Chapter 2

Background

2.1 Introduction to Optimal Coalition Structure

Optimal coalition structure has been the core of coalition formation research in the past decades. Coalition formation is the process that leads to cooperation among agents within a multi-agent system. Coalition formation was first studied in cooperative game theory by Von Neumann and Morgensterncite89. With the growth and maturity of multi-agent system research, coalition formation has gained more attention from researchers and has been regarded as an important area in multi-agent systems. This chapter explores coalition formation and optimal coalition structure below.

2.1.1 Coalition Formation

Coalition formation is the cooperative, versus the non-cooperative, side of game theory. Game theory studies decision making by multiple decision making units, which we shall refer to as **agents** henceforth, and whose decisions are inter-related. The goal of game theory is to find stable states in which none of the agents will want to change their decisions. Such a stable state is called an *equilibrium*. A principle that describes reasons which lead agents to an equilibrium is a *solution concept*. Game theory can be divided into *non-cooperative* games and *cooperative* games. In the non-cooperative game environment, agents are not allowed to collaborate or communicate with each other. A real world example is the anti-trust law that prohibits agents from forming cartels. Research in non-cooperative game theory seeks to identify strategies which are the best possible response to other agents' strategies A well known solution concept in this area is Nash Equilibrium [10]

in which none of the agents can benefit from changing their strategies.

In contrast to non-cooperative game theory, coalition formation allows for agents to communicate that leads them to cooperation [3] from which they can benefit more individually. Agents communicate in order to negotiate with regard to whom they can cooperate and how the joint benefits will be distributed among them. When several agents make a binding agreement to cooperate, we say a *coalition* has been formed. Mathematically, given set N of n agents, a coalition is a non-empty subset S of N, $S \subseteq N, S \neq \emptyset$. The set N itself is called the *grand coalition* while a coalition of one agent is called *singleton coalition.* Let $\mathbf{S}$ be the set of all coalitions, whose size of S is $2^n - 1$.

Given a set of 3 agents,

$$N = \{1, 2, 3\},$$

all the 7 coalitions are

$$\{1\}, \{2\}, \{3\}, \{1, 2\}, \{1, 3\}, \{2, 3\} \text{ and } \{1, 2, 3\}.$$

As in set theory, the *cardinality*, $|S|$, of S is the size of (the number of agents in) S. Once agents have formed coalitions, they can be viewed as if they have divided themselves into a mutually exclusive and exhaustive partitions. We define a *coalition structure*, CS, as a partition of N. A CS can be described by $CS = \{S_1, S_2, \ldots, S_m\}$. The set of all CS is denoted by $\mathcal{CS}$. For example, given $N = \{1, 2, 3\}$, all CS in $\mathcal{S}$ are

$$\{\{1\}, \{2\}, \{3\}\}, \{\{1, 2\}, \{3\}\}, \{\{1, 3\}, \{2\}\}, \{\{2, 3\}, \{1\}\}, \{\{1\}, \{2\}, \{3\}\}.$$

Mathematically, a CS has to satisfy three conditions [3]:

1) $S_j \neq \emptyset, j = 1, 2, \ldots, m,$
2) $S_i \cap S_j = \emptyset$ for all $i \neq j$, and
3) $\bigcup S_j = N.$

The joint benefit of a coalition is called the *coalition value*, which is a numeric value that usually represents the utility which accrues from their cooperation. It is quite common in coalition formation that the coalition value is money value, e.g., dollars. coalition formation assumes that there is a *characteristic function* [3], V that assigns a real number to each S, $V : 2^n \rightarrow \mathbb{R}$. We shall denote the coalition value of S with V_S. Hence, a cooperative n-person game in characteristic function form is defined by the pair $(N; V)$ [3]. The portion of V_S given to agent i is the *payoff*, U_i, of the agent for which the agent plays the game. The collection of payoffs to each agent is the payoff vector, $U = (U_1, U_2, \ldots, U_n)$, which specifies the payoff for each respective

agent. Putting together a coalition structure and a payoff vector is the *payoff configuration* [3], $(U; CS)$, which describes a possible outcome of the game. For example,the payoff configuration $(5, 10, 5; \{1, 3\}, \{2\})$ means agents 1 and 3 have formed a coalition and receive payoff for 5 dollars each while agent 2 remains a singleton coalition and receives 10 dollars payoff on its own.

