have the stated consequences?
CQ3	Assuming the circumstances and that the action has the stated consequences, will the action bring about the desired goal?
CQ4	Does the goal realise the social-value stated?
CQ5	Are there alternative ways of realising the same consequences?
CQ6	Are there alternative ways of realising the same goal?
CQ7	Are there alternative ways of promoting the same social-value?
CQ8	Does doing the action have a side effect which demotes the social-value?
CQ9	Does doing the action have a side effect which demotes some other social-value?
CQ10	Does doing the action promote some other social-value?
CQ11	Does doing the action preclude some other action which would promote some other social-value?
CQ12	Are the circumstances as described possible?
CQ13	Is the action possible?
CQ14	Are the consequences as described possible?
CQ15	Can the desired goal be realised?
CQ16	Is the social-value indeed a legitimate social-value?
CQ17	Is the agent guaranteed to perform its part of the joint-action?

TABLE 2.3: The full list of critical questions associated with argumentation schema AS1.

Definition 45: *The* **AS1** *scheme for practical reasoning is as follows:*

In the current circumstances R

We should perform action A

Which will result in new circumstances S

Which will realise goal G

Which will promote the social-value V

This scheme uses social-values to describe a social interest that an agent has, which will be promoted by moving to a state in which goal G becomes true [19]. An agent may propose an action (the conclusion of the scheme) and its justification by instantiating this scheme. Other agents can then challenge instantiations by posing critical questions (CQ) associated with the scheme. Seventeen critical questions are associated with the above scheme [10, 12], which raise potential issues with: the validity of the elements instantiated in the scheme; the connections between the elements of the scheme; the side effects of the action; and the possible alternative actions. These seventeen critical questions are provided in Table 2.3.

Each of these critical questions identifies a source of disagreement between two or more agents concerning an instantiation of the argumentation scheme AS1. Some critical questions disagree with the premises of the scheme, for example CQ14 disagrees with the premise of the

new consequences. Other critical questions offer an alternative conclusion, for example, CQ6 offers a new action to achieve the required goal.

The disagreements highlighted by critical questions against an argumentation scheme can be modeled in a value-based argumentation framework or simply an argumentation framework (if no social-values were used), to reason logically over what is the best conclusion. In the AS1 case, value-based argumentation frameworks would be used to reason over what is the best action to perform. A modified version of AS1 and its associated seventeen critical questions are used later in Chapter 3 for agents to persuade others to form a coalition to undertake a joint-action.

2.6.4 Value-based Alternating Transition Systems

To help with the generation of argumentation schemes, a Value-based Alternating Transition System (VATS) can be used, because VATS model the environment using propositions and state transitions that can be formalised into argumentation schemes, such as in the models of [9, 21]. A VATS is a modified version of an Action-based Alternating Transition System (AATS) [135], which has been extended to enable the inclusion of social-values. When the agents are using the same environmental model (the VATS), argumentation schemes instantiated from parts of each agents VATS can be used to reason over possibly contradictory information. A VATS is defined as:

Definition 46: *A* **VATS** *for an agent i, denoted* VATS^i*, is a 9-tuple* $\langle Q^i, q_0^i, Ac^i, Av^i, \rho^i, \tau^i, \Phi^i, \pi^i, \delta^i \rangle$ *s.t.:*

- Q^i *is a finite set of* states*;*
- $q_0^i \in Q^i$ *is the designated* initial state*;*
- Ac^i *is a finite set of* single actions*;*
- Av^i *is a finite set of* social-values*;*
- $\rho^i : 2^{Ac^i} \mapsto 2^{Q^i}$ *is an* action precondition function*, which for each joint-action* $\mu = \langle ac^x, ..., ac^y \rangle$ *(where* $\forall ac^x \in \mu, ac^x \in Ac^i$*), defines the set of states* $\rho(\mu)$ *from which* μ *may be executed;*
- $\tau^i : Q^i \times 2^{Ac^i} \mapsto Q^i$ *is a partial* system transition function*, which defines the state* $\tau^i(q_x, \mu)$ *that would result from the performance of joint-action* $\mu = \langle ac^x, ..., ac^y \rangle$ *from state* q_x*. As this function is partial, not all joint-actions are possible in all states;*
- Φ^i *is a finite set of* atomic propositions*;*
- $\pi^i : Q^i \mapsto 2^{\Phi^i}$ *is an interpretation function, which gives the set of primitive propositions satisfied in each state: if* $p \in \pi^i(q_y)$*, then this means that the propositional variable* p *is satisfied (equivalently, true) in state* q_y*; and*

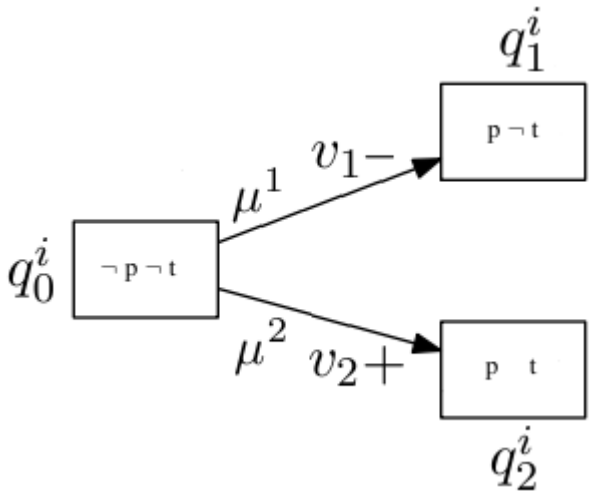

FIGURE 2.7: An example Value-based Alternating Transition System for an agent i. The rectangles represent the different states and contain the propositions true or false at that state. Each state is given an identifier that is adjacent to that state, for example the left most state has the identifier q_0^i. The arrows represent the transitions between the states and are labeled by the joint action needed for the transition and the social-value promoted/demoted by the transition. For example the lower arrow requires joint-action μ^2 to be performed and this transition will promote social-value v_2.

- $\delta^i : Q^i \times Q^i \times Av^i \mapsto \{+, -, =\}$ *is a* valuation function *which defines the* status *(promoted (+), demoted (−), or neutral (=)) of a social-value* $v \in Av^i$ *ascribed by the agent to the transition between two states; i.e.* $\delta^i(q_x, q_y, v)$ *labels the transition between* q_x *and* q_y *with respect to the value* $v \in Av^i$.

Note, $Q^i = \emptyset \leftrightarrow Ac^i = \emptyset \leftrightarrow Av^i = \emptyset \leftrightarrow \Phi^i = \emptyset$.

Figure 2.7 illustrates an example VATS for an agent i with: three states (i.e. $Q^i = \{q_0^i, q_1^i, q_2^i\}$); one designated initial state q_0^i; four single actions (i.e. $Ac^i = \{ac^1, ac^2, ac^3, ac^4\}$ that combine to give two possible joint-actions $\mu^1 = \langle ac^1, ac^2 \rangle$ and $\mu^2 = \langle ac^3, ac^4 \rangle$); two social-values (i.e. $Av^i = \{v_1, v_2\}$); two pre-conditions (i.e. $\rho(\mu^1) = q_0^i$ and $\rho(\mu^2) = q_0^i$); two transition functions (i.e. $\tau(q_0^i, \mu^1) = q_1^i$ and $\rho(q_0^i, \mu^2) = q_2^i$); two propositions p and t (i.e. $\Phi = \{p, t\}$); three interpretations (i.e. $\pi(q_0^i) = \{\neg p, \neg t\}$, $\pi(q_1^i) = \{p, \neg t\}$ and $\pi(q_2^i) = \{p, t\}$); and two valuation functions (i.e. $\delta(q_0^i, q_1^i, \mu^1) = -$ and $\rho(q_0^i, q_2^i, \mu^2) = +$).

VATS can be used to formally instantiate argumentation schemes. For example, the AS1 scheme for practical reasoning introduced in the previous section can now formally become:

Definition 47: *An* **AS1-argument** *constructed by an agent* i *from its VATS*^{i} *is a 6-tuple* $AS1 = \langle q_x, \mu, q_y, p, v, s \rangle$ *s.t.: the current circumstances are* $q_x = q_0^i$*; the proposed action is* μ*; the new circumstances will be* $\tau^i(q_x, \mu) = q_y$*; the goal realised is* $p \in \pi^i(q_y)$*; the value is* $v \in Av^i$*; and the value will be affected by* $\delta^x(q_x, q_y, v) = s$ *where* $s \in \{+, -, =\}$.

In Chapter 3, VATS are used to allow agents to create new instantiated argumentation schemes that allow the agents to logically argue for/against a coalition to form to undertake a joint-action.

2.6.5 Reasoning Over Current Beliefs

When reasoning over what to do, agents may need to inquire about beliefs on the current state of the world, especially when their environment is represented in some qualitative form. To do this, agents can have epistemic knowledge (beliefs). One popular approach for representing such beliefs is Garcia and Simari's Defeasible Logic Programming (DELP) [57]. The following definitions from [21], which makes use of DELP, provide a formal framework for modeling beliefs:

Definition 48: *A* **defeasible rule** λ *is denoted* $\alpha_1 \wedge \ldots \wedge \alpha_{n-1} \rightarrow \alpha_n$ *where* α_i *is a literal for* $1 \leq i \leq n$*. A* **defeasible fact** *is denoted* α *where* α *is a literal. A* **belief** *is either a defeasible rule or a defeasible fact. The following functions are defined:* $\mathsf{DefeasibleSection}(\lambda) = \{\alpha_1, \ldots, \alpha_{n-1}\}$*;* $\mathsf{DefeasibleProp}(\lambda) = \alpha_n$*.*

Definition 49: *A* **belief base** *of an agent* j *is represented as a set denoted* Σ^j*, which contains all the defeasibled rules or defeasible facts that an agent is aware of.*

Example 24: *An example defeasible rule is* $r \wedge s \rightarrow t$ *where the following could hold:* $r =$ grass is wet*,* $s =$ no garden sprinklers *and* $t =$ it has been raining*. Additionally* r *and* s *can be viewed as defeasible facts if they are known to be true. For instance, if there were two agents* i *and* j*, their belief bases could be:* $\Sigma^i = \{r \wedge s \rightarrow t\}$ *and* $\Sigma^j = \{r, s\}$*. If the two belief bases were combined then* t *would be found to hold.*

The definition of a defeasible derivation is adapted from [57] in [21], to work with the assumption that all beliefs are defeasible:

Definition 50: *Let* Ψ *be a set of beliefs and* α *a literal. A* **defeasible derivation** *of* α *from* Ψ*, denoted* $\Psi \mid\sim \alpha$*, is a finite sequence* $\alpha_1, \alpha_2, \ldots, \alpha_n$ *of literals s.t.:* α_n *is* α*; and each literal* α_m *(*$1 \leq m \leq n$*) is in the sequence because either* α_m *is a defeasible fact in* Ψ*, or there exists a defeasible rule* $\beta_1 \wedge \ldots \wedge \beta_j \rightarrow \alpha_m$ *in* Ψ *s.t. every literal* β_i *(*$1 \leq i \leq j$*) is an element* α_k *preceding* α_m *in the sequence (*$k < m$*).*

Example 25: *An example defeasible derivation is* $\Psi = \{r, s, q, r \wedge s \rightarrow t, q \rightarrow t, t \rightarrow \alpha\} \mid\sim \alpha$ *where the following could hold:* r*,* s *and* t *remain the same as the previous example;* $q =$ john saw it rain*; and* $\alpha =$ the garden does not need hosing.

A b-argument is a minimally consistent set of beliefs from which a claim can be defeasibly derived.

Definition 51: *A* $\mathcal{B}$**-argument** *is denoted* $B = \langle \Phi, \phi \rangle$ *where* ϕ *is a defeasible fact and* Φ *is a set of beliefs s.t: 1)* $\Phi \mid\sim \phi$*; 2)* $\forall \phi, \phi'$ *s.t.* $\Phi \mid\sim \phi$ *and* $\Phi \mid\sim \phi'$*, it is not the case that* $\phi \cup \phi' \vdash \bot$ *(where* $\vdash$ *represents classical implication); and there is no subset of* Φ *satisfying (1 and 2).* Φ *is called the* **support** *of the b-argument and* ϕ *is called the* **claim***.*

Example 26: *An example* $\mathcal{B}$*-argument given the defeasible derivation of the previous example is* $\langle \Phi, \phi \rangle$*, where* $\Phi = \{q, q \rightarrow t, t \rightarrow \alpha\} \mid\sim \alpha$*,* $\phi = \alpha$*, and* q*,* t *and* α *remain the same as the previous example.*

In Chapter 3 of this thesis, defeasible rules, defeasible facts, defeasible derivations and $\mathcal{B}$-arguments are used to allow the agents to find their state of the world before they argue and reason over what coalitions to form.

2.6.6 Argumentation Applied to Coalition Formation

Several studies have combined argumentation with coalition formation (e.g. [4, 24, 25, 28, 29, 50]); two of the papers that are most relevant to the work presented in this thesis are discussed here.

Firstly, in Dung's seminal paper that introduced the notion of Argumentation Frameworks (AFs) [50], he used characteristic function games to demonstrate the correctness of AFs. He showed that AFs can be used to represent core and von Neumann-Morgenstein stable solution concepts (where the grand coalition is assumed to form). The following theorem was given:

Theorem 2.1. *Let Imp be the set of imputations of a coalitional game G and let $attacks$ be the corresponding domination relation between them*[4]*. Given the argumentation framework $AF = (Imp, attacks)$, then the core of G coincides with all acceptable imputations defended by the empty set.*

This Theorem details how all the core solutions can be found for a characteristic function game with transferable utility. Amgoud expanded on Dung's work to show how argumentation frameworks can be used to reason over what *coalition structures* to form [4]. The formal framework was as follows:

Definition 52: *A* **Coalition Structure Framework** *(CSF): is a framework for generating coalition structures denoted $\langle \mathcal{C}, \mathcal{R}, \succ \rangle$. In this framework: $\mathcal{C}$ is a set of possible coalitions; $\mathcal{R}$ is a binary relation $\mathcal{R} \subseteq \mathcal{C} \times \mathcal{C}$ representing the attack relationship between coalitions (the reasoning for the attack is left undefined); and $\succ$ is a (partial or complete) pre-ordering on $\mathcal{C}$, representing the preferences of the coalitions, where a coalition C^x that attacks C^y only defeats C^y if $C^y \succ C^x$ does* not *hold.*

Due to the nature of argumentation frameworks, Amgoud outlined three classes of coalitions:

1. The class $\mathcal{A}_{\mathcal{R},\succ}$ of acceptable coalitions.
2. The class $\mathcal{R}_{\mathcal{R},\succ}$ of rejected coalitions.
3. The class $\mathcal{B}_{\mathcal{R},\succ}$ of coalitions neither accepted or rejected, the so-called class of coalitions in abeyance, where $\mathcal{B}_{\mathcal{R},\succ} = \mathcal{C} \backslash (\mathcal{A}_{\mathcal{R},\succ} \cup \mathcal{R}_{\mathcal{R},\succ})$.

These three classes can be described as follows: the *acceptable* coalitions are coalitions that are not attacked *or* coalitions defended by other acceptable coalitions; the set of coalitions in *abeyance* are not defeated by the acceptable coalitions but may conflict with each other; and the *rejected* coalitions are the set defeated by the acceptable coalitions. A two agent dialogue game was also detailed to determine whether a coalition is acceptable or not, which allowed one coalition's acceptability to be checked without having to know the full set of acceptable coalitions.

Sometimes the set of acceptable coalitions may be empty. Additionally choosing a subset of coalitions, from the coalitions in abeyance, to join with the acceptable coalitions to create a

[4] See Section 2.1 for the definition of domination and imputations.

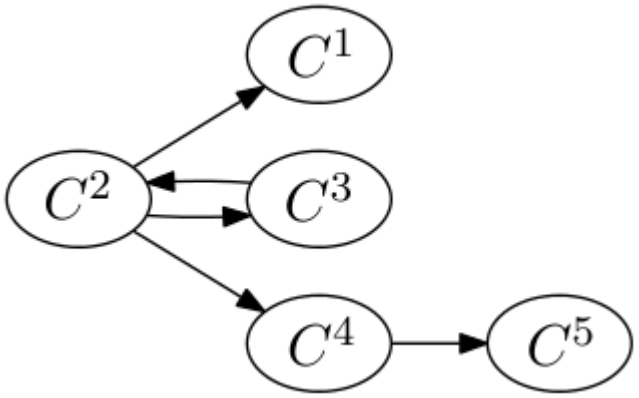

FIGURE 2.8: An example coalition structure framework.

coalition structure may not be an obvious task. To help with these issues, Amgoud introduced semantics for the coalition structure framework, as well as a modified definition of how a set of arguments are conflict free:

Definition 53: *For a CSF, a set of coalitions $S \subseteq \mathcal{C}$ is* **conflict free***, iff $\neg\exists C^x, C^y \in S$ where $R(C^x, C^y)$ and it is not the case that $C^y \succ C^x$.*

Definition 54: *For a CSF, a set of coalitions $S \subseteq \mathcal{C}$ is a* **preferred extension***, iff:*

- *S is conflict free*
- *S defends all its elements*
- *S is maximal (wrt set inclusion).*

Each CSF has at least one preferred extension:

Example 27: *Consider the following coalition structure framework, represented in Figure 2.8: $\langle \mathcal{C}, \mathcal{R}, \succ \rangle$ where $\mathcal{C} = \{C^1, C^2, C^3, C^4, C^5\}$, $\mathcal{R} = \{(C^2, C^1), (C^2, C^3), (C^2, C^4), (C^3, C^2), (C^4, C^5)\}$, $C^2 \succ C^1$, $C^4 \succ C^2$ and $C^4 \succ C^5$ (as the $\succ$ relationship is partial, all pairs of coalitions not given a preference order are assumed to have equal preference). The following set of coalitions are conflict free: $\emptyset$, $\{C^1\}$, $\{C^2\}$, $\{C^3\}$, $\{C^4\}$, $\{C^5\}$, $\{C^1, C^3\}$, $\{C^1, C^4\}$, $\{C^1, C^5\}$, $\{C^2, C^4\}$, $\{C^2, C^5\}$, $\{C^3, C^4\}$, $\{C^3, C^5\}$, $\{C^1, C^3, C^4\}$ and $\{C^1, C^3, C^5\}$. There exists two preferred extensions: $\{C^1, C^3, C^4\}$ (because the three coalitions do not attack each other and defend themselves from attacks) and $\{C^2, C^4\}$ (because even though C^2 does attack C^4, this attack does not succeed because $C^4 \succ C^2$).*

The class of acceptable coalitions is therefore $\mathcal{A}_{\mathcal{R},\succ} = \{C^4\}$ because C^4 can never be defeated. The class of rejected coalitions is $\mathcal{R}_{\mathcal{R},\succ} = \{C^5\}$ because C^5 is defeated by an acceptable coalition. This leaves the coalitions in abeyance $\mathcal{B}_{\mathcal{R},\succ} = \{C^1, C^2, C^3\}$ because these coalitions may or may not be part of the chosen coalition structure, depending on what other coalitions of the abeyance class are chosen.

Coalition Structure Frameworks are based on *preference-based argumentation theory (PBAT)* [48]. The key computational issues from PBAT that relate to Coalition Structure Frameworks are:

- Theorem 4 of [48] states that deciding whether an argument a is accepted by at least one preference order in a PBAT is NP-hard. In the context of CSFs, this means that finding out if a coalition is accepted given at least one preference order is NP-Hard.
- Proposition 3 of [48] states that deciding whether an argument a is accepted by all preference orders in a PBAT is coNP-hard. In the context of CSFs, this means that finding out if a coalition is accepted given all the possible preference orders is coNP-Hard.
- In Section 5 of [48] it is shown that a stable extension of arguments can be found in a PBAT, given a single preference order, in polynomial time. In the context of CSFs, this means that finding a set of coalitions that are stable can be found in polynomial time.

In summary, there are issues that still remain in argumentation methods for coalition formation:

- Detailing an individual imputation as one argument, as done in [50], can create a vastly complex argumentation framework, as a real value utility of a coalition can be distributed in a theoretically infinite number of ways (which, due to rounding, becomes a computationally very large but finite number of ways).
- Additionally, [50] does not detail how agents can collaboratively choose one payoff vector imputation out of all the possible core (or von-Neuman Morgenstein) stable solutions (in a possibly decentralised manner).
- Amgoud [4] does not detail what happens if the agents have their own individual preferences over which coalition to join, only one static system wide preference order is used.
- In [4], how the single static system wide preference order is found is not detailed.
- Finally, [4] does not detail what happens if the payoff is transferable, thus preferences between the agents over which coalition to join can change, depending on their received payoff.

2.7 Summary and Conclusions

In this chapter, topics within the fields of characteristic function games, cooperative game theory, coalition formation in multi-agent systems, agent communication and argumentation relevant to the research in this thesis have been presented. The remaining chapters focus on the creation of new methods for decentralised coalition formation, drawing on the research presented within this literature review. Work within decentralised environments suggest that methods for coalition formation should be presented that (among other properties): (i) make use of the distributed computational resources; (ii) minimise costly communication; (iii) deal with possibly conflicting information between the agents; and (iv) find solutions that a centralised arbiter with complete knowledge would prescribe.

The following chapters are organised as follows. Chapter 3 begins with an investigation into how argumentation and dialogue game tools (detailed in Section 2.6 and Section 2.5 respectively) can be used for the agents to reason over conflicting information in *qualitatively* described environments, to find a conclusion over what coalitions to form. Additionally Chapter 3 has parallels to the non-transferable utility game literature of Section 2.3.1.

Chapter 4 shows how the distributed computational resources of agents can be exploited when calculating all of the coalitions' values. It describes the problem as a characteristic function game, as described in Section 2.1, and focuses in depth on the literature of distributed coalition value calculation algorithms, detailed in Section 2.4.1. Chapter 5, builds on Chapter 4's work to show how cooperative game theory stable solutions can be found with distributed knowledge and minimal communication. Chapter 5 uses the characteristic function game model of Section 2.1, the cooperative game theory solution concepts of Section 2.2, and draws on the literature of coalition structure generation and payoff distribution, as described in Sections 2.4.2 and Sections 2.4.3 respectively.

Finally Chapter 6 details a new formal model of coalitional games when the agent differ on their predicted coalition values. Chapter 6 focuses in detail on the valuation disagreement coalitional game literature, described in Section 2.3.2.

For a discussion on each Chapter's contributions compared to the related work, see Section 1.4.

Chapter 3

Chapter 3

Forming Coalitions with Argumentation Schemes and Critical Questions

This chapter discusses an argumentation-based method that agents can use to form coalitions in a decentralised manner. In this method, the agents can: (i) engage in an *inquiry* dialogue, where they can use arguments over beliefs to find the current state of the world; and (ii) engage in a *persuasion* dialogue, where they can use arguments over actions to persuade others over whether or not to form a coalition. Both of these dialogue types are multi-agent dialogues. This chapter focuses on situations where the agents represent their world in qualitative terms by a set of propositions, where coalitions can achieve qualitative changes in the environment through joint-actions between the agents of the coalition. The qualitative representation of the world is assumed to be in Value-based Alternating Transition System (VATS) form[1], which itself was an extended version of AATS [135]. In the model of this chapter, the use of joint-actions (orginally embedded within the AATS themselves) allows the agents to reason over which coalitions may conflict with each others capability.

Due to the qualitative representation, inquiry and persuasion dialogues were chosen to allow the agents to form coalitions. The contributions of this chapter are an *inquiry dialogue* over beliefs that allows the agents (with possible heterogeneous knowledge bases) to reason over the current state of the world. Once a logical conclusion over the state of the world is found, the next contribution is a *persuasion dialogue* that will allow the agents to complete the coalition formation process by reasoning over what coalitions to form and what joint-actions these coalitions should undertake. Both dialogues are defined in the style of a dialogue game. Dialogue games usually consist of a set of communicative acts called moves, a set of rules detailing the moves that are legal to make at any time in a dialogue (the protocol), a set of rules stating the effect of making each move, and a set of rules that detail when a dialogue terminates (e.g. [72]).

In this chapter, the contribution of the inquiry dialogue (which is an extension of the Black & Hunter model of [22]) includes: (a) clarification on how the agents compare their current

[1] See Section 2.6.4 for the formal definition of a VATS.

states; and (b) the definition of a protocol that agents can use to find the set of legal moves in the dialogue. Both of these contributions were presented within [101]. Furthermore, the contribution of the persuasion dialogue (which is an extension of the Black & Atkinson model of [21]) includes: (i) an argumentation scheme to allow for coalitions to perform joint-actions; (ii) the formalisation of the critical questions associated with the scheme of (i); (iii) a full protocol that allows the agents to find the set of legal moves in the dialogue; and (iv) a model specific argument evaluation method to find the acceptable coalitions to form. Contribution (i) is based on [98], influenced by the argumentation scheme of [10]. Contribution (ii) is based on [101]. Contribution (iii) is an extended and modified version of the protocol in [21]. Finally, contribution (iv) is a modified version of the acceptability definitions presented in [19].

This chapter is structured as follows: Section 3.1 details how the agents find their state of the world. Section 3.2 details the practical reasoning model used. Section 3.3 details the inquiry and persuasion dialogues, including the formalisation of all the critical questions. Section 3.4 gives a protocol for the inquiry dialogue and a protocol for the persuasion dialogue, where both protocols find all the legal moves for an agent in their respective dialogues. Section 3.5 details the method to evaluate the practical reasoning arguments communicated in the persuasion dialogue, to find the acceptable coalitions. Section 3.6 gives a full example of both dialogues and the argument evaluation method. Finally Section 3.7 concludes.

3.1 Finding the State of the World

Before reasoning qualitatively over what coalition to join, it is reasonable to suggest that each agent will want to find the current state of the world. This requires each agent to inquire over the true value of the propositions it uses to represent the world that are likely to influence it's preference over which coalition to join. The inquiry dialogue, formally detailed in Section 3.3, allows the agents to communicate *defeasible facts*, *defeasible rules* and $\mathcal{B}$*-arguments*[2]. The result of these inquiries will identify $\mathcal{B}$-arguments that each agent should use to find a conclusion over the current state of the world.

Reasoning over these $\mathcal{B}$-arguments may be complicated because scenarios could arise whereby $\mathcal{B}$-arguments may claim logical contradictions (e.g. p and $\neg p$). In a decentralised environment, it is possible that some agents would put forth $\mathcal{B}$-arguments that are due to an erroneous knowledge base or a manipulation attempt. In this situation, the agents could be equipped with a weighting system based on trust to help resolve which agent's arguments should be prioritised. The exact details of this system is out of the scope of this thesis.

To find a conclusion for each inquired proposition, the agents can look at the corresponding $\mathcal{B}$-arguments for and against the proposition's correct Boolean value being true to find the most likely Boolean valuation of each proposition. For the purposes of simplifying the discussion in the rest of the chapter, it is assumed that no contradictions on the Boolean value of a proposition occur.

[2]See Section 2.6.5 for the formal definition of defeasible facts, defeasible rules and $\mathcal{B}$-arguments.

3.1.1 Comparing States

One particular issue that arose when implementing the inquiry dialogue was the need for a method to clarify how agents using different propositions to represent the state of the world can accurately compare states (since agents' VATS reflect only an individual's representation of the world). The solution is described in this subsection.

Two agents, i and j can compare their respective current states (q_i and q_j), using their respective propositional representation of the world (Φ^i and Φ^j), and assume they are equal (denoted $q_i \approx q_j$) iff $\pi(q_i) \cap \Phi^j = \pi(q_j) \cap \Phi^i$. Otherwise the states are different (denoted $q_i \neq q_j$). This equality test utilises an intersection to eliminate propositions that reside in only one of the agent's propositional representations of the world. When the above approximation holds, the two states q_i and q_j cannot reasonably be said to be different, as both states will agree for each shared proposition. However, these two states may not be identical since the same conclusion can be reached irrespective of the Boolean assignments of the distinct propositions. If the comparison does not hold, then the states are definitely different due to both agents holding inconsistent Boolean values for their shared propositions. This comparison requires: either each agent to have an internal model of the other agent's beliefs; or for the agents to make (at least some) of the Boolean values of Φ^i public. In this chapter the latter option is chosen due to the revelation of each agent's current state being a requirement of the practical reasoning argumentation scheme (introduced in the next section). If any agent i has any privacy concerns over some propositions that it uses to represent its state, then i can choose to not make these propositions public. But privacy concerns should be balanced with knowledge accuracy concerns, because the fewer propositions communicated by i, the more likely other agents will believe they are in different states to i, yet will not be able to form $\mathcal{B}$-arguments to indicate this, because i did not reveal enough information to trigger a $\mathcal{B}$-argument construction.

The following example shows how the state comparison definition works, when every agent reveals all their propositions:

Example 28: *Consider the following propositional representations of the world and the current states for agents i and j: $\Phi^i = \{p, q, r, t\}$; $\Phi^j = \{p, r, v\}$; $q_i = [p, \neg q, \neg r, t]$; and $q_j = [p, \neg r, v]$. Given the state comparison definition $\pi(q_i) \cap \Phi^j = \pi(q_j) \cap \Phi^i$, the substitution $\{p, t\} \cap \{p, r, v\} = \{p, v\} \cap \{p, q, r, t\}$ gives $\{p\} = \{p\}$. Therefore, as $q_i \approx q_j$, the conclusion is that there is no evidence to suggest the states are necessarily different.*

3.2 The Practical Reasoning Model

A Value-based Alternating Transition System (VATS) [9] is an appropriate representation for reasoning over a qualitative coalition formation process as the underlying AATS [135] was initially used to show how (already formed) coalitions, performing joint-actions, change the state of the world. The VATS will be used in this chapter to help the agents of the system find acceptable coalitions, given their beliefs and social-value preference order.

To help the agents form coalitions, the following new function is added to the VATS representation of agent i, given the full set of agents of the coalitional game (denoted N):

- $\zeta^i : N \times Ac^i \rightarrow \{\top, \bot\}$ is a representation function, which defines whether agent i believes if agent $j \in N$ will perform the given action (therefore $\top$ is returned) or not (therefore $\bot$ is returned).

The Practical Reasoning Argumentation Scheme for Coalition Formation (PRASCF), modified from the AS1 scheme[3] of [21], and presented in [98] is:

PRASCF*: In the current circumstances R, joint action J should be performed, by coalition C, which will result in the new circumstances S, which will realise goal G and promote/demote the social-value V.*

Circumstances *R* and *S* are represented as tuples of propositions. Joint action *J* is a tuple μ of single actions, denoted $\mu = \langle ac_m, ..., ac_n \rangle$. The coalitions are assumed to be able to coordinate these joint-actions themselves[4]. Coalition $C = \{i, ..., j\}$ is a set of agents where a tuple ξ matches agents to single actions, denoted $\xi = \langle (i, ac_p), ..., (j, ac_q) \rangle$. The intended interpretation for ξ is that if this coalition is accepted, then each agent in *C* will perform the single action that it is paired to in ξ. An agent can put forth PRASCF arguments, to persuade others that a coalition should be formed to undertake a joint-action, by instantiating this scheme. The goal to be achieved, and the social-value to be promoted, provides the justification for the coalition to undertake the joint-action.

Due to the nature of decentralised environments, the following conditions are enforced when an agent i is creating a new instantiation of the argumentation scheme:

- The ξ tuple cannot have been previously proposed in an argumentation scheme and rejected unless the interpretation of the current state has changed, through an agent *talking by doing* (i.e. an agent performing an action or a coalition performing a joint-action) or through an agent *doing by talking* (i.e. an agent asserting a new $\mathcal{B}$-arguments that could influence another agent's opinion of the current state). This stops instantiations of argumentation schemes being repeated indefinitely, yet allows for situations where new information is uncovered that indicates the current reasoning was flawed.

- Every agent of the coalition must be an agent in the ξ tuple. This condition requires each member of the coalition not to be a *dummy player* [136], i.e. each agent of the coalition must contribute to the coalition it is in.

- Each agent must occur once and only once in ξ. This condition ensures that an agent is not given multiple single actions to perform at the same time.

- The coalition argued for/against does not have to be complete, i.e. the ξ tuple does not have to be fully instantiated initially. This condition is due to an agent i potentially having only partial knowledge of the system.

A formally instantiated version of the PRASCF scheme is defined as:

[3] The AS1 scheme is described in Section 2.6.3.

[4] With this assumption the joint actions operate in much the same manner as actions in the AS1 schema (the only different is that a certain number of agents are required to perform a joint-action).

Definition 55: *A PRASCF is denoted* $\mathcal{C} = \langle q_x, \mu, \xi, q_y, p, v, s \rangle$ *where* q_x *is the current state,* μ *is the joint action,* ξ *is the tuple that matches agents to a joint action,* q_y *is the new state,* p *is the goal in the new state,* v *is the social-value associated with this state transition and* s *(where* $s = \{+, -, =\}$*) is the sign indicating whether the value is promoted/demoted/not affected respectively.*

In the PRASCF scheme, the coalition C is implicit in the ξ tuple, found by: $\forall \langle i, ac \rangle \in \xi$ then $i \in C$. A $\mathcal{C}$-argument will represent a *proposal* if there exist single actions of μ not matched to an agent in ξ. This situation is a proposal because the coalition does not yet have a sufficient number of members to carry out the joint-action and so requires others to complete the coalition. A $\mathcal{C}$-argument will represent an *assertion* if all single actions of μ have been assigned to an agent, because the coalition now has enough members to carry out the joint-action and is therefore ready to form.

Additionally agents will be able to communicate critical questions attacking any proposal, assertion or another critical question. A formalised critical question is instantiated as a modified version of a $\mathcal{C}$-argument intended to reflect the question it represents, in a logical form. These critical questions are modifications of the critical questions of the AS1 scheme, presented in Section 2.6.3.

3.3 Using Dialogues for Inquiry and Persuasion to Form Coalitions

The new dialogues presented in this chapter assume a decentralised multi-agent system, where the agents collaborate in coalitions to achieve mutual goals that promote social-values the agents are concerned with. To find the best joint-action that achieves a dialogue initiator's goal, the agents enter a *coalitional persuasion over action* (C-$pAct$) dialogue, which provides the agents with an opportunity to persuade others by putting forward arguments for/against coalitions to form to undertake joint-actions.

However, before these arguments can be communicated, each agent i may want to inquire over some propositions to find their Boolean value so that the agent's initial state can be found. Once this has occurred, the correct $\mathcal{C}$-arguments for the current system state can be uttered. If an agent i wants to find the Boolean value of a proposition, then it will open a *coalitional inquiry* (C-inq) dialogue with other agents in the system. The result of each C-inq dialogue is for each agent participating in the C-inq dialogue to find a single point belief on the Boolean value for all of agent i's propositions that the C-inq dialogue was opened for. As stated in [34], the single-point belief assumption is helpful because: probabilistic beliefs may not be available; single point beliefs are easier to form; reasoning over single point beliefs is computationally less complex; and single point beliefs are a natural assumption in many real-world situations.

In the C-$pAct$ and C-inq dialogues, six different moves are allowed for all of the agents' needs, where the agents broadcast these moves to all the others when required. These moves are as follows:

- *Open*: The open move allows an agent i to start a new C-$pAct$ or C-inq dialogue.

Move	Format
$open$	$\langle i, open, dialogue(\theta, \gamma)\rangle$
$join$	$\langle i, join, dialogue(\theta, \gamma)\rangle$
$propose$	$\langle i, propose, \Upsilon, dialogue(\theta, \gamma)\rangle$
$assert$	$\langle i, assert, \Upsilon, dialogue(\theta, \gamma)\rangle$
$close$	$\langle i, close, dialogue(\theta, \gamma)\rangle$
$leave$	$\langle i, leave, dialogue(\theta, \gamma)\rangle$

TABLE 3.1: The format for moves used in this dialogue, where N represents the full set of agents of the coalitional game, i is the agent making the move and $i \in N$. Then either $\theta = C\text{-}pAct$ and γ is a proposition (representing the dialogue goal), or $\theta = C\text{-}inq$ and γ is a set of propositions (that i is inquiring over). The set Υ is either a set of $\mathcal{C}$-arguments and critical questions (if $\theta = C\text{-}pAct$) or Υ is a set of $\mathcal{B}$-arguments, defeasible facts and defeasible rules (if $\theta = C\text{-}inq$).

- *Join*: The join move allows an agent i to join a dialogue that another agent j has created.
- *Propose*: The propose move allows an agent i to argue for/against the formation of a coalition to undertake a joint-action, where all elements of the joint-action *are not* assigned to agents.
- *Assert*: The assert move allows an agent i to either: argue for/against the formation of a coalition to undertake a joint-action, where all elements of the joint-action *are* assigned to agents (if the dialogue type is $C\text{-}pAct$); or communicate defeasible facts, defeasible rules and $\mathcal{B}$-arguments (if the dialogue is $C\text{-}inq$).
- *Close*: The close move allows an agent i to tell the other agents that it currently has no other information to add to the dialogue. If all agents of the dialogue perform a close move without any new arguments being communicated then the dialogue is over.
- *Leave*: The leave move allows an agent i to leave its current dialogue.

The format of the moves that the agents can perform is presented in Table 3.1. The set of all moves meeting the format defined in Table 3.1 is denoted $\mathcal{M}$. Within this chapter, a dialogue $\mathcal{D}_r^t$, is a sequence of moves $m_r, \ldots, m_t$ where $r, \ldots, t \in \mathbb{N}$ represents the time-point at which each move was made, with r being the starting point of the dialogue and t the end point. The collection of all on-going dialogues of the system are referred to as Ω.

Each agent i's proposals and assertions are stored in its public commitment store ($CoSt_i$) that grows monotonically over time:

Definition 56: **Commitment store update.** *For a dialogue with participants N, denoted $\mathcal{D}_r^t$, a commitment store of agent i at time-point t is denoted $CoSt_i^t$ and updated as follows:*

$$CoSt_i^t = \begin{cases} \emptyset & \textit{iff } t = 0, \\ CoSt_i^{t-1} \cup \Upsilon & \textit{iff } m_t = \langle i, proposal, \Upsilon, dialogue(\theta, \gamma)\rangle, \\ CoSt_i^{t-1} \cup \Upsilon & \textit{iff } m_t = \langle i, assert, \Upsilon, dialogue(\theta, \gamma)\rangle, \\ CoSt_i^{t-1} & \textit{otherwise.} \end{cases}$$

As the dialogues of this chapter are decentralised, the public commitment store of each agent is assumed to be maintained by every agent internally. Each agent i's public commitment store is empty at time $t = 0$. For any time point $t > 0$, any instantiated arguments that agent i communicates as proposals or assertions are added to the public commitment store of agent i, so that all agents can reason over all arguments communicated:

Definition 57: *The union of all the commitment stores at timepoint t is defined as:*

$$CoSt^t = \bigcup_{\forall i \in N} CoSt_i^t.$$

Even thought the union of all commitment stores includes arguments that have been publicly asserted, agents may still hold their own private information, which will remain private until used in an argument that is asserted publically.

The following functions operate over and within a dialogue:

- $\mathsf{Initiator}(\mathcal{D}_r^t)$ returns the agent i who opened the dialogue $\mathcal{D}_r^t$.
- $\mathsf{Type}(\mathcal{D}_r^t)$ returns the type of the dialogue $\mathcal{D}_r^t$ (i.e. $C\text{-}pAct$ or $C\text{-}inq$).
- $\mathsf{Ags}(\mathcal{D}_r^t)$ returns the set of agents in the dialogue D_r^t.
- $\mathsf{Topic}(\mathcal{D}_r^t)$ returns the goal the agents are trying to achieve **iff** the dialogue type is $C\text{-}pAct$, i.e. $\mathsf{Type}(D_r^t) = C\text{-}pAct$. $\mathsf{Topic}(\mathrm{D}_r^t)$ returns the set of propositions which the agents are jointly trying to find the truth value of **iff** the dialogue type is $C\text{-}Inq$, i.e. $\mathsf{Type}(D_r^t) = C\text{-}inq$.
- $\mathsf{DiaLive}(dialogue(\theta, \gamma))$ returns the ongoing dialogue D_r^t **if** it is of the type θ and its topic matches with γ. I.e. D_r^t is returned if there $\exists \mathcal{D}_r^t \in \Omega$ where $\mathsf{Type}(\mathcal{D}_r^t) = \theta$ and $\mathsf{Topic}(\mathcal{D}_r^t) = \gamma$ **else** the empty set $\emptyset$ is returned.
- $\mathsf{Valid}(A)$ returns $\top$ if A is a defeasible fact, defeasible rule, $\mathcal{B}$-argument *or* A is an argument for a coalition to form and each element of the joint-action is given to one and only one agent. Otherwise $\bot$ is returned.
- $\mathsf{AgAction}(j, \xi)$ returns the single action ac agent j is assigned to perform should an argument including ξ be accepted. The empty set $\emptyset$ is returned if j is not assigned an action in ξ.
- $\mathsf{StartState}(A)$ returns q_x for any argument A where A is a $\mathcal{C}$-argument or critical question. Returns $\emptyset$ if there is no q_x in A.
- $\mathsf{Action}(A)$ returns μ for any argument A where A is a $\mathcal{C}$-argument or critical question. Returns $\emptyset$ if there is no μ in A.
- $\mathsf{CoalAct}(A)$ returns ξ for any argument A where A is a $\mathcal{C}$-argument or critical question. Returns $\emptyset$ if there is no ξ in A.
- $\mathsf{Coalition}(A)$ returns C for any argument A where A is a $\mathcal{C}$-argument or critical question, and for all $\langle ac, j \rangle \in \mathsf{CoalAct}(A)$ then $j \in C$. Returns $\emptyset$ if there is no ξ in A.

- $\mathsf{Coalition}(\xi)$ returns C for any tuple ξ where for all $\langle ac, j\rangle \in \xi$ then $j \in C$. Returns $\emptyset$ if there is no ξ in A.
- $\mathsf{EndState}(A)$ returns q_y for any argument A where A is a $\mathcal{C}$-argument or critical question. Returns $\emptyset$ if there is no q_y in A.
- $\mathsf{Goal}(A)$ returns p for any argument A where A is a $\mathcal{C}$-argument or critical question. Returns $\emptyset$ if there is no p in A.
- $\mathsf{Value}(A)$ returns v for any argument A where A is a $\mathcal{C}$-argument or critical question. Returns $\emptyset$ if there is no v in A.
- $\mathsf{Polarity}(A)$ returns s for any argument A where A is a $\mathcal{C}$-argument or critical question. Returns $\emptyset$ if there is no s in A.

Dialogues commence when an agent i desires to move to another state (defined by its VATS). This agent should perform the *open* move to open either: a $C\text{-}inq$ dialogue to inquire over some or all of i's propositions used to represent the world (if a $C\text{-}inq$ dialogue to inquire over the same propositions is not currently on-going); or a $C\text{-}pAct$ dialogue if i is ready to communicate $\mathcal{C}$-arguments, to achieve some goal to promote some social-value (if a $C\text{-}pAct$ dialogue to achieve the same proposition is not currently on-going). Then the agent should use the relevant protocol for the dialogue type, either the $C\text{-}inq$ protocol of Section 3.4.1 or the $C\text{-}pAct$ protocol of Section 3.4.2, to find out what possible moves are available. The pre and post conditions of the moves are listed in Table 3.2.

Additionally, each agent i should check the Ω list of currently ongoing dialogues to see if there is any that i finds relevant to join. Agent i should find a $C\text{-}inq$ dialogue relevant if it is discussing one (or more) of the propositions that i uses to represent the world in Φ^i. Agent i should find a $C\text{-}pAct$ dialogue relevant if there is a possibility that i could persuade some agents of the dialogue to join a coalition with i to achieve a goal that will promote some social-value that i desires.

There is only one way for a dialogue to be terminated *successfully*. This is when all agents have consecutively made a close move one after another without a different move separating them, as this ensures that the dialogue does not terminate until none of the agents have anything more they want to add. Alternatively a dialogue could be terminated *unsuccessfully*, which occurs when the dialogue has not been completed successfully *and* all the agents have left the dialogue.

Once the *C-pAct* dialogue has terminated successfully, each agent evaluates the arguments to determine the acceptable set. To evaluate the arguments during the dialogue would be redundant computation because the *C-pAct* protocol is designed to uncover all relevant arguments for the current dialogue. An argument is relevant in a *C-pAct* dialogue if it is a critical question to another argument in the dialogue *or* it is a $\mathcal{C}$-argument for achieving the dialogue goal p. It is assumed that the agents think rationally and want the best coalitions to form given all the relevant arguments. Only when all these arguments have been uncovered, do the agents then need to find the acceptable set of arguments and thus the acceptable coalitions.

Move	Format	Pre-conditions	Post-conditions
$open$	$\langle i, open, dialogue(\theta, \gamma)\rangle$	$\mathsf{DiaLive}(dialogue(\theta, \gamma)) = \emptyset$	Dialogue commenced, i.e. $\mathsf{DiaLive}(dialogue(\theta, \gamma)) = dialogue(\theta, \gamma)$. Also $i \in \mathsf{Ags}(dialogue(\theta, \gamma))$.
$join$	$\langle i, join, dialogue(\theta, \gamma)\rangle$	$\mathsf{DiaLive}(dialogue(\theta, \gamma)) = \mathcal{D}$ and $i \notin \mathsf{Ags}(\mathcal{D})$	$i \in \mathsf{Ags}(\mathcal{D})$
$propose$	$\langle i, propose, \Upsilon, dialogue(\theta, \gamma)\rangle$	$\mathsf{DiaLive}(dialogue(\theta, \gamma)) = \mathcal{D}$, $i \in \mathsf{Ags}(\mathcal{D})$ and $\forall \mathcal{A} \in \Upsilon$, $\mathsf{Valid}(\mathcal{A}) = \bot$.	$CoSt_i^t = CoSt_i^{t-1} \cup \Upsilon$.
$assert$	$\langle i, assert, \Upsilon, dialogue(\theta, \gamma)\rangle$	$\mathsf{DiaLive}(dialogue(\theta, \gamma)) = \mathcal{D}$, $i \in \mathsf{Ags}(\mathcal{D})$ and $\forall \mathcal{A} \in \Upsilon$, $\mathsf{Valid}(\mathcal{A}) = \top$.	$CoSt_i^t = CoSt_i^{t-1} \cup \Upsilon$.
$close$	$\langle i, close, dialogue(\theta, \gamma)\rangle$	$\mathsf{DiaLive}(dialogue(\theta, \gamma)) = \mathcal{D}$ and $i \in \mathsf{Ags}(\mathcal{D})$	$\mathsf{DiaLive}(dialogue(\theta, \gamma)) = \emptyset$ **iff** all agents have performed a close move in a row (without another move inbetween).
$leave$	$\langle i, leave, dialogue(\theta, \gamma)\rangle$	$\mathsf{DiaLive}(dialogue(\theta, \gamma)) = \mathcal{D}$ and $i \in \mathsf{Ags}(\mathcal{D})$	$\mathsf{DiaLive}(dialogue(\theta, \gamma)) = \emptyset$ **iff** $\mathsf{Ags}(\mathcal{D}) = \emptyset$. Otherwise $i \notin \mathsf{Ags}(\mathcal{D})$.