Games Environments

Classical research in coalition formation considers games within the *superadditive* [3] environment in which the value of a coalition is at least as much as the sum of the values of each pair of its subcoalitions, e.g.,

$$V_{S \cup T} \geq V_S + V_T \text{ for all } S, T \subseteq N \text{ such that } S \cap T = \emptyset.$$

In contrast to superadditive, the *subadditive* environment is one in which the coalition value of a given coalition is strictly less than the sums of the coalition value of each pair of its subcoalitions, e.g.,

$$V_{S \cup T} < V_S + V_T \text{ for all } S, T \subseteq N \text{ such that } S \cap T = \emptyset.$$

In both environments, there is monotonicity in coalition value based on the size of coalitions. However, a *non-superadditive* [24] environment is one in which coalition values have no relationship to the size of coalitions at all. They are arbitrarily random. This environment is similar to the real world. It is less explored in coalition formation but has recently received more attention in multi-agent systems research recently [18, 4, 5, 21, 24, 22, 1, 23, 25].

2.1.2 Example of Coalition Formation

We now consider a classic game of coalition formation, the Sandal Maker game [3, 8, 20]. In this game, there are five sandal makers, whom we shall refer to as agents. Agent 1 and 2 make only left sandals, while agents 3, 4, and 5 make right sandals. (Although no sandal makers in the world would produce sandals in this bizarre manner, this convincing example provides a fruitful model of coalition formation because they need to cooperate in order to sell their product.) In one cycle, a left sandal maker can produce 17 sandals, while a right sandal maker can produce 10 sandals. Obviously, a single sandal is worth nothing, only a pair of left and right sandals can be sold for 20 dollars. All the scrap leather and unused sandals are thrown away at the end of each cycle. In this game, agents need to form coalitions of left and right sandal makers in order to create value to their coalitions and divide the money. Given this simple information, we can determine coalition

values as following. Since an agent alone cannot sell its sandals, the value of a singleton coalition is 0. Also, a coalition of agents that produces the same side sandals is worth 0. Apart from this, a coalition consisting of both agents capabled of producing each moiety (side) will be basically constrained by the smallest number of either moiety. For example, the value of a coalition of two agents, each of a moiety, is limited by the smaller number of right sandals. The value of a coalition of three agents, one of which is of the left moiety, is limited by the number of left sandals. The value of a coalition of four members, one of which of the right moiety, is limited by the number of right sandals. Finally the coalition value of the grand coalition is limited by the number of left sandals. Note that we will denote a coalition S with the list of its member to designate the coalition value as well. For example, V_1 refers to the value of coalition $S = \{1\}$ while $V_{1,2}$ refers to the value of coalition $S = \{1, 2\}$. Below is the characteristic function that summarizes coalition values:

$$
\begin{aligned}
&V_1 = V_2 = V_3 = V_4 = V_5 = 0 \\
&V_{1,2} = V_{3,4} = V_{3,5} = V_{4,5} = V_{3,4,5} = 0 \\
&V_{1,3} = V_{1,4} = V_{1,5} = V_{2,3} = V_{2,4} = V_{2,5} = 200 \\
&V_{1,3,4} = V_{1,3,5} = V_{1,4,5} = V_{2,3,4} = V_{2,3,5} = V_{2,4,5} = 340 \\
&V_{1,2,3} = V_{1,2,4} = V_{1,2,5} = 200 \\
&V_{1,3,4,5} = V_{2,3,4,5} = 340 \\
&V_{1,2,3,4} = V_{1,2,3,5} = V_{1,2,4,5} = 400 \\
&V_{1,2,3,4,5} = 600
\end{aligned}
$$

If agent 1 and 3 agree to make a deal, while player 2 and 4 agree on another deal, a payoff configuration could be $(100, 50, 100, 150, 0; \{1, 3\}, \{2, 4\}, \{5\})$. Is this, however, a solution of the game? Since agents are self-interested, reaching such agreement may not always be this easy because there may be a chance that some agents are still looking to increase their payoffs. In the following, we shall explore solution concepts that bring stability to the game.

2.2 Optimal Coalition Structures

Searching for optimal coalition structures has gained much attention from researchers recently. It is so important for two reasons: *i*) it indicates the optimal solution of a given system, and *ii*) it helps determining the core of the system (collective rationality). In the following, we shall discuss the overview of the problem as it is presented in the literature [18].