TABLE 3.2: The moves available to the agents

The agents find the acceptable set of asserted arguments using a *Value-Based Argumentation Framework* (VAF) [19]; the details of this evaluation and how the agents find acceptable coalitions to form given their different social-value orders is described in Section 3.5. For any two coalitions C and C' that are recommended to form, then $C \cap C' = \emptyset$ is assumed. This assumption fits in with the traditional understanding of coalitional games. Additionally in the context of this chapter, it makes sense for an agent to not be assigned to perform different actions at the same time (which would occur if this assumption did not hold).

Finally, the memory requirements for each agent in inquiry and persuasion dialogues presented in this Chapter, is according to the size of storing: (i) their VATS; (ii) their associated beliefs; (iii) the utterances in the dialogues; (iv) the attack relations between the argumentation schemes; and (v) all the agents' preference orderings. The size of (i) to (v) can vary greatly given different domains and different agents.

3.3.1 Extending the Formalisation of Critical Questions

The dialogue system set out in [21] handled only three of the possible seventeen critical questions (CQs) associated with the AS1 schema that the PRASCF schema is based on. These three CQs were: '*are there alternative ways of realising the same consequences to promote some other social-value*[5]*?*' (CQ6); '*does doing the action have a side effect which demotes some*

[5]Where *social-value* is a social interest an agent is concerned over, as described in [19].

other social-value?' (CQ9); and '*does doing the action promote some other social-value?*' (CQ10). In this chapter the dialogue system is extended by not only modifying the AS1 schema so that coalitions can form, but by also increasing the CQs that the agents can use. The CQs formalised in this section that follow the PRASCF definition are also $\mathcal{C}$-arguments. All CQs can be communicated by any agent to challenge a practical reasoning argument of any other agent (including itself).

Unlike the previous work of [10], this chapter's CQs do not argue over the starting state. This is to add flexibility for a coalition to form even if the members of that coalition think they are in different starting states. This can occur after the $C\text{-}inq$ dialogue due to the possible different methods used to reason over the asserted $\mathcal{B}$-arguments. Yet believing they are in different starting states should not be a hindrance to coalition formation, as there maybe joint-actions that all of the agents of the coalition believe will be successful from all of their believed starting states. For this reason, CQ1 and CQ12 of [10] are not in the dialogue system detailed in this chapter, because they both require specific agreement on the starting state.

Within the definitions below, Arguments Over Actions (AOAs) refers to $\mathcal{C}$-arguments and CQs attacking $\mathcal{C}$-arguments. The critical questions that allow a new coalition to be suggested are CQ5, CQ6, CQ7 and CQ11, because these critical questions allow alternative joint-actions to be proposed/asserted compared to the ones already existing in the dialogue. Accompanying the definitions are figures that illustrate a situation where each CQ could be posed. The same two agents appear in all of the figures, with the assumption being they are arguing over whether the coalition $\{1, 2\}$ should form, and that agent 1 has put forward a $\mathcal{C}$-argument for $\{1, 2\}$ to form. All of the figures have each agent's initial state on the left-hand side. Some of the figures for a CQ may not include explicit joint-actions or social-values because they do not occur in the definition of that critical question.

Definition 58: **A cq2-argument** *answers the question* 'Does the joint-action have the stated consequences?'. *It is constructed from VATS*^{i} *and denoted* $\langle \mu, \xi, q_y \rangle$ *s.t.* $\mu = \langle ac_p, ..., ac_q \rangle$; $\forall ac_k \in \mu$, $ac_k \in Ac^i$; $i \in \mathsf{Coalition}(\xi)$; $\tau^i(q_0^i, \mu) = q_y$. *It challenges any AOA including* μ', ξ', q'_y *iff* $\mu = \mu'$, $\xi = \xi'$, $q_y \neq q'_y$.

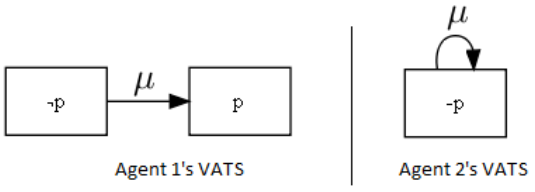

FIGURE 3.1: Illustration of a cq2-argument (Definition 58). Agent 2 uses a cq2-argument as it thinks that the joint-action $\mu = \mu'$ will not bring about the consequences $[p]$ from $[\neg p]$.

Definition 59: **A cq3-argument** *answers the question* 'Assuming the joint-action has the stated consequences, will the joint-action bring about the desired goal?'. *It is constructed from VATS*^{i} *and denoted* $\langle \mu, \xi, q_y, \emptyset \rangle$ *s.t.* $\mu = \langle ac_p, ..., ac_q \rangle$; $\forall ac_k \in \mu$, $ac_k \in Ac^i$; $i \in \mathsf{Coalition}(\xi)$; $\tau^i(q_x, \mu) = q_y$. *It challenges any AOA that includes* μ', ξ', q'_y, p' *iff* $\mu = \mu'$, $\xi = \xi'$, $q_y \approx q'_y$, $p' \notin (q_y)$.

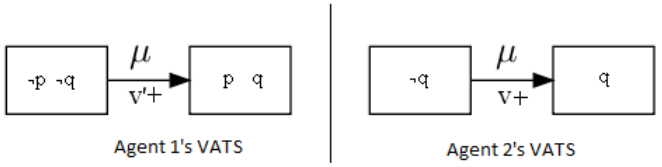

FIGURE 3.2: Illustration of a cq3-argument (Definition 59). Agent 2 uses a cq3-argument as it thinks that the joint-action $\mu = \mu'$ used at $[\neg q] \approx [\neg p, \neg q]$ will not bring about a state that includes the goal p because $\tau^2([\neg q], \mu) = [q]$.

Definition 60: **A cq4-argument** *answers the question* 'Assuming the joint-action has the stated consequences, will the joint-action bring about the desired social-value?'. *It is constructed from VATS^i and denoted* $\langle \mu, \xi, q_y, v, s \rangle$ *s.t.* $\mu = \langle ac_p, ..., ac_q \rangle$; $\forall ac_k \in \mu$, $ac_k \in Ac^i$; $i \in \mathsf{Coalition}(\xi)$; $\tau^i(q_0^i, \mu) = q_y$; $v \in Av^i$; $\delta^i(q_0^i, q_y, v) = s$; $s \in \{=, -\}$. *It challenges an AOA that includes* μ', ξ', q'_y, v', s' *iff* $\mu = \mu'$, $\xi = \xi'$, $q_y \approx q'_y$, $v = v'$, $s' = +$.

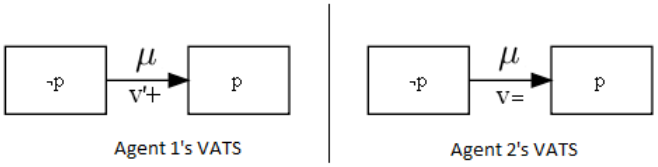

FIGURE 3.3: Illustration of a cq4-argument (Definition 60). Agent 2 uses a cq4-argument as it thinks that the joint-action $\mu = \mu'$ used at state $[\neg p] \approx [\neg p]$ to achieve state $[p] \approx [p]$ does not promote the social-value $v = v'$, i.e. $\delta^2([\neg p], [p], v) \neq +$.

Definition 61: **A cq5-argument** *answers the question* 'Are there alternative ways of realising the same consequences?'. *It is constructed from VATS^i and denoted* $\langle \mu, \xi, q_y \rangle$ *s.t.* $\mu = \langle ac_p, ..., ac_q \rangle$; $\forall ac_k \in \mu$, $ac_k \in Ac^i$; $i \in \mathsf{Coalition}(\xi)$; $\forall j \in \mathsf{Coalition}(\xi)$ *then* $\zeta^i(j, \mathsf{AgAction}(j, \xi)) = \top$; $\tau^i(q_0^i, \mu) = q_y$. *It challenges an AOA that includes* μ', q'_y *iff* $\mu \neq \mu'$, $q_y \approx q'_y$.

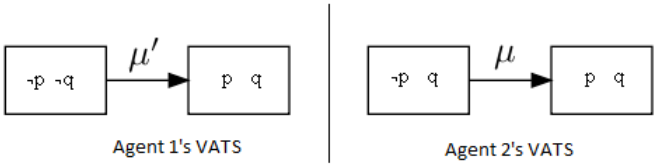

FIGURE 3.4: Illustration of a cq5-argument (Definition 61). Agent 2 uses a cq5-argument as it thinks that there is another joint-action $\mu \neq \mu'$ that can be used from its current state $[\neg p, q]$ to achieve state $[p, q] \approx [p, q]$.

Definition 62: **A cq6-argument** *answers the question* 'Are there alternative ways of realising the same consequences to promote some other social-value?'. *It is constructed from VATS^i and denoted* $\langle \mu, \xi, q_y, v, + \rangle$ *s.t.* $\mu = \langle ac_p, ..., ac_q \rangle$; $\forall ac_k \in \mu$, $ac_k \in Ac^i$; $i \in \mathsf{Coalition}(\xi)$; $\forall j \in \mathsf{Coalition}(\xi)$ *then* $\zeta^i(j, \mathsf{AgAction}(j, \xi)) = \top$; $\tau^i(q_0^i, \mu) = q_y$; $v \in Av^i$; $\delta^i(q_0^i, q_y, v) = +$. *It challenges an AOA that includes* μ', q'_y v', s' *iff* $\mu \neq \mu$, $q_y \approx q'_y$, $v \neq v'$, $s' = +$.

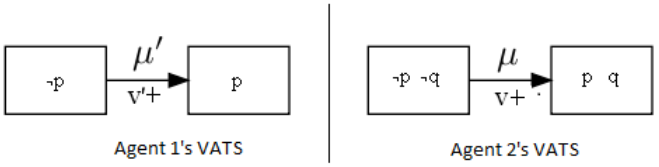

FIGURE 3.5: Illustration of cq6-argument (Definition 62). Agent 2 uses a cq6-argument as it thinks that there is another joint-action $\mu \neq \mu'$ that can be used from its current state $[\neg p, \neg q]$ to achieve state $[p, q] \approx [p]$ while promoting a different value $v \neq v'$.

Definition 63: **A cq7-argument** *answers the question* 'Are there alternative ways of promoting the same social-value?'. *It is constructed from VATS*^{i} *and denoted* $\langle \mu, \xi, q_y, v, + \rangle$ *s.t.* $\mu = \langle ac_p, ..., ac_q \rangle$; $\forall ac_k \in \mu$, $ac_k \in Ac^i$; $i \in \mathsf{Coalition}(\xi)$; $\forall j \in \mathsf{Coalition}(\xi)$ *then* $\zeta^i(j, \mathsf{AgAction}(j, \xi)) = \top$; $\tau^i(q_0^i, \mu) = q_y$; $v \in Av^i$; $\delta(q_0^i, q_y, v) = +$. *It challenges an AOA that includes* μ', q'_y, v', s' *iff* $\mu \neq \mu'$, $q_y \neq q'_y$, $v = v'$, $s' = +$.

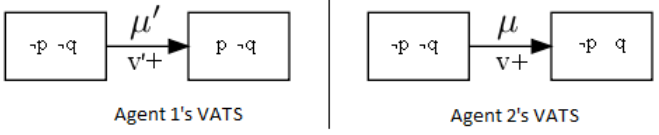

FIGURE 3.6: Illustration of cq7-argument (Definition 63). Agent 2 uses a cq7-argument as it thinks that there is another joint-action $\mu \neq \mu'$ that can be used from its current state $[\neg p, \neg q]$ to achieve state $[\neg p, q] \neq [p, \neg q]$ while achieving the social-value $v = v'$.

Definition 64: **A cq8-argument** *answers the question* 'Does doing the joint-action have a side effect which demotes the social-value?'. *It is constructed from VATS*^{i} *and denoted* $\langle \mu, \xi, q_y, v, - \rangle$ *s.t.* $\mu = \langle ac_p, ..., ac_q \rangle$; $\forall ac_k \in \mu$, $ac_k \in Ac^i$; $i \in \mathsf{Coalition}(\xi)$; $\tau^i(q_0^i, \mu) = q_y$; $v \in Av^i$; $\delta^i(q_0^i, q_y, v) = -$. *It challenges an AOA that includes* μ', ξ', q'_y, v', s' *iff* $\mu = \mu'$, $\xi = \xi'$, $q_y \approx q'_y$, $v = v'$, $s' = +$.

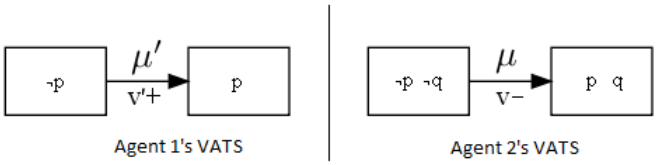

FIGURE 3.7: Illustration of a cq8-argument (Definition 64). Agent 2 uses a cq8-argument as it thinks that the joint-action $\mu = \mu'$ from its current state $[\neg p, \neg q]$ will produce the side effect of q in state $[p, q] \approx [p]$, which agent 1 did not recognise, that will demote the social-value $v = v'$.

Definition 65: *A* **cq9-argument** *answers the question* 'Does doing the joint-action have a side effect which demotes some other social-value?'. *It is constructed from VATS*^{i} *and denoted* $\langle \mu, \xi, q_y, v, - \rangle$ *s.t.:* $\mu = \langle ac_p, ..., ac_q \rangle$; $\forall ac_k \in \mu$, $ac_k \in Ac^i$; $i \in \mathsf{Coalition}(\xi)$; $\tau^i(q_0^i, \mu) = q_y$; $v \in Av^i$; $\delta^i(q_0^i, q_y, v) = -$. *It challenges an AOA that includes* μ', ξ', q'_y *iff* $\mu = \mu'$, $\xi = \xi'$, $q_y \approx q'_y$, $v \neq v'$, $s' = +$.

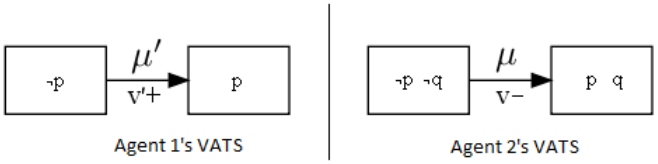

FIGURE 3.8: Illustration of a cq9-argument (Definition 65). Agent 2 uses a cq9-argument as it thinks that the joint-action $\mu = \mu'$ from the its current state $[\neg p, \neg q]$ will produce the side effect of q in state $[p, q] \approx [p]$, which agent 1 did not recognise, that will demote the social-value $v \neq v'$.

Definition 66: *A* **cq10-argument** *answers the question* 'Does doing the joint-action promote some other social-value?'. *It is constructed from VATS^i and denoted* $\langle \mu, \xi, q_y, v, + \rangle$ *s.t.:* $\mu = \langle ac_p, ..., ac_q \rangle$; $\forall ac_k \in \mu$, $ac_k \in Ac^i$; $i \in \mathsf{Coalition}(\xi)$; $\tau^i(q_0^i, \mu) = q_y$; $v \in Av^i$; $\delta^i(q_0^i, q_y, v) = +$. *It challenges an AOA that includes* μ', ξ', q'_y v', s' *iff* $\mu = \mu'$, $\xi = \xi'$, $q_y \approx q'_y$, $v \neq v'$, $s' = +$.

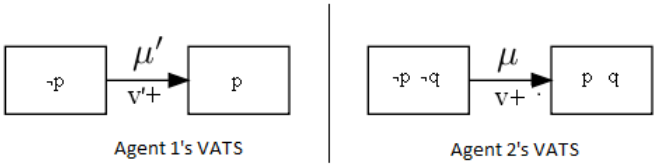

FIGURE 3.9: Illustration of cq10-argument (Definition 66). Agent 2 uses a cq10-argument as it thinks that the joint-action $\mu = \mu'$ from the state $[\neg p, \neg q]$ will achieve state $[p, q] \approx [p]$ and will promote the social-value $v \neq v'$.

Definition 67: **A cq11-argument** *answers the question* 'Does doing the joint-action preclude some other joint-action which would promote some other social-value?'. *It is constructed from VATS^i and denoted* $\langle \mu, \xi, q_y, v, + \rangle$ *s.t.* $\mu = \langle ac_p, ..., ac_q \rangle$; $\forall ac_k \in \mu$, $ac_k \in Ac^i$; $i \in \mathsf{Coalition}(\xi)$; $\forall j \in \mathsf{Coalition}(\xi)$ *then* $\zeta^i(j, \mathsf{AgAction}(j, \xi)) = \top$; $\tau^i(q_0^i, \mu) = q_y$; $v \in Av^i$; $\delta(q_0^i, q_y, v) = +$. *It challenges an AOA that includes* μ', q'_y, v', s' *iff* $\mu \neq \mu'$, $q_y \neq q'_y$, $v \neq v'$, $s' = +$.

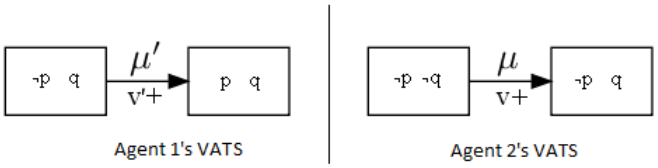

FIGURE 3.10: Illustration of a cq11-argument (Definition 67). Agent 2 uses a cq11-argument as it thinks that performing the joint-action μ' will preclude μ being used (where $\mu \neq \mu'$) to achieve a different state $[\neg p, q] \neq [p, q]$, which will promote a social-value $v \neq v'$.

Definition 68: **A cq13-argument** *answers the question* 'Is the joint-action possible?'. *It is constructed from VATS^i and denoted* $\langle \neg \mu \rangle$ *s.t.* $\exists ac \in \mu$ *where* $ac \notin Ac^i$. *It challenges any AOA that includes* μ' *iff* $\mu = \mu'$.

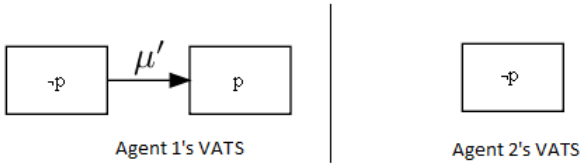

FIGURE 3.11: Illustration of a cq13-argument (Definition 68). Agent 2 uses a cq13-argument as it does not recognise that joint-action $\mu = \mu'$ is possible.

Definition 69: **A cq14-argument** *answers the question* 'Are the consequences as described possible?'. *It is constructed from VATS*^{i} *and denoted* $\langle \neg q_y \rangle$ *s.t.* $\forall q \in q_y$ *then* $q \in \Phi^i$*; and* $\neg \exists q_z \notin Q^i$ *where* $q_z \approx q_y$*. It challenges any AOA that includes* q'_y *iff* $q_y \approx q'_y$*.*

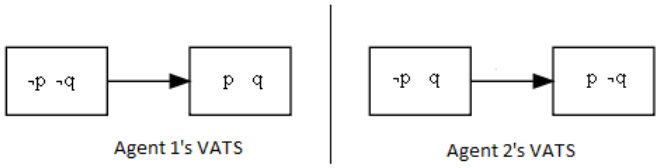

FIGURE 3.12: Illustration of a cq14-argument (Definition 69). Agent 2 uses a cq14-argument as it shares the propositions p and q with agent 1, yet thinks agent 1's end state $[p, q]$ can never be achieved.

Definition 70: **A cq15-argument** *answers the question* 'Can the desired goal be realised?'. *It is constructed from VATS*^{i} *and denoted* $\langle \neg p \rangle$ *s.t.* $p \in \Phi^i$*;* $(\forall q \in Q^i)(p \notin \pi(q))$*. It challenges an AOA that includes* p' *iff* $p = p'$*.*

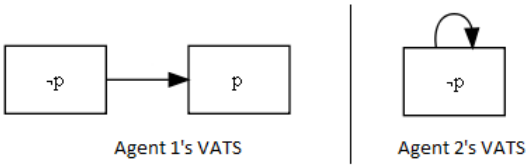

FIGURE 3.13: Illustration of a cq15-argument (Definition 70). Agent 2 uses a cq15-argument as even though it recognises p as a valid proposition, it believes that p cannot be achieved in any state.

Definition 71: **A cq16-argument** *answers the question* 'Is the social-value indeed a legitimate social-value?'. *It is constructed from VATS*^{i} *and denoted* $\langle \neg v \rangle$ *s.t.* $v \notin Av^i$*. It challenges an AOA that includes* v' *iff* $v = v'$*.*

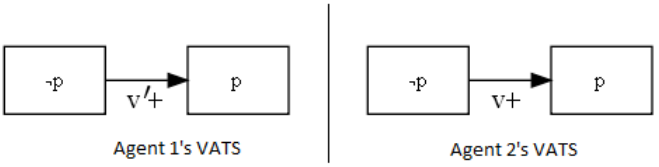

FIGURE 3.14: Illustration of a cq16-Argument (Definition 71). Agent 2 uses a cq16-argument as it does not recognise social-value $v = v'$ as a valid social-value.

Definition 72: **A cq17-argument** *answers the question* 'Is another agent guaranteed to execute its part of the desired joint-action?'. *It is constructed from VATS*^{i} *and denoted* $\langle \mu, \neg \xi \rangle$ *s.t.* $\forall ac_k \in \mu$*,* $ac_k \in Ac^i$*;* $\exists j \in \xi$ *where* $\zeta^i(j, \mathsf{AgAction}(j, \xi)) = \bot$*. It challenges an AOA that includes* μ' *iff* $\mu = \mu'$*.*

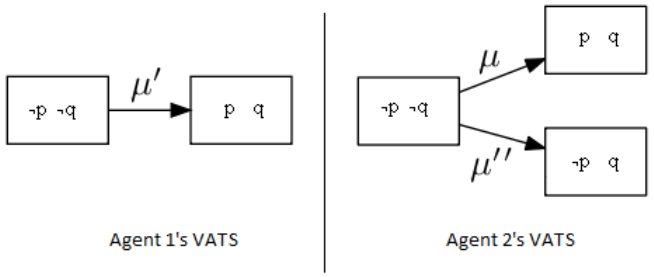

FIGURE 3.15: Illustration of a cq17-Argument (Definition 72). Agent 2 uses a cq17-argument as it has reason to doubt that the joint-action $\mu = \mu'$ will be executed because agent 2 doubts an agent j will perform its action in the coalition as described in ξ. Instead agent 2 expects the joint-action $\mu'' = \mu - \mathsf{AgAction}(j, \xi)$ to be performed. Where this doubt comes from is left undefined, be it from a trust issue or something else.

3.4 The Dialogue Protocols

Now that the formalism of all the possible arguments has been detailed, to find out what arguments are possible in the dialogue at any time, two protocols have been defined: the $C\text{-}inq$ protocol in Section 3.4.1 that finds all the possible defeasible facts, defeasible rules and $\mathcal{B}$-arguments that can be communicated in the $C\text{-}inq$ dialogue; and the $C\text{-}pAct$ protocol in Section 3.4.2 that finds all the possible $\mathcal{C}$-arguments and critical questions that can be communicated in the $C\text{-}pAct$ dialogue. It is then left to the agents themselves to autonomously decide which (if any) of the possible moves to use.

3.4.1 Defining the Inquiry Protocol

An inquiry protocol to find the state of the world, is now formally defined in Definition 73 (below) through the Ξ function, using the protocol format detailed in [21]. This Ξ protocol function returns the set of possible moves denoted $\wp^I(\mathcal{M})$ (from the set of all possible moves $\mathcal{M}$) that are legal for each agent (from N) that has joined the dialogue ($\mathcal{D}_r^t$) of the type $C\text{-}Inq$.

This protocol will not allow any proposition to become a claim of a $\mathcal{B}$-argument without supporting evidence. Supporting evidence takes the form of defeasible facts or a fully supported defeasible rule. A defeasible rule λ is fully supported when there is a defeasible derivation for the consequence of the rule that includes the rule and can be constructed from the union of all the commitment stores.

The protocol works by initially allowing each agent j to identify all of its *relevant beliefs* that can be asserted in the current dialogue that are not already present in the commitment store (see $\Xi_{\mathsf{a}}(D_1^t, j)$ in the following definition). A belief can be either a defeasible rule or a defeasible fact (see $\Xi_{\mathsf{a}}(D_1^t, j)$ part (1)). A defeasible fact is *relevant* to the current dialogue (see $\Xi_{\mathsf{a}}(D_1^t, j)$ part (2)) if: (2)(i) it is not already present in the commitment store and is present in agent j's belief base (i.e. Σ^j); and (2)(ii,a) it is an element of the dialogue topic or (2)(ii,b) it is an element of a defeasible rule in the combined commitment store of all of the agents. A defeasible rule is *relevant* to the current dialogue (see $\Xi_{\mathsf{a}}(D_1^t, j)$ part (3)) if: (3)(i) it is not already present in the commitment store and is present in agent j's belief base (i.e. Σ^j); and (3)(ii,a) its consequent

returns a defeasible fact that is an element of the dialogue topic or (3)(ii,b) its consequent returns a defeasible fact that is an element of another defeasible rule in the $CoSt$.

Next, the agent checks to see if any of its asserted beliefs (denoted ϕ) are now fully supported by a set of defeasible facts and defeasible rules (denoted Φ) in the commitment store, i.e $\Phi \subseteq CoSt$ (see $\Xi_{\mathsf{b}}(D_1^t, j)$ in the following definition). If some asserted beliefs are found to be fully supported, these beliefs can be asserted as $\mathcal{B}$-arguments in the form $\mathcal{B} = \langle \Phi, \phi \rangle$, as long as they have not been asserted already (i.e. $\mathcal{B} \notin CoSt$). Each agent j can only assert a $\mathcal{B}$-argument with claim ϕ if it asserted the belief that included ϕ (i.e. $\phi \in CoSt_i^t$). This is to eliminate multiple assertions of $\mathcal{B}$-arguments. Lastly if the agents cannot assert anything new then the moves that are returned are the 'close' move to be used when the agents want to remain in the dialogue to hear what the other agents have to say, or the 'leave' move to be used if the agent wants to leave the dialogue completely.

Definition 73: *The* **C-inq protocol** *for a C-inq dialogue is a function $\Xi : \mathcal{D} \times N \mapsto \wp^I(\mathcal{M})$. If $j \in N$ is the given agent and D_r^t is the given dialogue where $\mathsf{Ags}(D_r^t) = N$, $CoSt = \bigcup_{\forall k \in N} CoSt_k^t$, $\mathsf{Type}(D_r^t) = C\text{-}Inq$ and $1 \leq t$, then $\Xi(D_r^t, j)$ is*

$\Xi_{\mathsf{a}}(D_1^t, j) \cup \Xi_{\mathsf{b}}(D_1^t, j) \cup \{\langle j, close, dialogue(C\text{-}Inq, \mathsf{Topic}(D_r^t))\rangle\} \cup$
$\{\langle j, leave, dialogue(C\text{-}Inq, \mathsf{Topic}(D_r^t))\rangle\}$

where

$\Xi_{\mathsf{a}}(D_1^t, j) = \{\langle j, assert, \Psi\rangle |$
(1) $\Psi \neq \emptyset$ *where* Ψ *is a set of beliefs, and*
(2) $\forall \phi \in \Psi$ *where* ϕ *is a defeasible fact:*
(i) $\phi \notin CoSt, \phi \in \Sigma^j$ *, and*
either (ii,a) $\phi \in \mathsf{Topic}(D_r^t)$ *,*
or (ii,b) $\exists \lambda \in CoSt$ *s.t.* $\phi \in \mathsf{DefeasibleSection}(\lambda)$
(3) $\forall \lambda \in \Psi$ *where* λ *is a defeasible rule:*
(i) $\lambda \notin CoSt, \lambda \in \Sigma^j$ *, and*
either (ii,a) $\mathsf{DefeasibleProp}(\lambda) \in \mathsf{Topic}(D_r^t)$ *,*
or (ii,b) $\exists \lambda' \in CoSt$ *s.t.* $\mathsf{DefeasibleProp}(\lambda) \in \mathsf{DefeasibleSection}(\lambda')$

$\Xi_{\mathsf{b}}(D_1^t, j) = \{\langle j, assert, \Upsilon\rangle |$
(1) $\Upsilon \neq \emptyset, \Upsilon$ *is a set of $\mathcal{B}$-arguments , and*
(2) $\forall \mathcal{B} \in \Upsilon$*:* $\mathcal{B} = \langle \Phi, \phi \rangle$ *where:* $\Phi \subseteq \Psi \cup CoSt$*,* $\phi \in \Psi \cup CoSt_i^t$ *and* $\mathcal{B} \notin CoSt$

If every agent is using the C-inq protocol then a C-inq dialogue, just like the inquiry dialogue of [22], is guaranteed to terminate according to the following Theorem:

Theorem 3.1. *If each agent in a C-inq dialogue is using the C-inq protocol, then the dialogue is guaranteed to terminate (given each agent's beliefs is of finite size).*

Proof. Assume that all n agents in a C-inq dialogue game are using the C-inq protocol but the dialogue never terminates. For this to occur, the agents must be constantly asserting arguments of defeasible rules or defeasible facts (because n leave moves will terminate the dialogue as

there will be no agents left and n close moves will terminate the dialogue as no more agents have any more arguments to add).

Yet according to the $C\text{-}inq$ protocol condition (2)(i) and (3)(i), an argument for a defeasible fact or defeasible rule can only be asserted if it has not previously been asserted by any agent. As it is given that each agent's beliefs is of finite size, then in the worst case, each agent will assert all of the defeasible rules and defeasible facts that it is aware of. Once this has occurred, the only moves the agents will have left is the close or leave move and n leave moves or n close moves in a row will terminate the dialogue, which contradicts the assumption. □

Thus, as every belief of an agent could theoretically be communicated, the tractability of the $C\text{-}inq$ dialogues (like the inquiry dialogues in [22]) depends on the total number of possible beliefs of all the agents.

3.4.2 Extending the pAct Protocol

A *Persuasion for Action* Protocol for finding arguments for coalition formation is now formally defined in Definition 74 through the Π function, using the protocol format detailed in [21]. This protocol returns the set of possible moves denoted $\wp^P(\mathcal{M})$ (from the set of all possible moves $\mathcal{M}$) that are legal for each agent (from N) that has joined the dialogue ($\mathcal{D}_r^t$) of the type $C\text{-}pAct$.

The protocol works by initially allowing each agent i to assert or propose all its *relevant $\mathcal{C}$-arguments and critical questions* to the current dialogue that are not already present in the commitment store (see $\Xi_{\mathsf{a}}(D_1^t, i)$ in the following definition). A $\mathcal{C}$-argument and critical question is relevant if it achieves the dialogue topic (given by $\mathsf{Topic}(D_r^t)$) or attacks an argument present in the combined commitment store $CoSt$. The agent should use an *assert* move for arguments with completed ξ tuples, or the *propose* move for non-completed ξ tuples. Any arguments returned that are either a $\mathcal{C}$-argument or one of the critical questions CQ5, CQ6, CQ7 or CQ11 allow the proposal or assertion of a new coalition in the ξ tuple, because this is an argument where the condition $\mathsf{CoalAct}(A') = \xi'$ ($\xi = \xi'$) for an argument $A' \in CoSt$ is not enforced. The other CQs use a previously proposed or asserted coalition to argue in favour or against it.

Lastly, if all the agents cannot propose or assert anything new, then the moves that are returned are either the 'close' move to be used if the agents want to remain in the dialogue to hear what the other agents have to say, or the 'leave' move used when the agent wants to leave the dialogue completely.

Definition 74: *The* **C-pAct protocol** *for a $C\text{-}pAct$ dialogue is a function $\Pi : \mathcal{D} \times N \mapsto \wp^P(\mathcal{M})$. If agent $j \in N$ is the agent given and D_r^t is the dialogue given where: $\mathsf{Ags}(D_r^t) = N$, $CoSt = \bigcup_{\forall k \in N} CoSt_k^t$, $\mathsf{Type}(D_r^t) = C\text{-}pAct$, $\mathsf{Topic}(D_r^t) = p$ and $1 \leq t$, then $\Pi(D_1^t, j)$ is*

$\Pi_{\mathsf{a}}(D_1^t, j) \cup \{\langle j, close, dialogue(C\text{-}pAct, \mathsf{Topic}(D_r^t))\rangle\} \cup$
$\{\langle j, leave, dialogue(C\text{-}pAct, \mathsf{Topic}(D_r^t))\rangle\}$

where

$\Pi_{\mathsf{a}}(D_1^t, j) = \{\langle j, propose \vee assert, \Psi\rangle |$
(1) $\Psi \neq \emptyset$*, and*

(2) $\forall A \in \Psi$*:*

(i) $A \notin CoSt$*, and*

either (ii,c) $A = \langle q_x, \mu, \xi, q_y, p, v, + \rangle$ *where A is a* $\mathcal{C}$*-*$argument$

or (ii,2) $A = \langle \mu, \xi, q_y \rangle$ *where A is a* $cq2$*-*$argument$ *and* $\exists A' \in CoSt$ *s.t.*

$\mathsf{Action}(A') = \mu'$, $(\mu = \mu')$

$\mathsf{CoalAct}(A') = \xi'$ $(\xi = \xi')$,

$\mathsf{EndState}(A') = q'_y$ $(q_y \neq q'_y)$,

or (ii,3) $A = \langle \mu, \xi, q_y, \emptyset \rangle$ *where A is a* $cq3$*-*$argument$ *and* $\exists A' \in CoSt$ *s.t.*

$\mathsf{Action}(A') = \mu'$, $(\mu = \mu')$

$\mathsf{CoalAct}(A') = \xi'$ $(\xi = \xi')$,

$\mathsf{EndState}(A') = q'_y$ $(q_y \approx q'_y, p \notin q_y, p \in q'_y)$

or (ii, 4) $A = \langle \mu, \xi, q_y, v, \{=, -\} \rangle$ *and where A is a* $cq4$*-*$argument$ $\exists A' \in CoSt$ *s.t.*

$\mathsf{Action}(A') = \mu'$, $(\mu = \mu')$

$\mathsf{CoalAct}(A') = \xi'$ $(\xi = \xi')$,

$\mathsf{EndState}(A') = q'_y$ $(q_y \approx q'_y)$

$\mathsf{Value}(A') = v$ *(v = v')*, $\mathsf{Polarity}(A') = +$,

or (ii, 5) $A = \langle \mu, \xi, q_y \rangle$ *where A is a* $cq5$*-*$argument$ *and* $\exists A' \in CoSt$ *s.t.*

$\mathsf{Action}(A') = \mu'$, $(\mu \neq \mu')$

$\mathsf{EndState}(A') = q'_y$ $(q_y \approx q'_y)$

or (ii, 6) $A = \langle \mu, \xi, q_y, v, + \rangle$ *where A is a* $cq6$*-*$argument$ *and* $\exists A' \in CoSt$ *s.t.*

$\mathsf{Action}(A') = \mu'$, $(\mu \neq \mu')$

$\mathsf{EndState}(A') = q'_y$ $(q_y \approx q'_y)$

$\mathsf{Value}(A') = v$ *(*$v \neq v'$*)*, $\mathsf{Polarity}(A') = +$,

or (ii, 7) $A = \langle \mu, \xi, q_y, v, + \rangle$ *where A is a* $cq7$*-*$argument$ *and* $\exists A' \in CoSt$ *s.t.*

$\mathsf{Action}(A') = \mu'$, $(\mu \neq \mu')$

$\mathsf{EndState}(A') = q'_y$ $(q_y \neq q'_y)$

$\mathsf{Value}(A') = v$ *(v = v')*, $\mathsf{Polarity}(A') = +$,

or (ii, 8) $A = \langle \mu, \xi, q_y, v, - \rangle$ *where A is a* $cq8$*-*$argument$ *and* $\exists A' \in CoSt$ *s.t.*

$\mathsf{Action}(A') = \mu'$, $(\mu = \mu')$

$\mathsf{CoalAct}(A') = \xi'$ $(\xi = \xi')$,

$\mathsf{EndState}(A') = q'_y$ $(q_y \approx q'_y)$

$\mathsf{Value}(A') = v$ *(v = v')*, $\mathsf{Polarity}(A') = +$,

or (ii,9) $A = \langle \mu, \xi, q_y, v, - \rangle$ *where A is a* $cq9$*-*$argument$ *and* $\exists A' \in CoSt$ *s.t.*

$\mathsf{Action}(A') = \mu'$, $(\mu = \mu')$

$\mathsf{CoalAct}(A') = \xi'$ $(\xi = \xi')$,

$\mathsf{EndState}(A') = q'_y$ $(q_y \approx q'_y)$

$\mathsf{Value}(A') = v$ *(*$v \neq v'$*)*, $\mathsf{Polarity}(A') = +$,

or (ii,10) $A = \langle \mu, \xi, q_y, v, + \rangle$ *where A is a* $cq10$*-*$argument$ *and* $\exists A' \in CoSt$ *s.t.*

$\mathsf{Action}(A') = \mu'$, $(\mu = \mu')$

$\mathsf{CoalAct}(A') = \xi'$ $(\xi = \xi')$,

$\mathsf{EndState}(A') = q'_y$ $(q_y \approx q'_y)$

$\mathsf{Value}(A') = v$ *(*$v \neq v'$*)*, $\mathsf{Polarity}(A') = +$,

or (ii, 11) $A = \langle \mu, \xi, q_y, v, + \rangle$ *where A is a* $cq11\text{-}argument$ *and* $\exists A' \in CoSt$ *s.t.*

$\mathsf{Action}(A') = \mu'$, $(\mu \neq \mu')$

$\mathsf{EndState}(A') = q'_y$ $(q_y \neq q'_y)$

$\mathsf{Value}(A') = v$ *(*$v \neq v'$*)*, $\mathsf{Polarity}(A') = +$,

or (ii, 13) $A = \langle \neg\mu \rangle$ *where A is a* $cq13\text{-}argument$ *and* $\exists A' \in CoSt$ *s.t.*

$\mathsf{Action}(A') = \mu'$ *(*$\mu = \mu'$ *and* $\exists ac \in \mu$ *where* $ac \notin Ac^i$*)*

or (ii,14) $A = \langle \neg q_y \rangle$ *where A is a* $cq14\text{-}argument$ *and* $\exists A' \in CoSt$ *s.t.*

$\mathsf{EndState}(A') = q'_y$ $(q_y \approx q'_y$ *and* $\neg\exists q_z \in Q^i$ *where* $q_y \approx q_z)$,

or (ii, 15) $A = \langle \neg p \rangle$ *where A is a* $cq15\text{-}argument$ *and* $\exists A' \in CoSt$ *s.t.*

$\mathsf{Goal}(A') = p'$, *(*$p = p'$, $p \in \Phi^i$ *and* $\neg\exists q_z \in Q^j$ *where* $p \in \pi(q_z)$*)*

or (ii, 16) $A = \langle \neg v \rangle$ *where A is a* $cq16\text{-}argument$ *and* $\exists A' \in CoSt$ *s.t.*

$\mathsf{Value}(A') = v'$ *(*$v = v'$ *and* $v \notin Av^i$*).*

or (ii, 17) $A = \langle \mu, \neg\xi \rangle$ *where A is a* $cq17\text{-}argument$ *and* $\exists A' \in CoSt$ *s.t.*

$\mathsf{Action}(A') = \mu'$, $(\mu = \mu'$ *and* $\exists j \in \xi$ *where* $\zeta^i(j, \mathsf{AgAction}(j, \xi)) = \bot)$

else $\Pi(D_1^t, j) = \emptyset$.

If every agent is using the $C\text{-}pAct$ protocol then a $C\text{-}pAct$ dialogue, just like the persuasion dialogue of [21], is guaranteed to terminate according to the following Theorem:

Theorem 3.2. *If each agent in a* $C\text{-}pAct$ *dialogue is using the* $C\text{-}pAct$ *protocol, then the dialogue is guaranteed to terminate (given each agent's VATS is of finite size).*

Proof. Assume that all n agents in a $C\text{-}pAct$ dialogue game are using the $C\text{-}pAct$ protocol but the dialogue never terminates. For this to occur, the agents must be constantly proposing or asserting arguments for coalitions to form (because n leave moves will terminate the dialogue as there will be no agents left and n close moves will terminate the dialogue as no more agents have any more arguments to add).

Yet according to the $C\text{-}pAct$ protocol condition (2)(i), an argument for a coalition to form can only be proposed or asserted if it has not previously been proposed or asserted by any agent. As it is given that each agent's VATS is of finite size, then in the worst case, each agent will propose or assert every possible argument it can make out of its VATS. Once this has occurred, the only moves the agents will have left is the close or leave move and n leave moves or n close moves in a row will terminate the dialogue, thus contradicting the assumption. □

Thus, as every possible argument in a VATS of an agent could theoretically be communicated, the tractability of the $C\text{-}pAct$ dialogue game (like the persuasion dialogue game in [21]) depends on the total number of possible arguments that can be generated from all of the agent's VATS.

3.5 Coalition Argument Evaluation

Each agent i can evaluate the $\mathcal{C}$-arguments and critical questions uttered in the dialogue that include i, by placing them in a *Value-Based Argumentation Framework*[6] (VAF) [19]. For a VAF, it is assumed that each agent/audience i has a preference order over its *social-values*, of the form $v_1 \succ ... \succ v_k$ (where $k = |Av^i|$), that ranks i's known social-values into an order where v_1 is the most preferred and v_k the least. This gives each agent i an Audience specific Value-based Argumentation Framework (AVAF), denoted VAF$_i$.

The classical VAF definitions of *objective acceptance* and *subjective acceptance*[7], to find the acceptable arguments, are modified in this chapter to handle arguments for coalitions, as several issues were identified whilst developing a model to select the final recommended coalitions to form, from the communicated arguments in the C-$pAct$ dialogue. These issues were:

1. **Coalition Membership** - In this chapter it is assumed that each agent i only reasons over arguments for coalitions where i is a member of that coalition. This condition is used so that an agent cannot try to manipulate other agents not in its coalition with false arguments over that coalition.

2. **Incomplete Coalitions** - There may be some arguments communicated in the dialogue, through the propose move, that have incomplete coalitions. These arguments should be discounted from consideration.

3. **Agents Leaving** - Arguments that include an agent who has left the dialogue before the dialogue completed should be removed from consideration.

4. **Arguments Without Social-Values** - Since several critical questions concern *problem formulation* issues [10], not all arguments have an associated social-value (i.e. those derived from CQ13, CQ14 and CQ15). Instead, these arguments are taken to automatically defeat any other argument that they attack in the AVAF, by being assigned the social-value 'truth', which always ranks higher than any other social-value [19], provided that these arguments are *agreeable*. An *agreeable* argument for an agent i is one that matches the information in agent i's VATS, i.e.: an agreeable argument can be formulated from VATS^i if required. For example, CQ16 against the social-value v_7 is only *agreeable* to i if $v_7 \notin Av^i$. The same argument is *not agreeable* to j if $v_7 \in Av^j$. In future work, this process could be modeled within *uniform argumentation frameworks* [11] where value-based and value independent arguments are modeled together.

5. **Coalition to Coalition Attacks** - Two different arguments in favour of two distinct coalitions should attack each other if they share an agent, due to a traditional assumption of coalition formation that an agent can only be in one coalition at any one time [136]. Furthermore, given the model presented in this chapter, two different arguments A1 and A2 for two different coalitions should attack each other if EndState$(A1) \neq$ EndState$(A2)$,

[6] Described in background Section 2.6.2.
[7] Both described in Section 2.6.2.

which indicate that the two coalitions will attempt to bring about a state with at least one conflicting proposition. This is a logical contradiction.

After considering these issues, for an agent i to construct an AVAF to reason over its acceptable coalition(s) given the arguments communicated during the dialogue, each agent i should: (1) find the arguments with coalitions including i; (2) remove the arguments with incomplete coalitions; (3) remove the arguments that include an agent j who has left the dialogue; (4) remove the non-agreeable arguments that promote the value 'truth'; (5) and add the critical question attacks (which include the coalition to coalition attacks). Each agent i can then evaluate the VAF_i it has created, using its social-value ordering to find its preferred extension(s).

In this thesis, due to theses identified issues, model specific acceptability definitions are used:

Definition 75: Coalitionally Objectively-Stable: *Given a VAF, $\langle Args, R, V, val, N\rangle$, a coalition $C \subseteq N$ is coalitionally objectively-stable if and only if: $\forall i \in C$, $\exists A \in Args$ that is in every preferred extension of VAF_i where Coalition$(A) = C$.*

Example 3.1. *Consider $N = \{1, 2, 3\}$ agents discussing what coalitions to form, where there are three possible coalitions that can form:*

- *Coalition $\{1, 2\}$ in argument A_1 that promotes social-value v_3;*
- *Coalition $\{1, 3\}$ in argument A_2 that promotes social-value v_1; and*
- *Coalition $\{2, 3\}$ in argument A_3 that promotes social-value v_2.*

In this example, all arguments attack every other argument. The social-value ordering of the agents are as follows:

- *Agent 1 prefers v_1 over v_2 over v_3;*
- *Agent 2 prefers v_3 over v_2 over v_1; and*
- *Agent 3 prefers v_1 over v_2 over v_3.*

Given that the agents only reason over coalitions that involve itself, agent 1 has a preferred extension (PE) of $PE_1^1 = \{A_2\}$, agent 2 has $PE_1^2 = \{A_1\}$ and agent 3 has $PE_1^3 = \{A_2\}$.

In this case, it can be seen that coalition $\{1, 3\}$, in argument A_2 is coalitionally objectively-stable because the coalition exists in all the preferred extensions of all the agents that are included in it. For example, A_2 is in PE_1^1 and PE_1^3 while no more PEs exist for agents 1 or 3.