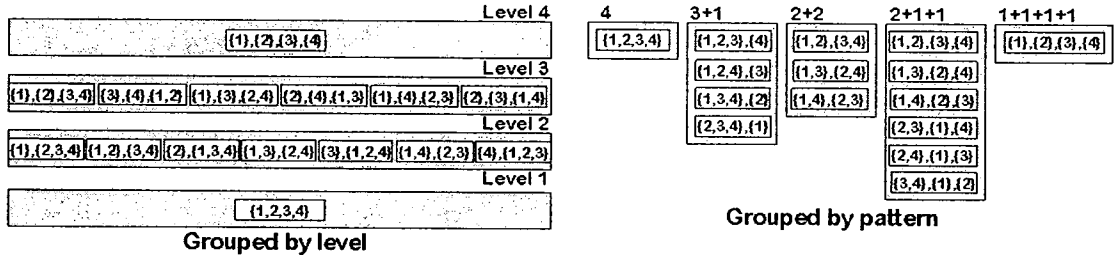

Figure 2.1: **Configuration Bounds**

Given a CS, we define its value,

$$V(CS) = \sum_{S \in CS} V_S,$$

which indicates the system's utility yielded by that partitioning. An optimal coalition structure is a CS^* such that

$$CS^* = argmax_{CS \in L} V(CS)$$

The number of all coalition structures can be determined by B_n [6], *Bell Number* which is the size of the whole search space. Since the value of B_n can be very large for a small value of n, existing algorithms tend to divide the search space into small portions. There are two divisional methods. Firstly, we can categorize CSs by the number of coalitions within them [18]. We denote the set of CSs, whose number of coalitions of each CS is $1 \leq i \leq n$, by L_i. Each L_i is known as a layer. The number of CSs in L_i is known as the *Stirling Number of the Second Kind* [6],

$$S(n,i) = \frac{1}{i!} \sum (-1)^{ki}{}_{k=0} \binom{i}{k} (i-k)^n, \text{where} \binom{i}{k} \equiv \frac{i!}{(i-k)!k!}$$

Hence, the set of all CSs is $L = \bigcup_{i=1}^{n} L_i$.

Alternatively, we can categorize CSs by the integer partition of n that describes the number of coalitions and their cardinalities. Each instance j of such a partition is known as a "pattern" [28, 26] or a "configuration" [11], G_j, which is usually written in the form $b_1 + \ldots + b_k$, where $\sum_{l=1}^{k} b_l = n$. Given a set of 4 agents, all the configurations are 4, 3+1, 2+2, 2+1+1 and 1+1+1+1. Figure 2.1 shows all coalition structures of 4 agents categorized by level and by pattern (configurations).

Table 2.1: Search Space in Coalition Structure where "B_n" is the number of coalition structures, "Largest L_i" is the largest layer i, "$S(n,i)$" is the number of CS in that layer i, "# of Config." is the number of configuration, "Conf Max" is the configuration which has the largest number of CSs, "CS Max" is the number of CSs in "Conf Max".

# Agents	26	27	28	29	30
B_n	4.96E+19	5.46E+20	6.16E+21	7.13E+22	8.47E+23
Largest L_i	10	10	10	11	11
$S(n,i)$	1.32E+19	1.43E+20	1.54E+21	1.81E+22	2.15E+23
# of Config.	2436	3010	3718	4565	5604
Config Max	4+4+3(6)	3(9)	4+3(8)	4+4+3(7)	3(10)
CS Max	3.72E+14	2.98E+15	2.08E+16	1.51E+17	1.21E+18

2.2.1 The Analysis of the Problem

Sandholm et al. [18] show that computing the optimal coalition structures in a non-superadditive environment is non-trivial; it is NP-hard because B_n can be very large for a small n. Table 2.2.1 shows approximate numbers of coalition structures for $11 \leq n \leq 30$. For $n = 11$ agents, B_{11} is relatively small but is very large for 30 agents, $B_{30} = 8.47 \times 10^{23}$. Let us consider the size of search space by levels. Given n agents, the number of coalition structures in each level increases exponentially, around $6^n(roughly)$, as the level gets higher. It reaches the peak around the middle level and decreases exponentially towards the top level. For $n = 11$ agents, level 5, L_5, has the highest number of coalition structures, i.e., approximately $S(11,5) = 2.74 \times 10^5$ coalition structures. For $n = 30$ agents, the largest level is 11 whose search space is approximately 2.15×10^{23}. The whole search space for any given n agents is slightly higher than that of largest level. In case of dividing the search space into configurations, each divided search space can be relatively small. For 11 agents, there are 56 configurations. The largest configuration is 3+3+3+2 whose number of coalition structures is approximately 3.69×10^3. For 30 agents, there are 5604 configurations. The largest configuration is 3+3+3+3+3+3+3+3+3+3 whose number of coalition structures is 1.21×10^{18}.