Definition 76: Coalitionally Subjectively-Stable: *Given a VAF, $\langle Args, R, V, val, N\rangle$, a coalition $C \subseteq N$ is coalitionally subjectively-stable if and only if: $\forall i \in C$, $\exists A \in Args$ that is in at least one preferred extension of VAF_i where Coalition$(A) = C$; and $\neg\exists A' \in Args$ where Coalition$(A') \neq$ Coalition(A), $R(A', A)$ and for all $j \in$ Coalition(A'), A' is in* at least one *preferred extension of VAF_j.*

Example 3.2. *Consider $N = \{1, 2, 3\}$ agents discussing what coalitions to form, where there are four possible coalitions that can form:*

- *Coalition* $\{1, 2\}$ *in argument* A_1 *that promotes social-value* v_1*;*
- *Coalition* $\{1, 3\}$ *in argument* A_2 *that promotes social-value* v_1*;*
- *Coalition* $\{2, 3\}$ *in argument* A_3 *that promotes social-value* v_2*; and*
- *Coalition* $\{1, 2, 3\}$ *in argument* A_4 *that promotes social-value* v_3*.*

In this example, all arguments attack every other argument. The social-value ordering of the agents are as follows:

- *Agent 1 prefers* v_1 *over* v_2 *over* v_3*;*
- *Agent 2 also prefers* v_1 *over* v_2 *over* v_3*; and*
- *Agent 3 prefers* v_3 *over* v_2 *over* v_1*.*

Given that the agents only reason over coalitions that involve itself, agent 1 has two preferred extensions (PEs) of $PE_1^1 = \{A_1\}$ *or* $PE_2^1 = \{A_2\}$*, agent 2 has only* $PE_1^2 = \{A_1\}$ *and agent 3 has only* $PE_1^3 = \{A_4\}$*.*

In this case, it can be seen that the coalition $\{1, 2\}$*, in argument* A_1 *is coalitionally subjective-stable because even though it is not in an argument in all the preferred extensions of agent 1 (that is agent 1 thinks it has another option available), it is the only coalition option that is also in an argument in the other coalition member's preferred extension. For example,* A_1 *in* PE_1^1 *and* PE_1^2 *while* $\{1, 3\}$ *(in argument* A_2*) is not in any PE of agent 3.*

Definition 77: Coalitionally Subjectively-Unstable: *Given a VAF,* $\langle Args, R, V, val, N\rangle$*, a coalition* $C \subseteq N$ *is coalitionally subjectively-unstable if and only if:* $\underline{\forall i \in C}$*,* $\underline{\exists A \in Args}$ *that is in* <u>at least one</u> *preferred extension of* VAF_i *where* $Coalition(A) = C$*; and* $\underline{\exists A' \in Args}$ *where* $Coalition(A') \neq Coalition(A)$*,* $R(A', A)$ *and for all* $j \in \mathsf{Coalition}(A')$*,* A' *is in* at least one *preferred extension of* VAF_j*.*

Example 3.3. *Consider* $N = \{1, 2, 3\}$ *agents discussing what coalitions to form, where there are three possible coalitions that can form:*

- *Coalition* $\{1, 2\}$ *in argument* A_1 *that promotes social-value* v_1*;*
- *Coalition* $\{1, 3\}$ *in argument* A_2 *that promotes social-value* v_1*; and*
- *Coalition* $\{2, 3\}$ *in argument* A_3 *that promotes social-value* v_2*.*

In this example, all arguments attack every other argument. The social-value ordering of the agents are as follows:

- *Agent 1 prefers* v_1 *over* v_2*;*
- *Agent 2 prefers* v_1 *over* v_2*; and*
- *Agent 3 also prefers* v_1 *over* v_2*.*

Given that the agents only reason over coalitions that involve itself, agent 1 has two preferred extensions (PEs) of $PE_1^1 = \{A_1\}$ *or* $PE_2^1 = \{A_2\}$*, agent 2 has only* $PE_1^2 = \{A_1\}$ *and agent 3 has only* $PE_1^3 = \{A_2\}$.

In this case, it can be seen that coalition $\{1, 2\}$ *in argument* A_1*, and coalition* $\{1, 3\}$ *in argument* A_2*, are coalitionally subjectively-unstable because both of the coalitions exists in at least one argument of one of the preferred extensions of all their members. For example,* A_1 *is in* PE_1^1 *and* PE_1^2*, while* A_2 *is in* PE_2^1 *and* PE_1^3.

Any coalition C that is coalitionally objectively-stable should form, as all agents believe that C is acceptable to them given their value order under *every* preferred extension of their AVAF, and they do not have another equally acceptable coalition (otherwise C would be either coalitionally subjectively-stable or coalitionally subjectively-unstable due to the coalition to coalition attack issue).

Any coalition C that is coalitionally subjectively-stable should form, as all agents believe that C is acceptable to them given their value order under *at least one* preferred extension of their AVAF, and they do not have another equally acceptable coalition C' to form that every agent of C' finds equally acceptable (otherwise C would be coalitionally subjectively-unstable due to the coalition to coalition attack issue).

If CS is the set of coalitions that should form after the VAF evaluation, then: for every coalition C^o, that is coalitionally objectively-stable, $C^o \in CS$ should hold; and for every coalition C^s that is coalitionally subjective-stable, $C^s \in CS$ should hold.

The remaining coalitions that should be considered for inclusion in CS are those coalitions that are coalitionally subjectively-unstable (denoted Δ). That is, a coalition C that is acceptable to every agent $i \in C$ under *at least one* preferred extension of i, even though agent i has other coalitions of the same acceptability level. Firstly any subset $\Lambda \subseteq \Delta$ chosen to be added to CS should have the following conditions so that Λ does not conflict with itself or the other coalitions in CS: (1) no coalitions in CS or Λ should share an agent; and (2) no coalitions in CS or Λ should attempt to bring about a conflicting end state. That is, CS should always remain conflict free.

The idea of partitioning the coalitions into three sets resembles the three set partition of Amgoud's work in [4], where *acceptable*, *abeyance* and *rejected* coalitions were identified. Yet in [4], only one preference order for the coalitions was used. Having multiple preference orders gives rise to different degrees of acceptability (identified in this chapter as coalitionally objectively-stable, coalitionally subjectively-stable and coalitionally subjectively-unstable).

To resolve which coalitions Λ should be chosen from Δ, inspiration could be taken from qualitative coalitional games (QCGs). In these games, the minimal coalition(s) that are successful are chosen. In this chapter, the minimal successful coalitions are the ones that satisfy as many agents as possible, i.e. $\neg\exists\Lambda' \subseteq \Delta$ where $|\Lambda'| > |\Lambda|$. Yet this method of selection does not take into account the preferences of the agents.

Alternatively, to resolve which coalitions to choose from Δ, inspiration could be taken from qualitative coalitional games with preferences. In these games, using the stability definition given in Section 2.3.1, a stable subset $\Lambda \subseteq \Delta$ could be identified where there does not exist a

coalition $S \in \Delta$, $S \notin \Lambda$, where every member of S prefers coalition S compared to its coalition in Λ.

To conclude, exactly which arguments of Δ should be chosen to be in the final recommended coalition structure CS should depend on what the dialogue designer wants to achieve, for example: are a minimal number of coalitions desired?; or are the most stable coalitions desired?

3.6 Dialogue and Argument Evaluation Example

In order to illustrate the model introduced in this chapter, the following example is used that demonstrates: the $C\text{-}inq$ dialogue; the $C\text{-}pAct$ dialogue; and the evaluation over the asserted arguments for/against coalitions.

3.6.1 Example Preliminaries

Consider a system with agents $N = \{1, 2, 3, 4\}$ where their VATS are modeled in the Figures 3.16 and 3.17, which show the state transitions and the associated value assignments.

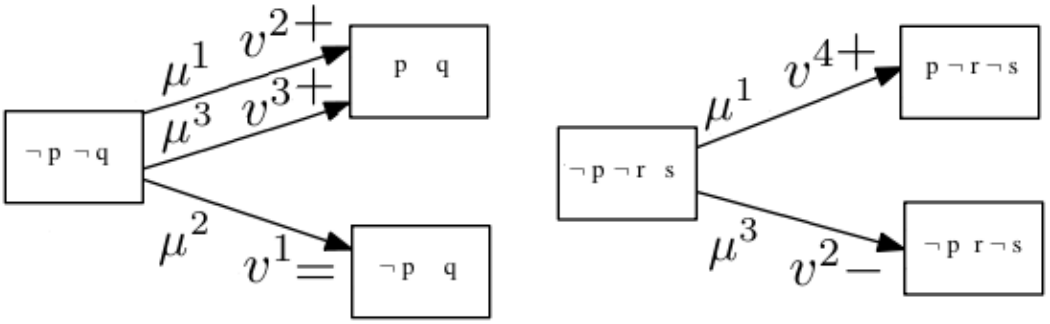

FIGURE 3.16: Agent 1's VATS^1 (left side) and agent 2's VATS^2 (right side).

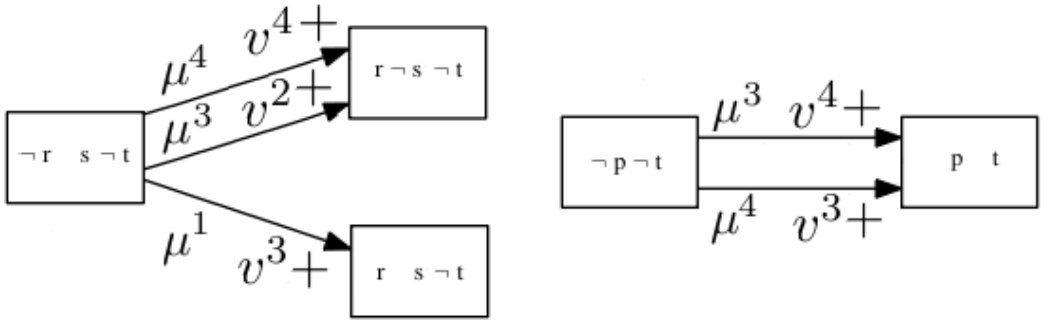

FIGURE 3.17: Agent 3's VATS^3 (left side) and agent 4's VATS^4 (right side).

Given the VATS we find the agent's propositional representations of the world are the following: $\Phi^1 = \{p, q\}$; $\Phi^2 = \{p, r, s\}$; $\Phi^3 = \{r, s, t\}$; and $\Phi^4 = \{p, t\}$. Additionally each agent's belief base before any dialogue commences are: $\Sigma^1 = \{\neg p, \neg q\}$; $\Sigma^2 = \{\neg p \rightarrow \neg r, \neg p \rightarrow s\}$; $\Sigma^3 = \{\}$; and $\Sigma^4 = \{s \rightarrow \neg t\}$.

Finally the joint-actions are composed like so: $\mu^1 = \langle ac^1, ac^2 \rangle$; $\mu^2 = \langle ac^1, ac^5 \rangle$; $\mu^3 = \langle ac^1, ac^3, ac^4 \rangle$; and $\mu^4 = \langle ac^3, ac^4 \rangle$. For simplicity in this example, it is assumed all agents

know that: agent 1 can perform ac^1; agent 2 can perform ac^2; agent 3 can perform ac^3; and agent 4 can perform ac^4.

3.6.2 Inquiry Dialogues

In this example, agent 1 initially knows the value of all of its propositions, obviating the need to inquire over them. However, as agent 2 does not know the current value of any of its propositions it is aware of, it opens an $C\text{-}inq$ dialogue over its propositions p, r and s, denoted $dialogue(C\text{-}inq, \{p, r, s\})$. In this inquiry dialogue, agent 2 can communicate the defeasible rules $\neg p \rightarrow \neg r$ and $\neg p \rightarrow s$. Agent 1 can then respond with the defeasible fact $\neg p$ and the $\mathcal{B}$-argument $\langle \emptyset, \neg p \rangle$, to support the previous defeasible rules communicated. Agent 2 can now communicate the $\mathcal{B}$-arguments $\langle \{\neg p, \neg p \rightarrow \neg r\}, \neg r \rangle$ and $\langle \{\neg p, \neg p \rightarrow s\}, s \rangle$ to indicate that it is now aware that its starting state is $[\neg p, \neg r, s]$[8].

The full informal description of $dialogue(C\text{-}inq, \{p, r, s\})$, is now presented, followed by the formal notation (where the redundant close moves have been removed in both tables):

Move Num.	Move Meaning
1	Agent 2 wants to open a $C\text{-}inq$ dialogue to discuss the Boolean values of propositions p, q and r.
2	Agent 1 wants to join this $C\text{-}inq$ dialogue.
3	Agent 3 wants to join this $C\text{-}inq$ dialogue.
4	Agent 4 wants to join this $C\text{-}inq$ dialogue.
5	Agent 2 believes that: if $\neg p$ holds then $\neg r$ holds; and if $\neg p$ holds then s holds.
6	Agent 1 believes that: $\neg p$ holds true.
7	Agent 2 now has enough information to believe that: $\neg r$ and s hold true.
8	Agent 3 has no more relevant information to communicate.
9	Agent 4 has no more relevant information to communicate.
10	Agent 1 has no more relevant information to communicate.
11	Agent 2 has no more relevant information to communicate. As *all* the agents have communicated that they have no more information to add, then the dialogue is completed successfully.

The above informal description of $dialogue(C\text{-}inq, \{p, r, s\})$ translates to the following formal notation:

Move Num.	Move Notation
1	$\langle 2, open, dialogue(C\text{-}inq, \{p, r, s\}) \rangle$
2	$\langle 1, join, dialogue(C\text{-}inq, \{p, q, r\}) \rangle$
3	$\langle 3, join, dialogue(C\text{-}inq, \{p, q, r\}) \rangle$

[8] Assuming agent 2 trusts agent 1's assertions enough (or trusts agent 1's assertions above any evidence to the contrary).

4	$\langle 4, join, dialogue(C\text{-}inq, \{p, q, r\})\rangle$
5	$\langle 2, assert, \{\neg p \rightarrow \neg r, \neg p \rightarrow s\}, dialogue(C\text{-}inq, \{p, q, r\})\rangle$
6	$\langle 1, assert, \{\neg p, \langle \emptyset, \neg p\rangle\}, dialogue(C\text{-}inq, \{p, q, r\})\rangle$
7	$\langle 2, assert, \{\langle\{\neg p, \neg p \rightarrow \neg r\}, \neg r\rangle, \langle\{\neg p, \neg p \rightarrow s\}, s\rangle\}, dialogue(C\text{-}inq, \{p, q, r\})\rangle$
8	$\langle 3, close, dialogue(C\text{-}inq, \{p, q, r\})\rangle$
9	$\langle 4, close, dialogue(C\text{-}inq, \{p, q, r\})\rangle$
10	$\langle 1, close, dialogue(C\text{-}inq, \{p, q, r\})\rangle$
11	$\langle 2, close, dialogue(C\text{-}inq, \{p, q, r\})\rangle$

This $C\text{-}inq$ dialogue should update the beliefs of the agents of the system to include the $\mathcal{B}$-arguments asserted within it.

Agent 3 now does not need to open another $C\text{-}inq$ dialogue over the propositions r and s, because all of the relevant defeasible facts, defeasible rules and $\mathcal{B}$-arguments have been communicated for r and s. Instead, agent 3 can just open a $C\text{-}inq$ dialogue over proposition t, denoted $dialogue(C\text{-}inq, \{t\})$. Agent 4 has the only information on t and so communicates the $\mathcal{B}$-argument $\langle\{\neg p, \neg p \rightarrow s, s \rightarrow \neg t\}, \neg t\rangle$, where the support for $\neg t$ is in the union of agent 4's belief base (i.e. Σ^4), and the arguments communicated in the previous $C\text{-}inq$ dialogue (i.e. $dialogue(C\text{-}inq, \{p, q, r\})$).

The full informal description of $dialogue(C\text{-}inq, \{t\})$, is now presented, followed by the formal notation (where the redundant close moves have been removed in both tables):

Move Num.	**Move Meaning**
1	Agent 3 wants to open a $C\text{-}inq$ dialogue to discuss the Boolean value of proposition t.
2	Agent 1 wants to join this $C\text{-}inq$ dialogue.
3	Agent 2 wants to join this $C\text{-}inq$ dialogue.
4	Agent 4 wants to join this $C\text{-}inq$ dialogue.
5	Agent 4 believes that: if s holds then $\neg t$ holds. In $dialogue(C\text{-}inq, \{p, r, s\})$, a $\mathcal{B}$-argument was communicated for s, and agent 4 trusts the agent who communicated this argument, so agent 4 now has enough information to believe that $\neg t$ holds.
6	Agent 1 has no more relevant information to communicate.
7	Agent 2 has no more relevant information to communicate.
8	Agent 3 has no more relevant information to communicate.
8	Agent 4 has no more relevant information to communicate. As *all* the agents have communicated that they have no more information to add, then the dialogue is completed successfully.

The above informal description of $dialogue(C\text{-}inq, \{p, r, s\})$ translates to the following formal notation:

Move Num.	Move Notation
1	$\langle 3, open, dialogue(C\text{-}inq, \{t\})\rangle$
2	$\langle 1, join, dialogue(C\text{-}inq, \{t\})\rangle$
3	$\langle 2, join, dialogue(C\text{-}inq, \{t\})\rangle$
4	$\langle 4, join, dialogue(C\text{-}inq, \{t\})\rangle$
5	$\langle 4, assert, \{s \rightarrow \neg t, \langle\{\neg p, \neg p \rightarrow s, s \rightarrow \neg t\}, \neg t\rangle\}, dialogue(C\text{-}inq, \{t\})\rangle$
6	$\langle 1, close, dialogue(C\text{-}inq, \{t\})\rangle$
7	$\langle 2, close, dialogue(C\text{-}inq, \{t\})\rangle$
8	$\langle 3, close, dialogue(C\text{-}inq, \{t\})\rangle$
9	$\langle 4, close, dialogue(C\text{-}inq, \{t\})\rangle$

All agents now have enough information to form single point beliefs over each of their Boolean propositions used to represent the world. If we assume that each agent i trusts the communicated defeasible facts, defeasible rules and $\mathcal{B}$-arguments in both the $C\text{-}inq$ dialogues, then the starting states of each of the agent's VATS are found to be: $q_0^1 = [\neg p, \neg q]$ for agent 1; $q_0^2 = [\neg p, \neg r, s]$ for agent 2; $q_0^3 = [\neg r, s, \neg t]$ for agent 3; and $q_0^4 = [\neg p, \neg t]$ for agent 4.

3.6.3 Persuasion Dialogue

Now that each agent's starting state has been established, the agents can start to reason over what coalitions to form. Consider Agent 1 starting a $C\text{-}pAct$ dialogue to achieve p, denoted $dialogue(C\text{-}pAct, p)$. The following is a description of what would happen if all the agents joined this dialogue and each agent i communicated all arguments identified by the $C\text{-}pAct$ protocol relating to arguments for/against the formation of a coalition including i (where these arguments are listed in Table 3.7). The argument evaluation for this example, in Section 3.6.4, makes clear which arguments and argument attacks are relevant to which agent.

In $dialogue(C\text{-}pAct, p)$, agent 1 initially identifies the joint-actions that it thinks will achieve the goal p to promote a social-value. Given agent 1's VATS^1, these joint-actions are μ^1 and μ^3 that promote v^2 and v^3 respectively. Agent 1 identifies those agents that it believes can help it achieve μ^1 and μ^3. It believes the coalition $\{1, 2\}$ can perform μ^1 and the coalition $\{1, 3, 4\}$ can achieve μ^3. Agent 1 puts this identified information into $\mathcal{C}$-argument form, denoted **A1**$= \langle \mu^1, \langle\langle 1, ac^1\rangle, \langle 2, ac^2\rangle\rangle, [p, q], p, v_2, +\rangle$ and **A2** $= \langle \mu^3, \langle\langle 1, ac^1\rangle, \langle 3, ac^3\rangle, \langle 4, ac^4\rangle\rangle, [p, q], p, v_3, +\rangle$, and *asserts* them to the other agents of the respective coalitions. Agent 1 cannot form an argument for μ^2 as μ^2 does not achieve the goal that is the subject of the persuasion dialogue (or give a relevant critical question to another argument in the dialogue). Even if μ^2 could achieve goal p, an argument including μ^2 could only be *proposed* because agent 1 does not know of an agent capable of performing the single action ac^5.

Both arguments A1 and A2 are conflicting because the coalitions in both arguments cannot form at the same time (as agent 1 cannot be in two coalitions at once). Therefore A1 and A2 pose the following critical questions to each other: **CQ5** = 'Are there alternative ways of

Arg Num	**Ag**	μ	ξ	q_y	p	v	**s**	**Attacks**
A1	1	μ^1	$\langle\langle 1, ac^1\rangle, \langle 2, ac^2\rangle\rangle$	$[p, q]$	p	v_2	$+$	A2(CQ5&6), A3(CQ10), A4(CQ5&6), A5(CQ5&6), A6(CQ5&6), A7(CQ5&6).
A2	1	μ^3	$\langle\langle 1, ac^1\rangle, \langle 3, ac^3\rangle, \langle 4, ac^4\rangle\rangle$	$[p, q]$	p	v_3	$+$	A1(CQ5&6), A3(CQ5&6), A4(CQ10), A5(CQ5&6), A6(CQ10), A7(CQ5)
A3	2	μ^1	$\langle\langle 1, ac^1\rangle, \langle 2, ac^2\rangle\rangle$	$[p, \neg r, \neg s]$	p	v_4	$+$	A1(CQ10), A2(CQ5&6), A4(CQ11), A5(CQ7), A6(CQ5&6), A7(CQ5,CQ6)
A4	3	μ^3	$\langle\langle 1, ac^1\rangle, \langle 3, ac^3\rangle, \langle 4, ac^4\rangle\rangle$	$[r, \neg s, \neg t]$	$\emptyset$	v_2	$+$	A1(CQ5), A2(CQ3,CQ10), A3(CQ11), A5(CQ5&6), A6(CQ2), A7(CQ11)
A5	3	μ^4	$\langle\langle 3, ac^3\rangle, \langle 4, ac^4\rangle\rangle$	$[r, \neg s, \neg t]$	$\emptyset$	v_4	$+$	A1(CQ5&6), A2(CQ5&6), A3(CQ7), A4(CQ5&6), A6(CQ7), A7(CQ2)
A6	4	μ^3	$\langle\langle 1, ac^1\rangle, \langle 3, ac^3\rangle, \langle 4, \alpha^4\rangle\rangle$	$[p, t]$	p	v_4	$+$	A1(CQ5&6), A2(CQ10), A3(CQ5), A4(CQ2), A5(CQ7), A7(CQ5&6)
A7	4	μ^4	$\langle\langle 3, ac^3\rangle, \langle 4, ac^4\rangle\rangle$	$[p, t]$	p	v_3	$+$	A1(CQ5&6), A2(CQ5), A3(CQ5&6), A4(CQ11), A5(CQ2), A6(CQ5&6)
A8	4					$\neg v_2$		A1(CQ16), A4(CQ16)
A9	1					$\neg v_4$		A3(CQ16), A5(CQ16), A6(CQ16)
A10	3			$\neg[t]$				A6(CQ14), A7(CQ14)

TABLE 3.7: The instantiated argumentation schemes asserted in the example C-$pAct$ dialogue of Section 3.6.3, where the attack column lists *all* the calculated attacks, found once the C-$pAct$ dialogue has been completed successfully.

realising the same consequences?' because Action$(A1) \neq$ Action$(A2)$, and EndState$(A1) \approx$ EndState$(A2)$; and **CQ6** = 'Are there alternative ways of realising the same consequences to promote some other social-value?' because Action$(A1) \neq$ Action$(A2)$, EndState$(A1) \approx$ EndState$(A2)$, Value$(A1) \neq$ Value$(A2)$ and Polarity$(A1) =$ Polarity$(A2) = +$. Consequently agent 1 needs to reason over the arguments presented by the other agents to find out which (if any) of the two arguments communicated are acceptable to the agents of the potential coalitions $\{1, 2\}$ and $\{1, 3, 4\}$.

Given agent 1's communicated arguments, agent 2 can formulate an additional argument to *assert* that poses different critical questions. The argument, denoted **A3** $= \langle \mu^1, \langle\langle 1, ac^1\rangle, \langle 2, ac^2\rangle\rangle, [p, \neg r, \neg s], p, v_4, +\rangle$, poses the critical questions: **CQ5 to A2**; **CQ6 to A2**; and **CQ10** = 'Does doing the joint-action promote some other social-value?' **to A1** because Action$(A1)$ = Action$(A3)$, CoalAct$(A1)$ = CoalAct$(A3)$, EndState$(A1) \approx$ EndState$(A3)$, Value$(A1) \neq$ Value$(A3)$ and Polarity$(A1) =$ Polarity$(A3)$ $+$. Agent 2 cannot formulate an argument for μ^3 because it cannot perform any single actions of μ^3.

Agent 3 can now respond to the previous arguments by formulating and *asserting* two new arguments. The first argument, denoted **A4** $= \langle \mu^3, \langle\langle 1, ac^1\rangle, \langle 3, ac^3\rangle, \langle 4, ac^4\rangle\rangle, [r, \neg s, \neg t], \emptyset, v_2, + \rangle$, poses the critical questions: **CQ3 =** 'Assuming the joint-action has the stated consequences, will the joint-action bring about the desired goal?' **to A2** because Action$(A2)$ = Action$(A4)$, CoalAct$(A2)$ = CoalAct$(A4)$, EndState$(A2) \approx$ EndState$(A4)$, Goal$(A2) \notin$ EndState$(A4)$ and Goal$(A2) \in$ EndState$(A2)$; **CQ5 to A1**; **CQ10 to A2**; and **CQ11 =** 'Does doing the joint-action preclude some other joint-action which would promote some other social-value?' **to A3** because Action$(A3) \neq$ Action$(A4)$, EndState$(A3) \neq$ EndState$(A4)$, Value$(A3) \neq$ Value$(A4)$ and Polarity$(A3)$ = Polarity$(A4)$ = $+$. The second argument of agent 3, denoted **A5** $= \langle \mu^4, \langle\langle 3, ac^3\rangle, \langle 4, ac^4\rangle, [\neg r, \neg s, t], \emptyset, v_4, + \rangle$, poses the critical questions (to the arguments already communicated): **CQ5 to A1, A2 and A4**; **CQ6 to A1, A2 and A4**; and **CQ7 =** 'Are there alternative ways of promoting the same social-value?' **to A3** because Action$(A3) \neq$ Action$(A5)$, EndState$(A3) \neq$ EndState$(A5)$, Value$(A3)$ = Value$(A5)$ and Polarity$(A3)$ = Polarity$(A5) = +$.

Agent 4 now responds by formulating three additional arguments that attack other arguments previously asserted via critical questions. The first argument agent 4 formulates, denoted **A6** $= \langle \mu^3, \langle\langle 1, ac^1\rangle, \langle 3, ac^3\rangle, \langle 4, ac^4\rangle\rangle, [p, t], p, v_4, + \rangle$, poses the critical questions: **CQ2 =** 'Does the joint-action have the stated consequences?' to **A4** because Action$(A4)$ = Action$(A6)$, CoalAct$(A4)$ = CoalAct$(A6)$, EndState$(A4) \neq$ EndState$(A6)$; **CQ5 to A1 and A3**; **CQ6 to A1**; **CQ7 to A5**; and **CQ10 to A2**. The second argument agent 4 formulates, denoted **A7** $= \langle \mu^4, \langle\langle 3, ac^3\rangle, \langle 4, ac^4\rangle\rangle, [p, t], p, v_3, + \rangle$, poses the critical questions: **CQ2 to A5**; **CQ5 to A1, A2, A3 and A6**; **CQ6 to A1, A3, A6**; and **CQ11 to A4**.

Agent 4's last argument, denoted **A8** $= \langle \neg v_2 \rangle$, poses the critical question: **CQ16 =** 'Is the social-value a legitimate social-value?' to **A1** (and **A4**) because Value$(A8) = \neg$ Value$(A1)$ and Value$(A1) \notin Av^4$. Argument A8 is associated to the social-value truth (the highest ranking social-value) for an agent i, if i finds A8 agreeable, i.e. i can form A8 in its VATS^i. Agent 1 can assert a similar argument denoted **A9** $= \langle \neg v_4 \rangle$, to poses the critical question: **CQ16 to A3, A5 and A6**. Argument A9 is also associated to the social-value truth.

Finally, agent 4's arguments allow agent 3 to *assert* one more argument, denoted **A10** $= \langle \neg [t] \rangle$, that poses the critical question: **CQ14 =** 'Are the consequences as described possible?' to **A6 and A7** because $\neg \exists q_z \in Q^4$ where $q_z \approx$ EndState$(A6) \approx$ EndState$(A7) \approx$ EndState$(A10)$. Again, argument A10 is associated to the social-value truth for an agent i, if i finds A10 agreeable.

This is all the possible arguments that could be generated using the agents' VATS of Section 3.6.1, where each agent communicates only arguments for or against a coalition that includes itself. As no more moves are possible, the agents close the dialogue, and $dialogue(C\text{-}pAct, p)$ ends. Table 3.7 lists all the arguments communicated within $dialogue(C\text{-}pAct, p)$, where the attacks between the arguments have been re-calculated after the $C\text{-}pAct$ dialogue has completed successfully.

The full informal description of $dialogue(C\text{-}pAct, p)$, is now presented, followed by the formal notation (where the redundant close moves have been removed in both tables):

Move Num.	Move Meaning
1	Agent 1 wants to open a $C\text{-}pAct$ dialogue to achieve goal p while promoting a value.
2	Agent 2 wants to join this $C\text{-}pAct$ dialogue.
3	Agent 3 wants to join this $C\text{-}pAct$ dialogue.
4	Agent 4 wants to join this $C\text{-}pAct$ dialogue.
5	Agent 1 believes that: if coalition $\{1, 2\}$ forms and undertakes joint-action μ^1, then goal p will be achieved and value v_2 will be promoted; and if coalition $\{1, 3, 4\}$ forms and undertakes joint-action μ^3, then goal p will be achieved and value v_3 will be promoted.
6	Agent 2 believes that: if coalition $\{1, 2\}$ forms and undertakes joint-action μ^1, then goal p will be achieved and value v_4 will be promoted.
7	Agent 3 believes that: if coalition $\{1, 3, 4\}$ forms and undertakes joint-action μ^3, goal p will *not* be achieved but value v_2 will still be promoted; and if coalition $\{3, 4\}$ forms and undertakes joint-action μ^4, then goal p will *not* be achieved but value v_4 will be promoted anyway.
8	Agent 4 believes that: if coalition $\{1, 3, 4\}$ forms and undertakes joint-action μ^3, goal p will be achieved and value v_4 will be promoted; if coalition $\{3, 4\}$ forms and undertakes joint-action μ^4, then goal p will be achieved and value v_3 will be promoted; and finally that the value v_2 does not exist.
9	Agent 1 believes that the value v_4 does not exist.
10	Agent 3 believes that there does not exist a state where t is true.
11	Agent 1 has no more relevant information to communicate.
12	Agent 2 has no more relevant information to communicate.
13	Agent 3 has no more relevant information to communicate.
14	Agent 4 has no more relevant information to communicate. As all the agents have communicated that they all have no more information to add, then the dialogue is completed successfully.

The above informal description of $dialogue(C\text{-}inq, \{p, r, s\})$ translates to the following formal notation:

Move Num.	Move Notation
1	$\langle 1, open, dialogue(C\text{-}pAct, p)\rangle$
2	$\langle 2, join, dialogue(C\text{-}pAct, p)\rangle$
3	$\langle 3, join, dialogue(C\text{-}pAct, p)\rangle$
4	$\langle 4, join, dialogue(C\text{-}pAct, p)\rangle$
5	$\langle 1, assert, \{\langle \mu^1, \langle\langle 1, ac^1\rangle, \langle 2, ac^2\rangle\rangle, [p, q], p, v_2, +\rangle,$ $\langle \mu^3, \langle\langle 1, ac^1\rangle, \langle 3, ac^3\rangle, \langle 4, ac^4\rangle\rangle, [p, q], p, v_3, +\rangle\}, dialogue(C\text{-}pAct, p)\rangle$

6	$\langle 2, assert, \{\langle \mu^1, \langle\langle 1, ac^1\rangle, \langle 2, ac^2\rangle\rangle, \quad [p, \neg r, \neg s], p, v_4, +\rangle\}, dialogue(C\text{-}pAct, p)\rangle$
7	$\langle 3, assert, \{\langle \mu^3, \langle\langle 1, ac^1\rangle, \langle 3, ac^3\rangle, \langle 4, ac^4\rangle\rangle, [r, \neg s, \neg t], \emptyset, v_2, +\rangle, \langle \mu^4, \langle\langle 3, ac^3\rangle, \langle 4, ac^4\rangle\rangle, [\neg r, \neg s, t], \emptyset, v_4, +\rangle\}, dialogue(C\text{-}pAct, p)\rangle$
8	$\langle 4, assert, \{\langle \mu^3, \langle\langle 1, ac^1\rangle, \langle 3, ac^3\rangle, \langle 4, ac^4\rangle\rangle, [p, t], p, v_4, +\rangle, \langle \mu^4, \langle\langle 3, ac^3\rangle, \langle 4, ac^4\rangle\rangle, [p, t], p, v_3, +\rangle \ \langle \neg v_2\rangle\}, dialogue(C\text{-}pAct, p)\rangle$
9	$\langle 1, assert, \{\langle \neg v_4\rangle\}, dialogue(C\text{-}pAct, p)\rangle$
10	$\langle 3, assert, \{\langle \neg [t]\rangle\}, dialogue(C\text{-}pAct, p)\rangle$
11	$\langle 1, close, dialogue(C\text{-}pAct, p)\rangle$
12	$\langle 2, close, dialogue(C\text{-}pAct, p)\rangle$
13	$\langle 3, close, dialogue(C\text{-}pAct, p)\rangle$
14	$\langle 4, close, dialogue(C\text{-}pAct, p)\rangle$

3.6.4 Argument Evaluation

This section evaluates the arguments presented in the example C-$pAct$ dialogue using the acceptability definitions introduced in Section 3.5, which are used to determine which coalitions should be formed. To find either *coalitionally objectively-stable, coalitionally subjectively-stable or coalitionally subjective unstable* arguments, each agent i should generate their audience specific value-based argumentation framework, denoted VAF_i, that contains arguments in the C-$pAct$ dialogue that: (i) have i in the coalition; or (ii) include *agreeable* arguments to i (i.e. arguments that include information that is within agent i's VATS).

Looking initially at VAF_1 in Figure 3.18 there are two main issues to notice: (a) Not all of the arguments from the C-$pAct$ dialogue are present because only some of the arguments in the dialogue included agent 1; and (b) the majority of the attack relationships are reciprocal, due to the nature of the critical questions between these arguments.

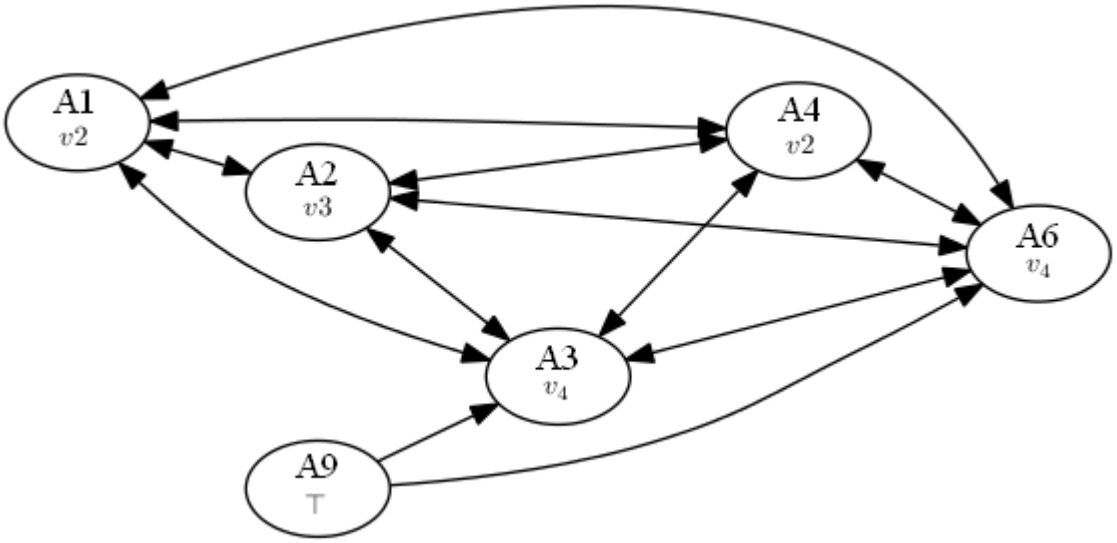

FIGURE 3.18: Agent 1's VAF taken from the example C-$pAct$ dialogue. Each node is labeled with the argument ID and the associated value. If there is no associated value then the label $\top$ is attached to the node to indicate the social-value 'truth'.

Agent 1 can evaluate these arguments using its value-ordering. As Agent 1 does not recognise the value v_4, arguments using this value are defeated in VAF_1 by argument A9. Assuming agent 1's value ordering is $v2 \prec v3$, then agent 1 has two preferred extensions: $PE_1^1 = \{A1\}$ and $PE_2^1 = \{A4\}$. As Coalition$(A1) = \{1, 2\}$ and Coalition$(A4) = \{1, 3, 4\}$, agent 1 is currently undecided on which coalition it wants to form because there is no coalition that is coalitionally objectively-stable, due to there being no coalition in all of agent 1's preferred extensions.

To find out if $\{1, 2\}$ is acceptable then VAF_2 (Figure 3.19) has to be evaluated. This evaluation is very simple because there are only two arguments in the example $C\text{-}pAct$ dialogue that involve a coalition that agent 2 is a member of. Both of these arguments promote different social-values, whilst suggesting that coalition $\{1, 2\}$ should form. Therefore, whatever value ordering agent 2 choses, there will be one argument with the coalition $\{1, 2\}$ in the preferred extension. It is now known that $\{1, 2\}$ is acceptable to its members. To find out if $\{1, 2\}$ is coalitionally subjectively-stable or coalitionally subjectively-unstable, agent 1 needs to know if the coalition $\{1, 3, 4\}$ is also acceptable, and therefore a valid option.

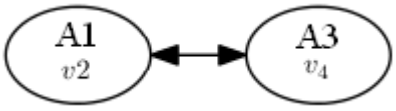

FIGURE 3.19: Agent 2's VAF taken from the example $C\text{-}pAct$ dialogue. Each node is labeled with the argument ID and the associated value.

To find out agent 3's acceptable coalitions (and to continue investigating if $\{1, 3, 4\}$ is acceptable) then VAF_3 (Figure 3.20) has to be evaluated. In VAF_3, argument A10 removes A6 and A7 from consideration due to these instantiations suggesting an end state that conflicts with VATS[3]. Therefore agent 3 is left with a choice between: coalition $\{1, 3, 4\}$ supported by arguments A2 and A4; and coalition $\{3, 4\}$ supported by argument A5. In this case, because different social-values are promoted by the arguments; only one preferred extension is possible. The argument in the preferred extension depends on agent 3's value ordering. Before discussing how the different value orderings will effect the system's outcome, VAF_4 will be introduced (Figure 3.21).

It can be seen that VAF_4 is similar to agent 3's apart from agent 4 has A8 (instead of A10) to eliminate arguments from consideration, where A8 is used to show that agent 4 does not recognise v_2. Out of the remaining arguments, A5 and A7 support coalition $\{3, 4\}$, while arguments A2 and A6 support coalition $\{1, 3, 4\}$. Assuming agent 1's value ordering is $v_2 \prec v_3$, then to get the *final recommended coalition structure* to be $CS = \{\{1, 2\}, \{3, 4\}\}$, agent 3's value order should have v_4 as the most preferred, whereas agent 4 can have any value order. This would make both coalitions of CS coalitionally subjectively-stable as no other coalitions would have the same level of acceptability. The coalitions of CS are not coalitionally objectively-stable because neither coalitions of CS are present in all preferred extensions of agent 1 and agent 4.

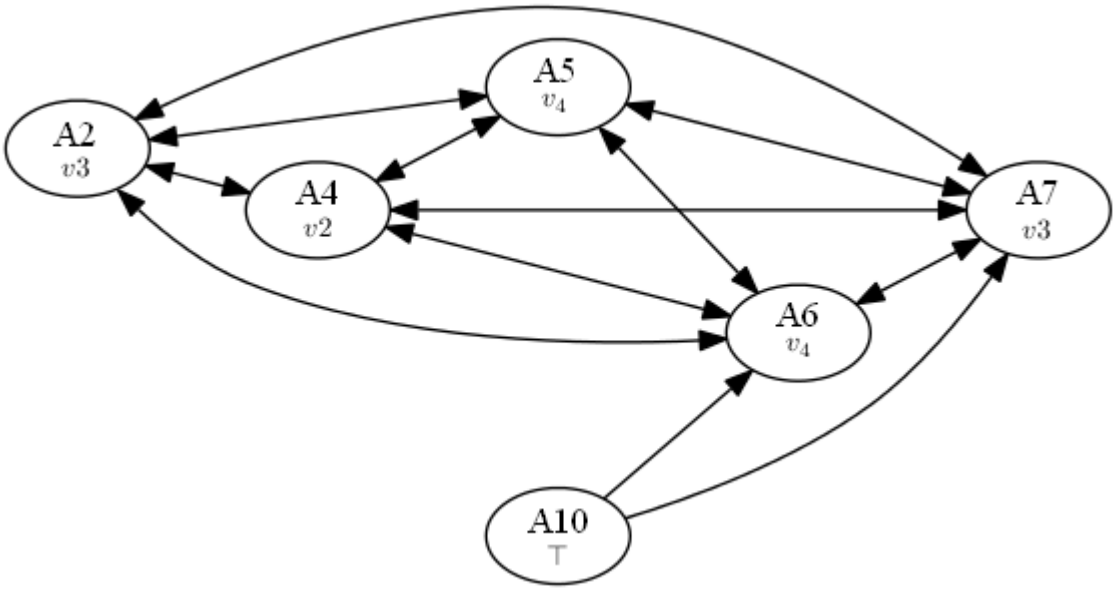

FIGURE 3.20: Agent 3's VAF taken from the example $C\text{-}pAct$ dialogue. Each node is labeled with the argument ID and the associated value. If there is no associated value then the label $\top$ is attached to the node to indicate the social-value 'truth'.

Alternatively, coalitions $\{1, 2\}$ and $\{1, 3, 4\}$ would be coalitionally subjective-unstable if agent 1's value order remained the same, agent 3 favoured v_2 or v_3, and agent 4 had any value order. In this case, both $\{1, 2\}$ and $\{1, 3, 4\}$ would exist in at least one (but not all) preferred extension(s) of every one of its members. As both coalitions would be coalitionally subjective-unstable and conflict with each other by sharing a member, then only one of them can form. This concludes the example.

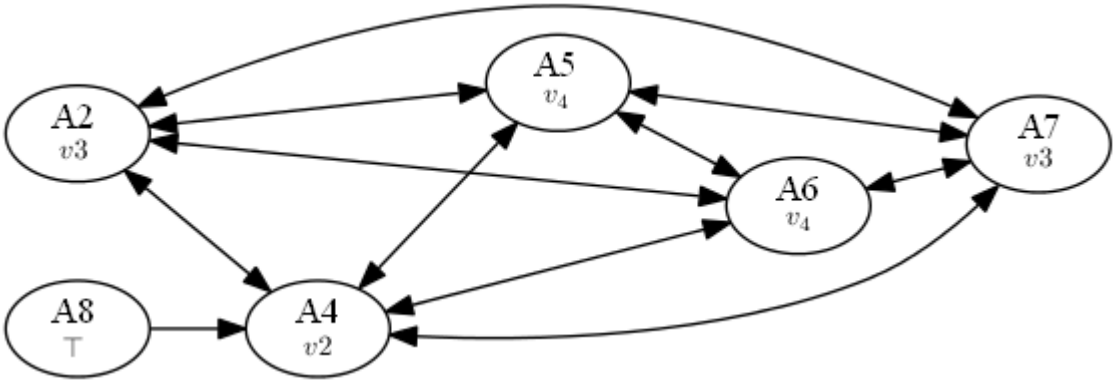

FIGURE 3.21: Agent 4's VAF taken from the example $C\text{-}pAct$ dialogue. Each node is labeled with the argument ID and the associated value. If there is no associated value then the label $\top$ is attached to the node to indicate the social-value 'truth'.

3.7 Summary

This chapter assumes that the agents have a *qualitative* method to value the coalitions. The qualitative method used was *social-values*, as recent work in argumentation suggests that agent systems can be more richly described with the inclusion of these social-values [10, 125] as opposed to just describing systems with goals. Social-values provide reasons why agents may sometimes want to achieve one goal in one situation, and achieve a different (and perhaps contradictory) goal in another situation.

However, using a purely qualitative method to value the coalitions raises several issues:

1. **What is the best set of coalitions to choose?**: When analysing a VAF to find acceptable arguments for coalitions, the agents in the coalitions of arguments that are *coalitionally objectively-stable* and *coalitionally subjectively-stable* have no reasonable logical alternative. But agents in the coalitions of arguments that are *coalitional subjective-unstable* have at least one logical alternative, which raises the following further issues:

2. **Differentiating between coalitions promoting the same social-value**: Some coalitions may achieve the same goal and promote the same value, but not to the same degree. For example, say some agents join together in a coalition to repair a resource that they will mutually benefit from, thereby promoting the social-value *efficiency of resource*. Yet there may be a few ways to promote this social-value that will improve the efficiency on different scales. Consideration of the degree of social-value promotion/demotion is left as a future avenue for research.

3. **How to offer incentives to choose a coalition**: If no acceptable coalitions can be formed using the definitions given in Section 3.5, then at least one agent needs to be incentivised to change its value ordering for a coalition to become acceptable to all of its members. But how the agent can be incentivised in this model is unclear. In transferable utility games [83], an agent i can be incentivised to join a coalition C through the rest of the members of C offering i more proportion of the transferable utility value payoff of C. This idea of using transferable utility as an incentive to form a coalition can help with the reasoning on what coalitions should be chosen. Yet this transferable utility incentivisation cannot be achieved in the qualitative representation presented in this chapter.

4. **How to defined the maximum social welfare**: It can be desirable to design an agent system to maximise social welfare, even in the self-interested multi-agent systems context. For example, the cooperative game theory solution concept of the *core* is defined for self-interested agents and maximises social welfare [83]. Yet there is no way to define the maximal social welfare in this model because there is no method to compare between coalitions promoting the same social-value and no method to compare the utility difference between one coalition promoting one social-value with another coalition promoting another social-value.

All these identified issues can be solved using a quantitative transferable utility based approach to coalition formation. Quantitative transferable utility approaches: (i) define the best set

of coalitions to choose via cooperative game theory *solution concepts*[9]; (ii) allow all coalitions to be differentiated by the utility value that they are assigned; (iii) allow incentives to be offered to agents to join a coalition via the transferable utility property; and (iv) define the maximum social welfare by the total utility value that all the coalitions achieve.

Therefore in the next chapter, the coalition formation process will be discussed from a quantitative valuation approach. Chapter 4 and Chapter 5 assume that each agent holds the same valuation for each coalition, while Chapter 6 will look at situations where valuation disagreements between the agents could occur.

[9] See Section 2.2 for a discussion on the different types.

Chapter 4

Chapter 4

Distributing Coalition Value Calculations

This chapter investigates the best way to distribute the coalition value calculations in characteristic function games to the agents of a multi-agent system.

In a characteristic function game, each coalition's value should be calculated as an agent can potentially join many different coalitions, and so must choose which coalition to join. As the number of coalitions grows exponentially with the number of agents, the burden of requiring every agent to calculate all of the coalition values is high[1]. Furthermore, the complexity of calculating an individual coalition's value can vary, and potentially be exponential [106]. Thus, even if each agent only calculates the value of those coalitions that it can be a member of (i.e. $\frac{2^n}{2}$), then this can still result in a significant overlap of calculations, such that this redundancy can converge to 100%, as $\lim_{n\to\infty} \frac{2^n-1}{n2^{(n-1)}} = 0$.

This chapter's *distributed coalition generation* (DCG) algorithm divides the coalition value calculations between the agents of the system, and has the properties of: (i) no communication is required among the agents; (ii) no coalition's value is calculated more than once; (iii) the coalition value calculations are divided into shares, one for each agent, in a full and approximately equal manner; (iv) agents with equal sized shares have the same number of operations to perform; and (v) each coalition in an agent's share includes that agent as a member. No other algorithm apart from DCG has all of these properties, as discussed in detail in Section 4.3. The motivation of the DCG algorithm was initially outlined in [102], while the underlining theory was presented in [99]. The proof that the DCG algorithm generates all coalitions *once and only once* was developed with Paul Dunne, given in Appendix A. In Chapter 5, it will be described how the DCG algorithm can be used as input to a distributed algorithm to find an optimal coalition structure and a payoff vector in the weak-least core.

The rest of the Chapter is structured as follows: Section 4.1 provides the preliminary machinery for a new method to order and distribute the coalitions, while also providing an introductory example. Section 4.2 details the DCG algorithm. Section 4.3 uses an implementation and theoretical results to analyse the properties of the DCG algorithm when compared to the

[1]The exact number of coalitions that can form given a population of n agents is $2^n - 1$.

other coalition value calculation algorithms. Section 4.4 analyses the runtime performance of the algorithm. Finally Section 4.5 discusses how the DCG could be used in general coalition formation.

4.1 Preliminaries and Introductory Example

The DCG algorithm exploits a novel method for representing and ordering coalitions, so that different coalitions can be allocated to each agent, in such a way as to facilitate the construction of *shares* (one per agent) that eliminate redundant coalition value calculations.

4.1.1 The New Ordering Method

In this chapter, a coalition $C \subseteq \{1, 2, ..., n\}$ is represented as an ordered sequence of identifiers (IDs) that form a *coalition array*, where no agent appears more than once in any coalition, and where $s = |C|$. An *integer increment value* between two contiguous agents i and j in a coalition array corresponds to the difference in the agents' IDs[2]. For example, if we have a coalition array $[3, 6, 1]$, then there are two integer increment values between the ID pairs $3, 6$ and $6, 1$. There is an additional increment between the last and the first agent IDs in the array; i.e. the ID pair $1, 3$. The *integer increment value* between two agents i and j can be decomposed into a *baseline increment* (which is assumed to be 1, since agent IDs are unique) and an *offset increment*, denoted $t_i = (j - i) - 1 \bmod n$ (i.e. integers modulo n). Thus, if $t_i = 0$, the difference between the IDs for agents i and j corresponds only to the baseline increment; whereas if $t_i \neq 0$, then t_i represents an additional offset increment. An *increment array* (IA) denoted $\underline{t} = \langle t_0, t_1, \ldots, t_{s-1} \rangle$ therefore represents the offset increments between the identifiers of the coalition array. For example, given the coalition array $[3, 6, 1]$, the corresponding IA will be $\langle 2, 0, 1 \rangle$.

An *integer partition* of k is a combination of positive integers that add up to exactly k. The DCG algorithm uses integer partitions to identify the offset increments between consecutive pairs of IDs in the coalition array. The full set of integer partitions is denoted $\mathcal{I}(n - s)$; for example, given $n = 6$ and $s = 3$, $\mathcal{I}(n - s) = \{\{3\}, \{2, 1\}, \{1, 1, 1\}\}$. Increment arrays can be formed from an integer partition I for coalitions of size s, *only* when $I \in \mathcal{I}(n - s)$ and $|I| \leq s$, by including additional zero values to satisfy the property:

$$\sum_{i=0}^{s-1} t_i = (n - s)$$

For example, when $n = 6$ and $s = 3$, the integer partition $\{2, 1\}$ could be used to form various possible increment arrays: $\langle 2, 1, 0 \rangle$, $\langle 2, 0, 1 \rangle$, etc. The integer increment values corresponding to the increment array $\langle 2, 1, 0 \rangle$ result from the two coalition arrays $[1, 4, 6]$ and $[2, 5, 1]$, as the ID pairs $1, 4$ and $2, 5$ share $(2 + 1)$, whereas the ID pairs $4, 6$ and $5, 1$ share $(1 + 1)$. As IAs are

[2] As agent IDs are in the range $[1..n]$, IDs modified using an integer increment will result in an ID modulo n. The agent ID n will be returned when the ID 0 is found because $0 \equiv n \pmod{n}$.

	L_3			
	$L_{3,\langle 3,0,0\rangle}$	$L_{3,\langle 2,1,0\rangle}$	$L_{3,\langle 2,0,1\rangle}$	$L_{3,\langle 1,1,1\rangle}$
CV_1	1,5,6	1,4,6	1,4,5	
CV_2	2,6,1	2,5,1	2,5,6	
CV_3	3,1,2	3,6,2	3,6,1	
CV_4	4,2,3	4,1,3	4,1,2	4,6,2
CV_5	5,3,4	5,2,4	5,2,3	5,1,3
CV_6	6,4,5	6,3,5	6,3,4	

TABLE 4.1: Coalition value calculation shares (CV) for all $s = 3$ agent coalitions in an $n = 6$ agent coalition-game.

shared between coalition arrays, the new ordering method introduced in this paper divides the coalitions into 2-dimensional lists $L_{s,\underline{t}}$.

4.1.2 Example

Each *increment array* $\underline{t}$ represents the necessary offset increments from one agent ID of the coalition array to the next. For agent i to generate a coalition C assigned to itself using $\underline{t}$, the first element of the coalition array will be i to *motivate* i to compute the coalition's value. The second agent ID j in the coalition array will be $= (i + t_0 + 1) \bmod n$; and the third agent ID k will be $= ((i + t_0 + 1) + t_1 + 1) \bmod n$. This continues until the coalition's size s limit has been reached.

Table 4.1 presents a subset of the coalition arrays, grouped by IAs of size $s = 3$ for $n = 6$ agents. Each column represents a single list $L_{s,\underline{t}}$ for some IA $\underline{t}$, whereas the rows present the *coalition value calculation shares* (CVs) comprising the different coalition arrays with a common first element (CV_i is agent i's share). The table represents all coalition arrays necessary for coalitions of size $s = 3$. To assign all of the coalitions, multiple IAs are needed; however, every coalition is assigned once and only once. Note that an integer partition may form more than one increment array; for example the two increment arrays $\langle 2, 1, 0\rangle$ and $\langle 2, 0, 1\rangle$ are formed from the $\{2,1\}$ integer partition.

Four different IAs are required for all the coalitions to be allocated in the above example. The IA $\underline{t}^x = \langle 2, 0, 1\rangle$ is valid as $\{2, 1\}$ is a candidate integer partition of $\mathcal{I}(6 - 3)$ that satisfies $|\{2, 1\}| \leq s = 3$. Yet as $|\{2, 1\}| \neq 3$, additional zeros are needed to fill up the IA to make the IA the required size s. If agent 1 used $\underline{t}^x$, the coalition array would comprise:

$$
\begin{aligned}
&= \{1, (i + t_0^x + 1) \bmod n, ((i + t_0^x + 1) + t_1^x + 1) \bmod n\} \\
&= \{1, (1 + 2 + 1) \bmod 6, ((1 + 2 + 1) + 0 + 1) \bmod 6\} \\
&= \{1, 4, 5\}
\end{aligned}
$$

Alternatively if agent 2 used $\underline{t}^x$, the coalition array would comprise:

$$\begin{aligned}
&= \{i, (i + t_0^x + 1) \bmod n, ((i + t_0^x + 1) + t_1^x + 1) \bmod n\} \\
&= \{2, (2 + 2 + 1) \bmod 6, ((2 + 2 + 1) + 0 + 1) \bmod 6\} \\
&= \{2, 5, 0\} \equiv \{2, 5, 6\}
\end{aligned}$$

In the above example, the ID 0 was generated. As $0 \equiv n \pmod n$, this is replaced with ID $= n = 6$ in this coalition.

Each unique IA should be used n times (once for each agent) unless the IA includes a sequence that is repeated throughout the IA. In Table 4.1, $\langle 1, 1, 1\rangle$ is the only IA with a repeated sequence, with $\{1\}$ being repeated $m = 3$ times. The number of times an IA with a repeating sequence should be used relates to the size of the repeating sequence and is given by r (introduced in the next subsection). The choice of agents that should use this type of IA will depend on the allocation of other coalitions; the DCG algorithm balances the number of coalitions assigned to each agent's share so that an agent's share can never be greater than two coalitions bigger than another agent' share.

Thus, if any other IA was used other than the ones listed in Table 4.1, it would result in a coalition's value being calculated more than once. For example, if agent 6 used $t^y = \langle 1, 2, 0\rangle$, the coalition array $[6, 2, 5]$ would be generated despite this coalition being generated by agent 2 using $t^x = \langle 2, 0, 1\rangle$.

4.1.3 A Distributed Method for Coalition Generation

The *period of* $\underline{t}$, denoted by $\pi(\underline{t})$ is defined as:

Definition 78: *The* period of $\underline{t}$*:*

$$\min_{1 \le p \le s} \underline{t} = \langle t_0, t_1, \ldots, t_{p-1}, t_0, t_1, \ldots, t_{p-1}, \ldots, t_0, t_1, \ldots, t_{p-1}\rangle$$

Hence, $\underline{t}$ is formed by m identical copies of a sequence of length $\pi(\underline{t})$. The period of $\underline{t}$ is therefore the length of the smallest subsection of $\underline{t}$ that is repeated throughout $\underline{t}$. For example, if $\underline{t} = [3, 1, 3, 1, 3, 1]$ then the period of $\underline{t}$ is 2 because the subsection $[3, 1]$ is the smallest subsection repeated throughout $\underline{t}$. Additionally if $\underline{t} = [3, 1, 3, 1, 3, 1]$, then $m = 3$ because the subsection $[3, 1]$ is repeated 3 times within $\underline{t}$.

Definition 79: Calculating an agent ag's coalition C given an IA array $\underline{t}$:

Given $C \subseteq \{1, 2, \ldots, n\}$*, agent* ag *generates* C *from* ag *if* $C = \{ag_1, ag_2, \ldots, ag_s\}$ *and:*

$$ag_i = \begin{cases} ag & \text{if} \quad i = 1 \\ (ag + \phi_i) \bmod n & \text{if} \quad (ag + \phi_i) \bmod n \neq 0 \\ n & \text{if} \quad (ag + \phi_i) \bmod n = 0 \end{cases}$$

where:

$$\phi_i = \sum_{k=0}^{i-2} t_k + (i-1)$$

Additionally, $C(ag, \underline{t})$ denotes the subset of $\{1, 2, \ldots, n\}$ generated by the IA $\underline{t}$ from agent ag. The proofs in Appendix A demonstrate that each $\underline{t}$ only needs to be used by $r = (n \times \pi(\underline{t}))/s$ different agents. If more than r agents use $\underline{t}$ to generate a coalition, then repeated coalitions will be generated. For example, if the chosen IA from Table 4.1 is $\underline{t}^q = \langle 1, 1, 1 \rangle$ then $r = (n \times \pi(\underline{t}^q))/s = (6 \times 1)/3 = 6/3 = 2$ agents should use $\underline{t}^q$, which is true as any other agent using $\underline{t}^q$ would repeat either coalition $\{1, 3, 5\}$ or $\{2, 4, 6\}$. Finally, if $\underline{t}^x$ and $\underline{t}^y$ generate the same coalition C for two different agents $i, j \in C$ (i.e. $C(i, \underline{t}^x) = C(j, \underline{t}^y)$), then $\underline{t}^x$ and $\underline{t}^y$ are classified as belonging to the same *equivalence class*, denoted $\underline{t}^x \approx \underline{t}^y$. For example, the IAs $C(2, \langle 2, 0, 1 \rangle) = C(6, \langle 1, 2, 0 \rangle) = \{2, 5, 6\}$ belong to the same equivalence class. We write $\underline{t}^x \approx \underline{t}^y$ when $t^x = \langle t^y_k, ..., t^y_{s-1}, t^y_0, ..., t^y_{k-1} \rangle$ for some $0 \leq k \leq s-1$. To get $\underline{t}^y$ from $\underline{t}^x$, it can be said that $\underline{t}^x$ is *shifted* $(s-k)$ steps. Appendix A proves that rather than considering *every* possible IA, it suffices only to consider a *single* representative from each equivalence class $\approx$. The DCG algorithm is designed to only consider a single representative of each class, the one that is the smallest lexicographically, named the *canonical representative* for its equivalence class and denoted $[\underline{t}]_\approx$.

4.2 The Distributed Coalition Generation (DCG) Algorithm

Given the definitions presented in the previous section, the DCG algorithm for distributed coalition value calculations can now be presented.

The DCG algorithm focuses on finding the *canonical representative* for each equivalence class, i.e. the lexicographically smallest IA in each equivalence class. The intuition of the algorithm is as follows, given the number of agents n, for each coalition size s, the DCG seeks to find all IAs that total $(n-s)$ that are not a member of an equivalence class previously found.

One method of guaranteeing that all the canonical representatives are found, is to generate all possible IAs that total $(n-s)$ and then test all of them to make sure that they are the lexicographically smallest. To do this an indexing scheme could be created. This indexing scheme relies on the function, $place : \underline{t} \rightarrow \mathbb{N}_0$ that maps an IA, $\underline{t} = \langle t_0, \ldots, t_{s-1} \rangle$ to a non-negative integer by:

$$place(\langle t_0, t_1, \ldots, t_{s-1} \rangle) \;=\; \sum_{i=0}^{s-1} (t_i \times (n-s+1)^i)$$

Informally, $place(\underline{t})$ treats $\underline{t}$ as an integer expressed in base $n-s+1$. With this convention, the number of distinct IAs is bounded by $(n-s+1)^s$. But generating every possible IA and testing it, is a great inefficiency for even small numbers of n as there may be many IAs in the

same equivalence class. For example when $n = 5$, $\sum_{s=1}^{s=n}(n-s+1)^s = (5+16+27+16+1) = 65$. Yet as detailed in Section 4.4, for $n = 5$ there are only 7 canonical representative IAs[3].

To find all the canonical representatives, DCG generates the IAs in lexicographical order and tests a subset of them to check that each tested IA is the lexicographically smallest in its equivalence class. As n increases, DCG skips over an ever increasing number of IAs in the lexicographical order. To describe how this is done, the `correct` function in Algorithm 4 will be introduced. To understand the intuition behind Algorithm 4, the following example is used:

Example 4.1. *Given $n = 17$ and $s = 6$, suppose the last IA generated was $\underline{t} = \langle 0, 3, 0, 5, 3, 0\rangle$. It can be seen that $\underline{t}$ is not the canonical representative because there is a way that the numbers in $\underline{t}$ can be* shifted *to make another IA $\underline{u}^x$ so that $\underline{u}^x \approx \underline{t}$ and $[\underline{u}^x]_{\approx}$. In this case $\underline{u}^x = \langle 0, 0, 3, 0, 5, 3\rangle$.*

To try to find the next canonical representative from $\underline{t}$, the next IA in lexicographical order could be generated. Remember this is not *$\underline{u}^y = \langle 0, 3, 0, 5, 3, 1\rangle$ because now $\underline{u}^y$ sums to greater than $(n - s)$. So the next IA in lexicographical order from $\underline{t}$ is formed via some increment and decrement of positions in $\underline{t}$. In this case the `correct` function increments $t_3 = 6$, decrements $t_4 = 0$ and increments $t_5 = 2$, giving $\underline{t}^1 = \langle 0, 3, 0, 6, 0, 2\rangle$. But $\underline{t}^1$ is not the canonical representative of its class because $\underline{t}^1$ could be shifted two places to the right, giving $\underline{u}^z = \langle 0, 2, 0, 3, 0, 6\rangle$. As $\underline{t}^1$ failed the canonical representative check, the `correct` function would find the next IA in lexicographical order by incrementing and decrementing positions of $\underline{t}^1$ to give $\underline{t}^2 = \langle 0, 3, 0, 6, 1, 1\rangle$. The IA $\underline{t}^2$ is a canonical representative as there is no way to shift $\underline{t}^2$ to generate another IA that is lexicographically smaller.*

Now how the `correct` function skips over some IAs is as follows. Given $\underline{t}^3 = \langle 0, 3, 0, 6, 2, 0\rangle$ for $n = 17$ and $s = 6$, the `correct` function would generate $\underline{t}^4 = \langle 0, 3, 0, 7, 0, 1\rangle$ and set a boolean variable $hold_3 = true$ to indicate the lowest part of the IA that has just been incremented (i.e. t_3^4). Now if a canonical representative is not found for any IA starting with the prefix $[0, 3, 0, 7]$ then incrementing the same t_3^4 will not find a canonical representative either[4]. So in this case no IA starting with the prefix $[0, 3, 0, 8]$ is generated or checked. Instead the `correct` function skips over t_3^4 and increments t_2^4, giving $t^5 = [0, 3, 1, 0, 0, 7]$ (which will now get tested and the algorithm will continue).

Now that an example has been detailed, the `correct` function can be introduced after the following variables and notational conventions:

msd: The value t_0, i.e the most significant "digit" of the IA $\underline{t}$. To begin with $msd := 0$.

`EditIA()`: This function takes $(\underline{t}, j, msd)$ as input and returns $\underline{t}'$ where: $t'_x = msd$, $\forall t'_x$ where $x \in \{j + 1, j + 2, ..., s - 2\}$; $t'_y = t_y$, $\forall t'_y$ where $y < j + 1$; and $t'_{s-1} = (n - s) - \sum_{i=0}^{s-2} t'_i$.

$complete$: This is a global boolean variable that is set as $false$ until the `correct` function can find no more possible IAs.

[3] For completeness, these 7 are: $[4]$; $[0, 3]$; $[1, 2]$; $[0, 0, 2]$; $[0, 1, 1]$; $[0, 0, 0, 1]$; and $[0, 0, 0, 0, 0]$.

[4] This holds true apart from the special case when $t_3^4 <= \frac{n-s-1-msd}{2}$

Algorithm 4: Finding the next IA to be lexicographically tested.

```
1:  function correct (t) returns s-tuple;
2:  Input: ⟨t⟩; where t is the last IA tested to see it if its the canonical representative.
3:  Output: ⟨t_0, t_1, ..., t_{s-1}⟩; the next IA to be lexicographically tested.
4:  begin
5:  int sum := t_{s-1};
6:  for (int j := s − 2; j >= 0; j − −) do
7:     if (t_j + sum == (n − s)) and (hold_j ≠ true) then
8:        hold_j = true;
9:        t_j + +;
10:       sum − −;
11:       if j == 0 then
12:          msd + +;
13:       end if
14:       return EditIA(t, j, msd);
15:    else if (sum ≠ 0) and ((hold_j ≠ true) or (t_j < (n−s−1−msd)/2)) then
16:       hold_j := true;
17:       t_j + +;
18:       sum − −;
19:       return EditIA(t, j, msd);
20:    end if
21:    sum := sum + t_j
22: end for
23: complete := true;
24: return null;
25: end
```

hold: This is a global boolean array variable that is set in the build function to $false$ for every position, until the `correct` function changes parts of it to $true$ to help skip IAs.

The `correct` function in Algorithm 4 begins by being given an IA $\underline{t}$ that is not a canonical representative and setting the current sum to equal the last value in the IA (line 5). The `correct` function starts from the second to last digit of $\underline{t}$ and continues down to the first digit, until a digit is found that can be incremented by one (line 6). For every loop the current sum of all the digits considered is recorded (line 21). A digit can be incremented by one if it is non-zero, it is in the smallest position of the IA for all the non-zero digits, and this position has not previously been incremented (since the last canonical representative IA was found) (line 7). Additionally a digit can be incremented if the current sum is greater than zero, this digit is not in a position that has already been incremented (since the last canonical representative IA was found) and this digit is less than $\frac{n-s-1-msd}{2}$ (to make sure all canonical representatives get tested) (line 15). If the digit incremented was the digit at position zero, then msd must be incremented as this is the smallest value any digit can now take (line 12). Regardless of which digit is incremented by one, the correct $hold$ position is set to $true$ (lines 8 and 16), the sum is decremented by one (lines 10 and 18) and the rest of the IA is edited to become the smallest that it can be in lexicographical order (lines 14 and 19). Finally if no digits satisfy the increment conditions then $complete$ is set to $true$ and $null$ is returned.

Algorithm 5: Finding canonical representative IAs from equivalence classes of $\approx$.

1: **function** `build` ($\underline{t}$) **returns** s-tuple;
2: **Input:** $\langle \underline{t} \rangle$; where $\underline{t}$ is the last canonical representative IA used.
3: **Output:** $\langle t_0^k, t_1^k, \ldots, t_{s-1}^k \rangle$; the next canonical representative IA of a class not used so far.
4: **begin**
5: $t_{s-1} - -$;
6: $t_{s-2} + +$;
7: **if** $(s \neq 2)$ **and** $(t_{s-1} > msd)$ **then**
8: **return** $\underline{t}$;
9: **else if** $(s == 2)$ **and** $(t_{s-2} \leq t_{s-1})$ **and** $(t_{s-1} > msd)$ **then**
10: **return** $\underline{t}$;
11: **else if** $(s == 2)$ **then**
12: $completed := true$;
13: **return** $null$;
14: **end if**
15: $hold := \langle false, false, ..., false \rangle$;
16: **repeat**
17: $\underline{t} =$ `correct`($\underline{t}$);
18: **until** $((\underline{t} == [\underline{t}]_{\approx})$ **or** $(completed == true))$
19: **return** $\underline{t}$;
20: **end**

Now that the `correct` function has been introduced, the `build` function that finds the next canonical representative of each class can be detailed, as seen in Algorithm 5. Remember that canonical representatives are found in lexicographical order. For example, given $n = 6$ and $s = 4$, the canonical representative IA $\langle 0, 0, 0, 2 \rangle$ is generated first and then $\langle 0, 0, 1, 1 \rangle$.

The simplest manner to generate the next canonical representative IA is to decrement the last digit of the IA by one and increment the last but one digit by one. When the size of the coalition is greater than two, then there will exist a digit with the msd value in at least the first position. Therefore the only way an IA generated in this manner cannot be an canonical representative IA is when the last digit is less than or equal to msd. In this case, the IA can be shifted one place to the right to generate a lexicographically smaller IA. For example when $n = 5$, $s = 3$ and $msd = 0$, the IA $\underline{t}^x = \langle 0, 1, 1 \rangle$ is the canonical representation, but the next IA generated $\underline{t}^y = \langle 0, 2, 0 \rangle$ is not because $t_2^y = msd$ and therefore $\underline{t}^z = \langle 0, 0, 2 \rangle$ is lexicographically smaller. This exception is caught in the if statement of line 7.

When the coalition is of size two, then an IA generated in this manner may not be a canonical representative IA, when the first digit becomes greater than the last digit. This is because the lexicographical smallest IA of this class has already been generated, as can be seen by swapping the digits around. For example if the IA generated is $\langle 3, 2 \rangle$, the canonical representation of this class is $\langle 2, 3 \rangle$ and was generated previously. This exception is dealt with in the if statements of lines 9 and 11.

Now the later part of the `build` function, from line 15 onwards, is only reached if this simple generation method described in the preceding paragraphs has not found a canonical representative IA. In this case, the `correct` function is called to generate another IA, where some

Algorithm 6: The Distributed Coalition Generation (DCG) method.

```
1:  function DCG (int n, i)
2:  Input: ⟨n, i⟩ (1 ≤ i ≤ n); where n is the number of agents and i the agent ID.
3:  begin
4:  int bal := 1;
5:  v({i});
6:  j := i + 1 mod n;
7:  v(N\{j});
8:  for (int s = 2; s ≤ n/2; s++) do
9:     boolean completed := false;
10:    int msd := 0;
11:    int[] t := ⟨0, 0, ..., n − s⟩;
12:    while (complete == false) do
13:       p := π(t);
14:       r := (n × p)/s;
15:       if (bal ≤ i < bal + r) or (bal + r > n and 1 ≤ i < bal + r − n) then
16:          v(C(i, t));
17:          if s ≠ n/2 then
18:             v(N\C(j, t));
19:          end if
20:       end if
21:       bal := (bal + r) mod n;
22:       t = build(t);
23:    end while
24: end for
25: if (i == bal) then
26:    v(N);
27: end if
28: end
```

possible IAs may be skipped. The IA generated via the `correct` function is lexicographically tested to see if it is the canonical representative class (line 18). If IA is not the lexicographically smallest IA of its class then the `correct` function is called again until a canonical representative is found or no more possible IAs can be made.

Now finally the approach each agent i uses to generate all of its coalitions in its coalition value calculation share is presented in the `DCG` algorithm in Algorithm 6, which exploits the `build` and `correct` functions. The `DCG(n,i)` algorithm starts by initializing the *bal* pointer (line 4), which is used to distribute the coalitions in an approximately equal manner. Next each agent i should calculate its own value and the value of the coalition of size $n-1$ that does not include agent j (line 5-7). The agents now look for canonical representative IAs for sizes 2 to $\lfloor \frac{n}{2} \rfloor$ (line 8). The agents do not look for canonical representatives of every size as it is quicker to find the complement of agent j, where agent j is one increment greater than agent i's ID (when $\bmod n$ is used). The complement of agent j's coalition will include agent i because of the baseline increment assumption described previously.

For each s used to search for canonical representative IAs, the first canonical representative is always the lexicographical minimal (line 11). Then the while loop (lines 12-23) finds out how

many agents should calculate a coalition using $\underline{t}$ (line 14) and allows the agent to use the IA if that agent is one of the next r agents according to the bal pointer (line 15 and 16). Additionally, if the agent was allowed to calculate the value, it will also calculate the value of a coalition of size $n - s$, if $s \neq \frac{n}{2}$ (lines 17-19). The while loop then continually calls the `build` function of Algorithm 5 (line 22) to find the next canonical representative IA. When $complete$ has been set to true in the `build` function or the `correct` function, this indicates that all the coalitions of size s for agent i's calculation share have been found. Finally, the grand coalition is assigned to the agent who, according to the bal pointer, is the next agent to calculate a coalition (line 26).

A note here should be made on how the memory requirements of the DCG algorithm grows linearly as the number of agents n increases. This is because the maximum size of the $\underline{t}$ and $hold$ array is $\frac{n}{2}$ (according to line 8), while the maximum size of a coalition generated is n (according to line 26).

4.3 Discussion

The following is a discussion on how the DCG algorithm presented in this chapter, compares to the DCVC algorithm [89, 90], the VBFR algorithm [126] and the SK algorithm [117], when judged on the following properties identified in the introduction of this chapter: (i) elimination of communication; (ii) elimination of coaltion value calculation redundancy; (iii) approximately equal coalition value calculation shares for each agent; (iv) agents with equal sized shares have the same number of operations to perform; and (v) each coalition in an agent's share includes that agent as a member.

(i) **Elimination of communication:** The method to split up the calculations of the coalition values in the DCG algorithm has no communication costs (as you can see there is no where in the DCG algorithm that requires communication between the agents). No communication costs occur because, like the DCVC and VBFR algorithms, the agents have: (1) a pre-agreed ordering of the coalitions; and (2) a pre-agreed algorithm to dictate which exact share of this ordering each agent should calculate. Conditions (1) and (2) are not the case for the SK algorithm and so agents using this algorithm need to communicate to find out which coalitions will be in each agent's share. Yet for any distributed coalition value calculation algorithm, if the agents want to complete the decentralised coalition formation process, communication costs will be incurred later when agents communicate to each other their best coalitions or coalition structures found (e.g. [1, 73, 104]). Even when the best coalition structure is known, more communication will occur if the agents want to negotiate on their final utility payoff (e.g. [1, 7, 30, 59, 67, 118, 137]). How the agents can communicate the correct coalitions (given coalition shares according to the DCG algorithm), constrain communication costs and achieve a certain stable outcome is described in Chapter 5.

(ii) **Elimination of Redundancy:** In the DCG algorithm: (1) one and only one representative increment array from each equivalence class is used. This single representative increment

array $\underline{t}$ is only used: (2) $(n \times \pi(\underline{t}))/s$ times; and (3) no more than once for each agent. In Appendix A it is proved that a repeated coalition can only be generated from an increment array if any of (1), (2) and (3) are not abided to. The DCVC algorithm and VBFR algorithm also do not have any redundant coalition value calculation issues because the agents are aware how the coalitions are ordered in both algorithms and therefore each agent's share does not accidentally overlap. The SK algorithm does involve an exponentially large redundancy because, as stated in [90], each agent commits to calculate the value of a set of coalitions with limited knowledge on the other agent's commitments.

(iii) **Approximately equal coalition value calculation shares:** The DCG algorithm, like the DCVC algorithm, gives approximately equal shares. The DCG algorithm has a maximum difference of two between the agent's coalition value calculation share due to the each agent generating 2 coalitions at a time. The DCG algorithm achieves this through the use of the bal pointer, while the DCVC algorithm uses the α pointer (where the α pointer is described in Section 2.4.1).

The bal pointer is initialised to 1 in line 4 of the DCG algorithm. Then bal lets the next r agents calculate a maximum of two coalition values, after that, bal is corrected (line 21). As bal only allows a maximum of two coalitions to be calculated at a time, an agent can only be a maximum of two coalition assignments ahead of another agent.

The VBFR algorithm does not give balanced shares to the agents (in the case where all the agents are assumed to be in a fully connected graph). In this case, for n agents, agent 1 is always assigned $2^n/2$ coalitions because there are $2^n/2$ coalitions where agent ID 1 is the smallest ID of that coalition. Agent n on the other hand is always assigned only one coalition because there is only one coalition with agent n as the smallest ID, which is the singleton coalition $\{n\}$. Finally the SK algorithm has no guarantees on the maximum difference between the agent's shares, yet the average difference grows exponentially with the number of agents, as detailed in [90].

(iv) **Equal number of operations for agents with equal sized shares:** Only the DCG algorithm and the DCVC algorithms guarantee that the agents coalition value calculation shares will be approximately equal. Yet [90] shows that the DCVC algorithms do not guarantee an equal number of operations[5] to generate each agent's coalition value calculation share, even when each agent's share is equal sized. This is due to the lexicographical ordering of the coalitions that the DCVC algorithm uses. For the DCG algorithm equal operations will be performed by agents that are allocated an equal number of coalitions of each size, as each IA of the same size requires the exact same number of operations of additions to find the corresponding coalition (see Definition 79). This is because the DCG algorithm, unlike the DCVC algorithm, does not rely on lexicographical order to generate the coalitions. Instead two-dimensional lists based on increment arrays are used. For every increment array of size s, an agent needs to use $s-1$ additions to generate the associated coalition, even if the increment arrays are different. The increment arrays used by the agents are the only

[5]Where the operations are comparisons and additions.

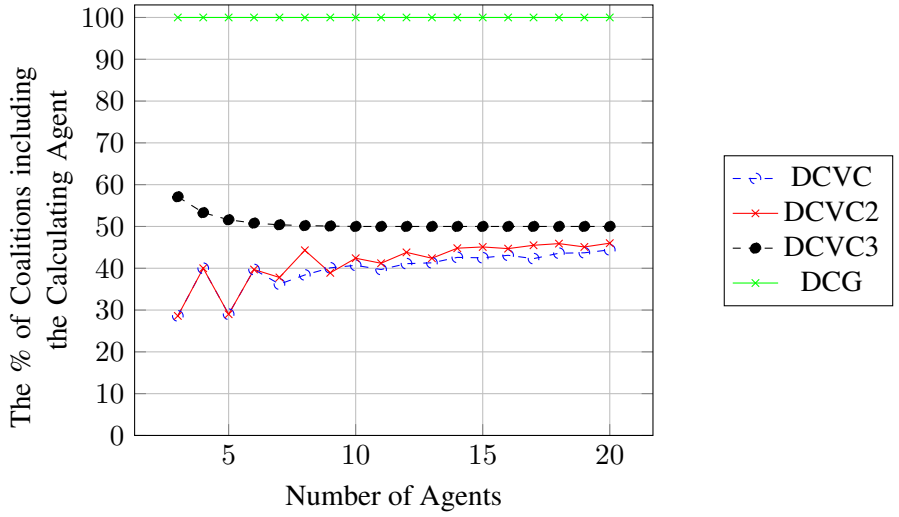

FIGURE 4.1: This graph shows, for an n agent game, the percentage of coalitions in all $CV_{i \in N}$ that include agent i (for all sizes $1 \leq s \leq n$). It was produced from a java implementation of all the algorithms, where each agent kept a count of each coalition assigned to itself that included itself.

variable points of the DCG algorithm. Therefore for any two agents that generate the same number of coalitions for every s, then the same number of increment arrays are used, thus the same number of operations are used.

(v) **Agents are members of all their assigned coalitions:** The DCG algorithm, like the SK and VBFR algorithms before it, guarantee that every coalition distributed to an agent i, includes i as a member. In the DCG algorithm, this is achieved by assigning agent i to be the first agent in every coalition that it generates (see Definition 79).

This is not the case in any of the DCVC algorithms, as can be seen in Figure 4.1 (that shows the three different types of the DCVC algorithm described in Section 2.4.1). This is a weakness for the DCVC algorithm for certain domains. Consider any e-commerce environment where agents (representing a single business) will join with others in a temporary coalition if they can gain price discounts and economics of scale. As there maybe anti-monopoly penalties or other considerations, such as communication and logistic costs, then this problem cannot simply be solved by forming the grand coalition. Therefore the agents together will have to calculate the values of the different coalitions to find their most preferable ones. If the variables involved in the calculation of each coalition's value are public knowledge, then this situation can be treated as a characteristic function game and the coalition value calculation costs can be shared around the agents of the system. In this situation it makes no sense for an agent to calculate the value of a coalition C that does not include itself because the agent will gain no benefit if C were to form.

When using the DCG algorithm with self-interested agents, possible manipulations could occur by for example, misreporting the value of a certain coalition. In this case the tools of mechanism design [80] may have to be used, where the DCG algorithm can be wrapped

Algorithm	Eliminates Communication	Eliminates Redundancy	Equal Shares	Equal Operations	Agent in All its Assigned Coalitions
SK	No	No	No	No	Yes
DCVC	Yes	Yes	Yes	No	No
VBFR	Yes	Yes	No	No	Yes
DCG	Yes	Yes	Yes	Yes	Yes

TABLE 4.2: A summary of the comparison between the DCVC, VBFR, SK and DCG algorithms that distribute coalition value calculations.

in some form of mechanism that constitutes a Bayesian game. If this mechanism has Bayesian equilibria and the optimal strategy for each agent is to misreport information, then via the *revelation principle* [76, 119] their exists another payoff-equivalent individually rational mechanism that has an equilibrium where the players truthfully report their types[6] [76, 119].

This discussion, summarised in Table 4.2, has shown that the DCG algorithm satisfies all the above properties, which none of the DCVC, VBFR or SK algorithms can do.

4.4 Performance Evaluation

To evaluate the performance of the DCG algorithm, the running time of the algorithm is compared to the benchmark of the running time of generating all coalitions that include the same agent, which is named *AgentInAll*[7]. Generating every coalition that an agent is in, is exactly what has to occur for the agent with the lowest ID in the VBFR algorithm. SK was not included in the performance evaluation as the exponientially increasing communication and memory requirements significantly slow the algorithm down compared to the AgentInAll approach, as discussed in [90].

By evaluating the time that AgentInAll takes, the benefits of balancing the computation around the agents of the system using IAs, while keeping the key property that every agent is assigned coalitions in which it is a member, can be tested. For completeness, DCG is additionally compared to DCVC, the fastest algorithm for coalition value calculations, but the reader is reminded that DCVC may assign coalitions to agents that do not include that agent.

Three figures as used to present the coalition value calculation time of the algorithms, Figure 4.2 for a coalition valuation function of $O(s)$ complexity, Figure 4.3 for a coalition valuation function of $O(s^2)$ complexity and Figure 4.4 for a coalition valuation function of $O(2^s)$ complexity[8]. For all the experiments of each complexity class, for all the algorithms, the coalition

[6] The search for this truthful mechanism is left for future work.

[7] The PC on which the simulations were run had a processor: Intel(R) Core(TM) i7-4810MQ 2.80 GHz, with 8GB of RAM.

[8] All Figures use a 95% confidence interval, which are all small relative to the y axis (that displays the milliseconds each algorithm takes to complete).

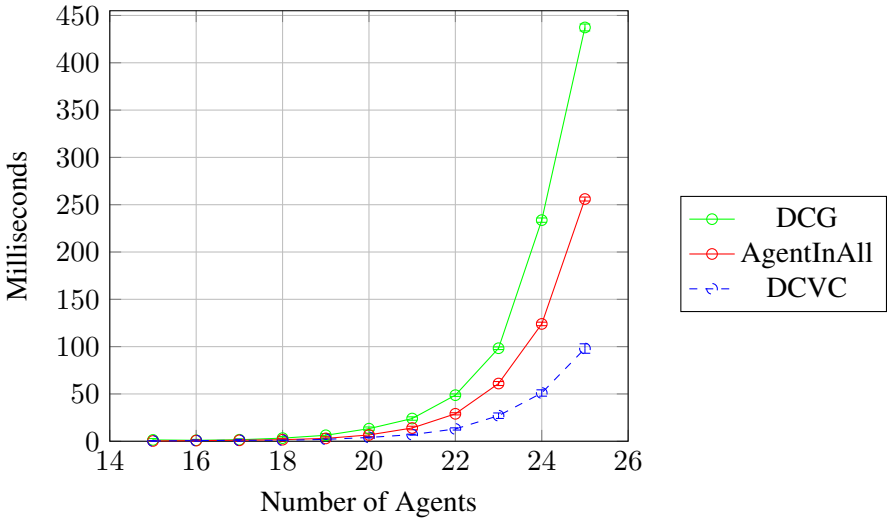

FIGURE 4.2: This graph shows the computation time for the agents calculating their coalition values using different possible algorithms, when the cost of calculating each coalition's value is $O(s)$.

value calculation function was exactly the same, consisting of s, s^2 or 2^s multiplication operations.

Starting with a negative result, it can be seen by looking at Figure 4.2 that the AgentInAll algorithm is quicker than DCG when the complexity of the coalition value calculation function is linear according to the size of the coalition. This result can be explained by the fact that the additional overhead of finding all canonical representative IAs has a larger cost than calculating the value of the full set of coalitions an agent is in, when the valuation function is linear according to the coalition's size. So the current implementation of DCG should not be used when the coalition value calculation function cost is relatively inexpensive. That said, a quicker algorithm than DCG that finds each canonical representative IA could be found in the future.

Figure 4.2 also shows that the DCVC algorithm remains the quickest coalition value calculation algorithm, but as all the algorithms involve the distribution of an exponentially increasing number of coalitions, all the different algorithms' computation times will increase exponentially.

Moving onto positive results, consider Figure 4.3, where it can be seen that there is an advantage of using the DCG algorithm instead of calculating all coalitions the agent is in. For a coalition value calculation function of $O(s^2)$, the DCG algorithm delays the significant time cost of calculating each agent's assigned coalitions longer than the AgentInAll algorithm, while the DCVC algorithm delays the significant time cost longer than the DCG algorithm. Now looking at Figure 4.4, it can be seen that the advantage of using the DCG algorithm instead of calculating all the coalitions the agent is in, increases the more complex the coalition valuation function is. The AgentInAll algorithm's computation time grows a lot faster than the other two algorithms. For instance at $n = 25$, the AgentInAll algorithm took approximately $10\frac{1}{2}$ minutes to complete, compared to DCG that took approximately 4 seconds and DCVC that took approximately 0.1 seconds. In this figure, DCVC has again not increased its execution time to the level of the other algorithms as it has still managed to delay the significant exponential time growth until after $n = 25$.

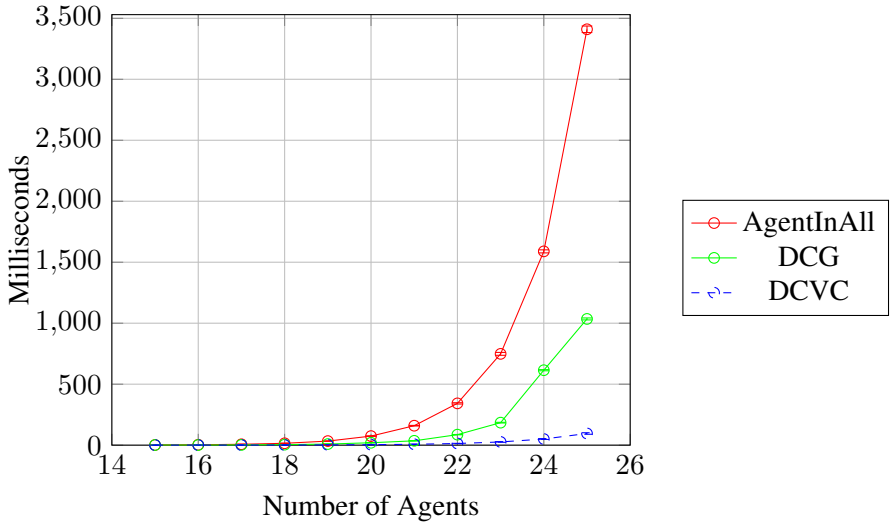

FIGURE 4.3: This graph shows the computation time for the agents calculating their coalition values using different possible algorithms, when the cost of calculating each coalition's value is $O(s^2)$.

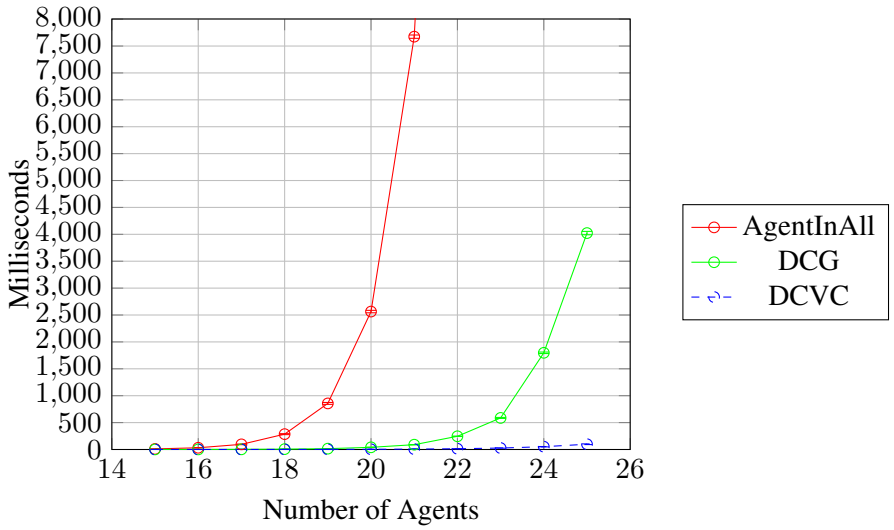

FIGURE 4.4: This graph shows, the computation time for the agents calculating their coalition values using different possible algorithms, when the cost of calculating each coalition's value is $O(2^s)$.

Finally the number of canonical representative IAs needed to distribute all coalitions were found through the implementation of the DCG algorithm. These results are presented in Figure 4.5 for low number of n because the growth of the canonical representative IAs, like the coalitions themselves are exponential according to the number of agents n. The main issues to point out here are the following:

- The number of canonical representative IAs are significantly less than the number of possible coalitions $2^n - 1$ and are approximately $\frac{2^n-1}{n}$. For example the number of canonical representative IAs for $n = 14$ is 1178, while $\frac{2^n-1}{n} = \frac{16383}{14} = 1170$.

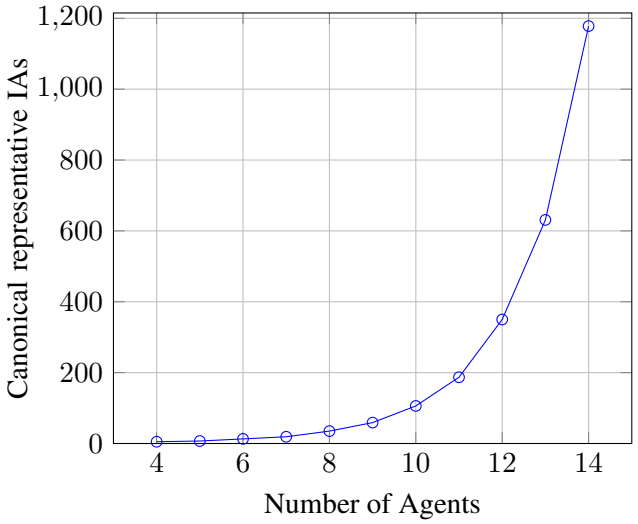

FIGURE 4.5: This graph shows the number of canonical representative IAs.

- The number of canonical representative IAs are significantly less than the bound on the number of possible IAs, which is $(n - s + 1)^s$, as described earlier.

The DCG does not need to explicitly find all these canonical representative IAs. Each agent using the DCG algorithm does not use an IA to generate its singleton coalition and any coalition of size greater than $\lfloor \frac{n}{2} \rfloor$, as explained in Section 4.2.

4.5 Towards Distributed Coalition Formation

The DCG algorithm is only designed to organise agents for the first stage of coalition formation. To complete the coalition formation process in a desirable manner, the agents will need to collaborate to find an optimal coalition structure and a stable payoff vector. Additionally, the approach that the agents take, needs to take into account the distributed knowledge of the coalition values that the DCG algorithm has created. This is the aim of the *distributed dynamic programming* (DDP) algorithms introduced in the next chapter.

Chapter 5

Chapter 5

A Distributed Search for the Superadditive Cover Least Core

The computational cost of calculating the value of every coalition grows exponentially as the number of agents in the system increases. To tackle this issue, these calculations can be distributed among the agents of the system, as the previous chapter suggests. Yet distributing the value calculations means that each agent of the system will have only partial, incomplete knowledge of the characteristic function game.

Given this distributed knowledge, this chapter investigates and provides a method to solve the following issues[1]: (1) how can a stable core-based solution to a characteristic function game be guaranteed to be found in a decentralised manner?; and (2) how can this stable solution be found when the ϵ values of the stable core-based solutions are not known beforehand?

The decentralised dynamic programming algorithm introduced in this chapter, named DDP, is a modification of the dynamic programming (DP) algorithm introduced in [138] and detailed previously in Section 2.4.2. The DDP algorithm shows how the agents can complete all three stages of the coalition formation process in a distributed manner. It can be used for any characteristic function game, and is guaranteed (through Lemma 5.1, Theorem 5.2 and Corollary 5.3) to find a coalition structure and payoff vector distribution in the solution concept of the *weak least core* for the superadditive cover of the characteristic function game when cross-coalition side payments are allowed[2]. This guarantee comes from the fact that the DP and DDP algorithm can be used to identify the coalitions in the synergy coalitional group representation [41, 81].

The contributions of the DDP algorithm are that: (i) a stable solution is found with distributed knowledge, which is equivalent to the complete knowledge solution; (ii) this stable solution is guaranteed to be in the *weak-least core*$^+$ solution concept for the superadditive cover of the characteristic function game, where the plus denotes that cross-coalition side payments are allowed; and (iii) every agent is motivated to perform its part of the algorithm.

[1] An early attempt at solving issues (1) and (2) of this Chapter was presented via an argumentation-based dialogue game in [100].

[2] I.e. The total payoff the agents of a formed coalition receive can be less than the coalition's total value when cross-coalition side payments are allowed.

The rest of this chapter is structured as follows: Section 5.1 provides a detailed discussion on the solution concept guarantees of the DDP algorithm; Section 5.2 details the DDP algorithm; Section 5.3 describes how the communication costs of DDP can be lowered and constrained; Section 5.4 provides an example of how the DDP algorithm works, once the communication costs have been constrained; Section 5.5 evaluates the DDP algorithm according to the communicated information, the number of agent operations, and the solution concept success rate; Finally Section 5.6 concludes.

5.1 Guaranteeing Stable Solutions

When the characteristic function game is superadditive (see Section 2.1 for definition), then the grand coalition is the optimal coalition structure [36]. In this chapter, the *synergy coalitional group* (SCG) representation (detailed in Section 2.4.1) is used to guarantee that a least core stable solution is found. In SCGs, a coalition and coalition value pair (i.e. $(C, v(C))$) is explicitly represented within the set of *synergy coalitions*, denoted W, if that coalition C earns more utility from forming then any possible partition of C could receive. The values of any other coalition S not represented within W is found by finding the maximum value partition of S made up of coalitions explicitly represented within W. In Lemma 2 and Theorem 4 of [41] it is proven that the set W allows a core solution to be found if the core is non-empty. This Lemma and Theorem have been modified below to show that a payoff vector in the weak least core can also be guaranteed to be found for the grand coalition using the W set:

Lemma 5.1. *Given a superadditive characteristic function game $\mathcal{G} = \langle N, v \rangle$ and a payoff vector x (where $x(N) = v(N)$), let ϵ^w be the maximum weak excess value that still gives a blocking coalition for x given full knowledge on each coalition's value. Using only the coalition values in W, a blocking coalition D for this maximum weak excess value will be present within W.*

Proof. Suppose x is blocked by a coalition C through $v(C) - |C|\epsilon^w$ so that $v(C) - |C|\epsilon^w > x(C)$ and $\forall \epsilon^{w\prime} > \epsilon^w$ there is no blocking coalition. If $(C, v(C)) \in W$ then this proves the Lemma. If $(C, v(C)) \notin W$ then from the definition of a SCG, it is known that the value of the coalition can be found through its maximum value partition, i.e. $v(C) - |C|\epsilon^w = \sum_{1 \leq p \leq q}(v(C_p) - |C_p|\epsilon^w)$, for some set of coalitions $\{C_1, ..., C_q\}$ where:

1. $\bigcup_{p=1}^{q} C_p = C$
2. $C_i \cap C_j = \emptyset$ for any $i, j \in \{1, ..., q\}$ where $i \neq j$
3. $(C_p, v(C_p)) \in W$, for all $C_p \in \{C_1, ..., C_q\}$

Via substitution, it follows that $\sum_{1 \leq p \leq q}(v(C_p) - |C_p|\epsilon^w) > x(C)$ and hence for at least one C_p then $v(C_p) - |C_p|\epsilon^w > x(C_p)$. Therefore if the value of a coalition C that is not in the SCG representation blocks a payoff vector x given the maximum weak excess ϵ^w penalty that still gives a blocking coalition, it has been shown that there exists a coalition $C_p \subset C$, that also blocks x given the maximum weak excess ϵ^w penalty, yet C_p is represented explicitly within the SCG representation (i.e. $(C_p, v(C_p)) \in W$). Thus the proof of the Lemma is complete. □

Lemma 5.1 shows that given a superadditive characteristic function game, the grand coalition and a payoff vector distributing the grand coalition's value, a coalition that blocks the payoff vector for the maximum weak excess (that allows a blocking coalition) will be present in the SCG representation. For further understanding, consider the following example:

Example 5.1. *Consider a characteristic function game with $N = \{1, 2, 3, 4\}$ agents, where the coalition $C = \{1, 2, 3\}$ has a value of $v(\{1, 2, 3\}) = 25$. Given a payoff vector x, that gives the coalition C the total value $x(C) = 18$, it is known that the maximum (integer)* weak *excess to still give a blocking coalition is $\epsilon^{w\prime} = 2$. For instance $x(C) = 18 < v(C) - |C|\epsilon^{w\prime} = 25 - (3 \times 2) = 25 - 6 = 19$, and so coalition $\{1, 2, 3\}$ is a blocking coalition for x when $\epsilon^{w\prime} = 2$ but not a blocking coalition when $\epsilon^{w\prime} = 3$.*

If $C \in W$ then a blocking coalition for the maximum weak excess value (that still admits a blocking coalition) has been found in the SCG representation. If $C \notin W$ then there must be coalitions within W that make up a partition of C and have an equal or greater combined value then C. In this example assume $(\{1, 2\}, v(\{1, 2\}) = 16), (\{3\}, v(\{3\}) = 9) \in W$. The values of these coalitions have been chosen because $v(\{1, 2\}) + v(\{3\}) = v(\{1, 2, 3\})$, i.e. the values are the minimal needed to make sure that coalition $\{1, 2, 3\}$ is not in the SCG representation.

Given theses preliminaries, to stop coalition $\{1, 2\}$ being a blocking coalition when $\epsilon^{w\prime} = 2$, then $x(\{1, 2\})$ must be greater than or equal to $v(\{1, 2\}) - |\{1, 2\}|\epsilon^{w\prime} = 16 - (2 \times 2) = 12$. Assume that $x(\{1, 2\}) = 12$ (i.e. the minimal payoff to satify the coalition has been given).

Recall $x(C) = 18$. Therefore $x_3 = x(C) - x(\{1, 2\}) = 18 - 12 = 6$. But this gives $x(\{3\}) = 6 < v(\{3\}) - |\{3\}|\epsilon^{w\prime} = 9 - (1 \times 2) = 9 - 2 = 7$ and so the singleton coalition $\{3\}$ is a blocking coalition for x when $\epsilon^{w\prime} = 2$ and this blocking coalition has been found in the SCG representation.

In conclusion this example shows that if a blocking coalition C with the maximum weak excess (that still admits a blocking coalition) is not *present in the SCG representation, then a coalition $C' \subset C$ with the same weak excess value will be present in the SCG representation.*

The following theorem shows that given the grand coalition for a superadditive game, the coalitions in the SCG representation can be used to find a payoff vector within the weak least core:

Theorem 5.2. *For the grand coalition, a payoff vector that minimises the maximum weak excess can be found using only the coalitions in W.*

Proof. The linear program to minimise ϵ^w is:

$$\min \epsilon^w \text{ subject to:} \tag{5.1}$$

$$x_i \geq 0 \text{ for each } i \in N \tag{5.2}$$

$$x(N) = v(N) \tag{5.3}$$

$$x(C) \geq v(C) - |C|\epsilon^w, \text{ for all } (C, v(C)) \in W \tag{5.4}$$

Only the coalitions in W are needed to be used in this linear program, according to Lemma 5.1. □

Before introducing the corollary to this theorem, a formal definition regarding cross-coalition side payments are required, which are shown through the weak ϵ-core$^+$ and weak least core$^+$ definitions:

Definition 80: *The* **weak ϵ-core$^+$***:- For a characteristic function game $\mathcal{G} = \langle N, v \rangle$, a coalition structure and payoff vector pair $\langle CS^*, x \rangle$ is in the weak ϵ-core$^+$ iff:*

$$\sum_{i \in N} x_i = CS^* \tag{5.5}$$

$$\sum_{i \in C} x_i \geq v(C) - |C|\epsilon, \ \forall C \subseteq N \tag{5.6}$$

The difference of the weak ϵ-core$^+$ compared to the weak ϵ-core of Section 2.2.1 is that a weak ϵ-core$^+$ payoff vector totals the value of the optimal coalition structure, which may not be the grand coalition. The difference of the weak ϵ-core$^+$ compared to the ϵ-CS-core of Section 2.2.3 is that the weak ϵ-core$^+$ does *not* have the condition that all the payoff of each coalition in the coalition structure is given to that coalition (i.e. $x(C) = v(C), \forall C \in CS^*$ does *not* have to hold). Given the definition of the weak ϵ-core$^+$, the weak least core $^+$ can be defined:

Definition 81: **weak least core$^+$***:- For a characteristic function game $\mathcal{G} = \langle N, v \rangle$, a coalition structure and payoff vector pair $\langle CS^*, x \rangle$ is in the weak ϵ-core$^+$ iff:*

$$\langle CS^*, x \rangle \text{ is in the weak } \epsilon\text{-core}^+$$
$$\forall \epsilon' < \epsilon, \text{ the weak } \epsilon'\text{-core}^+ \text{ is empty}$$

Corollary 5.3. *For the grand coalition of a superadditive cover game, a payoff vector that minimises the maximum weak excess of the superadditive cover can be found using only the coalitions in W.*

Proof. Replace the characteristic function v with the superadditive cover characteristic function v^* in Lemma 5.1 and Theorem 5.2 (meaning the grand coalition's value will be equal to the optimal coalition structure), to guarantee a weak least core$^+$ solution for the superadditive cover. □

Given Corollary 5.3, the DDP algorithms introduced in this chapter identify the coalitions in the *synergy coalitional group* (SCG) representation and the coalition structure that maximises the value of the superadditive cover of the grand coalition to guarantee that a weak least core$^+$ stable solution is found. If cross-coalition side payments were not used, then it would be much more difficult for the agents to reason over what is the most stable coalition structure and payoff vector pair, as [112] showed the optimal coalition structure may not be the most stable in this situation and so multiple coalition structures will have to be compared via not just their value, but also their stability.

Searching all possible coalition structures is a highly complex task because the total possible number of coalition structures grows at a significantly higher rate than the number of potential

coalitions [104]. For n agents, the number of possible coalition structures is found using the Bell number B_n, which is $\sim \theta(n^n)$ and so significantly larger then the $\theta(2^n)$ growth of possible coalitions [104]. It is of great benefit to the agents to significantly minimise the number of possible coalition structures to compare.

Additionally, as noted in [2, 66], allowing cross-coalition side payments can benefit multi-agent systems, as it was argued that introducing cross-coalition side payments can be considered a more fair payoff mechanism then disallowing cross-coalition side payments The additional fairness comes from eliminating the effect of the coalition structure on agent payoffs. The example given in [2] is that it may be possible that in the optimal coalition structure some agents $M \subset N$ are by themselves in singletons, or are members of a comparatively small coalition compared to the size of the other coalitions in the coalition structure. When cross-coalition side payments are not allowed, these M agents do not benefit from the cooperation of others, even when the M agents are in many potential coalitions that have a high value, that the agents $N \backslash M$ may have used to negotiate for a better payoff. Yet, for the greater good of the population, i.e., maximizing social welfare, the M agents may be forced to stay in the optimal coalition structure.

In this chapter, the DDP algorithm assumes cross-coalition side payments are allowed because it significantly reduces the computation costs of the agents as multiple coalition structures will not have to be compared according to their stability.

5.2 Finding a Superadditive Cover Least Core Solution

In this section, the fully decentralised DDP algorithm for finding a solution within the weak least core$^+$ is introduced. When designing decentralised algorithms for coalition formation, it is critical to reduce the number of coalitions and coalition structures to be considered by each agent to minimise computation costs. In the DDP algorithm, a decentralised filtering method is used to transform a characteristic function game with an exponential representation into a superadditive cover with a synergy coalition group representation (SCG) [41, 81]. Even though in the worst case the SCG representation lists every coalition, in the majority of cases this transformation lowers the number of explicitly represented coalitions (as can be seen in the experimental evaluation of Section 5.5.1). The explicit listing of a lower number of coalitions reduces the number of constraints used to find a stable payoff vector. Developing algorithms for characteristic function games with constraints is an active research area (e.g. [5, 66, 92]). This chapter shows how agents with distributed knowledge on coalition values can create their own constraints for any possible characteristic function game, using their own self-interest as their motivator.

5.2.1 The Filtering Method

The proofs of Section 5.1 show that for any random payoff vector x, the coalition values of only the coalitions within the SCG representation need to be used in a linear program to guarantee that a weak least core$^+$ stable payoff vector can be found. Therefore the filtering in the DDP

algorithm distributes the *checking* of each $C \subseteq N$ coalition to see if C is in the SCG representation. So, instead of $|C \subseteq N|$ linear program constraints to find a stable payoff vector solution, there will now be $|W|$ constraints, where W is the set of publicly known coalition and coalition value pairs, individually denoted $(C, v(C))$.

The agents can then use the filtered coalitions in W to find the most stable payoff vector for the superadditive cover of the given characteristic function game by using the following function:

Definition 82: *The function* ***StablePayoff****$(v^*(N), W)$, given the superadditive cover of the grand coalition and the synergy coalition group representation, returns a payoff vector satisfying the least weak core$^+$ for the superadditive cover, by returning the solution to the following linear program:*

$$\textit{min } \epsilon^w \textit{ subject to:} \tag{5.7}$$

$$x_i \geq 0 \textit{ for each } i \in N \tag{5.8}$$

$$x(N) = v^*(N) \tag{5.9}$$

$$x(S) \geq v(S) - |S|\epsilon^w \textit{ for each } S \in W \tag{5.10}$$

The linear program used in the `StablePayoff` function is different to the traditional least core finding linear program, detailed in Section 2.4.3. Firstly, the new linear program divides the value of the optimal coalition structure of the characteristic function game (which is now also the superadditive cover of the grand coalition). Secondly, the exponential constraint $S \subseteq N$ has been replaced by the filtered W. In the worst case, W holds the same number of constraints as $|S \subseteq N|$, yet this number is significantly reduced experimentally, as show in Section 5.5, for standard coalition-value distributions in the literature [93].

The distribution of the task of checking each coalition $S \subseteq N$ to see if S is in the set W uses elements of the DCG algorithm of Chapter 4. The DCG algorithm is used to give each agent i a coalition value calculation share (CV_i) that is:

(a) approximately equal to another agent j's share, i.e. for all $i, j \in N$ then $|CV_i| = |CV_j| + p$ where $-2 \leq p \leq 2$;

(b) contains only coalitions where i is a member, i.e. for all $C \in CV_i$ then $i \in C$.

Point (a) is used to share the coalition value calculations around the group. Point (b) is used as an additional incentive for the agent to calculate the value of the coalition it has been assigned.

In the DDP algorithm, agents only communicate a coalition value to each other from their value calculation share, when this coalition is a member of the explicit SCG representation of the characteristic function game. A coalition and coalition value $(C, v(C))$ *should be explicitly included in the W set if $v(C)$ is of greater value than all possible partitions of C containing coalitions only from within W.*

Algorithm 7: A function to detail to the agents what information to communicate between themselves within the DDP algorithm.

1: **function** `Communicate` (i, s, n, bal)
2: **Input:** $\langle i, s, n, bal \rangle$; where i is the agent ID, s the size of the coalitions to evaluate, n is the total number of agents and bal is the balance pointer.
3: **Output:** $\langle \Gamma \rangle$ where Γ is a vector of objects containing coalition values or coalition structures to communicate to the other agents.
4: **vector** $\Gamma^i = \langle\rangle$
5: **if** $s == 1$ **then**
6: $\Gamma^i_0 := v(\{i\})$; // If size = 1, report this information to all the agents
7: *add* $(\{i\}, v(\{i\}))$ to W; // All coalitions of size 1 belong in the SCG representation
8: **else**
9: **int** $k := 0$;
10: **for** each $C \in$ `DCGSingleSize`(i, s, n, bal) **do**
11: $f_2[C] := max\{f_2[C'] + f_2[C \backslash C'] : C' \subset C$ and $1 \leq |C'| \leq 1/2|C|\}$; // $f_2[C]$ is the maximum value of any two-set partition of C
12: **if** $f_2[C] < v(C)$ **then**
13: *set* $f_1[C] := C$ and $f_2[C] := v(C)$; // C is greater in value than all its two-set partitions
14: *set* $\Gamma^i_k := v(C)$; // So the value of the coalition should be reported to all agents...
15: *add* $(C, v(C))$ to W; // ...because C is in the SCG
16: **else**
17: *set* $f_1[C] := C^*$ where C^* maximises $f_2[C]$;
18: *set* $\Gamma^i_k := C^*$ // The best two-set partition of C is reported to the agents
19: **end if**
20: $k++$;
21: **end for**
22: **end if**
23: **return** Γ^i;
24: **end**;

5.2.2 The Distributed Dynamic Programming (DDP) Algorithm

The DDP algorithm is split into three main functions for clarity. The first function, named `Communicate` (detailed in Algorithm 7), shows how the agents divide the parts of the original DP algorithm (see Section 2.4.2) up between themselves and how the agents decide what to communicate to the others. The second function, named `decodeDDP` (detailed in Algorithm 9), shows how each agent takes the communicated information from every other agent and uses this information to update its own knowledge base. Finally the full `DDP` algorithm is detailed in Algorithm 10. Through the DDP algorithm, the v table stores the values of the original characteristic function game, the values of the superadditive cover is stored in the f_2 table, while the f_1 table stores the partition of each coalition that maximises its superadditive cover.

In more detail[3], the `Communicate` function of Algorithm 7 describes to each agent i what information it should communicate to the others (in the vector Γ), regarding the coalitions in

[3]The following description does not take into account the issue of communication costs. This issue is dealt with in Section 5.3.

Algorithm 8: The Single sized variant of the Distributed Coalition Generation (DCG) method.

1: **function** `DCGSingleSize` (**int** i, s, n, bal)
2: **Input:** $\langle i, s, n, bal \rangle$ ($1 \leq i \leq n$); where i the agent ID, s is the size of the coalitions, n is the number of agents, and bal is the balance pointer.
3: **Ouput:** $\langle CV_{i,s} \rangle$; where $CV_{i,s}$ is the ordered set of coalitions of size s assigned to agent i, denoted $CV_{i,s} = \langle C_1, C_2, ..., C_k \rangle$.
4: **begin**
5: **if** $s < n$ **then**
6: **int** $count := 1$;
7: **boolean** $completed := false$;
8: **int** $msd := 0$;
9: **int[]** $\underline{t} := \langle 0, 0, ..., n - s \rangle$; // Get the first canonical representative IA
10: **while** ($complete == false$) **do**
11: $p := \pi(\underline{t})$;
12: $r := (n \times p)/s$;
13: **if** ($bal \leq i < bal + r$) **or** ($bal + r > n$ **and** $1 \leq i < bal + r - n$) **then**
14: $C(i, \underline{t}) = C_{count} \in CV_{i,s}$; // Add the coalition to the set of coalitions to return
15: $count++$;
16: **end if**
17: $bal := (bal + r) \bmod n$;
18: $\underline{t} =$ `build`$(\underline{t})$; // Get the next canonical representative IA
19: **end while**
20: **else if** ($s == n$) **and** ($i == bal$) **then**
21: $CV_{i,n} = \{N\}$;
22: **end if**
23: **end**

CV_i that are of size s. These coalitions are found using the `DCGSingleSize` function, which is a single sized variant of the DCG algorithm displayed in algorithm 8 (that uses the function `build` from the previous chapter).

In the `Communicate` function, if $s = 1$ then each agent should simply communicate the value of the singleton coalition of itself and represent this coalition explicitly within the SCG representation (lines 5 to 7). If $s > 1$ then the `Communicate` function assigns each agent a coalition value calculation share via the `DCGSingleSize` function. For each coalition of size s in i's coalition value calculation share, agent i checks if the value of C adds any synergy for the agents of C. If there *is* some additional synergy found (lines 12 to 15) then f_1 is set equal to C and f_2 is set equal to $v(C)$, indicating that the agents of C *cannot* gain any payoff from splitting into smaller coalitions, meaning C should be in the SCG representation. In this situation the agent i should add the value of $v(C)$ to Γ^i. Alternatively, if there is *no* additional synergy found (lines 16 to 18) f_1 is set equal to C^* and f_2 is set equal to $v(C^*)$, indicating that the agents of C can maximise their value by splitting into the smaller coalitions detailed in the set $C^* = \{C^*_1, ..., C^*_k\}$, where:

1. $\bigcup_{p=1}^{k} C^*_p = C$
2. $C^*_i \cap C^*_j = \emptyset$ for any $i, j \in \{1, ..., k\}$ where $i \neq j$

Algorithm 9: An function showing how the agents should decode the information communicated to them within the DDP algorithm.

1: **function** `decodeDDP` (i, s, n, bal)
2: **Input:** $\langle i, s, n, bal \rangle$; where i is the agent ID, s the size of the coalitions to evaluate, n is the total number of agents and bal the variable used to balance the agents' coalition value calculation shares.
3: **if** $s == 1$ **then**
4: **for** $j := 1$ to n where $j \neq i$ **do**
5: set $f_1[\{j\}] := \{j\}$ and $f_2[\{j\}] := v(\Gamma_0^j)$; // Set the value of the singleton coalitions to the communicated values
6: add $(\{j\}, v(\{j\}))$ to W; // Add all the communicated coalitions of size 1 to the SCG
7: **end for**
8: **else**
9: **int**$[n]$ $counter := [0, ..., 0]$;
10: **int** $m := 0$;
11: $\underline{t}^m := \langle 0, 0, ..., n - s \rangle$; // Generate the first canonical representative IA
12: **while** $\underline{t}^m \neq null$ **do**
13: $p := \pi(\underline{t}^m)$; // Find the period of the IA
14: $r := (n * p)/s$; // Find the number of agents to use the IA
15: **for** each $j \in N$ where $j \neq i$ **do**
16: **if** $bal \leq j < bal + r$ or $(bal + r > n$ and $1 \leq j < bal + r - n)$ **then**
17: $C := C(j, \underline{t}^m)$; // Get the coalition for j if j was one of the next r agents
18: **if** $\Gamma^j_{counter_{j-1}} == v(C)$ **then**
19: $f_1[C] := [C]$ and $f_2[C] := v(C)$; // Parse j's communicated data correctly
20: add $(C, v(C))$ to W; // Add C to the SCG
21: **else**
22: $f_1[C] := \Gamma^j_{counter_{j-1}}$ and $f_2[C] := f_2[v(\Gamma^j_{counter_{j-1}})]$; // Parse j's communicated data correctly
23: **end if**
24: $counter_{j-1} + +$; // Increment agent j's counter
25: **end if**
26: **end for**
27: $bal := bal + r$; // Increment pointer
28: **if** $bal > n$ **then**
29: $bal := bal - n$; // Correct the pointer
30: **end if**
31: $\underline{t}^m =$ `build`$(\underline{t}^m)$; // Get next canonical representative IA
32: **end while**
33: **end if**

In this situation the agent i should add the collection of coalitions C^* to Γ^i. The `Communicate` function then returns Γ^i for agent i to communicate. Later in Section 5.3, it is shown how the cost of communicating Γ can be significantly reduced.

Now given the communicated data on all coalitions of size s, in the form Γ^j for all $j \in N$, each agent i needs a method to decode the data, detailed in the `decodeDDP` function of Algorithm 9. If $s = 1$ then each agent knows that only the value of each singleton coalition was communicated, which is straightforward to decode (lines 4 to 7). If $s > 1$ then this is more

Algorithm 10: Agent i's section of the decentralised DDP algorithm to find a stable characteristic function game outcome given distributed knowledge on the coalition values.

1: **function** `DDP` (i, N)
2: **Input:** $\langle i, N \rangle$; where i is the agent ID and N is the set of agent IDs.
3: **Output:** $\langle CS^*, x^k \rangle$ where CS^* is the optimal coalition structure and x^k is a weak least core$^+$ stable payoff vector.
4: **int** $bal := 1$;
5: **set** $W := \{\}$;
6: **for int** $s := 1$ to n **do**
7: **int** $tempBal := bal$;
8: *send* `Communicate` (i, s, n, bal) to all agents;
9: *wait* until Γ^j received from all agents $j \in N \backslash \{i\}$;
10: `decodeDDP` $(i, s, n, tempBal)$; // Decode all data received from the other agents
11: **end for**
12: Set $CS^* := \{N\}$;
13: **Boolean** $edits := true$;
14: **while** $edits == true$ **do**
15: $edits := false$;
16: **for** each $C \in CS^*$ **do**
17: **if** $f_1[C] \neq C$ **then**
18: *set* $CS^* := (CS^* \backslash \{C\}) \cup \{f_1[C]\}$; // Replace C in the coalition structure, with its best two-set partition
19: $edits := true$; // This is set to true so the agents will look again through CS^* to see if any other coalitions in CS^* can be broken down further into two-set partitions
20: **end if**
21: **end for**
22: **end while**
23: $x^i :=$ `StablePayoff` (CS^*);
24: *Collaboratively choose* an x^k to implement;
25: **return** $\langle CS^*, x^k \rangle$;
26: **end**;

complicated to decode as: (i) each agent may have calculated the value of multiple coalitions; (ii) each agent may have searched for the optimal two-set partition of multiple coalitions; and (iii) not all agents may have used the same increment arrays $\underline{t}$'s to generate their coalitions (due to the use of the bal pointer in the `DCGSingleSize` function).

When $s > 2$, the `decodeDDP` proceeds by using every canonical representative increment array (lines 11 and 31), to generate the coalitions that the agents $N \backslash \{i\}$ calculated the value for (line 17). As not all of the agents may have used every increment array due to the repeating sequences (line 13 and 14), the following are used: (a) the bal pointer (line 16 and 27 to 29); (b) the r value (line 14); and (c) the $counter$ array (line 9, 18 and 22). Both (a) and (b) are used for the same reasons as in the `DCG` algorithm; to work out which agents use which increment array, while (c) is used so the agent can keep track of what position of each Γ^j is to be used next.

When agent i, using the `decodeDDP` function, finds that another agent generated a coalition C (at line 17), if i finds the next element of Γ^j to be a coalition value, then this indicates to agent i that C is more valuable than any possible partition of C. To indicate this synergy: $f_1[C]$ is

set to equal C; $f_2[C]$ is set equal to $v(C)$ to indicate the value of this synergy; and $(C, v(C))$ is to added to the list of coalitions explicitly represented in the SCG representation. If the next element of Γ^j is instead set to a partition of C that has the maximum value, then (just like the DP algorithm [138]): $f_1[C]$ is set equal to this partition; and f_2 is set equal to the partition's value. Once all possible increment arrays have been generated, then the decoding of the different data sent by the agents is complete.

Finally, the full `DDP` algorithm can be introduced in Algorithm 10, which takes an agent ID i from the set N as input and outputs a characteristic function game outcome of an optimal coalition structure CS^* (representing the coalition structure that maximises the superadditive cover of the grand coalition) and a weak least core$^+$ stable payoff vector x^k.

The `DDP` algorithm begins with the agents sharing the required coalition values and/or coalition partitions and decoding this communication for every potential size of a coalition (lines 6 to 11). The same pointer value is used for both the `Communicate` and `decodeDDP` functions to keep them synchronised. Next all the agents find an optimal coalition structure using the method first detailed in the DP algorithm of [138] (lines 16 to 21). Lastly all the agents find a stable payoff vector using the linear program detailed in the `StablePayoff` function described in Section 5.2.1, which only includes the coalitions in the SCG representation as constraints. Together the agents should choose the final single payoff vector in accordance to some agreed criteria as the least-weak-CS core$^+$ may contain multiple different payoff vectors. For example, the payoff vector chosen could be: from an agent, picked according to some form of lottery; or the most commonly suggested payoff vector; or the one that satisfies the majority of agents to some degree; or the one that has the smallest deficit vector[4]. In the next section, modifications to the DDP algorithm are detailed that significantly reduce the communication costs.

5.3 Lowering the Communication Costs

If bandwidth is costly, the agents have three main choices on how to lower communication costs (and for the algorithm to still remain decentralised and distributed):

1. Give each two-set partition of a coalition an index value.
2. Indicate whether a coalition's value creates synergy or not.
3. State a worst case number of bits to communicate for every possible coalition.

The advantage of giving the two-set partitions of each coalition an index value can be seen from the following discussion. For any coalition C of size s, if the agents of the two-set partition are required to be explicitly stated, then it will take a worst case of $\lfloor \frac{s}{2} \rfloor \times \lceil log_2(s) \rceil$ bits to detail this because it will take $\lceil log_2(s) \rceil$ bits to represent one agent, and the smallest part of the two-set partition of C contains a maximum of $\lfloor \frac{s}{2} \rfloor$ agents. Yet this is significantly more bits compared to using indexes.

For any coalition C of size s, there are $2^{s-1} - 1$ different possible two-set partitions of C [36]. The formula $2^{s-1} - 1$ is equivalent to the $(s-1)$th *Mersenne number* [75]. In

[4]Definition in Section 2.2.1

[53], function $posn(\underline{C}, \underline{n}, s)$ is used for indexing a coalition $\underline{C} = \langle c_1, ...c_s \rangle$ of size s, given $\underline{n} = \{ag_1, ag_2, ..., ag_n\}$ agents, where $\underline{C} \subseteq \underline{n}$. The $posn$ function gives an index relative only to coalitions of the same size s, and can be build upon to index every coalition $\underline{C}$ relative to coalitions of all sizes (that are all subsets of $\underline{n}$), through the following $totalposn$ function:

$$totalposn(\underline{C}, \underline{n}, s) = \sum_{1 \leq k \leq s-1} \binom{|\underline{n}|}{k} + posn(\underline{C}, \underline{n}, s)$$

Now $totalposn$ can be used to give the index of any two-set partition of a coalition $C = \underline{n}$ by giving the index value of the first set partition C_1, while the second set partition is all the agents not in the first set partition, i.e. $C_2 = C \backslash C_1$. It is assumed that C_1 is either: (a) the smallest coalition of the partition, i.e. $|C_1| < |C_2|$; or (b) both coalitions are of equal size and C_1 is smaller lexicographically, i.e. $|C_1| = |C_2|$ and $C_1 <_{lex} C_2$. Additionally, the function $totalposn^{-1}$ (that makes use of the $posn^{-1}$ function from [53]) returns a coalition given an index m:

$$totalposn^{-1}(m, \underline{n}) = indexCorrect(m, \underline{n}, 1)$$

$$indexCorrect(m, \underline{n}, s) = posn^{-1}(m, \underline{n}, s) \text{ if } m - \binom{|\underline{n}|}{s} < 0 \text{ else}$$

$$indexCorrect(m, \underline{n}, s) = indexCorrect(m - \binom{|\underline{n}|}{s}, s + 1)$$

To show the benefits of communicating an index for the two-set partition, compared to explicitly stating the agents, Table 5.1 shows the number of bits required for both methods. Table 5.1 clearly shows that indexing is the better method because the number of bits required for explicitly stating the agents grows at a higher rate than the indexing method, which grows linearly.

Alternatively, if indicating the best two-set partition of each coalition was not required, the communication costs can be lowered further. In this case, each agent j could send to every agent i (where $i \in N \backslash \{j\}$), a single bit for each of its coalitions in CV_j to indicate whether this coalition adds synergy (bit sent is 1) or not (bit sent is 0). Yet this method will require each agent i to either: (i) re-calculate the value of the coalition, which agent j has already completed (if bit sent is 1); or (ii) re-calculate the best two-set partition of the coalition, which agent j again has already completed (if bit sent is 0).

A compromise between the two approaches of using an index value or just sending a single bit, is to use a pre-agreed limit on the number of bits to be communicated per coalition, denoted b_c. In this scenario, the first bit should be reserved to indicate that the coalition adds synergy (the first bit is set to 1) or not (the first bit is set to 0). Then the remaining bits should be used to detail the coalition value (if the coalition creates synergy and the coalition value can be represented in the remaining bits) or the binary search instructions for the index value of the best two-set partition (if the coalition does not create any synergy). For example, when a coalition does not create any synergy, if there are m possible two-set partitions of a coalition, the second bit (the

s	two-set partitions	C bits	Indexing bits
2	1	1	1
3	3	2	2
4	7	4	3
5	15	6	4
6	31	9	5
7	63	9	6
8	127	12	7
9	255	16	8
10	512	20	9
11	1047	20	10
12	2047	24	11
13	4095	24	12
14	8191	28	13
15	16383	28	14
16	32767	32	15
17	65535	40	16
18	131071	45	17
19	262143	45	18
20	524288	50	19

TABLE 5.1: This table details the number of bits needed to represent the best two-set partition of a coalition, where: *two-set partitions* represent the total number of ways to split a coalition of size s into two partitions, found using the formula $2^{s-1}-1$; *C bits* is the worst case number of bits to explicitly represent each agent of one coalition of size $\lfloor \frac{s}{2} \rfloor$, found using the formula $\lfloor \frac{s}{2} \rfloor \times \lceil log_2(s) \rceil$; and *Indexing bits* is the worst case number of bits using the indexing method of the $totalposn$ function.

first bit indicates whether a coalition value creates synergy or not) states whether the index of the best partition is higher (bit is set to 1) or lower (bit is set to 0) than $m/2$. Then the next bit designates if the index of the best partition is higher or lower than the midpoint of the remaining index values, and so on until the best partition is found or the number of bits required reaches the pre-agreed limit. This compromise approach has the benefit of each agent i knowing that the worst case bits of information they will be communicating will be the pre-agreed limit on bits per coalition multiplied by the number of coalitions in i's value calculation share (i.e. $b_c \times CV_i$).

5.4 A DDP Algorithm Example

The following is a discussion of the characteristic function game example outlined in Table 5.2, where the original value of each coalition is given in $v[C]$, the superadditive cover of C is given in $f_2[C]$ and the index of the partition of C that maximises $f_2[C]$ is given in $f_1[C]$ (where the indexing scheme is described in Section 5.3). In the DDP algorithm, the agents go step by step through the coalitions of each size. The agents begin with the singleton coalitions, where they calculate their assigned value, add it to the synergy coalitional game (SCG) representation and communicate the value to the other agents. As all the singleton coalitions are in the SCG representation then their f_1 pointers are set to the index value of zero to indicate this. In Table 5.2

s	C	Ag	The evalations performed before setting f_1 and f_2		f_1	f_2	SCG
1	$\{1\}$	1	$v[\{1\}] = 1.0$		0	1.0	YES
	$\{2\}$	2	$v[\{2\}] = 1.1$		0	1.1	YES
	$\{3\}$	3	$v[\{3\}] = 0.9$		0	0.9	YES
	$\{4\}$	4	$v[\{4\}] = 0.9$		0	0.9	YES
2	$\{1,2\}$	1	$v[\{1,2\}] = 2.0$	$f_2[\{1\}] + f_2[\{2\}] = 2.1$	1	2.1	NO
	$\{1,3\}$	1	$v[\{1,3\}] = 2.0$	$f_2[\{1\}] + f_2[\{3\}] = 1.9$	0	2.0	YES
	$\{1,4\}$	4	$v[\{1,4\}] = 2.2$	$f_2[\{1\}] + f_2[\{4\}] = 1.9$	0	2.2	YES
	$\{2,3\}$	2	$v[\{2,3\}] = 2.2$	$f_2[\{2\}] + f_2[\{3\}] = 2.0$	0	2.2	YES
	$\{2,4\}$	2	$v[\{2,4\}] = 2.0$	$f_2[\{2\}] + f_2[\{4\}] = 2.0$	1	2.0	NO
	$\{3,4\}$	3	$v[\{3,4\}] = 2.0$	$f_2[\{3\}] + f_2[\{4\}] = 1.8$	0	2.0	YES
3	$\{1,2,3\}$	1	$v[\{1,2,3\}] = 3.0$	$f_2[\{1\}] + f_2[\{2,3\}] = 3.2$	1	3.2	NO
			$f_2[\{2\}] + f_2[\{1,3\}] = 3.1$	$f_2[\{3\}] + f_2[\{1,2\}] = 3$			
	$\{1,2,4\}$	4	$v[\{1,2,4\}] = 3.0$	$f_2[\{1\}] + f_2[\{2,4\}] = 3.0$	2	3.3	NO
			$f_2[\{2\}] + f_2[\{1,4\}] = 3.3$	$f_2[\{4\}] + f_2[\{1,2\}] = 3.0$			
	$\{1,3,4\}$	3	$v[\{1,3,4\}] = 3.3$	$f_2[\{1\}] + f_2[\{3,4\}] = 3$	0	3.3	YES
			$f_2[\{3\}] + f_2[\{1,4\}] = 3.1$	$f_2[\{4\}] + f_2[\{1,3\}] = 2.9$			
	$\{2,3,4\}$	2	$v[\{2,3,4\}] = 3.0$	$f_2[\{2\}] + f_2[\{3,4\}] = 3.1$	1	3.1	NO
			$f_2[\{3\}] + f_2[\{2,4\}] = 2.9$	$f_2[\{4\}] + f_2[\{2,3\}] = 3.1$			
4	N	3	$v[\{1,2,3,4\}] = 4.0$	$f_2[\{1\}] + f_2[\{2,3,4\}] = 4.1$	2	4.4	NO
			$f_2[\{2\}] + f_2[\{1,3,4\}] = 4.4$	$f_2[\{3\}] + f_2[\{1,2,4\}] = 4.2$			
			$f_2[\{4\}] + f_2[\{1,2,3\}] = 4.1$	$f_2[\{1,2\}] + f_2[\{3,4\}] = 4.1$			
			$f_2[\{1,3\}] + f_2[\{2,4\}] = 4$	$f_2[\{1,4\}] + f_2[\{2,3\}] = 4.4$			

TABLE 5.2: This table details a characteristic function game for the set of agents $N = \{1,2,3,4\}$. The agents find a stable least-weak-CS core$^+$ solution in a decentralised manner when using the DDP algorithm. The array f_1 is set to the lowest possible index value for each coalition.

it is shown, through the *Ag* column, which agents are assigned which coalitions in their value calculation share[5].

Now for all coalitions of sizes $2 \leq s \leq n$ the agent who is assigned to calculate the value of a coalition C will also have to compare the values of all the possible two-set partitions of C. If the value of the coalition is higher than the value of any possible two-set partition, the calculating agent needs to communicate this to the agents, set the index value to zero and add the coalition to the SCG representation. If the value of a two-set partition is higher, then the calculating agent also needs to communicate this, and set the index value to the correct number via the $totalposn$ function. For example in Table 5.2, $f_1[\{1,2\}]$ is set to 1 as there is only one way to split the coalition $\{1,2\}$ into a two-set partition and this partition has a greater combined value of 2.1 compared to the value of the coalition $\{1,2\}$. Alternatively, $f_1[\{1,2,4\}]$ is set to 2, because there are three different ways to split $\{1,2,4\}$ into a two-set partition, where the most valuable partition corresponds to $\{\{2\},\{1,4\}\}$, which is assigned an index value of 2 according to the $totalposn$ function. This index value of 2 occurs because the smallest coalition in the partition (i.e. $\{2\}$) is the second smallest coalition (in lexicographical order) of the smallest

[5]The canonical representative increment arrays used to assign the coalitions: for $s = 2$ were $[0,2]$ and $[1,1]$, while for $s = 3$, $[0,0,1]$ was used.

possible coalition size. When there is more than one partition with the same value, then the partition with the lowest index is assumed to be used.

In this example, it can be seen that 9 coalition values have been communicated, and therefore $(2^{4-1} - 1) - 9 = 6$ index values have been communicated by the agents. Given this communicated information, the agents find that there are two possible optimal coalition structures of the characteristic function game described in the v table of Table 5.2 that maximise the superadditive cover f_2 of the grand coalition: (a) $\{\{1,4\},\{2,3\}\}$; or (b) $\{\{2\},\{1,3,4\}\}$. It is assumed, that the optimal coalition structure with the lowest index value is chosen, which is: $\{\{2\},\{1,3,4\}\}$.

Using the 9 communicated coalition values in the linear program of the `StablePayoff` function will allow the agents to find a weak least core$^+$ solution of the superadditive cover. The grand coalition of the superadditive cover gives a value of $\epsilon^w = 0$ under the weak least core$^+$ definition, indicating that the core of the superadditive cover is non-empty. An example payoff vector that satisfies all the agents of the game described in Table 5.2 is: $x(1.0, 1.1, 1.1, 1.2)$.

5.5 Evaluation

This section will provide an evaluation through the discussion of: (a) the information communicated between the agents for two standard coalition-value distributions; (b) the number of split operations each agent needs to perform (i.e. the number of two-set partitions of every coalition); (c) how to modify DDP to make sure that the difference between the number of split operations between the agents are lowered; and (d) how successful DDP is at finding an outcome in different solution concepts.

For part (a), an experimental analysis was used as (a) is dependent on the coalition-value distribution. The experiments were run using: (i) different agent numbers from 4 to 12, where 30 runs for each agent number were used; and (ii) different coalition-value distributions, which were normal, uniform and NDCS, as is usual in the coalition structure generation literature [93]. These coalition-value distributions are defined as:

- *Uniform*:- Each coalition's value is determined by multiplying the number of agents in the coalition by a variable q picked from a uniform distribution between 0 and 1.0. Formally, $v(C) = max(0, |C| \times q)$, where: $q \in \mathcal{U}(a, b)$; $a = 0$; and $b = 1.0$.
- *Normal*:- Each coalition's value is determined by multiplying the number of agents in the coalition by a variable q picked from a normal distribution with mean 1 and variance 0.1. Formally, $v(C) = max(0, |C| \times q)$, where: $q \in \mathcal{N}(\mu, \sigma^2)$; $\mu = 1$; and $\sigma^2 = 0.1$.
- *NDCS*:- Each coalition's value is determined by a variable q picked from a normal distribution with mean of the coalition's size and variance also according to the coalition's size. Formally, $v(C) = max(0, q)$, where: $q \in \mathcal{N}(\mu, \sigma^2)$; $\mu = |C|$; and $\sigma^2 = |C|$.

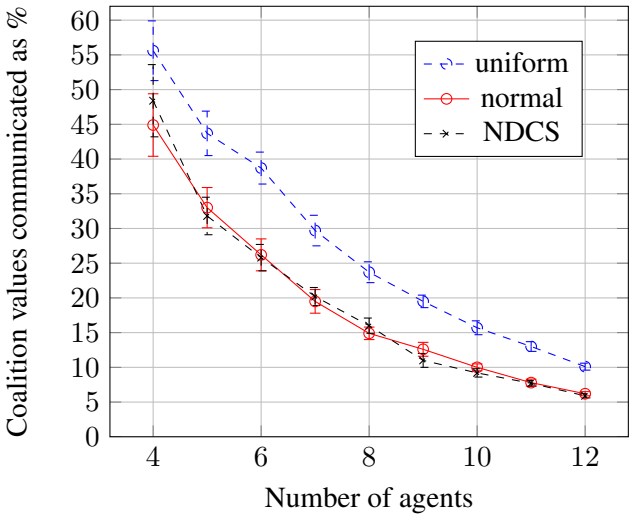

FIGURE 5.1: This figure shows the percentage of coalition values communicated for the uniform, normal and NDCS coalition-value distributions. A 95% confidence interval was used.

5.5.1 The Communicated Information

As discussed in Section 5.3, the DDP algorithm requires the agents to communicate at least one bit for each coalition they are assigned. To help estimate the cost of allowing agents to communicate all of the coalition values that DDP requires (according to the `Communicate` function), the number of coalition values communicated for the three standard coalition-value distributions is illustrated in Figure 5.1, where the x axis shows the total number of agents of the characteristic function game and the y axis shows the coalition values communicated as a percentage of the total coalitions. For example, using Figure 5.1, if it is known that the worst-case cost of communicating one coalition value in an 8 agent game is b_{cv}, and the coalition-value distribution is expected to be of a normal distribution, then an estimate for the total cost is $15 \times b_{cv}$.

Additionally, to estimate the cost of allowing agents to communicate an index for the best two-set partition of each coalition that does not create synergy, the average percentage of required index communications is 100% minus the average percentage of communicated coalition values. For example, in Figure 5.1, if it is known that the worst-case cost of communicating one coalition index in an 8 agent game is b_i, and the coalition-value distribution is expected to be of a normal distribution, then an estimate for the total cost is $(100 - 15) \times b_i$.

We can infer from Figure 5.1 that the number of the coalitions in the SCG representation decreases (as a percentage of the total coalitions) as n increases, for all the uniform, normal and NDCS coalition value distributions. This means that searching for a core or weak least core stable payoff vector using the *full set of coalitions* with these coalition value distributions is greatly inefficient (as the SCG representation includes all the coalitions to find a core or weak-least core payoff vector, see Section 5.1). This is why the DDP algorithm uses the decentralised filtering process.

Finally the experiments showed the normal coalition-value distribution (and the NDCS variant) to require less coalition values to be communicated. Upon closer inspection, this occurred

n	Total Operations	Highest Ag Ops	Lowest Ag Ops	Difference	Operations for N
2	1	1	0	1	1
3	6	4	1	3	3
4	25	11	4	7	7
5	90	30	15	15	15
6	301	72	41	31	31
7	966	192	129	63	63
8	3025	475	348	127	127
9	9330	1252	997	255	255
10	28501	3244	2733	511	511
11	86526	8796	7773	1023	1023
12	261625	23431	21384	2047	2047

TABLE 5.3: This table gives the number of split operations occurring in the DDP algorithm for different n values where: *Total Operations* is the total split operations by all the agents; *Highest Ag Ops* gives the maximum number of split operations an agent will perform; *Lowest Ag Ops* gives the lowest; *Difference* gives the difference between highest and lowest; while *Operations for N* gives the number of split operations required for the grand coalition.

because in general, the q value used in the normal (and NDCS) coalition-value calculations have a more compact distribution. Therefore, there is a higher chance in a uniform coalition-value distribution that a larger coalition will create synergy, which requires for that coalition to be communicated.

5.5.2 Agent Split Operations

For coalitions of increasing size, there are an exponentially increasing number of ways to split such coalitions into two partitions (henceforth known as *split operations*), as DDP requires. This exponentially increasing sequence is actually equivalent to the Stirling number of the second kind: $S(n+1,3)$ [104]. Even though each agent is assigned an approximately equal number of coalitions in its coalition value calculation share, each agent using DDP will have different numbers of split operations to perform. This is because the agents do not calculate an equal number of coalitions of every size, for example there is only ever one grand coalition to distribute.

As can be seen in Table 5.3, as the total number of agents increases, the gap between the highest and lowest number of split operations performed by an agent also increases. Yet highlighted in the final two columns of Table 5.3 is the fact that this gap is equivalent to the number of split operations required by the grand coalition N. Therefore a modification of DDP, named DDP*, is introduced here, that constricts the range of split operations required for each agent. The algorithm DDP* is the same as DDP except that the split operations of the grand coalition are divided among the agents. DDP* uses the `split` function presented in Algorithm 11 to achieve the distribution of the grand coalition split operations. The DDP algorithm is modified from the DDP* algorithm by replacing line 6 of the DDP algorithm with the following: "**for int** $s := 1$ to $n-1$ **do**" and immediately after this for loop the `split` function is called.

In the `split` function of Algorithm 11, firstly the number of split operations that *every* agent has to perform is calculated (line 3). This is found by dividing the $(n-1)$'th *Mersenne*

Algorithm 11: The function determining which two-set partitions of the grand coalition each agent should search in the DDP* algorithm.

1: **function** `split` (i, n, bal)
2: **Input:** $\langle i, n, bal \rangle$; where i is the agent ID, n is the number of agents and bal is the next agent to calculate an additional value
3: **int** $k := \lfloor \frac{2^{n-1}-1}{n} \rfloor$; // k is the number of two-set partitions for each agent
4: **if** $i \neq bal$ **then**
5: $f_2^i[N] := max\{f_2[C'] + f_2[N \backslash C'] : C' := totalposn^{-1}(m, N)$ for all $(k * i) - k < m \leq k * i\}$; // m is all the indexes of the two-set partitions to check
6: **else**
7: $f_2^i[N] := max\{f_2[C'] + f_2[N \backslash C'] : C' := totalposn^{-1}(m, N)$ for all $(k * i) - k < m \leq k * i$ and $k * n < m\}$; // In this case, agent i gets additional two-set partitions to check
8: **end if**
9: $f_1^i[N] := C^*$ where C^* maximises $f_2^i[N]$; // Now $C*$ maximises only the two-set partitions of N that agent i searched
10: *send* $f_1^i[N]$ and $f_2^i[N]$ to all agents;
11: $f_2[N] = max\{f_2^j[N] : \forall j \in N\}$; // The real maximum partition of N is now found
12: **if** $i == bal$ **then**
13: *calculate* $v(N)$; // agent with ID bal is assigned to calculate the value
14: *send* $v(N)$ to all agents;
15: **else**
16: *wait* to receive $v(C)$;
17: **end if**
18: **if** $f_2[N] < v(N)$ **then**
19: $f_1[N] := N$ and $f_2[N] := v(N)$;
20: add $(N, v(N))$ to W; //N is added to the SCG if its value is greater than any two-set partition
21: **else**
22: $f_1[N] := f_1^j[N]$ where $f_2[N] == f_2^j[N]$;
23: **end if**
24: **end**;

number (which corresponds to the number of ways to split the grand coalition into two, shown by Table 5.1) by n. If the grand coalition split operations divides by the number of agents to give an integer value, then each agent gets assigned an equal number of split operations for the grand coalition (line 5 and 7). If the grand coalition split operations do not divide equally, then the agent who is to calculate the grand coalition value is assigned the remainder of the grand coalition split operations (line 7). Each agent i then sets its $f_1^i[N]$ and $f_2^i[N]$ pointers correctly (in lines 5, 7 and 9) and sends them to the others (line 10). The agents proceed to find the largest $f_2^j[N]$ pointer (line 11) and use it to see if the grand coalition creates any synergy or not (line 18 to 22), once the grand coalition value has been calculated by the agent assigned to it (line 12 to 17).

This distribution of grand coalition split operations will not create any new strategic issues when cross-coalition side payments are allowed, because: (i) each agent is in the grand coalition, so each agent has a motivation to maximise the value of the grand coalition partition; and (ii) all

n	DDP op	DDP* op^*	DDP Range	DDP* Range
2	$op(1,0)$	$op^*(1,0)$	[100%,0%]	[100%,0%]
3	$op(4,1,1)$	$op^*(2,2,2)$	[66.7%,16.7%]	[33.3%,33.3%]
4	$op(5,5,11,4)$	$op^*(6,6,8,5)$	[44%,16%]	[32%,20%]
5	$op(30,15,15,15,15)$	$op^*(18,18,18,18,18)$	[33.3%,16.7%]	[20%,20%]
6	$op(48,48,72,$ $43,43,47)$	$op^*(53,53,47,$ $49,49,52)$	[23.9%,14.3%]	[17.6%,15.6%]
7	$op(192,129,129,$ $129,129,129,129)$	$op^*(138,138,138,$ $138,138,138,138)$	[19.9%,13.4%]	[14.3%,14.3%]
8	$op(349,349,373,373,$ $379,379,475,348)$	$op^*(364,364,388,388,$ $394,394,370,363)$	[15.7%,11.5%]	[13%,12%]
9	$op(1000,1000,1000,$ $1028,1028,1028,$ $1252,997,977)$	$op^*(1028,1028,1028,$ $1056,1056,1056,$ $1028,1025,1025)$	[13.4%,10.7%]	[11.3%,11%]
10	$op(2860,2860,3244,$ $2733,2733,2747,$ $2747,2859,2859,2859)$	$op^*(2911,2911,2785,$ $2784,2784,2798,$ $2798,2910,2910,2910)$	[11.4%,9.6%]	[10.2%,9.8%]
11	$op(8796,7773,7773,$ $7773,7773,7773,7773,$ $7773,7773,7773,7773)$	$op^*(7866,7866,7866,$ $7866,7866,7866,7866,$ $7866,7866,7866,7866)$	[10.2%,9%]	[9.1%,9.1%]
12	$op(21775,21895,23431,$ $21384,21512,21512,$ $21514,21514,21770,$ $21770,21774,21774)$	$op^*(21945,22065,21561,$ $21554,21682,21682,$ $21684,21684,21940,$ $21940,21944,21944)$	[9%,8.2%]	[8.4%,8.2%]

TABLE 5.4: This table explicitly details the deterministic operation vectors for the DDP and DDP* algorithm, where the range columns detail the numbers of operations an agent has to perform as a percentage of the total operations (where the total operation number is found from Table 5.3).

agents will want to find the optimal superadditive cover of the grand coalition because all agents will then have more utility to distribute between themselves. Alternatively, if utility transfers are *not* allowed between the coalitions, then distributing the grand coalition split operations may introduce strategic considerations, as an agent i may want to hide a grand coalition partition that leads to a coalition structure with itself in a small coalition that has little utility to distribute between its member agents.

The differences between the number of split operations each agents computes can be analysed in Table 5.4, where the split operations are grouped into a vector op (or op^*) where agent i will perform op_i (or op_i^*) number of split operations. Analysing Table 5.4, it can be seen that the split operation range for the DDP* algorithm has been significantly restricted compared to the range for the DDP algorithm. For example, the 6 agent split operation vector for the DDP algorithm is $op(48,48,72,43,43,47)$, while Table 5.3 shows that the total number of split operations for 6 agents is 301 (i.e. $\sum_{1\leq i\leq 6} op_i = 301$), meaning the range of this operation vector is between $\frac{op_3=72}{301} = 23.9\%$ and $\frac{op_4=op_5=43}{301} = 14.3\%$. Using the DDP* algorithm to redistribute the 31 split operations required for the grand coalition approximately evenly around the agents of the system, gives the operation vector $op^*(53,53,47,49,49,52)$. In

op^*, agent 1 gets 53 split operations because $53 = op_1 + \lfloor \frac{31}{6} \rfloor = 48 + 5$. Additionally in op^*, the agent assigned to calculate the grand coalition (agent 3), gets 47 split operations because $47 = op_1 - 31 + \lfloor \frac{31}{6} \rfloor + (31 - (\lfloor \frac{31}{6} \rfloor \times 6)) = 72 - 31 + 5 + 1$. The redistribution of the grand coalition split operations has restricted the range, compared to the operation vector of the DDP algorithm, as it is now between $\frac{op_1^*=op_2^*=53}{301} = 17.6\%$ and $\frac{op_3=47}{301} = 15.6\%$.

Additionally it can be seen that for some agent numbers, the number of split operations required by the DDP* algorithm is equal for each agent (i.e. when $n \in \{3, 5, 7, 11\}$). Analysis of this interesting property led to the following theorem:

Theorem 5.4. *The number of split operations per agent for the DDP* algorithm is exactly equal whenever there are n agents and n is an odd prime.*

Proof. It is easy to show that for every odd prime n and $0 < s < n$, then n divides $\binom{n}{s} = \frac{n!}{s!(n-s)!}$ (written $n|\binom{n}{s}$), because in $\frac{n!}{s!(n-s)!}$ the numerator is divisible by n whereas the denominator is not (since it is a product of numbers smaller than n and n is a prime). The number $\binom{n}{s}$ corresponds to the number of coalitions of size s, when n is the number of agents in the characteristic function game. Therefore because $n|\binom{n}{s} \in \mathbb{N}$ (when $0 < s < n$), each agent receives an equal number of coalitions of the same size in its coalition value calculation share, and so an equal number of associated split operations for every size s where $0 < s < n$.

To complete the proof, the number of grand coalition split operations must be the same for each agent. This is proven through *Fermat's Little Theorem* [75] which states that if n is a prime number then $2^{n-1} \equiv 1(mod\ n)$ holds. Fermat's Little Theorem can be re-written as $\frac{2^{n-1}-1}{n} \in \mathbb{N}_{>0}$, which is the number of ways the grand coalition can be split into two [36], divided by the number of agents, resulting in a natural number.

Now the proof has shown that when n is an odd prime, each agent has an equal number of split operations for any s where $0 < s < n$ and each agent has an equal number of split operations for the grand coalition (i.e $s = n$), thus proving the theorem. □

Corollary 5.5. *When n is an odd prime, then n divides the Sterling number of the second kind $S(n + 1, 3)$.*

Proof. Theorem 5.4 shows that whenever n is an odd prime then the total number of split operations can be equally divided between the agents. The total number of ways to split every coalition $C \subseteq N$ into two parts is $S(n + 1, 3)$, showing the corollary. □

Thus Theorem 5.4, Table 5.4 and the previous discussion shows that by redistributing the split operations of the grand coalition around the agents, the computational burden for the agents of completing the coalition formation process becomes more balanced (and exactly equal when the number of agents is an odd prime).

5.5.3 Solution Concept Success Rate

As detailed in Section 5.1, if the DDP algorithms finds all coalitions that are explicitly detailed in the synergy coalitional group representation, then the payoff vector returned is guaranteed

to be in the weak least core$^+$ of the superadditive cover (where the plus indicates that cross-coalition side payments are allowed). The DDP algorithms find every explicit coalition in the SCG representation in lines 7 to 15 of the `Communicate` function (see Algorithm 7), lines 6 and 20 of the `decodeDDP` function (see Algorithm 9) and line 20 of the `Split` function, if DDP* is used (see Algorithm 11). Then the linear program of the `StablePayoff` function is solved for the grand coalition of the superadditive cover to find a weak least core$^+$ stable payoff vector of the superadditive cover.

If the core of the superadditive cover is non-empty then it will correspond to the weak least core$^+$ of the superadditive cover. Yet the success rate for the following solution concepts is not guaranteed using the DDP algorithm: (a) the least-strong-CS core$^+$ of the superadditive cover; or (b) the least-CS core. In [112], an example was detailed that showed that solutions within (b) may *not* include the optimal coalition structure. This means that the DDP algorithms may not find solution (b).

An outcome is not guaranteed for strong least core$^+$ of the superadditive cover due to the ϵ in the strong-ϵ core definition (of Section 2.2.1) not being proportional to the number of agents in the coalition and so a blocking coalition maybe missing from the SCG representation:

Theorem 5.6. *Given a superadditive characteristic function game $\mathcal{G} = \langle N, v\rangle$ and a payoff vector x (where $x(N) = v(N)$), let ϵ^w be the maximum weak excess that still gives a blocking coalition for x given full knowledge on each coalition's value. Using only the coalition values in W, a blocking coalition D for the weak excess value of $\epsilon^{s\prime}$ is guaranteed to be present within W, where $\epsilon^{s\prime}$ is in the bound $\frac{\epsilon^s}{n-1} \leq \epsilon^{s\prime} \leq \epsilon^s$.*

Proof. Suppose x is blocked by a coalition C through $v(C) - \epsilon^s$ so that $v(C) - \epsilon^s > x(C)$ and $\forall \epsilon^{s\prime\prime} > \epsilon^s$ there is no blocking coalition. If $(C, v(C)) \in W$ then this proves the Lemma. If $(C, v(C)) \notin W$ then from the definition of a SCG, it is known that the value of the coalition can be found through its maximum value partition, i.e. $v(C) - \epsilon^s = \sum_{1 \leq p \leq q} v(C_p) - \epsilon^s$, for some set of coalitions $\{C_1, ..., C_q\}$ where:

1. $\bigcup_{p=1}^{q} C_p = C$
2. $C_i \cap C_j = \emptyset$ for any $i, j \in \{1, ..., q\}$ where $i \neq j$
3. $(C_p, v(C_p)) \in W$, for all $C_p \in \{C_1, ..., C_q\}$

Via substitution, it follows that $\sum_{1 \leq p \leq q} v(C_p) - \epsilon^s > x(C)$ and hence for at least one C_p then $v(C_p) - \frac{\epsilon^s}{q} > x(C_p)$. As the game is superadditive the grand coalition is known to form, hence the largest coalition that can have an excess value is $S \subset N$ where $|S| = n - 1$, therefore $0 < q < n$.

Using this information, a coalition with the strong excess value of $\epsilon^{s\prime}$ is guaranteed to be in the SCG representation, where: $\frac{\epsilon^s}{n-1} \leq \epsilon^{s\prime} \leq \epsilon^s$. Thus the theorem is proved. □

Therefore searching for the strong least core using only the coalitions in W may return erroneous results (i.e. a solution not in the strong least core) because the agents may conclude that a payoff vector is stable when inspecting the excess values of only the coalitions in W but

actually this payoff vector is not stable when inspecting the coalitions not in W. For further understanding, consider the following example:

Example 5.2. *Consider a characteristic function game with $N = \{1, 2, 3, 4\}$ agents, where a coalition $C = \{1, 2, 3\}$ has a value of $v(\{1, 2, 3\}) = 25$. Given a payoff vector x, that gives the coalition C the total value $x(C) = 22$, it is known that the maximum (integer)* strong *excess to give a blocking coalition is $\epsilon^{s\prime} = 2$. For instance $x(C) = 22 < v(C) - \epsilon^{s\prime} = 25 - 2 = 23$, and so coalition $\{1, 2, 3\}$ is a blocking coalition for x when $\epsilon^{s\prime} = 2$ but not a blocking coalition when $\epsilon^{s\prime} = 3$.*

If $C \in W$ then a blocking coalition for the maximum strong excess value (that still admits a blocking coalition) has been found. If $C \notin W$ then there must be coalitions within W that make up a partition of C and have an equal or greater combined value then C. In this example assume $(\{1, 2\}, v(\{1, 2\}) = 16), (\{3\}, v(\{3\}) = 9) \in W$. The values of these coalitions have been chosen because $v(\{1, 2\}) + v(\{3\}) = v(\{1, 2, 3\})$, i.e. the values are the minimal needed to make sure that coalition $\{1, 2, 3\}$ is not in the SCG representation.

Given theses preliminaries, to stop coalition $\{1, 2\}$ blocking when $\epsilon^{s\prime} = 2$, then $x(\{1, 2\})$ must be greater than or equal to $v(\{1, 2\}) - \epsilon^s = 16 - 2 = 14$. Assume that $x(\{1, 2\}) = 14$ (i.e. the minimal payoff to satify the coalition has been given).

Recall $x(C) = 22$. Therefore $x_3 = x(C) - x(\{1, 2\}) = 22 - 14 = 8$. But this gives $x(\{3\}) = 8 > v(\{3\}) - \epsilon^s = 9 - 2 = 9 - 2 = 7$. As the payoff of 3 is currently greater than its singleton value minus $\epsilon^{s\prime}$, the singleton coalition $\{3\}$ does not become a blocking coalition for x under $\epsilon^{s\prime} = 2$.

In conclusion this example shows that if a blocking coalition C with the maximum weak excess (that still admits a blocking coalition) is not *present in the SCG representation, then it is not guaranteed that a coalition $C' \subset C$ with the same weak excess value will be present in the SCG representation.*

5.6 Summary

This chapter presented two decentralised algorithms, DDP and DDP*, for agents to use to solve the coalition formation process in a completely distributed manner (i.e. without even shared memory access). Both algorithms are dynamic programming ones that are guaranteed to output a solution within the weak least core$^+$ of the superadditive cover of the given characteristic function game, where the '+' indicates that cross-coalition side payments are allowed. The algorithms distribute: (i) the coalition value calculations; (ii) the filtering of the coalitions into the synergy coalitional group representation; and (iii) the two-set partitions of each coalition to search. The agents communicate to each other information that allows each agent to *not* have to compute all of (i), (ii) and (iii) by itself, yet the agents still achieve an optimal solution as if the individual agents had full knowledge.

The DDP and DDP* algorithms have only one difference, which is that the DDP* algorithm distributes the split operations of the grand coalition to the agents. This difference restricts the range of two-set partition operations required by the agents and leads to the interesting property,

proven in Theorem 5.4, that whenever the number of agents n is an odd prime, then the agents receive an exactly equal number of split operations to perform.

Both the DDP and DDP* algorithms have been developed from the DP algorithm detailed in [138], as discussed in Section 2.4.2. The DP algorithm only searched for the optimal coalition structure (referred to in [138] as the 'best set partition') and did not concern itself with stable payoff vectors. Another variation of DP is the IDP algorithm [91], which like DP finds the optimal coalition structure, but by considering only 38.7% of the two-set partitions. Additionally, less memory was used for IDP compared to DP. The DDP algorithms followed the DP approach of considering all of the split partitions as this led to the interesting properties of: (a) being able to locate the exact coalitions in the synergy coalitional group representation; and (b) each agent having an equal number of split operations to consider when the agent number is an odd prime. Methods to reduce the memory requirements (described in [91]), such as removing the f_2 table through putting all operations for f_2 into the v table, can easily be added to the DDP algorithms.

Finally, the DDP algorithms use elements of the DCG algorithm of Chapter 4 because: (1) the DCG algorithm gives an approximately equal distribution for the agents coalition value calculation shares; and (2) every coalition in each agent i's share has agent i as a member. Point (2) is very important in the context of finding an accurate stable payoff vector distribution (in the context of complete knowledge), because without this point, then self-interested agents will be motivated to hide values of coalitions not including themselves in order to gain a larger payoff.

Chapter 6

Chapter 6

Valuation Disagreement Coalitional Games

So far in Chapters 4 and 5, coalitional games with quantitative valuations were investigated where the values of the coalitions were the same for every agent, and therefore there were no valuation disagreements. In this chapter, *coalitional games with valuation disagreements* are investigated (where each agent assigns quantitative values to the coalitions). These disagreements can arise from agents having potentially different methods to interpret their possibly heterogeneous knowledge bases, which can lead to the agents finding different conclusions on a coalition's quantitative value. Thus finding stable coalitional game solutions in these situations involves identifying coalitions and agent-payoff contractual agreements that each agent finds acceptable given its own beliefs on each coalition's quantitative value.

In this chapter, the new *valuation disagreement coalitional game* (VCG) model is introduced with the new stability solution concepts of: the *valuation-disagreement core*, the *valuation-disagreement ϵ^v-core* and the non-empty *least valuation-disagreement core*. All of the solution concepts consist of a triple containing: (i) a *coalition structure*; (ii) an *estimated payoff vector*; and (iii) a *contract function*. The estimated payoff vector details each agent i's expected payoff given all the agents' coalition valuations, while the contract function provides an agreement on the manner to distribute the true value of each coalition in the coalition structure, once the true value of each coalition is revealed.

The new solution concepts allow harsh punishment of agents who overestimate the true value of their formed coalition in the agreed coalition structure. This harsh punishment is used to encourage agents to report lower pessimistic valuations. Pessimistic valuations can benefit an agent system as it increases the likelihood that each agent will gain a profit over its expected payoff once the true values of the formed coalitions are revealed. This is beneficial as less loss/debt will have to be assigned to some agent(s) from their expected payoffs (compared to the agents reporting normal or optimistic valuations). Assigning less debt is a very positive property due to coalitional games being a microeconomic model of the complicated human economic world, and assigning loss/debt can have a wide-ranging social impact on people (such as poverty, job losses, etc).

Therefore the contributions of this chapter are as follows: (i) the *valuation disagreement coalitional game* model itself and its related solution concepts are introduced; (ii) these new solution concepts, when analysed empirically and compared quantitatively to *coalitional games with beliefs*[1], allow previously unattainable coalitions to form; (iii) *valuation disagreement coalitional games*, compared to *coalitional games with beliefs*, give solutions of greater expected value in the majority of cases (when the same valuations for both games are used); and (iv) *valuation disagreement coalitional game* solution concepts are shown to encourage pessimistic valuations, compared to a solution concept that uses a single percentage-based agreement.

This chapter is structured as follows: in Section 6.1, the valuation disagreement coalitional game model is introduced, which includes the estimated payoff vector and the associated stability solution concepts. In Section 6.2, the role of contract functions are described in detail. In Section 6.3, an experimental evaluation is detailed. Finally Section 6.4 summarises and concludes.

6.1 The Proposed Solution

The motivation for this work can be found in the many possible methods to distribute unknown payoffs that can be identified in real economic environments where there is disagreement on a coalition's true value, such as labour and financial markets. Perhaps the most popular method in coalitional game literature is that of Chalkiadakis *et al.* [33, 34] and Suijs *et al.* [122] who replace the traditional numeric payoff vector of characteristic function games with a fixed percentage-based demand vector. Yet this method is only a subset of the possible ways to distribute the unknown value of a coalition. In these environments each agent has three main variables to consider before accepting or rejecting an offer to move to a new coalition:

1. **Expected/estimated payoff and minimum payoff**: Given agent i's knowledge about the other agents of the coalition and the environment it is in, agent i estimates the worth of the coalition C, denoted $w^i(C)$ in the *valuation disagreement coalitional game* model. Given the agreed terms at the coalition valuation $w^i(C)$, agent i can find its *estimated (or expected) payoff*. The *minimum payoff* is the minimum estimated payoff an agent will accept to join a new coalition.
2. **Share of profits**: It is possible that all/some of the agent valuations could be an underestimation of the true value. In this case, agents of the coalition will need an agreement to share any possible gains over all/some of their estimated payoffs.
3. **Share of losses**: Alternatively the agents may over-estimate all/some of their valuations of the coalitions. In this case, agents of the coalition will need an agreement to share any possible losses over all/some of their estimated payoffs. This is separate to variable 2 as some agents can make a profit on their estimated payoff when others make a loss.

[1] See Section 2.3.2 for a description.

Agents have different considerations for each of these variables. For instance, an agent i will only consider joining a coalition C if the estimated payoff that is being offered to i at the coalition value $w^i(C)$ equals or exceeds agent i's required minimum payoff. Each agent i will also want to maximise its estimated payoff at the coalition value $w^i(C)$. Additionally an agent i may be willing to sacrifice a percentage of payoff over a certain valuation of C to an agent j (where this valuation would rationally be somewhere above $w^i(C)$), if $w^j(C)$ is higher than $w^i(C)$. Sacrificing payoff at this higher valuation may entice agent j to join the coalition, possibly in return for more payoff under a valuation that i finds more likely to occur.

Different approaches treat these three variables either all together (e.g. [33, 34, 122]), or exhaustively by creating a payoff vector for each different possible combination of coalition values (e.g. [65]). The method used in this chapter is different because it combines elements of the exhaustive payoff vector proposals [65] with those used in the single percentage demand vector proposals [33, 34, 122], as discussed in the next section.

6.1.1 The Valuation Disagreement Coalitional Game Model

The valuation disagreement coalitional game (VCG) model provides more flexibility in coalitional agreements than [33, 34, 122] by allowing different percentage distributions for the different possible true values of each coalition. Yet the VCG model does not state an explicit agreement for every possible true value, due to this requirement being extremely complex even for situations with small amounts of valuation disagreement [65]. Instead, VCG relies on a contract function that gives an implicit payoff agreement for every possible true value.

The VCG does not state where exactly the difference in each agent's opinion on a coalition's value comes from (be it from agent types, possible worlds, or another method). Yet this chapter's model, like the coalitional games with beliefs model of [34], uses the simplifying assumption of single point beliefs for the reasons stated in Section 2.3.2. These reasons are: probabilistic beliefs may not be available; single point beliefs are easier to form; reasoning over single point beliefs is computationally less complex; and single point beliefs are a natural assumption in many real-world situations. Given these preliminaries, valuation disagreement coalitional games are defined as:

Definition 83: *An* valuation disagreement coalitional game *is:* $G^v = (N; w)$ *where* $N = \{1, .., n\}$ *is the set of agents and* w *is the* valuation vector *denoted* $w = \langle w^1, ..., w^n \rangle$ *where each* $w^i \in w$ *assigns every possible coalition a value* ($w^i(2^N) \rightarrow \mathbb{R}$) *representing agent* i*'s reported best guess on each coalition's value. The notation* $w(C)$ *denotes all the valuations of* C *by the agents of* C.

In this chapter, it is assumed that each agent reports w^i so that it is public knowledge. Then the valuation disagreement coalitional game outcome can accurately reflect each agent's reported valuations (which might not be their truthful valuations). The justification of this can be seen in many real life negotiations where the future value of a coalition is disputed and individuals make known their valuations to improve negotiation. For example, in the popular international television programme "Dragons' Den", where business ideas are pitched to a

panel of venture capitalists, the two or more negotiating parties usually make known their future valuation of the business when demanding a percentage share (of future payoff) to form the coalition.

Reporting personal information can create strategic issues. In this chapter, unlike [78], the concern is not with designing *incentive compatible* mechanisms where the agents perform best if they truthfully reveal their private information. Finding an incentive compatible mechanism can be a computationally difficult task [119].

Instead, like [23], in this chapter it is assume that the agents report coalition valuations in a *near incentive compatible* manner; that is, the agents cannot determine how to change their coalition valuations (from their truthful valuations) to increase their final payoff, even though this maybe theoretically possible.

A simple way to ensure near incentive compatibility is for each agent $i \in N$ to have no knowledge on the other agent's coalition valuations when i reports its coalition values. To do this, each agent i should encrypt its valuations, send them to every other agent j (i.e. $j \in N, j \neq i$), and for the decryption key to be sent only when i has received every other agent j's encrypted valuations [23]. Now in this situation, each agent i will not know if raising a single coalition's value (above the coalition's truthful value) will give i more power to demand more payoff from some other agents *or* cause another coalition to become unstable and dissolve, thus decreasing the total possible payoff agent i can demand from other agents anyway. If each agent's coalition valuations were exchanged like this, then the agents would have complete knowledge on each agent's coalition valuations and could continue to reason over which coalitions to form via coalitional bargaining, a payoff vector transfer scheme or another type of distributed algorithm (for example, similar to the previous chapter's distributed algorithm).

To reason over outcomes of valuation disagreement coalitional games, it is useful to introduce a function $\pi(C, i)$ that takes as input a coalition C and an agent $i \in C$ and outputs all the agents $M \subseteq C$ (where $i \in M$), who believe the value of C to be equal to or less than $w^i(C)$, i.e. for all $j \in M$, $w^j(C) \leq w^i(C)$ holds. Using this function, estimated payoff vectors that are used to find acceptable outcomes for valuation disagreement coalitional games can be introduced, formalised as:

Definition 84: *An* **estimated payoff vector** *denoted* $e = \langle e_1, ..., e_n \rangle$ *for a coalition structure* $CS = \{C^1, ..., C^k\}$ *in a game* G^v *satisfies: (a)* $e_i \geq w^i(\{i\})$ *(i.e.* individual rationality*) for all* $i \in N$*; and (b)* $\sum_{j \in \pi(C,i)} e_j \leq w^i(C)$ *for each agent* $i \in N$*, where* $i \in C \in CS$*. The estimated payoff vector of a coalition* C *is denoted* $e(C)$*. Using* e*, the expected payoff for each agent* i *is given by* e_i*.*

Estimated payoff vectors give the agents more flexibility when negotiating for payoff for a coalition C, as each agent i only has to offer a distribution of $w^i(C)$ that satisfies all agents $M \subseteq C$ who value C less than or equal to i. The total estimated payoff to the M agents given by $\pi(C, i)$ must never be greater than $w^i(C)$ for any coalition $C \in CS$, which ensures that each agent i can achieve its estimated valuation should the coalition's real value be equal to or greater than $w^i(C)$.

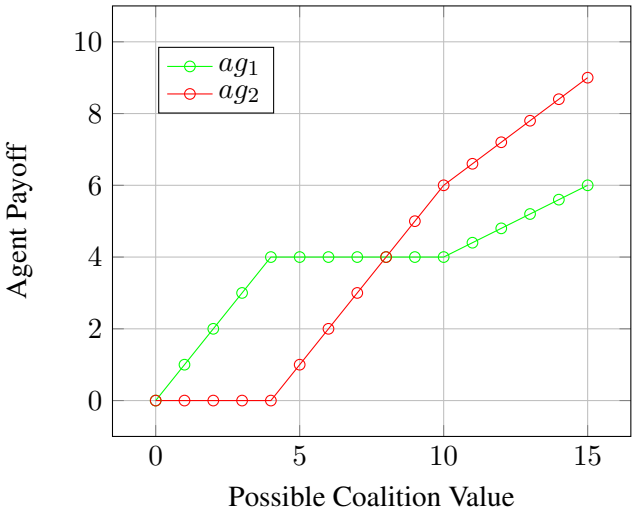

FIGURE 6.1: This figure shows how a simple contract changes agent ag_1 and ag_2's payoff given different possible future valuations of the coalition $\{ag_1, ag_2\}$, where ag_1 requires a payoff of 4 (or more) when the coalition is valued at 5 (or more) and ag_2 requires a payoff of 6 (or more) when the coalition is valued at 10 (or more).

The logic behind the estimated payoff vectors is simple. Consider two agents ag_1 and ag_2 negotiating over possibly joining together in a coalition. Agent ag_1 reports the coalition's value as 5 and requires an estimated payoff of 4 to join (i.e. 80% of 5). Agent ag_2 reports the coalition's value as 10 and requires an estimated payoff of 6 to join (i.e. 60% of 10). Then both agents will be satisfied (if the agents are risk neutral) with a *contract* that gives agent ag_1 a monotonically increasing payoff that will be ≥ 4 at the valuation 5 and gives agent ag_2 a monotonically increasing payoff that will be ≥ 6 at the valuation of 10. Additionally this contract needs to state how to work out each agent's payoff for any possible true valuation of the coalition. A simple contract (i.e. one of many possible contracts), such as the one presented in Figure 6.1, could assign all the payoff to the agent with the smallest valuation (ag_1) until its demand is reached, then all the payoff to the agent with the next smallest valuation (ag_2) until its demand is reached, and finally assign any payoff, over the total of both demands, proportionally to both agents.

These contracts are represented via a function κ that takes as input a tuple $\langle C, e(C), w(C), v(C)\rangle$ representing: the coalition (i.e. C); the estimated payoff of the agents in the coalition (i.e. $e(C)$); the reported agent valuations for that coalition (i.e. $w(C)$); and the coalition's true value (i.e. $v(C)$). The κ function outputs a payoff vector agreement $x(C)$ for the coalition at its true value $v(C)$.

More complicated contracts can be developed that distribute any possible future valuation of the coalition. These contracts could be designed for different purposes (or a combination of different purposes), such as: (i) encouraging conservative valuations; (ii) distributing the payoff in the most 'fair' manner; and (iii) distributing the payoff proportionally to some variable. The two restrictions assumed in contract functions (in this thesis) are:

1. Each agent's payoff is monotonically weakly increasing as the coalition value increases.

2. Each agent i must receive equal to or greater than its estimated payoff e_i when the coalition C is formed and $v(C) \geq w^i(C)$.

Restriction 1 means a situation cannot occur where an agent gets a payoff of y if $v(C) = z$ and a payoff of less than y if $v(C) > z$. Restriction 2 means that it is guaranteed that if any agent i correctly predicts or pessimistically values the coalition it forms, then i is to be given greater than or equal to its estimated payoff e_i.

Now that the estimated payoff vector and the contract function have been introduced, the outcome of a valuation disagreement coalitional game can be defined:

Definition 85: *An* outcome *of a valuation disagreement coalitional game is a triple consisting of a coalition structure, an estimated payoff vector and a contract function, denoted:* $\langle CS, e, \kappa \rangle$.

The different possible contract functions do not affect valuation disagreement coalitional game stability solution concepts (introduced in the next sub-section), yet may influence what valuations each agent reports. The valuation disagreement coalitional game stability solution concepts are not affected by the contract functions, because these concepts are only concerned with finding a stable outcome given the reported valuations. The contract functions are only used if the highest valuation of any coalition in the final coalition structure is wrong. In this case, some more (resp. less) payoff needs to be assigned to (resp. from) the coalition's members as the coalition's true value is higher (resp. lower) than expected.

6.1.2 Valuation Disagreement Stability Concepts

The solution concepts for valuation disagreement coalitional games that are provided in this chapter are the valuation-disagreement core (VC), the ϵ^v-valuation-disagreement core (ϵ^v-VC) and the least valuation-disagreement core (LVC) which are analogous to the CS-core, ϵ-CS-core and least-CS-core described in Section 2.2.3.

The VC is the set of stable outcomes where no subset of agents has an incentive to deviate. The ϵ^v-VC is a variant of the VC where the agents need an incentive greater than ϵ^v to deviate. The *estimated excess* of each coalition C is measured by:

Definition 86: *The* **estimated excess** *of a coalition C given an estimated payoff vector e, is denoted $\epsilon^v(C, e)$ and calculated by:*

$$\epsilon^v(C, e) = w^j(C) - \sum_{i \in C} e_i \text{ where } j \in C \text{ and } \pi(C, j) = C$$

Positive estimated excess, for a potential coalition C and the highest valuation of C by an agent (i.e. $w^j(C)$), means that the total estimated payoff of the members of C is lower than C's highest valuation. Negative estimated excess, for a potential coalition C and the highest valuation of C by an agent (i.e. $w^j(C)$), means that the total estimated payoff of the members of C is higher than C's highest valuation. The highest agent valuation was chosen because: (i) if $w^i(C) - \sum_{k \in C} e_k$ was used in the estimated excess formula for a lower

valuation $w^i(C) < w^j(C)$, then by definition a lower excess value would be returned; and (ii) if $w^i(C) - \sum_{k \in \pi(C,i)} e_k$ was used in the estimated excess formula for a lower valuation $w^i(C) < w^j(C)$, then any excess returned may be estimated payoff that is given to an agent m with a higher valuation (i.e. $m \in C$ and $m \notin \pi(C, i)$), to entice m to joint. Therefore it is the estimated excess that is propagated up to the top valuation that is of concern.

Definition 87: *The* **maximum estimated excess** ϵ^v *of a valuation disagreement coalitional game is defined as:*

$$\epsilon^v = \max_{C \subseteq N}(\epsilon^v(C, e))$$

In multi-agent systems, it is beneficial to view agents as being able to form many different coalition structures [36], hence the definitions for valuation disagreement coalitional games include a coalition structure. Both the VC and ϵ^v-VC for coalition structures can be defined using the following definition:

Definition 88: *The* **valuation-disagreement core/ϵ^v-valuation-disagreement core** *for a game G^v consists of all triples $\langle CS, e, \kappa \rangle$ where:*

1. $\sum_{i \in C} e_i = w^j(C)$, $\forall C \in CS$ *where* $j \in C$ *and* $\pi(C, j) = C$
2. $\sum_{i \in D} e_i \geq w^j(D) - \epsilon^v$, $\forall D \subseteq N$ *where* $D \notin CS$, $j \in C$ *and* $\pi(D, j) = D$.

When $\epsilon^v = 0$, the ϵ^v-VC definition is the same as the VC. For a triple $\langle CS, e, \kappa \rangle$ to be in the VC/ϵ^v-VC, conditions 1 and 2 must be satisfied. Condition 1 states that all the payoff of the highest valuation of each coalition C in the coalition structure CS must be distributed to all the agents of that coalition. The highest valuation for each coalition C in CS is used because if it was not, then at least one agent may believe there is more payoff that can be assigned. Condition 1 is therefore the *efficiency* condition. Additionally, due to the estimated payoff vector definition, the highest valuation of each coalition C should be distributed to its members in a manner such that every agent i can achieve its estimated payoff e_i by its valuation $w^i(C)$. The definition of the estimated payoff vector therefore ensures that a solution is *feasible*. Finally, condition 2 states that for every possible coalition D not in the coalition structure CS, the sum of estimated payoffs for $\pi(D, j)$ for an agent $j \in D$ with the highest valuation should be greater than or equal to j's valuation of D minus ϵ^v. If the second condition does not hold for some group of agents D, then D has a valid reason to deviate from the current CS.

The VC/ϵ^v-VC is similar to the CS-core for Beliefs (CSB) of [34]. The differences are that firstly, the CSB uses percentage demand vectors and the VC/ϵ^v-VC uses numeric estimated payoff vectors. Secondly, in the VC/ϵ^v-VC, agents can receive different percentages of the coalition's payoff at different valuations, to encourage cooperation. Finally, ϵ^v-VC is defined in the ϵ^v style, which allows the VC to be expanded if necessary, meaning that a stable solution can always be found, compared to the CSB which can sometimes be empty. A solution in the smallest non-empty ϵ^v-VC, known as the *least valuation-disagreement core* (LVC) is defined as:

Definition 89: *The* **least valuation-disagreement core** *for a game G^v consists of all triples $\langle CS, e, \kappa \rangle$ where:*

$$\langle CS, e, \kappa \rangle \text{ is in the } \epsilon^v\text{-valuation-disagreement core}$$
$$\forall \epsilon^{v\prime} < \epsilon^v, \text{ the } \epsilon^{v\prime}\text{-valuation-disagreement core is empty}$$

A triple $\langle CS, e, \kappa \rangle$ in the LVC minimises the maximum dissatisfaction of any potential coalition not in the LVC coalition structure.

6.1.3 Valuation-Disagreement Coaltional Game Properties

valuation disagreement coalitional games (VCG) essentially convert a coalitional game with valuation disagreements into a characteristic function game, by finding stability based on one value per coalition (the highest one). The difference from the traditional characteristic function game is that the estimated payoff vector adds some additional constraints on the final payoff each agent can receive (i.e. each agent i's payoff should be feasible given its coalition C and the valuations of the $\pi(C, i)$ set). The justification of this method will be discussed subsequent to the following proofs on the various properties of VCG outcomes compared to outcomes using a percentage demand vector.

Lemma 6.1. *For the same stable coalition structure CS where there* are no *valuation disagreements, then an estimated payoff vector and a percentage demand vector give the* same *total expected value.*

Proof. A percentage demand vector distributes 100% of the value of each coalition C in the coalition structure CS, giving a total of $\sum_{C \in CS} v(C)$. According to condition 1 of the VC/ϵ^v-VC, a stable estimated payoff vector does the same. □

Lemma 6.2. *For the same stable coalition structure CS where there* are *valuation disagreements, percentage demand vectors that distribute 100% of the payoff to the agents who value the coalitions in CS the highest, give the* same *total expected value as an estimated payoff vector.*

Proof. Assume a percentage demand vector distributes 100% of the highest valuation of each coalition C in the coalition structure CS, then the total is $\sum_{C \in CS} v^j(C)$ where $\pi(C, j) = C$. For the percentage demand vector to *not* total the estimated payoff vector then (according to condition 1 of the VC/ϵ^v-VC definition) for a least one coalition $C \in CS$ there must exist an agent i where $w^i(C) > w^j(C)$, but agent i cannot exist according to the assumption. □

Lemma 6.3. *For the same stable coalition structure CS where there* are *valuation disagreements, percentage demand vectors that* do not *distribute 100% of the payoff to the agents who value the coalitions the highest, give a* lower *total expected value compared to an estimated payoff vector.*

Proof. Assume a percentage demand vector d distributes equal (or greater) value to an estimated payoff vector (that distributes all of the top valuation for each coalition according to condition 1 of the VC/ϵ^v-VC definition), but d distributes less than 100% of the payoff of the highest valuation of a coalition $C \in CS$ to the agents. Then there exists a demand $0 < d_i \leq 1$ for an agent i who values C less than an agent j (i.e. $w^i(C) < w^j(C)$). As $d_i \times w^i(C) < d_i \times w^j(C)$ then the total value of the agents of the system would be increased by adding d_i to d_j. If this process is completed for all agents who do not value C the highest, then 100% of the highest valuation of C is distributed to the agents that value C the highest, meaning the new demand vector will *only now* give equal expected value compared to an estimated payoff vector (according to Lemma 6.2). □

Theorem 6.4. *For the same stable coalition structure where there* are *valuation disagreements, then an estimated payoff vector gives a higher (or equal) expected value than a percentage demand vector.*

Proof. See Lemma 6.2 and Lemma 6.3. □

Theorem 6.4 shows that solutions in the VC or ϵ^v-VC give equal or higher expected value than any percentage-based demand vector can give. In fact, according to Lemma 6.2 there is a very specific restriction placed on percentage demand vectors to give an equal expected value compared to estimated payoff vectors. This restriction is that 100% of the payoff must go to agents with the highest valuations. Therefore an agent with a lower valuation must receive zero expected payoff. This restriction seems unrealistic as rational agents should object if they are only offered zero payoff, especially if receiving zero payoff breaks the *individual rationality* condition[2].

Regarding the justification of this model; in situations where it is impossible/difficult to know the correct value for each coalition, the satisfaction of each agent according to its own beliefs should be prioritised. The proofs of this section and the experiments of Section 6.3, show that using a percentage-based demand vector is not the way to maximise the overall social welfare of the system in expectation because VCG estimated payoff vectors offer more satisfying solutions (in the form of higher expected value) for any realistic scenario for the multi-agent system.

So the first contribution of this Chapter's model is that in any *realistic* scenario with valuation disagreements between the agents[3], then estimated payoff vectors give a higher expected total payoff than a fixed percentage based model and thus collectively satisfies the agents more than the fixed percentage based model.

Another (perhaps more) important contribution of this model is that the contract functions can increase the likelihood that the agents report lower valuations and so increase the likelihood that the agents will be in profit over their expected payoff once the coalition forms and completes. This issue will be discussed in Section 6.2 after a brief example of the model is presented.

[2] See Section 2.1 for a definition.

[3] Realistic is used in this context to indicate that no agent will be satisfied with receiving zero expected value.

The trade-offs VCG make so that these additional contributions over the percentage based model can be achieved are: (i) complete knowledge on each agent's coalition valuations are assumed; and (ii) the complexity of the contract function, that is used to distribute the value of a coalition if that coalition's highest valuation was wrong, maybe more complex than the percentage based distribution method (but the exact complexity depends on the contract function designer).

It should be noted that in some percentage-based coalitional game models, such as [33, 34], an estimation of each other agent's valuations are kept in memory, so the VCG does not add any more memory requirements in this context, but can add more communication requirements depending on how many coalition valuations affect the stability of the VCG.

6.1.4 Example

In the valuation disagreement coalitional game model introduced in this chapter, the agents attempt to maximise their estimated payoff given the agents' reported valuations. This example will show the benefits of the new stability definitions presented in this chapter compared to the CS-core for beliefs (CSB) stability definition of the comparison model of [34], detailed in Section 2.3.2. This comparison model was chosen as it uses: single point beliefs (like valuation disagreement coalitional games); and percentage-based demand vectors.

Consider a 3-agent valuation disagreement coalitional game G_1^v as well as a 3-agent coalitional game with beliefs G_1^b, where the coalition values are the same for each agent in G_1^v and G_1^b, presented in Table 6.1.

Using Table 6.1's coalition valuations, the possible solutions in the game G_1^b (that uses percentage demand vectors, denoted d) are investigated. With the starting coalition structure of $\{\{1\},\{2\},\{3\}\}$, first consider whether the coalition $C = \{1,2,3\}$ can form, when each agent requires an increase of expected payoff from what they believe they can get by themselves. For any proposal for C to form, agent 1 will demand greater than 70% (i.e. $d_1 > 0.7$) of coalition C's payoff because $0.7 \times u_1(C) = u_1(\{1\})$ (i.e. because 70% of agent 1's valuation of C is equal to agent 1's valuation of itself). Agent 2 will demand greater than 40% (i.e. $d_2 > 0.4$) of coalition C's payoff because $0.4 \times u_2(C) = u_2(\{2\})$. Finally Agent 3 will demand greater than 5% (i.e. $d_3 > 0.05$) of coalition C's payoff because $0.05 \times u_3(C) = u_3(\{3\})$. Collecting all these lower bound demands into a demand vector we have $d(> 0.7, > 0.4, > 0.05)$, where $\sum_{i \in C} d_i > 1$ (i.e. the total demand is greater than 100% of coalition C's value). As C's lower bound on the total demand is >100%, then C is *infeasible* for rational agents and will not form in G_1^b.

On the other hand, coalition $C' = \{1,3\}$ can form in G_1^b because the agents of $\{1,3\}$ have lower bounds on demands that are $d'_1 > 0.7$ and $d'_3 > 0.1$ because $0.7 \times u_1(C') = u_1(\{1\})$ and $0.1 \times u_3(C') = u_3(\{3\})$, meaning the lower bound of $d'(C)$ is $\sum_{i \in C'} d'_i < 1$. This coalition is included in the best coalition structure in the CSB of G_1^b, which is $CS' = \{\{1,3\},\{2\}\}$. The coalition structure CS' can be matched with one of many possible demand vectors in the CSB solution concept. Yet the expected total payoff of any demand vector in the CSB, indicated by

Coalition	agent	Valuation
$\{1\}$	1	$w^1(\{1\}) = u_1(\{1\}) = 7$
$\{2\}$	2	$w^2(\{2\}) = u_2(\{2\}) = 6$
$\{3\}$	3	$w^3(\{3\}) = u_3(\{3\}) = 1$
$\{1,3\}$	1	$w^1(\{1,3\}) = u_1(\{1,3\}) = 10$
$\{1,3\}$	3	$w^3(\{1,3\}) = u_3(\{1,3\}) = 10$
$\{1,2,3\}$	1	$w^1(\{1,2,3\}) = u_1(\{1,2,3\}) = 10$
$\{1,2,3\}$	2	$w^2(\{1,2,3\}) = u_2(\{1,2,3\}) = 15$
$\{1,2,3\}$	3	$w^3(\{1,2,3\}) = u_3(\{1,2,3\}) = 20$

TABLE 6.1: All the coalition valuations not equal to zero for the valuation disagreement coalitional game G_1^v and the coalitional game with beliefs G_1^b, where $w^i(C)$ is agent i's valuation of C in G_1^v and $u_i(C)$ is agent i's valuation of C in G_1^b.

the vector p, will be $\sum_{i \in N} p_i = 10 + 6 = 16$ because both agents 1 and 3 believe $\{1,3\}$ is worth 10 and agent 2 believes the payoff by himself is 6.

Alternatively, using G_1^v, the coalition structure of the valuation-disagreement core contains the coalition $C = \{1,2,3\}$. A possible estimated payoff vector that can be present in the valuation-disagreement core with the coalition structure $CS = \{\{1,2,3\}\}$ is $e(8,7,5)$. This estimated payoff vector is possible because all agents receive a payoff that another coalition cannot improve on and all agents can achieve their estimated payoff by their valuation of C, i.e.: $e_1 \leq w^1(C)$, $e_1 + e_2 \leq w^2(C)$ and $e_1 + e_2 + e_3 \leq w^3(C)$.

Figure 7.2 shows different possible contracts functions that could be used for an valuation disagreement coalitional game outcome containing $CS = \{\{1,2,3\}\}$ and $e(8,7,5)$. The contract κ^1 satisfies the estimated payoff of the agent with the lowest valuation first, then the agent with the next lowest valuation and so on. Additionally κ^1 distributes the excess over the highest valuation to each agent i according to a value α_i^1 that represents i's estimated payoff divided by the highest valuation. For example, agent 1's payoff using κ^1 and the truthful valuation $v(C) = 40$ uses $\alpha_1^1 = \frac{e_1}{w^3(C)} = \frac{8}{20} = 0.4$ to find that $x_1 = e_1 + (\alpha_1^1 \times (v(C) - w^3(C))) = 8 + (0.4 \times (40 - 20)) = 16$.

Contract κ^2 is the same as κ^1 for each agent's valuation of C (i.e. when C is equal to 10, 15 or 20). Yet if the true value of the coalition $v(C)$ is lower than the lowest valuation of C (i.e. $w^1(C)$), then κ^2 divides the payoff according to $x_i = \frac{y_i}{w^1(C)} \times v(C)$ where y_i is the payoff i expected at $w^1(C)$ under κ^1. For example, agent 2's payoff at $v(C) = 6$ is $x_2 = \frac{2}{10} \times 6 = 1.2$. Additionally κ^2 distributes the excess over the highest valuation to each agent i according to a value α_i^2 (representing i's payoff on its own divided by i's valuation of C i.e. $\frac{w^i(\{i\})}{w^i(C)}$) divided by the sum of α^2 for C. For example, agent 1's payoff using κ^2 and the truthful valuation $v(C) = 40$ uses $\alpha_1^2 = \frac{w^i(\{1\})}{w^1(C)} = \frac{7}{10} = 0.7$ (additionally, $\alpha_2^2 = 0.47$ and $\alpha_3^2 = 0.05$) to find that $x_1 = e_1 + (\frac{\alpha_1^2}{\sum_{i \in C} \alpha_i^2} \times (v(C) - w^3(C))) = 8 + (\frac{0.7}{0.7+0.47+0.05} \times (40 - 20)) = 19.48$.

To summarise, this example shows that using this chapter's new stability concept, the agents are left with a coalition structure $CS = \{\{1,2,3\}\}$, which is not available using the CSB model of [34]. This coalition structure gives the agents a collective higher expected payoff when compared to the CSB, which accepts only the coalition structure $CS' = \{\{1,3\},\{2\}\}$,

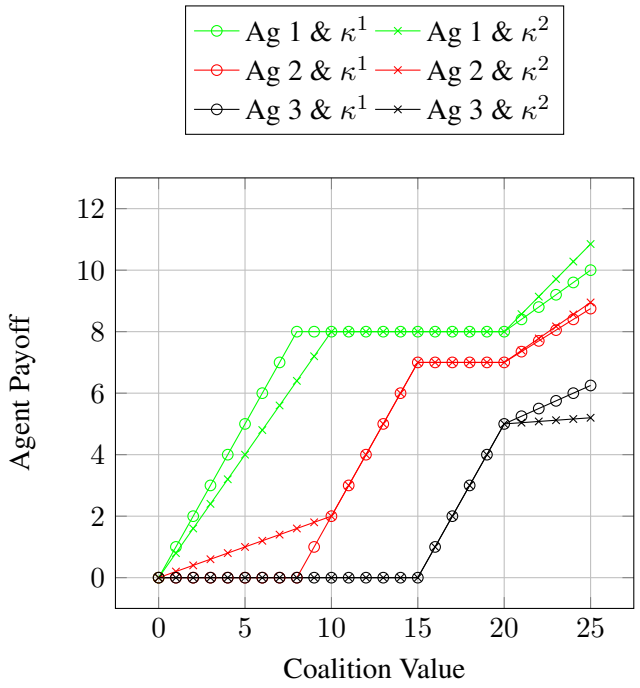

FIGURE 6.2: Shows the potential payoffs for the agents in the example game G_1^v using two different contracts functions κ^1 and κ^2.

i.e. $\sum_{i \in N} e_i = 20 > \sum_{i \in N} p_i = 16$. Meaning, in this example, the expected value of the solution in the LVC is 25% higher than the expected value of the CSB.

6.2 The Exposure to Debt/Distribution of Risk

So far, the expected value advantages of estimated payoff vectors over fixed percentage-based demand vectors have mostly been discussed (in situations where the agents reported the same valuations). In this section the exposure to debt, according to each agent's expected utility value, in valuation disagreement coalitional games is described.

In more detail, in valuation disagreement coalition games, each agent values every coalition a certain utility value. Then according to all the agent's values, each agent is assigned into a coalition and given an estimated payoff the agent can not object to. Now if the mindset is taken that each agent's estimated payoff is lent to the coalition while it is completing its task, then each agent has some debt exposure.

The distribution of the risk associated with this *estimated payoff debt exposure*[4] plays an important role in valuation disagreement coalitional games (VCGs). In this section it is shown how valuation disagreement coalitional games can be used to help lower the agents' valuations and increase the likelihood that the agents gain a profit (instead of a loss) over their expected payoff.

Gaining a profit can be very important when coalitional games are put in the context of the wider economy. When agents (individuals or businesses) commit to form a coalition for

[4]Risk in this context refers to the likelihood of the coalition defaulting on all or some of its debts to its member agents.

an expected payoff, in many cases, this payoff may be used in a budget to plan for the future. For example when a business sums its expected utilities for the next year and finds an expected profit, this expected profit may motivate one or more of the following: (i) the hiring of new employees; (ii) the commitment to expansion; (iii) the investment in new equipment. Yet if this expected profit turns into a real loss once the true values are known, the business may be forced to: (i) make employees redundant; (ii) cancel expansion plans mid-project; or (iii) default on payments for new equipment. Over-budgeting in this sense may have even worse consequences for individuals, such as: losing their job if businesses need to cut back; losing their home if they cannot afford mortgage payments anymore; and/or struggling to provide the basic needs for their family.

Essentially, in this section, it is shown how VCGs can lower the possibility of a 'market bubble'[5] appearing regarding the agents' coalition valuations. This is achieved through the distribution risk associated with the agents estimated payoff debt exposure.

6.2.1 Contract Functions

The requirements of contract functions place more risk on the estimated payoff debt exposure of the agents' with the higher valuations. These contract function requirements are: (i) that the contract function is monotonically weakly increasing; and (ii) an agent i's estimated payoff has to be achieved by agent i's valuation. Therefore if an agent over-estimates the value of a coalition, then that agent's true payoff has a higher chance of being lower than its estimated payoff. This is also true of percentage-based demand vectors but not to the same degree:

1. Using the percentage-based model where d is the demand vector and p is the vector of expected payoffs, where an agent i's expected payoff is $p_i = d_i \times u_i(C)$, i is a member of the coalition C whose true value is $v(C)$, this true coalition value is $z\%$ of $u_i(C)$ (where $0 \leq z \leq 100$), then agent i's true worst case payoff will be $x_i = p_i \times \frac{z}{100}$. Therefore $x_i = 0$ can only occur if: (a) agent i was offered no percentage share of the coalition's true value, i.e. $d_i = 0$ (which a rational agent should not agree to); (b) agent i valued the coalition at zero, i.e. $u_i(C) = 0$ (so a rational agent should not have agreed to join C in the first place); or (c) the true value of the coalition is zero, i.e. $v(C) = 0$ (i.e. the absolute worst case true value of the coalition).

2. Using the estimated payoff vector e where an agent i's expected payoff is e_i, i is a member of the coalition C whose true coalition value is $v(C)$, this true value is $z\%$ of $w^i(C)$ (where $0 \leq z \leq 100$), then agent i's true worst case payoff will be $x_i = max(0, e_i - (w^i(C) - v(C)))$. Now the value $x_i = 0$ can occur when: (a) agent i was offered no numeric share of the coalition's true value, i.e. $e_i = 0$ (which a rational agent should not agree to); (b) agent i valued the coalition at zero, i.e. $w^i(C) = 0$ (so a rational agent should not have agreed to join C in the first place); or (c) $e_i - (w^i(C) - v(C)) \leq 0$, which occurs when agent i over-estimates the value of the coalition by e_i or more.

[5] A market bubble is used here to mean that some/all of the reported coalition valuations are much higher than the true coalition valuations, leading to a 'crash' in some/all of the agents' valuations when the true value is revealed.

So considering the two points above and assuming no rational agent will join a coalition where either its payoff is zero (i.e. both 1(a) and 2(a) are irrational choices) or where it beliefs the coalition's valuation is zero (i.e. both 1(b) and 2(b) are irrational choices) we can conclude that agents can lose all their payoff significantly earlier (if they over-estimate their coalition's value) in an VCG compared to a coalitional game using a percentage-based demand vector.

A trivial example shows how contract functions "punish" higher valuations more than the percentage-based model. Take $w^i(C) = u_i(C) = 20$, $e_i = p_i = 5$ and $v(C) = 15$. Then the percentage-based model gives a worst case of $x_i = 5 \times 0.75 = 3.75$, while the VCG model gives a worst case of $x_i = max(0, 5 - (20 - 15)) = max(0, 0) = 0$. The VCG model with contract functions does give a best case payoff of 5 for this example, but this is only possible if there exists agents with a higher valuation that are punished in a harsher manner than in the percentage-based model.

6.3 Experimental Evaluation

The added value of this chapter's approach over the percentage based demand vector of [34] is evaluated empirically because the exact added utility value of the valuation disagreement coalitional game is dependent on each agent's coalition valuations. The different agent valuations can vary significantly due to the potentially significant difference in the method the agents use to interpret their possibly heterogeneous knowledge base. The fact that the valuation disagreement coalitional game model increases the risk associated with the estimated payoff debt exposure of the agents who value the coalitions higher is obvious from the discussion of Section 6.2 and therefore not analysed empirically.

For the experiments, different coalition value distributions popular in the coalition formation literature are used [93], defined as:

- *Uniform*:- Each coalition's value is determined by multiplying the number of agents in the coalition by a variable q picked from a uniform distribution between 0 and 1.0. Formally, for an agent i, $w^i(C) = u_i(C) = max(0, |C| \times q)$, where: $q \in \mathcal{U}(a, b)$; $a = 0$; and $b = 1.0$.
- *Normal*:- Each coalition's value is determined by multiplying the number of agents in the coalition by a variable q picked from a normal distribution with mean 1 and variance 0.1. Formally, for an agent i, $w^i(C) = u_i(C) = max(0, |C| \times q)$, where: $q \in \mathcal{N}(\mu, \sigma^2)$; $\mu = 1$; and $\sigma^2 = 0.1$.
- *NDCS*:- Each coalition's value is determined by a variable q picked from a normal distribution with mean of the coalition's size and variance also according to the coalition's size. Formally, $v(C) = max(0, q)$, where: $q \in \mathcal{N}(\mu, \sigma^2)$; $\mu = |C|$; and $\sigma^2 = |C|$.

These experimental conditions differ slightly compared to the conditions of the previous chapter. In the previous chapter the different distributions were used to give values to the *different coalitions* of the same size, values that differed according to their coalition value distribution.

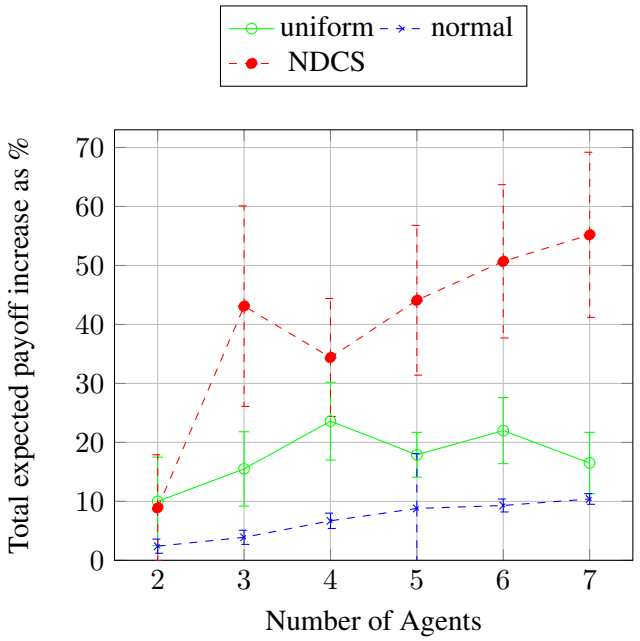

FIGURE 6.3: This figure shows the increase in the total expected payoff of the agents when using the LVC solution instead of the CSB solution. A 95% confidence interval was used.

Even though this is still the case, additionally now the values for the *same coalition* differ between the agents according to the coalition value distribution, which allows for coalition value disagreements to occur.

In the experiments of this chapter, normal (and NDCS) coalition-value distributions simulate the situations where agents have similar (but still possibly differing) opinions on the value of each coalition, as q is centered around a mean with a small standard deviation. Uniform coalition-value distributions on the other hand, simulate situations where the agents have quite different opinions on the value of each coalition, as q will in general be a lot more varied.

The following three hypotheses have been tested using both the coalition-value distributions:

(i) Outcomes in the least valuation-disagreement core (LVC) typically provide a higher total estimated payoff compared to the total expected payoffs in the CS-core for beliefs (CSB), when the same agent valuations are used for both solution concepts.

(ii) Coalitions/coalition structures can form in the LVC that cannot under the CSB solution concept, when the same agent valuations are used for both solution concepts.

(iii) LVC outcomes provide an improvement to the majority of the agents' expected payoff in the majority of cases when compared to CSB outcomes, when the same agent valuations are used for both solution concepts.

All the coalition-value distribution experiments were run for 2 to 7 agents. As the LVC is never empty, yet the CSB maybe empty, the comparisons between the two approaches were taken from the first 30 runs of each coalition-value distribution that returned a solution in the

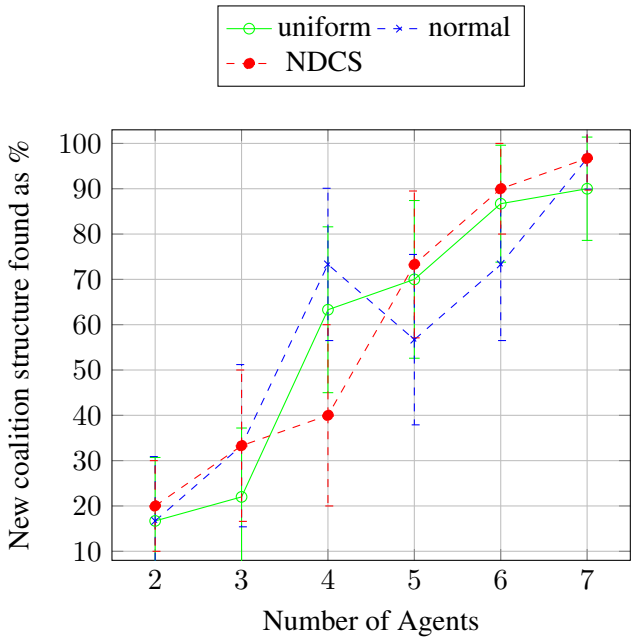

FIGURE 6.4: This figure shows the percentage number of times a stable coalition structure occurred in the LVC solution that was not in the CSB solution. A 95% confidence interval was used.

CSB. The number of runs matches previous work on coalition games in multi-agent systems (e.g. [32, 88]).

In Figure 6.3, the mean increase of the total expected payoff of the agents using the LVC over the CSB is shown. Additionally Figure 6.4 shows the mean number of experiments where the coalition structure in the resulting LVC was not in the CSB. Therefore Figure 6.4 shows the percentage number of times that the LVC allowed stable cooperation between agents that the CSB did not.

Figure 6.3 shows that the uniform coalition-value distribution yielded a greater mean improvement over the normal coalition-value distribution. The reason being that when a fixed percentage-based demand vector is used, then there will be more breakdowns of negotiations when the different valuations are spread out, meaning the CSB has less coalitions to find an acceptable solution with. Overall the greatest mean improvement was shown to be the NDCS distribution, as each individual coalition can be assigned more value using this distribution than the other two.

In summary, Figure 6.3 and Figure 6.4 show that: (i) with three well known coalitional-value distributions, the LVC stability solution concept gives higher expected payoffs for the agents compared to the CSB solution concept (when the same valuations are used for the LVC and CSB solutions); and (ii) with three well known coalitional-value distributions, the LVC stability solution concept allows new stable coalition structures to form that are not possible under the CSB solution concept (when the same valuations are used for the LVC and CSB solutions).

Thus these experiments show that from a system-wide perspective the LVC improves on the CSB. To show that the LVC also improves on the CSB on an individual level, Figure 6.5 is introduced. In this figure, the percentage of agents who increased their expected payoff in the

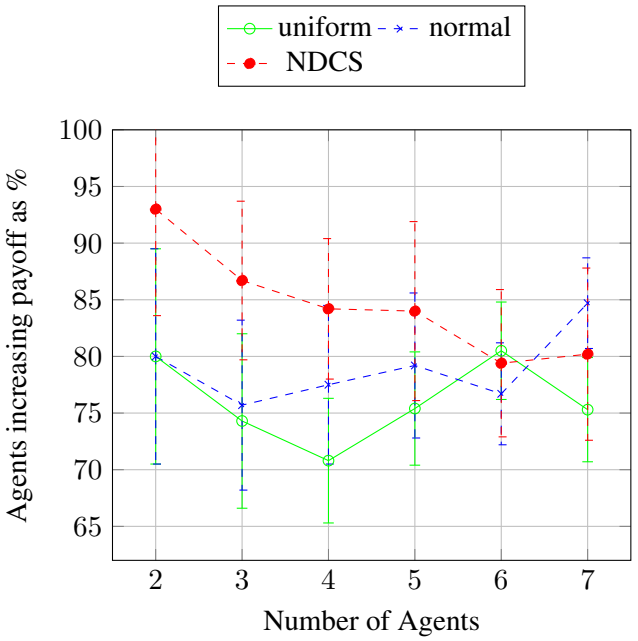

FIGURE 6.5: This figure shows the percentage of agents who increased their expected payoff in the experiments, using a LVC solution instead of a CSB solution. A 95% confidence interval was used.

experiments using a LVC solution instead of a CSB solution is plotted (the other agents lose out as there was not one experiment where any agent received the same expected payoff). The figure shows that for all the coalition-value distributions, there is an improvement to the majority of agents' expected payoffs in the LVC compared to CSB outcomes. Therefore, Figure 6.5 shows that, when the same valuations are used: (iii) using the LVC solution over the CSB will satisfy more agents regarding their expected payoffs.

6.4 Summary

This chapter has discussed how the agents can handle disagreements between themselves over the value of a coalition, when the coalitions are valued *quantitatively*, and so complements Chapter 3 that discussed how to handle disagreements when the coalitions are valued *qualitatively*.

To handle these quantitative value disagreements, a new coalitional game model, named *valuation disagreement coalitional games*, has been introduced as well as new solution concepts, one of which is never empty. These solution concepts detail stable coalition structures and estimated payoff vectors, matched with a contract function. This contract function allows a payoff vector solution for every possible valuation of every coalition in the coalition structure to be implicitly given. Detailing the future possible payoffs for any valuation via a function is computationally less complex compared to providing an explicit solution for every possible valuation, which is the approach of Bayesian Coalitional Games [65]. Additionally the implicit payoff vector solutions can give different percentage payoffs for the agents at different coalition valuations, allowing more flexibility in negotiation than fixed percentage-based models such as coalitional games with beliefs [34]. The flexibility of this chapter's approach over [34] allows

outcomes with a higher total expected value to be found that can include coalition structures that the approach of [34] does not allow to form. Finally due to the distribution of risk associated with the estimated payoff debt exposure in valuation disagreement coalitional games, Section 6.2 discussed how pessimistic valuations by the agents are encouraged, which is beneficial if the expected payoff to be gained by the formation of each coalition is of high importance.

Chapter 7

Chapter 7

Conclusions and Future Work

This Chapter presents the main contributions of this thesis in Section 7.1, followed by a discussion of a number of possible future work directions in Section 7.2.

7.1 Summary of Contributions

The aim of this thesis was to attempt to answer the research question defined in Section 1.3:

How can decentralised coalition formation methods be designed that: (i) balance the computational load; (ii) do not allow distributed knowledge to affect results; (iii) bound the communication costs; and (iv) handle valuation disagreements of a quantitative or qualitative manner?

All the contributions of this thesis relate to problems identifed by (i), (ii), (iii) and (iv) of the research question. Chapters 4 and 5 describe decentralised coalition formation methods that balance the computational load approximately equally, thereby achieving (i). Chapter 5 describes a decentralised coalition formation method that provides a guarantee that optimal and stable solutions will be found when information on coalition values is distributed, thereby achieving (ii). Chapters 4 and 5 describe decentralised coalition formation methods that constraint communication costs, thereby achieving (iii). Finally chapters 3 and 6 offer new resolution methods to find the best coalitions when there are disagreements on the values of the coalitions, thereby achieving (iv). Exactly how these chapters achieve this is expanded on below:

(i) **Balancing the computational load**. Two new methods for addressing this issue are given in this thesis, one method in Chapter 4, where the agents divide the coalition value calculations between themselves, and another method in Chapter 5 where the agents divide the optimal coalition structure and least core stable payoff vector search between themselves.

Chapter 4's method, named the DCG algorithm, is based on a new manner to list coalitions of size s in an n agent characteristic function game. This new listing method relates to the different (n, s)-increment arrays that are of size s and consists of: an integer partition of $(n - s)$ (of less than or equal size to s), and accompanying zeros if the chosen integer partition is less than size s. Using these new lists, every agent is assigned to calculate

the value of a coalition, only if that coalition includes itself as a member. It is shown in the proofs of Appendix A that each coalition's value is calculated once and only once. Additionally, it is seen from the discussion and analysing the algorithm itself, that if two agents calculate the exact same number of coalitions of each size s, then each agent has the exact same number of operations to perform.

Chapter 5's methods, named the DDP algorithms, are based on: (a) new proofs for synergy coalition groups (SCG); (b) having distributed coalition value calculations as input; (c) having an indexing scheme for two-set partitions of a coalition; and (d) modifying the previously introduced dynamic programming (DP) algorithm of [138], which is guaranteed to find the optimal coalition structure. Part (a) will be discussed in (ii). Given (b), provided by sections of the DCG algorithm, the agents can distribute the operations to split each coalition into two (required by the DDP algorithms and henceforth known as *split operations*) between themselves. Additionally when (c) is used to distribute the split operations of the grand coalition, the range of split operations given to the agents tightens and it is proven that the number of split operations becomes exactly equal if the number of agents is an odd prime. If the number of agents is not an odd prime, the DDP* algorithm that uses (c) still has a smaller range of split operations between the agents, compared to the DDP algorithm that does not use (c).

(ii) **Guaranteeing that optimal coalition formation solutions are found**. A new methods for addressing this issue is given in this thesis. The method is presented in Chapter 5 where the agents are guaranteed to find the weak least core$^+$ for the superadditive cover of any characteristic function game (where the $+$ indicates that cross-coalition side payments are allowed).

Chapter 5's method, as previously noted, relies on new proofs for synergy coalition groups (SCG). These proofs show that: (a) for any superadditive characteristic function game, using the coalitions only in the SCG representation, a solution in the weak least core is guaranteed to always be found, should a linear program be run to find a weak least core solution; and (b) for any superadditive characteristic function game, using the coalitions only in the SCG representation, a solution in the strong least core is not guaranteed to always be found, should a linear program be run to find a strong least core solution, yet a guaranteed bound around the strong least core is given. The DDP algorithms are used to guarantee that the coalitions in the SCG representation are located. Additionally, as the proofs apply also to the superadditive cover, then for any characteristic function game's superadditive cover, a solution in the weak least core$^+$ (where the $+$ indicates cross-coalition side payments are allowed), can always be found using the DDP algorithms.

(iii) **Bounding the communication costs**. Two new methods for addressing this issue are given in this thesis, one method in Chapter 4, where the agents complete the distributed calculation of all the coalition values with zero communication costs, and another method in Chapter 5, where a method to restrict the communication on each potential coalition is detailed when searching for the superadditive cover of a characteristic function game.

Chapter 4's method relies on the previously stated new listing method of the coalitions. As all agents know the listing method, and the DCG algorithm describes to each agent exactly what coalition of each list it should calculate the value of, then no communication is required between the agents to coordinate themselves.

Chapter 5's method, discussed in Section 5.3, shows how to restrict the communication between the agents when calculating the superadditive cover of a characteristic function game. Section 5.3 details three ways to restrict the communication costs: (1) assign each two-set partition of a coalition an index value to restrict the number of bits needed to communicate which partition has the best synergy; (2) communicate one bit per coalition indicating whether it has synergy or not; and (3) state a worst case number of bits to communicate per coalition.

(iv) **Allow coalition formation when there are disagreements on coalition values**. Two new methods for addressing this issue are given in this thesis, one method in Chapter 3, where the agents value the coalitions qualitatively, and another method in Chapter 6, where the agents value the coalitions quantitatively.

Chapter 3's method is a new argumentation-based approach where agents instantiate an argumentation scheme to suggest a coalition to form. Critical questions are provided to allow the agents to challenge the instantiations. To allow the agents to use this argumentation-based approach, two new dialogue game communication protocols are defined: (a) the $C\text{-}inq$ dialogue game that allows the agents to reason over the state of the world; and (b) the $C\text{-}pAct$ dialogue game that allows the agents to put forth and attack arguments for a coalition to form. Finally, to allow the agents to reason over these instantiated argumentation schemes and critical questions, a new argument evaluation method for value-based argumentation frameworks is detailed. This new argument evaluation method finds the best coalitions for the agents to form, in terms of the agents reported preferences.

Chapter 6's method is a new cooperative game theory model of coalitional games, named valuation disagreement coalitional games, where the agents disagree on the quantitative value of each coalition. This model provides new stability solution concepts that detail exactly which coalitions (and payoff vectors) are classified as the most stable, and therefore best to form. This chapter's model is shown to encourage pessimistic valuations in the agents because the model stacks the risk of the coalition not achieving its combined predicted value, on the agents who report a higher value than the real coalition value. Encouraging pessimistic valuations is a positive property in microeconomic models because assigning loss/debt can having a wide ranging social impact.

7.2 Future Directions

The results presented in this thesis have provided a number of possible directions for future work:

- **Hybrid qualitative and quantitative model**: In this thesis, a model for coalition formation in qualitative environments is outlined in Chapter 3 and then a quantitative environment model is used for the other chapters. In multi-agent systems, there are certain environments that are inherently quantitative while other environments are best described in a qualitative manner [132]. Yet there are many environments that can be described with a mix of both qualitative and quantitative models. For example, if a coalition achieves some goal or promotes some social-value, to what degree this is achieved is most likely best described in a quantitative form. Alternatively, if a coalition achieves some utility, exactly why it achieves this is most likely best described in propositional (i.e. qualitative) form.

  In this thesis, allowing the agents to intelligently reason in a qualitative environment was achieved through argumentation, while allowing the agents to intelligently reason in a quantitative environment was achieved through game theoretic analysis. This suggests that a good starting point for finding a hybrid qualitative and quantitative coalition formation model, is to investigate how the qualitative coalitional formation models of argumentation and the quantitative coalition formation models of game theory can be blended together.

- **Developing the DCG algorithm**: The DCG algorithm, given in Chapter 4, could be developed by adapting it: (i) for a dynamic number of agents; (ii) for repeating the calculation of certain coalitions whose values have changed; and (iii) for the possibly different computational resources of the agents. Additionally as DCG is essentially distributing all possible subsets to the agents, future work could investigate DCG in other applications apart from coalition formation. These new applications would all have to require the distribution of sets for some calculation.

- **Developing decentralised anytime algorithms**: The DDP algorithms to complete the coalition formation process presented in this thesis are dynamic programming algorithms and so give a solution only when the algorithm is run to conclusion. On the other hand, anytime algorithms can return a solution at anytime, which is guaranteed to be within a bound from the optimal solution [94]. In [94], it was shown that an anytime algorithm, named IP, can be faster than the DP dynamic programming algorithm of [138] for some standard coalition-value distributions. This is the case even though the worst case running time of DP is $O(3^n)$, while the worst case running time of IP is $O(n^n)$. With this is mind, an interesting future research direction would be to develop a decentralised anytime algorithm for coalition formation that has the guarantees of: (i) finding the optimal coalition structure; and (ii) finding a stable payoff vector in a least core related solution concept. A decentralised anytime algorithm for locating the optimal coalition structure does exist, named D-IP [73], yet D-IP does not provide any guarantees on whether (ii) can be achieved.

- **Incentive compatibility**: In both the DCG algorithm of Chapter 4 and the DDP algorithms of Chapter 5, the agents are given only coalitions that include themselves as a

member, to motivate them to follow the algorithm. Yet since self-interested agents may use the algorithms, it is not guaranteed that they will follow the algorithms completely. In this case the tools of mechanism design [80] may have to be used, where the DCG algorithm can be wrapped in some form of mechanism that is *incentive compatible*. An mechanism is incentive compatible if the agents fare best when they truthfully follow it [119], meaning rational self-interested agents will be guaranteed to comply with an incentive compatible mechanism. A mechanism is *near incentive compatible* if the agents cannot reason on how to lie profitably. Proving the incentive compatibility/near incentive compatibility of the algorithms in this thesis (or wrapping the algorithms in another mechanism so that they become incentive compatible/near incentive compatible [80]), is left for future work.

- **Controlling the risk in uncertain environments**: In the valuation disagreement coalitional games of Chapter 6, the agents are encouraged to report pessimistic valuations by heightening the risk of an agent losing all its expected payoff, the higher that agent values the coalition that forms. A very interesting future extension is to fully investigate different contract functions. Removing the assumption that contract functions have to allow an agent to achieve its expected payoff by its valuation of the coalition, could allow for the contract function designer to have better control over the multi-agent system's reported valuations. This additional control would come from allowing the contract designer more freedom to redistribute each coalition's true payoff, according to the agent's reported valuations. For example, contract functions without the previously mentioned assumption could: (i) assign a payoff penalty of varying percentage to every agent reporting a coalition value above the lowest reported valuation, where the percentage is dependent on how high the agent's valuation is in comparison to the other agents; and (ii) assign a valuation as a threshold and any reported valuation above this threshold value could be assigned a penalty. This threshold value could represent the mean, median or mode of the reported values. Both (i) and (ii) would allow the contract designer more control over the valuations the agents' report, controlling (to some degree) how low or high a valuation report is. This future work has comparisons to economic markets and these comparisons should be investigated.

These directions indicated above are just some of the possible options given the results of this thesis. I look forward to seeing any possible future research directions inspired by my thesis.

Appendix A

Proof for the Distributed Coalition Generation Algorithm

In this appendix, the definitions from Section 4.1.3 are restated. Following the definitions are the theorems and proofs that show that each coalition is assigned to *one and only one* agent when the DCG algorithm of Chapter 5 is used.

A.1 Definitions

Let n and s be natural numbers where n is the number of agents and s is the size of the coalitions where $s \leq n$. The notation $\underline{t}$ is used to denote an arbitrary sequence of non negative integers $\underline{t} = \langle t_0, t_1, \ldots, t_{s-1} \rangle$ such that $\sum_{i=0}^{s-1} t_i = n - s$. Such a sequence is named the *increment array*.

The *period of* $\underline{t}$, denoted by $\pi(\underline{t})$ is

$$\min_{1 \leq p \leq s} \quad \underline{t} = \langle t_0, t_1, \ldots, t_{p-1}, t_0, t_1, \ldots, t_{p-1}, \ldots, t_0, t_1, \ldots, t_{p-1} \rangle$$

That is, $\underline{t}$ formed by m identical copies of a sequence of length p.

Given $C \subseteq \{1, 2, \ldots, n\}$, agent ag (for some $1 \leq ag \leq n$) generates C from ag and $\underline{t}$ if $C = \{ag_1, ag_2, \ldots, ag_s\}$ and:

$$ag_i = \begin{cases} ag & \text{if} \quad i = 1 \\ (ag + \phi_i) \bmod n & \text{if} \quad (ag + \sum_{k=0}^{i-2} t_k + (i-1)) \bmod n \neq 0 \\ n & \text{if} \quad (ag + \sum_{k=0}^{i-2} t_k + (i-1)) \bmod n = 0 \end{cases}$$

Given an *arbitrary* sequence of non-negative integers, $\underline{b} = \langle b_0, \ldots, b_{s-1} \rangle$, the *n-correction* of $\underline{b}$ is the sequence, $corr(\underline{b}, n)$, obtained by replacing each $b_i > n$ with the value $(b_i \bmod n)$. If $\underline{c} = corr(\underline{b}, n)$ and $\underline{c} \neq \underline{b}$ then $\underline{b}$ is said to be *uncorrected* for n.

Additionally $C(ag, \underline{t})$ denotes the subset of $\{1, 2, \ldots, n\}$ generated by the increment array $\underline{t}$ from ag, where $C(ag, \underline{t})$ will have been $n\text{-}corrected$ if required. The proofs in A.2 demonstrate

that each $\underline{t}$ only needs to be used by $r = (n \times \pi(\underline{t}))/s$ different agents. If more than r agents use $\underline{t}$ to generate a coalition, then repeated coalitions will be generated.

Finally, if two non-negative integer sequences $\underline{t}^x$ and $\underline{t}^y$ generate the same coalition C for two different agents $i, j \in C$ (i.e. $C(i, \underline{t}^x) = C(j, \underline{t}^y)$), then $\underline{t}^x$ and $\underline{t}^y$ are classified as the same equivalence class, denoted $\underline{t}^x \approx \underline{t}^y$. The Proofs in Section A.2 show that rather than consider *every* possible increment array, it suffices only to consider a *single* representative from each equivalence class.

To simplify the notation let,

$$\varphi_r \;=\; \begin{cases} 0 & \text{if} \quad r = 1 \\ \sum_{k=0}^{r-2} \; t_k \;+\; (r-1) & \text{if} \quad 2 \leq r \leq s+1 \end{cases}$$

Note that $\varphi_{s+1} = n$.

A.2 Theorems

The main theorem is the following.

Theorem A.1. *For any increment array $\underline{t}$, and for all $1 \leq i \leq j \leq n$,*

$$C(i, \underline{t}) = C(j, \underline{t}) \quad \Leftrightarrow \quad \exists\, 0 \leq r \leq \frac{(n-i)s}{n\pi(\underline{t})} \;:\; j \;=\; i \;+\; r \times \left(\frac{n\pi(\underline{t})}{s} \right) \tag{A.1}$$

Proof. It is first shown that if $j = i + (rn\pi(\underline{t})/s)$ for r in the range stated then $C(i, \underline{t})$ and $C(j, \underline{t})$ are identical. Notice that the case $r = 0$ is trivial and it suffices to consider only $r = 1$. Note that no assumptions are made concerning i other than those prescribed by the requirement $1 \leq i \leq n - \pi(\underline{t})n/s$.

The (uncorrected with respect to n) sequence generated by $C(i, \underline{t})$ is,

$$\langle i + \varphi_1,\; i + \varphi_2,\; \ldots,\; i + \varphi_s \rangle$$

and that by $C(j, \underline{t})$ (again uncorrected with respect to n) is,

$$\langle i + (n\pi(\underline{t})/s) + \varphi_1,\; i + (n\pi(\underline{t})/s) + \varphi_2,\; \ldots,\; i + (n\pi(\underline{t})/s) + \varphi_s \rangle$$

Consider the term $\varphi_{\pi(\underline{t})+k}$ for $1 \leq k \leq s - \pi(\underline{t})$.

$$\begin{aligned} \varphi_{\pi(\underline{t})+k} \;&=\; \sum_{p=0}^{\pi(\underline{t})+k-2} \; t_p \;+\; \pi(\underline{t}) + k - 1 \\ &=\; \sum_{p=0}^{\pi(\underline{t})-1} \; t_p \;+\; \sum_{q=\pi(\underline{t})}^{\pi(\underline{t})+k-2} \; t_q \;+\; \pi(\underline{t}) + k - 1 \\ &=\; \varphi_{\pi(\underline{t})+1} \;+\; \sum_{q=0}^{k-2} \; t_q \;+\; k - 1 \\ &=\; \varphi_{\pi(\underline{t})+1} \;+\; \varphi_k \end{aligned}$$

The penultimate identity follows from $t_{\pi(\underline{t})+q} \;=\; t_q$.

Now consider $\varphi_{\pi(\underline{t})+1}$. We have,

$$\varphi_{\pi(\underline{t})+1} = \sum_{k=0}^{\pi(\underline{t})-1} t_k \;+\; \pi(\underline{t})$$

and

$$\begin{aligned} n - s \;=\; \sum_{k=0}^{s-1} t_k \quad &= \quad \sum_{r=0}^{s/\pi(\underline{t})-1} \; \sum_{k=r\pi(\underline{t})}^{(r+1)\pi(\underline{t})-1} t_k \\ &= \quad \sum_{r=0}^{s/\pi(\underline{t})-1} \; \sum_{k=r\pi(\underline{t})}^{(r+1)\pi(\underline{t})-1} t_{k-r\pi(\underline{t})} \\ &= \quad \frac{s}{\pi(\underline{t})} \sum_{p=0}^{\pi(\underline{t})-1} t_p \\ &= \quad \frac{s}{\pi(\underline{t})} \left(\varphi_{\pi(\underline{t})+1} - \pi(\underline{t})\right) \end{aligned}$$

We deduce, therefore, that

$$\varphi_{\pi(\underline{t})+1} \quad = \quad \frac{n\pi(\underline{t})}{s} \tag{A.2}$$

Hence from our earlier analysis of $\varphi_{\pi(\underline{t})+k}$ we obtain

$$\varphi_{\pi(\underline{t})+k} \quad = \quad \frac{n\pi(\underline{t})}{s} \;+\; \varphi_k \tag{A.3}$$

Returning to the (uncorrected) sequences defining $C(i, \underline{t})$ and $C(i + n\pi(\underline{t})/s, \underline{t})$, the former contributes

$$\bigcup_{k=1}^{s} \{i \;+\; \varphi_k\} \quad = \quad \bigcup_{r=0}^{s/\pi(\underline{t})-1} \; \bigcup_{k=1}^{\pi(\underline{t})} \{i + r\varphi_{\pi(\underline{t})+1} + \varphi_k\}$$

which is

$$\bigcup_{r=0}^{s/\pi(\underline{t})-1} \; \bigcup_{k=1}^{\pi(\underline{t})} \left\{i + \frac{rn\pi(\underline{t})}{s} + \varphi_k\right\} \tag{A.4}$$

And, by a similar analysis of $j = i + (n\pi(\underline{t})/s)$, in the latter case, we obtain, prior to correction, $C(j, \underline{t})$ as:

$$\bigcup_{r=0}^{s/\pi(\underline{t})-1} \; \bigcup_{k=1}^{\pi(\underline{t})} \left\{i + \frac{(r+1)n\pi(\underline{t})}{s} + \varphi_k\right\} \tag{A.5}$$

Comparing (4) and (5), the terms with $1 \leq r < s/\pi(\underline{t}) - 1$ clearly occur in both sets. The terms for $r = 0$ in (4) are

$$\bigcup_{k=1}^{\pi(\underline{t})} \{i + \varphi_k\}$$

Similarly, the terms corresponding to $r = s/\pi(\underline{t}) - 1$ in (5), are

$$\bigcup_{k=1}^{\pi(\underline{t})} \left\{ i + \frac{s}{\pi(\underline{t})} \frac{n\pi(\underline{t})}{s} + \varphi_k \right\}$$

These terms are prior to n correction, so that

$$i + \frac{s}{\pi(\underline{t})} \frac{n\pi(\underline{t})}{s} + \varphi_k \in C(j, \underline{t}) \;\; = \;\; i \; + \; n \; + \; \varphi_k \;\; = \;\; i + \varphi_k \in C(i, \underline{t})$$

This establishes the first part of the Theorem:

$$j \;=\; i + r \left(\frac{n\pi(\underline{t})}{s} \right) \;\; \Rightarrow \;\; C(i, \underline{t}) = C(j, \underline{t})$$

To complete the proof we show that

$$C(i, \underline{t}) = C(j, \underline{t}) \;\; \Rightarrow \;\; \exists\, 0 \leq r \leq \frac{(n-i)s}{n\pi(\underline{t})} \;:\; j \;=\; i \; + \; r \times \left(\frac{n\pi(\underline{t})}{s} \right)$$

First observe that if $C' = C(j, \underline{t}) \subseteq \{1, 2, \ldots, n\}$ of size s is the same set as $C = C(i, \underline{t}) = \{x_1, \ldots, x_s\}$ there must be some $2 \leq r \leq s$ such that

$$corr(\langle i + \varphi_r + \varphi_1, i + \varphi_r + \varphi_2, \ldots, i + \varphi_r + \varphi_p, \ldots, i + \varphi_r + \varphi_s \rangle, n)$$

is exactly the set $C(i, \underline{t})$ whose elements are

$$corr(\langle i + \varphi_1, i + \varphi_2, i + \varphi_3, \ldots, i + \varphi_p, \ldots, i + \varphi_s \rangle, n)$$

In summary from $C(i, \underline{t}) = C(j, \underline{t})$ we can deduce: $i \in C(i, \underline{t})$ and each element contributing to $C(i, \underline{t})$ (prior to n-correction) has the form $i + \varphi_k$ so that there is a choice, r, for which by $j \in \{i + \varphi_r, i + \varphi_r - n\}$.

The terms generated by $(i, \underline{t})$ are,

$$\beta(i) \;=\; corr(\langle i + \varphi_1, \ldots, i + \varphi_p, \ldots, i + \varphi_s \rangle, n)$$

The terms generated by $(i + \varphi_r, \underline{t})$ are

$$\beta(r) \;=\; corr(\langle i + \varphi_r + \varphi_1, \ldots, i + \varphi_r + \varphi_p, \ldots, i + \varphi_r + \varphi_s \langle, n)$$

From the premise, every term in $corr(\beta(r), n)$ must correspond to some term in $corr(\beta(i), n)$ That is,

$$\forall\, p\, \exists\, q \;\; : \;\; i + \varphi_p \;\in\; \{i + \varphi_r + \varphi_q, i + \varphi_r + \varphi_q - n\}$$

After some trivial rearrangement this is simply,

$$\forall\, p\, \exists\, q \;\; : \;\; \varphi_p \;\in\; \{\varphi_r + \varphi_q, \varphi_r + \varphi_q - n\}$$

Observe that for any sequence

$$\langle y_1, y_2, y_3, \ldots, y_s \rangle$$

generated via $(y_1, \underline{t})$ there is at most one index l for which

$$y_i \;\leq\; n \;\; \forall\, i \leq l \;\;;\;\; y_{l+1} \;>\; n \;\;;\;\; y_i - n \;\leq\; n \;\; \forall\, l+2 \leq i \leq s$$

In consequence, the values $\langle y_1, \ldots, y_l \rangle$ are strictly increasing, as are the values $\langle y_l - n, y_{l+1} - n, \ldots, y_s - n \rangle$. We can therefore deduce the following:

For φ_r witnessing the behaviour of $\underline{t}$ in the analysis above, there is a unique index p for which

$$\varphi_r + \varphi_q \;\; \text{is} \;\; \begin{cases} \leq n & \text{if} \quad q \leq p \\ > n & \text{if} \quad q > p \end{cases}$$

So that from

$$i + \varphi_r + \varphi_p \;\; = \;\; i + n \;\; = \;\; i + n + \varphi_1$$

It follows that,

$$\begin{array}{lcl}
\varphi_r + \varphi_p & = & n + \varphi_1 \\
\varphi_r + \varphi_{p+1} & = & n + \varphi_2 \\
 & \cdots & \\
\varphi_r + \varphi_{p+k} & = & n + \varphi_{k+1} \\
 & \cdots & \\
\varphi_r + \varphi_{p+(s-p)} & = & n + \varphi_{s-p+1} \\
 & & \\
\varphi_r + \varphi_{p-1} & = & \varphi_s \\
\varphi_r + \varphi_{p-2} & = & \varphi_{s-1} \\
 & \cdots & \\
\varphi_r + \varphi_{p-k} & = & \varphi_{s-k+1} \\
\varphi_r + \varphi_{p-(p-2)} & = & \varphi_{s-p+3} \\
\varphi_r + \varphi_{p-(p-1)} & = & \varphi_{s-p+2}
\end{array}$$

In consequence, s and r, uniquely determine the value of p as $s - r + 2$. Rearranging the expression above, gives

$$\begin{array}{lcl}
n + \varphi_1 & = & \varphi_r + \varphi_{(s-r+2)} \\
n + \varphi_2 & = & \varphi_r + \varphi_{(s-r+2)+1} \\
 & \cdots & \\
n + \varphi_{k+1} & = & \varphi_r + \varphi_{(s-r+2)+k} \\
 & \cdots & \\
n + \varphi_{s-(s-r+2)+1} & = & \varphi_r + \varphi_{(s-r+2)+(s-s-r+2)} \\
 & & \\
\varphi_{s-(s-r+2)+2} & = & \varphi_r + \varphi_{(s-r+2)-((s-r+2)-1)} \\
\varphi_{s-(s-r+2)+3} & = & \varphi_r + \varphi_{(s-r+2)-((s-r+2)-2)} \\
 & \cdots & \\
\varphi_{s-k+1} & = & \varphi_r + \varphi_{(s-r+2)-k} \\
 & \cdots & \\
\varphi_{s-1} & = & \varphi_r + \varphi_{(s-r+2)-2} \\
\varphi_s & = & \varphi_r + \varphi_{(s-r+2)-1}
\end{array}$$

Giving the final set of identities that the increment array $\underline{t}$ must satisfy in order for $C(i, \underline{t}) = C(i + \varphi_r, \underline{t})$ post n-correction (recall that $\varphi_{s+1} = n$).

$$\varphi_r \ + \ \varphi_q \ = \ \begin{cases} \varphi_{r+q-1} & \text{if} \quad 2 \leq q \leq s - r + 1 \\ \varphi_{s+1} + \varphi_{r+q-1-s} & \text{if} \quad s - r + 2 \leq q \leq s \end{cases}$$

where

$$\varphi_r \ = \ \begin{cases} 0 & \text{if} \quad r = 1 \\ \sum_{k=0}^{r-2} \ t_k \ + \ (r-1) & \text{if} \quad 2 \leq r \leq s \end{cases}$$

We first observe that these systems of identities can be expressed solely in terms of $\langle t_0, \ldots, t_{s-1} \rangle$: the terms on the left-hand side contribute $r - 1 + q - 1 = r - q - 2$ in addition to the t_k values. The terms on right-hand side, however, also contribute $r + q - 3$ (via $s + (r + q - 2 - s)$ when $s - r + 2 \leq q \leq s$) in addition to the t_k values, so that these cancel.

In total we have so far shown that if after n-correction, $C(i, \underline{t}) = C(i + \varphi_r, \underline{t})$ for some value r with $2 \leq r \leq s$, the increment array $\underline{t} = \langle t_0, \ldots, t_{s-1} \rangle$ is a solution for the system of identities

$$\sum_{k=0}^{r-2} t_k + \sum_{k=0}^{q-2} t_k = \sum_{k=0}^{r+q-3} t_k \qquad 2 \leq q \leq s - r + 1 \tag{A.6}$$

$$\sum_{k=0}^{r-2} t_k + \sum_{k=0}^{q-2} t_k = \sum_{k=0}^{s-1} t_k + \sum_{k=0}^{r+q-3-s} t_k \qquad s - r + 2 \leq q \leq s \tag{A.7}$$

We now show that any such solution must have $\pi(\underline{t}) \leq r - 1$.

Let $\underline{t}$ be an increment array that satisfies the system of identities given in (6) and (7) using r. In order to simplify the notation we use $\eta(n, s, r, q)$ to denote the relevant identity.

We analyse the general behaviours of

$$\begin{aligned} &\eta(n,s,r,r-p) \quad \text{for } 1 \le p \le r-2 \\ &\eta(n,s,r,r+p) \quad \text{for } 0 \le p \le s-r \end{aligned}$$

For $\eta(n,s,r,r-p)$ (1) and (2) yield the identities (post simplification and rearrangement)

$$\sum_{k=0}^{r-p-2} t_k = \sum_{k=r-1}^{2r-p-3} t_k \qquad 2r-s-1 \le p \le r-2 \tag{A.8}$$

$$\sum_{k=2r-p-2-s}^{r-p-2} t_k = \sum_{k=r-1}^{s-1} t_k \qquad 1 \le p \le 2r-s-2 \tag{A.9}$$

For $\eta(n,s,r,r+p)$ we obtain, in a similar manner,

$$\sum_{k=0}^{r-2} t_k = \sum_{k=r+p-1}^{2r+p-3} t_k \qquad 0 \le p \le s-2r+1 \tag{A.10}$$

$$\sum_{k=2r+p-2-s}^{r+p-2} t_k = \sum_{k=r-1}^{s-1} t_k \qquad s-2r+2 \le p \le s-r \tag{A.11}$$

Consider (8) and (9). As p *decreases* from its maximum value (for these cases) of $r-2$ to its minimum of 1 the system of identities follows the pattern,

$$\begin{aligned}
t_0 &= t_{r-1} && p=r-2, q=2 \\
t_0+t_1 &= t_{r-1}+t_r && p=r-3, q=3 \\
t_0+t_1+t_2 &= t_{r-1}+t_r+t_{r+1} && p=r-4, q=4 \\
&\cdots \\
t_0+t_1+t_2+\cdots+t_{r-k-2} &= t_{r-1}+t_r+t_{r+1}+\cdots+t_{2r-k-3} && p=r-k, q=k \\
&\cdots \\
t_0+t_1+t_2+\cdots+t_{r-3} &= t_{r-1}+t_r+t_{r+1}+\cdots+t_{2r-4} && p=1, q=r-1
\end{aligned}$$

We then have, via (10) and the first case of (11) (that is, $p=s-2r+2$, $q=s-r+2$), a sequence of $s-2r+3$ identities,

$$\begin{aligned}
t_0+t_1+\cdots+t_{r-2} &= t_{r-1}+t_r+t_{r+1}+\cdots+t_{2r-3} && q=r \\
t_0+t_1+\cdots+t_{r-2} &= t_r+t_{r+1}+t_{r+2}+\cdots+t_{2r-3}+t_{2r-2} && q=r+1 \\
t_0+t_1+\cdots+t_{r-2} &= t_{r+1}+t_{r+2}+\cdots+t_{2r-2}+t_{2r-1} && q=r+2 \\
&\cdots \\
t_0+t_1+\cdots+t_{r-2} &= t_{r+k-1}+t_{r+k}+\cdots+t_{2r+k-4}+t_{2r+k-3} && q=r+k \\
&\cdots \\
t_0+t_1+\cdots+t_{r-2} &= t_{s-r}+t_{s-r+1}+\cdots+t_{s-3}+t_{s-2} && q=s-r+1 \\
t_0+t_1+\cdots+t_{r-2} &= t_{s-r+1}+t_{s-r+2}+\cdots+t_{s-2}+t_{s-1} && q=s-r+2
\end{aligned}$$

Finally, from the remaining $r-2$ cases of (11), we obtain the identities,

$$\begin{array}{lcll}
t_1+t_2+t_3+\cdots+t_{r-2} & = & t_{s-r+2}+t_{s-r+3}+\cdots+t_{s-2}+t_{s-1} & q=s-r+3\\
t_2+t_3+\cdots+t_{r-2} & = & t_{s-r+3}+\cdots+t_{s-2}+t_{s-1} & q=s-r+4\\
\cdots & & & \\
t_k+\cdots+t_{r-2} & = & t_{s-r+k+1}+\cdots+t_{s-2}+t_{s-1} & q=s-r+k+2\\
\cdots & & & \\
t_{r-3}+t_{r-2} & = & t_{s-2}+t_{s-1} & q=s-1\\
t_{r-2} & = & t_{s-1} & q=s
\end{array}$$

It is clear, from these patterns, that the following equalities must hold in order for $\underline{t}$ to lead to $C(i,\underline{t})=C(i+\varphi_r,\underline{t})$ after n-correction.

From the first set of $r-2$ identities together with that for $q=r$ we obtain,

$$\begin{array}{lcl}
t_0 & = & t_{r-1}\\
t_1 & = & t_r\\
\cdots & & \\
t_{r-k-2} & = & t_{2r-k-3}\\
\cdots & & \\
t_{r-3} & = & t_{2r-4}\\
t_{r-2} & = & t_{2r-3}
\end{array} \tag{A.12}$$

For the identities which must hold when $r\leq q\leq s-r+2$, the left-hand of side of these is always $\sum_{k=0}^{r-2}t_k$ while the right-hand side of the identity for q has exactly two terms in common with the right-hand side of $q-1$ and two in common with $q+1$. In consequence we further deduce,

$$\begin{array}{lcl}
t_{r-1} & = & t_{2r-2}\\
t_r & = & t_{2r-1}\\
\cdots & & \\
t_{r+k-1} & = & t_{2r+k-2}\\
\cdots & & \\
t_{s-r} & = & t_{s-1}
\end{array} \tag{A.13}$$

Finally, in a similar manner to the identities in the first collection we deduce from the final set,

$$\begin{array}{lcl}
t_{s-1} & = & t_{r-2}\\
t_{s-2} & = & t_{r-3}\\
\cdots & & \\
t_{s-r+k+1} & = & t_k\\
\cdots & & \\
t_1 & = & t_{s-r+2}\\
t_0 & = & t_{s-r+1}
\end{array} \tag{A.14}$$

If we consider the effect of combining these and the consequences for $\langle t_0, \ldots, t_{s-1} \rangle$ we have the following.

For each j,

$$\begin{aligned} t_{(r-1)-j} &= t_{2(r-1)-j} \quad \text{from (12)} \\ t_{(r-1)+j} &= t_{2(r-1)+j} \quad \text{from (13)} \\ t_j &= t_{s-(r-1)+j} \quad \text{from (14)} \end{aligned}$$

Now suppose we write s as $s = b \times m$ where $b = gcd(s, r-1)$: note that since this allows $b = 1$ the value of b is always well defined for any s and $2 \leq r \leq s$.

In this case the conditions expressed in (12)–(14) indicate, from $s = b \times m$ and $(r-1) = b \times v$ that $\underline{t}$ conforms to the behaviour in Table A.1,

TABLE A.1: Values of $t_{i(r-1)+j}$ implied by (12)–(14), $s = m \times b$, $(r-1) = v \times b$

	0	1	$\cdots$	j	$\cdots$	$(b-2)$	$(b-1)$	$\cdots$	$(v-1)b$	$\cdots$	$vb-1$
0	t_0	t_1	$\cdots$	t_j	$\cdots$	$t_{(b-2)}$	$t_{(b-1)}$	$\cdots$	t_0	$\cdots$	$t_{(b-1)}$
1	t_0	t_1	$\cdots$	t_j	$\cdots$	$t_{(b-2)}$	$t_{(b-1)}$	$\cdots$	t_0	$\cdots$	$t_{(b-1)}$
$\cdots$	t_0	t_1	$\cdots$	t_j	$\cdots$	$t_{(b-2)}$	$t_{(b-1)}$	$\cdots$	t_0	$\cdots$	$t_{(b-1)}$
i	t_0	t_1	$\cdots$	t_j	$\cdots$	$t_{(b-2)}$	$t_{(b-1)}$	$\cdots$	t_0	$\cdots$	$t_{(b-1)}$
$\cdots$	t_0	t_1	$\cdots$	t_j	$\cdots$	$t_{(b-2)}$	$t_{(b-1)}$	$\cdots$	t_0	$\cdots$	$t_{(b-1)}$
$m-2$	t_0	t_1	$\cdots$	t_j	$\cdots$	$t_{(b-2)}$	$t_{(b-1)}$	$\cdots$	t_0	$\cdots$	$t_{(b-1)}$
$m-1$	t_0	t_1	$\cdots$	t_j	$\cdots$	$t_{(b-2)}$	$t_{(b-1)}$	$\cdots$	t_0	$\cdots$	$t_{(b-1)}$

That this is the behaviour imposed, follows from the conditions arising via (14) to the effect $t_j = t_{s-(r-1)+j}$, which is equivalent to the identity,

$$t_j = t_{bm-bv+j} = t_{b(m-v)+j}$$

indicating that values repeat in blocks of size b.

This, configuration, however, indicates that an increment array $\underline{t}$ for which $C(i, \underline{t}) = C(i + \varphi_r, \underline{t})$ after n-correction must have the form

$$\langle t_0, t_1, \ldots, t_{b-1}, t_0, t_1, \ldots, t_{b-1}, \ldots, t_0, t_1, \ldots, t_{b-1}, t_0, t_1, \ldots, t_{b-1} \rangle$$

That is to say, $\pi(\underline{t}) \leq b$. Now in order to complete the proof it remains to demonstrate that $\varphi_r = l \times (nb/s)$ for some choice of $1 \leq l \leq (n-i)s/nb$. Since $\pi(\underline{t}) \leq b$ and we have already shown that $\underline{t}$ repeats in blocks of length b, should $\pi(\underline{t}) < b$ then we must have b an exact multiple (> 1) of $\pi(\underline{t})$.

We have from our analysis above,

$$\begin{aligned}\varphi_r &= \sum_{k=0}^{r-2} t_k \ + \ r-1 \\ &= \frac{r-1}{b}\sum_{k=0}^{b-1} t_k \ + \ r-1 \\ &= \frac{r-1}{b}\left(\sum_{k=0}^{b-1} t_k \ + \ b\right) \\ &= \left(\frac{r-1}{b}\right)\varphi_{b+1}\end{aligned}$$

In addition, however,

$$\begin{aligned}n-s \ = \ \sum_{k=0}^{s-1} t_k \quad &= \quad \frac{s}{b}\sum_{k=0}^{b-1} t_k \\ &= \quad \frac{s}{b}(\varphi_{b+1}-b)\end{aligned}$$

So that

$$\varphi_{b+1} \ = \ \frac{b}{s}(n-s) \ + \ b \ = \ \frac{bn}{s}$$

Thereby giving φ_r as

$$\varphi_r \ = \ \left(\frac{r-1}{b}\right)\frac{bn}{s} \ = \ \frac{(r-1)n}{s} \ = \ l \times \left(\frac{n\pi(\underline{t})}{s}\right)$$

for some choice of l since $(r-1) = v \times b = l \times \pi(\underline{t})$.

This suffices to establish the result:

$$C(i,\underline{t}) = C(j,\underline{t}) \ \Rightarrow \ j = i + m \times \left(\frac{n\pi(\underline{t})}{s}\right)$$

□

Lemma A.2. *Let $C \subseteq \{1,2,\ldots,n\}$ with $|C| = s$. There is an increment array $\underline{t}$ and $i \in C$ such that $C = C(i,\underline{t})$.*

Proof. Let $C = \{x, x_0, x_1, \ldots, x_{s-2}\}$. Without loss of generality we may assume

$$x \ < \ x_0 \ < \ x_1 \ < \ \cdots \ < \ x_i \ < \ x_{i+1} \ < \ \cdots \ < \ x_{s-2}$$

It suffices to find $\underline{t} = \langle t_0, \ldots, t_{s-1} \rangle$ with $C(x, \underline{t}) = C$ and $\sum_{i=0}^{s-1} t_i = n - s$, i.e.

$$\begin{aligned} t_0 &= x_0 - (x+1) \\ &\cdots \\ t_k &= x_k - \left(\sum_{i=0}^{k-1} t_i + x + k + 1 \right) \\ &\cdots \\ t_{s-1} &= n - \sum_{i=0}^{s-2} t_i \end{aligned}$$

□

For increment arrays $\underline{t}$ and $\underline{u}$ we write $\underline{t} \approx \underline{u}$ if for some k with $0 \leq k \leq s-1$ we have

$$\langle u_0, u_1, \ldots, u_{s-1} \rangle \;=\; \langle t_k,\ t_{k+1}, \ldots, t_{s-1}, t_0, t_1, \ldots, t_{k-1} \rangle$$

It is easy to see that $\approx$ is an equivalence relation and that $\underline{t} \approx \underline{u} \Rightarrow \pi(\underline{t}) = \pi(\underline{u})$.

Lemma A.3. *If $\underline{t} \approx \underline{u}$ then*

$$\bigcup_{i=1}^{n} \{\, C(i, \underline{t}) \,\} \;=\; \bigcup_{i=1}^{n} \{\, C(i, \underline{u}) \,\}$$

Proof. Without loss of generality we may assume that $\underline{t} \approx \underline{u}$ is witnessed by the choice $k = s-1$, i.e.

$$\langle u_0, u_1, \ldots, u_{s-1} \rangle \;=\; \langle t_{s-1},\ t_0,\ t_1, \ldots, t_{s-3},\ t_{s-2} \rangle$$

Define φ_r for $1 \leq r \leq s+1$ as before and ψ_r for $1 \leq r \leq s+1$ via

$$\psi_r \;=\; \begin{cases} 0 & \text{if} \quad r = 1 \\ t_{s-1} \;+\; \sum_{k=0}^{r-3} t_k \;+\; r - 1 & \text{if} \quad 2 \leq r \leq s+1 \end{cases}$$

Comparing respective terms we see that for all $2 \leq k \leq s$

$$\psi_k \;=\; \varphi_k \;+\; (t_{s-1} \;-\; t_{k-2}) \tag{A.15}$$

We claim that this leads to the following,

$$C(i, \underline{t}) \;=\; \begin{cases} C(n - t_{s-1} + i - 1, \underline{u}) & \text{if} \quad 1 \leq i \leq t_{s-1} + 1 \\ C(i - t_{s-1} - 1, \underline{u}) & \text{if} \quad t_{s-1} + 2 \leq i \leq n \end{cases}$$

To see this, consider the case when $1 \leq i \leq t_{s-1} + 1$. We have,

$$C(i, \underline{t}) \;=\; \bigcup_{k=1}^{s} \{i + \varphi_k\}$$

which is claimed to be,

$$\begin{aligned} C(n - t_{s-1} + i - 1, \underline{u}) &= \bigcup_{k=1}^{s} \{n - t_{s-1} + i - 1 + \psi_k\} \\ &= \{n - t_{s-1} + i - 1\} \cup \bigcup_{k=2}^{s} \{n - t_{s-1} + i - 1 + \varphi_k + t_{s-1} - t_{k-2}\} \end{aligned}$$

Consider the terms

$$n - t_{s-1} + i - 1 + \varphi_k + t_{s-1} - t_{k-2}$$

For $2 \leq k \leq s$, from the fact that $\varphi_k = \sum_{j=0}^{k-2} t_j + k - 1$, these are equal to

$$n - t_{s-1} + i - 1 + \sum_{j=0}^{k-3} t_j \; + \; k - 1 \; + \; t_{s-1} \;\; = \;\; n + i + \varphi_{k-1}$$

In total we have, for $1 \leq i \leq t_{s-1} + 1$

$$\begin{aligned} C(i, \underline{t}) &= \bigcup_{k=1}^{s} \{i + \varphi_k\} \\ C(n - t_{s-1} + i - 1, \underline{u}) &= \{n - t_{s-1} + i - 1\} \; \cup \; \bigcup_{k=2}^{s} \{n + i + \varphi_{k-1}\} \end{aligned}$$

Noting that after n correction terms $n + i + \varphi_{k-1}$ become $i + \varphi_{k-1}$ all of which are elements of $C(i, \underline{t})$ the only terms unaccounted for are $\{n - t_{s-1} + i - 1\} \in C(n - t_{s-1} + i - 1, \underline{u})$ and $\{i + \varphi_s\} \in C(i, \underline{t})$. For these, however,

$$\begin{aligned} i + \varphi_s &= i + \sum_{j=0}^{s-2} t_j \; + \; s - 1 \\ &= i + (n - s - t_{s-1}) + s - 1 \\ &= i + n - t_{s-1} - 1 \end{aligned}$$

When $t_{s-1} + 2 \leq i \leq n$, it is claimed that

$$C(i, \underline{t}) \;\; = \;\; \bigcup_{k=1}^{s} \; \{i + \varphi_k\}$$

corresponded to

$$\begin{aligned} C(i - t_{s-1} - 1, \underline{u}) &= \bigcup_{k=1}^{s} \{i - t_{s-1} - 1 + \psi_k\} \\ &= \{i - t_{s-1} - 1\} \cup \bigcup_{k=2}^{s} \{i - t_{s-1} - 1 + \varphi_k + t_{s-1} - t_{k-2}\} \end{aligned}$$

Inspecting the terms for $2 \leq k \leq s$

$$i - t_{s-1} - 1 + \varphi_k + t_{s-1} - t_{k-2}$$

These, again, simplify to $i + \varphi_{k-1}$, so that

$$\begin{aligned} C(i, \underline{t}) &= \bigcup_{k=1}^{s} \{i + \varphi_k\} \\ C(i - t_{s-1} - 1, \underline{u}) &= \{i - t_{s-1} - 1\} \cup \bigcup_{k=2}^{s} \{i + \varphi_{k-1}\} \end{aligned}$$

When $1 \leq k \leq s-1$, the term $i + \varphi_k$ appears in both $C(i, \underline{t})$ and $C(i - t_{s-1} - 1, \underline{u})$, For the terms $i + \varphi_s \in C(i, \underline{t})$ and $i - t_{s-1} - 1 \in C(i - t_{s-1} - 1, \underline{u})$ we have already seen that $i + \varphi_s = i + n - t_{s-1} - 1$ which after n correction is $i - t_{s-1} - 1$ as required.

This establishes the property claimed: if $\underline{t}$ and $\underline{u}$ belong to the same equivalence class of $\approx$ then

$$\bigcup_{i=1}^{n} \{C(i, \underline{t})\} = \bigcup_{i=1}^{n} \{C(i, \underline{u})\}$$

□

Lemma A.3 shows that increment arrays belonging to the same equivalence class of $\approx$ generate exactly the same set of coalitions of size s from $\{1, \ldots, n\}$. Our next two results establish that this is the *only* way in which two distinct sequences can produce the same coalition.

Lemma A.4. *Let $C = C(x_i, \underline{t})$ and*

$$C = \{x_1,\ x_2, \ldots, x_i, \ldots, x_s\}$$

with $x_i < x_{i+1}$ for all $1 \leq i < s$. There is an increment array, $\underline{u}$, for which $\underline{t} \approx \underline{u}$ and $C(x_1, \underline{u})$ generates C in strictly increasing ordering of x_i, i.e.

$$x_i \ \in \ \left\{ x_1 \ + \ \sum_{k=0}^{i-2} u_i \ + \ i - 1,\ x_1 \ + \ \sum_{k=0}^{i-2} u_i \ + \ i - 1 - n \right\} \quad \forall\, 2 \leq i \leq s$$

Proof. Given $\underline{t}$, suppose

$$C(x_i, \underline{t}) = \{x_1, x_2, \ldots, x_i, \ldots, x_s\}$$

First observe that the terms

$$x_i \ + \ \sum_{k=0}^{r-2} t_k \ + \ r - 1 \ = \ x_i \ + \ \varphi_r$$

are strictly increasing. It follows that if $x_i \neq x_1$ there must be a *unique* index, p, for which

$$x_i \ + \ \varphi_r \quad \text{is} \quad \begin{cases} \leq \ n & \text{if } r < p \\ > n & \text{if } r \geq p \end{cases}$$

In consequence, $x_1 \ = \ x_i + \varphi_p - n$ for otherwise we cannot have $x_1 \in C(x_i, \underline{t})$. More generally, however, it must hold that,

$$\begin{aligned} x_k &= x_i + \varphi_{p+k-1} - n \quad \forall\, 1 \ \leq \ k \ \leq \ s - p + 1 \\ x_k &= x_i + \varphi_{p-(s-k)-1} \quad \forall\, s - p + 2 \ \leq k \ \leq s \end{aligned}$$

This, however, corresponds to the behaviour of the increment array, $\underline{u}$, whose definition is

$$\underline{u} \ = \ \langle t_{p+1}, \ t_{p+2}, \ \ldots, \ t_{p+k}, \ \ldots, \ t_s, \ t_0, \ \ldots, \ t_p \rangle$$

Clearly $\underline{u} \approx \underline{t}$ and $C(x_1, \underline{u}) = C(x_i, \underline{t})$ as claimed. □

As an easy consequence of Lemma A.4 we obtain,

Lemma A.5. *Let $\underline{t}$ and $\underline{u}$ be increment arrays for which $\underline{t} \not\approx \underline{u}$. In such cases,*

$$\bigcup_{i=1}^{n} \{C(i, \underline{t})\} \bigcap \bigcup_{i=1}^{n} \{C(i, \underline{u})\} \ = \ \emptyset$$

Proof. Suppose the contrary and that $C \ = \ \{x_1, \ldots, x_s\}$ can be generated by $C(x_i, \underline{t})$ and $C(x_j, \underline{u})$ for choices of $\underline{t}$ and $\underline{u}$ belonging to different equivalence classes of $\approx$. As a consequence of Lemma A.4 we know that there are increment arrays, $\underline{t}'$ and $\underline{u}'$ for which,

$$\begin{aligned} &\underline{t} \approx \underline{t}' \\ &\underline{u} \approx \underline{u}' \\ &C(x_1, \underline{t}') \ = \ C(x_i, \underline{t}) \ = \ C(x_j, \underline{u}) \ = \ C(x_1, \underline{u}') \end{aligned}$$

Furthermore, $C(x_1, \underline{t}')$ and $C(x_1, \underline{u}')$ produce the elements of C in increasing ordering of $x_i \in C$. This, however, is only possible if

$$x_i \ = \ x_1 \ + \ \sum_{k=0}^{i-2} t'_k \ + \ i - 1 \ = \ x_1 \ + \ \sum_{k=0}^{i-2} u'_k \ + \ i - 1$$

that is, $t'_i = u'_i$ for each $0 \leq i \leq s - 1$. This, however, implies that $\underline{t} \approx \underline{u}$ in contradiction to our starting premise. □

Theorem A.6. *For increment arrays, $\underline{t}$ and $\underline{u}$,*

$$\bigcup_{i=1}^{n} \{C(i, \underline{t})\} \bigcap \bigcup_{i=1}^{n} \{C(i, \underline{u})\} \ \neq \ \emptyset \quad \Leftrightarrow \quad \underline{t} \approx \underline{u} \tag{A.16}$$

$$\bigcup_{i=1}^{n} \{C(i, \underline{t})\} \ = \ \bigcup_{i=1}^{n} \{C(i, \underline{u})\} \quad \Leftrightarrow \quad \underline{t} \approx \underline{u} \tag{A.17}$$

Proof. That increment arrays belonging to the same equivalence class of $\approx$ generate identical sets of coalitions of size s over all agents (and hence such have a non-empty intersection) follows directly from Lemma A.3. This establishes the $\Leftarrow$ implication for (17) and hence (16). That this is the only way that two increment arrays can generate a coalition in common, i.e. that should $\underline{t}$ and $\underline{u}$ produce the same subset of $\{1, 2, \ldots, n\}$ then $\underline{t}$ and $\underline{u}$ belong to the same equivalence class of $\approx$ is immediate from Lemma A.4 and Lemma A.5. We thus have the $\Rightarrow$ implications of (16) and (17) thereby completing the theorem proof. □

Bibliography

[1] S. Airiau and S. Sen, *Distributed computation of kernel-stable coalition payoff distributions*, in Proceedings of the 1st Workshop on Cooperative Games in Multiagent Systems (CoopMAS), 2010, pp. 13–20.

[2] ———, *On the stability of an optimal coalition structure*, in Proceedings of the 19th European Conference on Artificial Intelligence (ECAI), 2010, pp. 203–308.

[3] K. Allan, *Felicity conditions on speech acts*, Encyclopedia of Language and Linguistics, Oxford: Pergamon Press, 1994.

[4] L. Amgoud, *An argumentation-based model for reasoning about coalition structures*, in Proceedings of the 2nd International Workshop on Argumentation in Multi-Agent Systems (ArgMAS), 2005, pp. 217–228.

[5] S. Arib and S. Aknine, *Preferences and constraints for agent coalition formation*, in Proceedings of the IEEE/WIC/ACM International Conference on Intelligent Agent Technology (IAT), 2013, pp. 130–137.

[6] Aristotle, *Topics*, Clarendon Press, Oxford, UK, 1997, Translated by R. Smith.

[7] T. Arnold and U. Schwalbe, *Dynamic coalition formation and the core*, Journal of Economic Behaviour and Organization **49** (2002), 363–380.

[8] K. Atkinson, *What should we do? Computational representation of persuasive argument in practical reasoning*, Ph.D. thesis, Department of Computer Science, University of Liverpool, 2005.

[9] K. Atkinson and T. Bench-Capon, *Action-based alternating transitions systems for arguments about action*, in Proceedings of the 22nd Conference on Artificial Intelligence (AAAI), 2007, pp. 24–29.

[10] ———, *Practical reasoning as presumptive argumentation using action based alternating transition systems*, Artificial Intelligence **171** (2007), 855–874.

[11] K. Atkinson, T. Bench-Capon, and P. E. Dunne, *Uniform argumentation frameworks*, in Proceedings of the 4th International Conference on Computational Models of Arguments (COMMA), 2012, pp. 165–176.

[12] K. Atkinson, T. Bench-Capon, and P. McBurney, *A dialogue game protocol for multi-agent argument over proposals for action*, Autonomous Agents and Multi-Agent Systems **11** (2005), 153–171.

[13] K. Atkinson, T. J. M. Bench-Capon, and P. McBurney, *Computational representation of practical argument*, Synthese **152** (2006), 157–206.

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

[15] R. J. Aumann and M. Maschler, *The bargaining set for cooperative games*, Advances in Game Theory (M. Dresher, L. S. Shapley, and A. W. Tucker, eds.), Princeton University Press, 1964, pp. 443–447.

[16] R. J. Aumann and M. Maschler, *Game-theoretic analysis of a bankruptcy problem from the Talmud*, Journal of Economic Theory **36** (1985), 195–213.

[17] J. L. Austin, *How to do things with words*, Oxford University Press, 1962.

[18] Y. Bachrach, E. Elkind, R. Meir, D. Pasechnik, M. Zuckerman, J. Rothe, and J. S. Rosenschein, *The cost of stability in coalitional games*, in Proceedings of the 2nd International Symposium on Algorithmic Game Theory (SAGT), 2009, pp. 122–134.

[19] T. Bench-Capon, *Persuasion in practical argument using value based argumentation frameworks*, Journal of Logic and Computation **13** (2003), 429–48.

[20] F. Bistaffa, Jesus Cerquides, and S. D. Ramchurn, *Anytime coalition structure generation on synergy graphs*, in Proceedings of the 13th International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS), 2014, pp. 13–20.

[21] E. Black and K. Atkinson, *Dialogues that account for different perspectives in collaborative argumentation*, in Proceedings of the 8th International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS), 2009, pp. 867–874.

[22] E. Black and A. Hunter, *An inquiry dialogue system*, Autonomous Agents and Multi-Agent Systems **19** (2009), 173–209.

[23] B. Blankenburg, R. K. Dash, S. D. Ramchurn, M. Klusch, and N. R. Jennings, *Trusted kernel-based coalition formation*, in Proceedings of the 4th International Conference on Autonomous Agents and Multiagent Systems (AAMAS), 2005, pp. 989–996.

[24] G. Boella, D. M. Gabbay, A. Perotti, and S. Villata, *Coalition formation via negotiation in multiagent systems with voluntary attacks*, in Proceedings of the 22nd Benelux Conference on Artificial Intelligence (BNAIC), 2010, pp. 25–32.

[25] G. Boella, L. v. d. Torre, and S. Villata, *Analyzing cooperation in iterative social network design*, Universal Computer Science **15** (2009), 2676–2700.

[26] A. Bogomolnaia and M. O. Jackson, *The stability of hedonic coalition structures*, Games and Economic Behaviour **28** (2002), 201–230.

[27] S. Branzei and K. Larson, *Coalitional affinity games and the stability gap*, in Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI), 2009, pp. 79–84.

[28] N. Bulling and J. Dix, *Modelling and verifying coalitions using argumentation and ATL*, Inteligencia Artificial **14** (2010), 45–73.

[29] C. Cayrol and M. C. Lagasquie-Schiex, *Coalitions of arguments: A tool for handling bipolar argumentation frameworks*, Intelligent Systems **25** (2010), 83–109.

[30] J. C. Cesco, *A convergent transfer scheme to the core of a TU-game*, Revista de Matemticas Aplicadas **19** (1998), 23–35.

[31] ———, *Fundamental cycles of pre-imputations in non-balanced TU-games*, International Journal of Game Theory **32** (2003), 211–222.

[32] G. Chalkiadakis, *A bayesian approach to multi-agent reinforcement learning and coalition formation under uncertainty*, Ph.D. thesis, Department of Computer Science, University of Toronto, 2007.

[33] G. Chalkiadakis and C. Boutilier, *Bayesian reinforcement learning for coalition formation under uncertainty*, in Proceedings of the 3th International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS), 2004, pp. 1090–1097.

[34] G. Chalkiadakis, E. Elkind, and N. R. Jennings, *Simple coalitional games with beliefs*, in Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI), 2009, pp. 85–90.

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

[36] G. Chalkiadakis, E. Elkind, and M. Wooldridge, *Computational aspects of cooperative game theory*, Morgan & Claypool Publishers, 2011.

[37] G. Chalkiadakis, E. Markakis, and C. Boutilier, *Coalition formation under uncertainty: bargaining equilibria and the bayesian core stability concept*, in Proceedings of the 6th International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS), 2007, pp. 64–81.

[38] A. Chapman, R. A. Micillo, R. Kota, and N. R. Jennings, *Decentralised dynamic task allocation using overlapping potential games*, The Computer Journal **53** (2010), 1462–1477.

[39] K. Chatterjee, B. Dutta, and K. Sengupta, *A noncooperative theory of coalitional bargaining*, Review of Economic Studies **60** (1993), 463–477.

[40] B. Cheng, G. Zeng, and A. Jie, *Adding branching temporal dimension to qualitative coalitional games with preferences*, Journal of Computers **5** (2010), 749–757.

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

[42] V. D. Dang, R. K. Dash, A. Rogers, and N. R. Jennings, *Overlapping coalition formation for efficient data fusion in multi-sensor networks*, in Proceedings of the 21st Conference on Artificial Intelligence (AAAI), 2006, pp. 635–640.

[43] V. D. Dang and N. R. Jennings, *Generating coalition structures with finite bound from the optimal guarantees*, in Proceedings of the 3th International Conference on Autonomous Agents and Multiagent Systems (AAMAS), 2004, pp. 564–571.

[44] S. Dasgupta, C. H. Papadimitriou, and U. V. Vazirani, *Algorithms*, McGraw-Hill Higher Education, 2006.

[45] M. Davis and M. Maschler, *The kernel of a cooperative game*, Naval Research Logistics Quarterly **12** (1965).

[46] F. Dignum, B. Dunin-Keplicz, and R. Verbrugge, *Agent theory for team formation by dialogue*, in Proceedings of the 7th International Workshop on Agent Theories, Architectures and Languages (ATAL), 2000, pp. 141–156.

[47] M. Dimand and R. W. Dimand, *The history of game theory, volume 1: from the beginnings to 1945*, Routledge, 1996.

[48] Y. Dimopoulos, P. Moraitis, and L. Amgoud, *Theoretical and computational properties of preference-based argumentation*, in Proceedings of the 18th European Conference on Artificial Intelligence, 2008, pp. 463–467.

[49] T. S. H. Driessen, *Cooperative games, solutions and applications*, Springer Netherlands, 1988.

[50] P. M. Dung, *On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games*, Artificial Intelligence **77** (1995), 321–357.

[51] P. E. Dunne, W. v. d. Hoek, S. Kraus, and M. Wooldridge, *Cooperative boolean games*, in Proceedings of the 7th International Conference on Autonomous Agents and Multiagent Systems (AAMAS), 2008, pp. 1015–1022.

[52] P. E. Dunne and M. Wooldridge, *Preferences in qualitative coalition games*, in Proceedings of the 6th Workshop in Game Theoretic and Decision Theoretic Agents (GTDT), 2004, pp. 29–38.

[53] Paul E. Dunne, *Indexing (n, s) combinations*, Tech. Report ULCS-15-001, University of Liverpool, 2015.

[54] B. Dutta and R. Vohra, *Incomplete information, credibility and the core*, Mathematical Social Sciences **50** (2005), 148–165.

[55] E. Elkind, G. Chalkiadakis, and N. R. Jennings, *Coalition structures in weighted voting games*, in Proceedings of the 18th European Conference on Artificial Intelligence (ECAI), 2008, pp. 393–397.

[56] R. Evans, *Coalitional bargaining with competition to make offers*, Games and Economic Behaviour **19** (1997), 211–220.

[57] A. J. García and G. R. Simari, *Defeasible logic programming an argumentative approach*, Theory and Practice of Logic Programming **4** (2004), 95–138.

[58] D. Gillies, *Some theorems on n-person games*, Ph.D. thesis, Princeton University, 1953.

[59] H. J. Goradia and J. M. Vidal, *A distributed algorithm for finding nucleolus-stable payoff divisions*, in Proceedings of the IEEE/WIC/ACM International Conference on Intelligent Agent Technology (IAT), 2007, pp. 395–398.

[60] T. Gordon, H. Prakken, and D. Walton, *The carneades model of argument and burden of proof*, Artificial Intelligence **171 (10-15)** (2007), 875–896.

[61] C. L. Hamblin, *Fallacies*, Methuen, London, UK, 1970.

[62] H. Hattori, T. Ito, T. Ozono, and T. Shintani, *An approach to coalition formation using argumentation-based negotiation in multiagent systems*, in Proceedings of the 14th International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems (IEA/AIE), 2001, pp. 687–696.

[63] B. Horling and V. Lesser, *A survey of multi-agent organizational paradigms*, Knowledge Engineering Review **19** (2004), no. 4, 281–316.

[64] S. Ieong and Y. Shoham, *Marginal contribution nets: a compact representation scheme for coalitional games*, in Proceedings of the ACM Conference on Electronic Commerce, 2005, pp. 193–202.

[65] ———, *Bayesian coalitional games*, in Proceedings of the 23rd Conference on Artificial Intelligence (AAAI), 2007, pp. 95–100.

[66] A Iwasaki, S. Ueda, and M. Yokoo, *Finding the core for coalition structure utilizing dual solution*, in Proceedings of the IEEE/WIC/ACM International Conference on Intelligent Agent Technology (IAT), 2013, pp. 114–121.

[67] E. Lehrer, *Allocation processes in cooperative games*, International Journal of Game Theory **31** (2003), 341–351.

[68] I. A. Letia and A. Groza, *Planning with argumentation schemes in online dispute resolution*, in Proceedings of the IEEE International Conference on Intelligent Computer Communication and Processing, 2007, pp. 17–24.

[69] W. F. Lucas, *A game with no solution*, Bulletin of the American Mathematical Society **7** (1968), 237–239.

[70] P. McBurney, D. Hitchcock, and S. Parsons, *The eightfold way of deliberation dialogue*, Intelligent Systems **22** (2007), 95–132.

[71] P. McBurney and S. Parsons, *Dialogue games for agent argumentation*, Argumentation in Artificial Intelligence, 2009, pp. 261–280.

[72] Peter McBurney and Simon Parsons, *Games that agents play: A formal framework for dialogues between autonomous agents*, Journal of Logic, Language and Information **11** (2002), no. 3, 315–334.

[73] T. Michalak, J. Sroka, T. Rahwan, M. Wooldridge, P. McBurney, and N. R. Jennings, *A distributed algorithm for anytime coalition structure generation*, in Proceedings of the 9th International Conference on Autonomous Agents and Multiagent System (AAMAS), 2010, pp. 1007–1014.

[74] P. J. Modi, W. M. Shen, M. Tambe, and M. Yokoo, *Adopt: Asynchronous distributed constraint optimization with quality guarantees*, Artificial Intelligence **161** (2006), 149–180.

[75] R. A. Mollin, *Fundamental number theory with applications, second edition*, Chapman and Hall/CRC, 2008.

[76] R. Myerson, *Incentive-compatibility and the bargaining problem*, Econometrica **47** (1979), 61–73.

[77] ———, *Cooperative games with incomplete information*, International Journal of Game Theory **13** (1984), 69–86.

[78] ———, *Virtual utility and the core for games with incomplete information*, Journal of Economic Theory **136** (2007), 260–285.

[79] J. v. Neumann and O. Morgenstern, *The theory of games and economic behavior*, Princeton University Press, 1944.

[80] N. Nisan, *Introduction to mechanism design (for computer scientists)*, Algorithmic Game Theory, Cambridge University Press, 2007, pp. 209–242.

[81] N. Ohta, V. Conitzer, R. Ichimura, Y. Sakurai, A. Iwasaki, and M. Yokoo, *Coalition structure generation utilizing compact characteristic function representations*, in Proceedings of the 15th International Conference on Principles and Practice of Constraint Programming (CP), 2009, pp. 623–638.

[82] A. Okada, *A noncooperative coalitional bargaining game with random proposers*, Games and Economic Behaviour **16** (1996), 97–108.

[83] M. J. Osbourne and A. Rubinstein, *A course in game theory*, MIT Press, 1994.

[84] W. Ouerdane, N. Maudet, and A. Tsoukias, *Argument schemes and critical questions for decision aiding process*, in Proceedings of the 2nd International Conference on Computational Models of Argument, 2008, pp. 285–296.

[85] H. Prakken, *Formal systems for persuasion dialogue*, Knowledge Engineering Review **21** (2006), 163–188.

[86] I. Rahwan, S. D. Ramchurn, N. R. Jennings, P. McBurney, S. Parsons, and L. Sonenberg, *Argumentation-based negotiation*, Knowledge Engineering Review **18** (2003), 343–375.

[87] I. Rahwan and G. R. Simari, *Argumentation in artificial intelligence*, Springer, 2009.

[88] T. Rahwan, *Algorithms for coalition formation in multi-agent systems*, Ph.D. thesis, School of Electronics and Computer Science, University of Southampton, 2007.

[89] T. Rahwan and N. R. Jennings, *Distributing coalition value calculations among cooperating agents*, in Proceedings of the 20th conference on artificial intelligence (AAAI), 2005, pp. 152–157.

[90] ———, *An algorithm for distributing coalition value calculations among cooperating agents*, Artificial Intelligence **171** (2007), 535–567.

[91] ———, *An improved dynamic programming algorithm for coalition structure generation*, In Proceedings of the 7th International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS) (2008), 1417–1420.

[92] T. Rahwan, T. Michalak, E. Elkind, P. Faliszewski, J. Sroka, M. Wooldridge, and N. R. Jennings, *Constrained coalition formation*, in Proceedings of the 25th Conference on Artificial Intelligence (AAAI), 2011, pp. 719–725.

[93] T. Rahwan, T. P. Michalak, and N. R. Jennings, *A hybrid algorithm for coalition structure generation*, in Proceedings of the 26th Conference on Artificial Intelligence (AAAI), 2012, pp. 1443–1449.

[94] T. Rahwan, S. D. Ramchurn, A. Giovannucci, and N. R. Jenning, *An anytime algorithm for optimal coalition structure generation*, Journal of Artificial Intelligence Research **34** (2009), 521–567.

[95] S. D. Ramchurn, E. Gerding, N. R. Jennings, and H. Jun, *Practical distributed coalition formation via heuristic negotiation in social networks*, in Proceedings of the 5th International Workshop on Optimisation in Multi-Agent Systems (OPTMAS), 2012.

[96] D. Ray, *A game-theoretic perspective on coalition formation*, Oxford University Press, 2007.

[97] C. Reed and D. Walton, *Argumentation schemes in dialogue*, in Proceedings of Dissensus and the Search for Common Ground (OSSA), 2010, pp. 1–11.

[98] L. Riley, *A persuasive dialogue game for coalition formation*, in Proceedings of the inaugural Imperial College Student Workshop (ICCSW), 2011, pp. 66–73.

[99] L. Riley, K. Atkinson, P. Dunne, and T. R. Payne, *Distributing coalition value calculations to coalition members*, in Proceedings of the 29th Conference on Artificial Intelligence (AAAI), 2015.

[100] L. Riley, K. Atkinson, and T. R. Payne, *A dialogue game for coalition structure generation with self-interested agents*, in Proceedings of the 4th International Conference of Computational Models of Argument (COMMA), 2012, pp. 311–322.

[101] L. Riley, K. Atkinson, T. R. Payne, and E. Black, *An implemented dialogue system for inquiry and persuasion*, in Proceedings of the 1st International Workshop on the Theory and Applications of Formal Argumentation (TAFA), 2011, pp. 67–84.

[102] L. Riley, T. R. Payne, T. Bench-Capon, and K. Atkinson, *Distributing coalition value calculations to self-interested agents*, in Proceedings of the 13th International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS), 2014, pp. 1431–1432.

[103] A. Rubinstein, *Perfect equilibrium in a bargaining model*, Econometrica **50** (1982), 97–110.

[104] T. W. Sandholm, K. S. Larson, M. Andersson, O. Shehory, and F. Tohme, *Coalition structure generation with worst case guarantees*, Artificial Intelligence **111** (1999), 209–238.

[105] T. W. Sandholm and V. Lesser, *Issues in automated negotiation and electronic commerce: extending the contract net protocol*, in Proceedings of the 1st International Conference on Multiagent Systems (ICMAS), 1995, pp. 328–335.

[106] T. W. Sandholm and V. R. Lesser, *Coalitions among computationally bounded agents*, Artificial Intelligence **94** (1997), 99–137.

[107] P. Scerri, D. Pynadath, and M. Tambe, *Adjustable autonomy for the real world*, Agent Autonomy **7** (2003), 211–241.

[108] D. Schmeidler, *The nucleolus of a characteristic function game*, SIAM Journal of applied mathematics **17** (1969), 1163–1170.

[109] A. S. Schulz and N. A. Uhan, *Sharing supermodular costs*, Operations Research **58** (2010), 1051–1056.

[110] J. R. Searle, *Speech acts: An essay in the philosophy of language*, Cambridge University Press, 1969.

[111] S. Sen and P. Dutta, *Searching for optimal coalition structures*, in Proceedings of the 4th International Conference on Multiagent Systems (ICMAS), 2000, pp. 287–292.

[112] T. Service, *Coalition structure generation in characteristic function games*, Ph.D. thesis, Vanderbilt University, 2012.

[113] L. S Shapley, *A value for n-person games*, Contributions to the Theory of Games, Volume II (Annals of Mathematics Studies) **28** (1953), 307–317.

[114] L. S. Shapley and M. Shubik, *Quasi-cores in a monetary economy with non-convex preferences*, Econometrica **34** (1966), 805–827.

[115] O. Shehory and S. Kraus, *Task allocation via coalition formation among autonomous-agents*, in Proceedings of the 14th International Joint Conference on Artificial Intelligence (IJCAI), 1995, pp. 655–661.

[116] ———, *Formation of overlapping coalitions for precedence-order task-execution among autonomous agents*, in Proceedings of the 2nd International Conference on Multiagent Systems (ICMAS), 1996, pp. 330–337.

[117] ———, *Methods for task allocation via agent coalition formation*, Artificial Intelligence **101** (1998), 165–200.

[118] ———, *Feasible formation of stable coalitions among autonomous agents in non-super-additive environments*, Computational Intelligence **15** (1999), 218–251.

[119] Y. Shoham and K. Leyton-Brown, *Multiagent systems: Algorithmic, game-theoretic and logical foundations*, Cambridge University Press, 2007.

[120] R. G. Smith, *The contract net protocol: High-level communication and control in a distributed problem solver,*, IEEE Transactions on Computers **29** (1980), 1104–113.

[121] R. E. Sterns, *Convergent transfer schemes for n-person games*, Transactions of American Mathematical Society **134** (1968), 449–459.

[122] J. Suijs, P. Borm, A. D. Waegenaere, and S. Tijs, *Cooperative games with stochastic payoffs*, European Journal of Operational Research **113** (1999), 193–205.

[123] A. Toniolo, T. J. Norman, and K. Sycara, *Argumentation schemes for collaborative planning*, in Proceedings of the 14th International Conference on the Principles and Practice of Multi-Agent Systems (PRIMA), 2011, pp. 323–325.

[124] S. Ueda, A. Iwasaki, M. Yokoo, M. C. Silaghi, K. Hirayama, and T. Matsui, *Coalition structure generation based on distributed constraint optimization*, Proceedings of the 24th Conference on Artificial Intelligence (AAAI), 2010, pp. 197–203.

[125] T. L. v. d. Weide, F. Dignum, J. J. Ch. Meyer, H. Prakken, and G. A. W. Vreeswijk, *Practical reasoning using values*, in Proceedings of the 6th International Workshop on Argumentation in Multi-agent Systems (ArgMAS), 2009, pp. 225–240.

[126] M. Vinyals, F. Bistaffa, A. Farinelli, and A. Rogers, *Coalitional energy purchasing in the smart grid*, in Proceedings of the IEEE International Energy Conference & Exhibition (ENERGYCON), 2012, pp. 848–853.

[127] T. Voice, S. D. Ramchurn, and N. R. Jennings, *On coalition formation with sparse synergies*, in Proceedings of the 11th International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS), 2012, pp. 223–230.

[128] D. N. Walton, *Argumentation schemes for presumptive reasoning*, Lawrence Erlbaum Associates, Mahwah, NJ, USA, 1996.

[129] D. N. Walton and E. C. W. Krabbe, *Commitment in dialogue: Basic concepts of interpersonal reasoning*, State University of New York Press, Albany NY, 1995.

[130] S. Wells and C. Reed, *Knowing when to bargain - the roles of negotiation and persuasion in dialogue*, in Proceedings of the 1st International Conference on Computational models of Argument (COMMA), 2006, pp. 235–246.

[131] R. Wilson, *Information, efficiency and the core of an economy*, Econometrica **46** (1978), 807–816.

[132] M. Wooldridge and P. E. Dunne, *On the computational complexity of qualitative coalition games*, Artificial Intelligence **158** (2004), 27–73.

[133] _____, *On the computational complexity of coalitional resource games*, Artificial Intelligence **170** (2006), 853–871.

[134] M. Wooldridge and N. R. Jennings, *Intelligent agents: Theory and practice*, Knowledge Engineering Review **10** (1995), 115–152.

[135] M. Wooldridge and W. v. d. Hoek, *On obligations and normative ability: Towards a logical analysis of the social contract*, Journal of Applied Logic **3** (2005), 396–420.

[136] Michael Wooldridge, *An introduction to multiagent systems second edition*, John Wiley & Sons, 2009.

[137] L. S.-Y. Wu, *A dynamic theory for the class of games with nonempty cores*, SIAM Journal on Applied Mathematics **32** (1977), 328–338.

[138] D. Y. Yeh, *A dynamic programming approach to the complete set partitioning problem*, BIT Numerical Mathematics **26** (1986), 467–474.