2.2.2 Coalition Value Distribution

In traditional algorithms of this problem, there are 4 four environments that are considered: normal, uniform, superadditive and subadditive. These environments involve the distribution of coalition values only. It is obvious that the structure of CS^* depends on the distribution of coalition values and so

does the performance of an algorithm. Small coalitions (e.g., of size 1 or 2, for example) tend to be in the optimal coalition structures if forming larger coalitions does not increase the value high enough. In this case, it is better for the system if most agents remain singleton coalitions. However, many real world environments involve cooperation among agents. Small coalitions do have enough resources to perform tasks or create any coalition value. Composite web services is a simple example. Agents have to form coalitions in order to create joint values which will be divided among coalition members. Hence, coalitions of small cardinality tend to be useless in composition of optimal coalition structures. As shown in table 2.1, the number of coalition structures in certain areas, e.g. Largest L_i and Config. Max, can be very large. Hence, it is important that the algorithm be consistently efficient compared with various distribution forms. In the following section, the previous algorithms for finding optimal coalition structures will be discussed.

2.3 Previous Works in Optimal Coalition Structures

Due to the fact that the search space of the optimal coalition structure problem is very large and the search terrain is arbitrarily random, the algorithm to solve the problem needs to perform efficiently. Previous algorithms tend to divide the whole search space into smaller parts. The division is based on the structure of coalitions in CSs and a lexicographic order of agents within coalitions.

Sandholm, Larson, Andersson, Shehory and Tohm [18] propose an anytime algorithm (that can yield an approximate answer, whose quality depends on the computation executed, at any time) that guarantees improvement of the worst-case bound as the algorithm proceeds. The large search space of all coalition structures is divided into levels, each of which is $L_i, 1 \leq i \leq n$ (a level where each CS has i coalitions). The algorithm advances through levels $1, 2, n, n-1, \ldots, 3$ and search through all CSs in each L_i in the breadth-first search manner. The algorithm can guarantee that the solutions that have been found after finishing the first two levels are within the bound $k = n$ from the optimal solution. Although this bounds drops as more levels have been completed, the search space in many levels is still large. Dang and Jennings [2] improve the performance of Sandholm et al.'s algorithm. Having finished the first three levels (1, 2 and n), Dang et al.'s algorithm then generates a list of indicators that will be used to determine various configurations across levels $n-1$ and 3. The indicator is simply used to choose

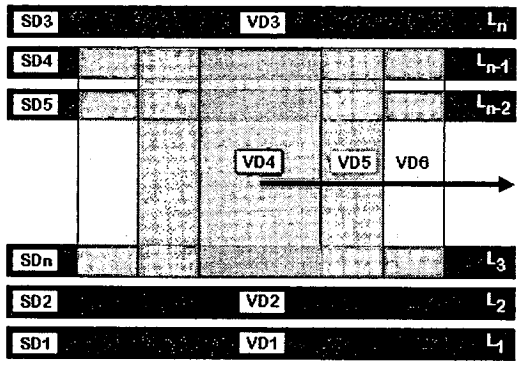

Figure 2.2: **Search Direction in Divided Search Space**
Sandholm et al. [18] divides search space into levels. The direction of search is $L_1, L_2, L_n, L_{n-1}, \ldots L_3$. Dang et al. [2] follow the first two steps of Sandholm et al. but search through portions of remaining levels.

any configuration that contains at least a coalition of a certain cardinality. Given a set of 20 agents, for example, the next cycle after L_n is to search through all coalition structures of any configuration that has at least one coalition of cardinality 16, 15, 13 and 10. Dang et al.'s algorithm guarantees that it reaches bounds closest to the optimal faster than Sandholm et al.'s algorithm. This is due to the greater selectivity in searching through L_i. Although these two algorithms can guarantee improving solutions as time elapses, they need to exhaustively search through the whole search space in each cycle to guarantee optimality. Figure 2.2 illustrates the search direction in the divided search space.

The common problem of Sandholm et al.'s and Dang et al.'s algorithms is that they rely on the completion of the search on the first two levels. For a small number of agents, e.g., 20-40 agents, the search space over the two levels is practically small, e.g., around 10^6 and 10^{12} coalitions respectively. Once the number of agents grows larger, e.g., 60 agents or more, the search space of the first two levels becomes too large to complete, e.g., around 10^{18}, let alone the remaining layers. Moreover, the solution obtained within the elapsed time can be bad. Furthermore, accessing the coalition values can be a problem. One can compute coalition values every time a coalition structure is generated but this can cost too much time because the same coalition value has to be computed again and again. An alternative to this is to keep the coalition values in memory. This, however, requires a large space of memory that no single computer can offer.

Sombattheera and Ghose [26] propose the idea of partitioning the search

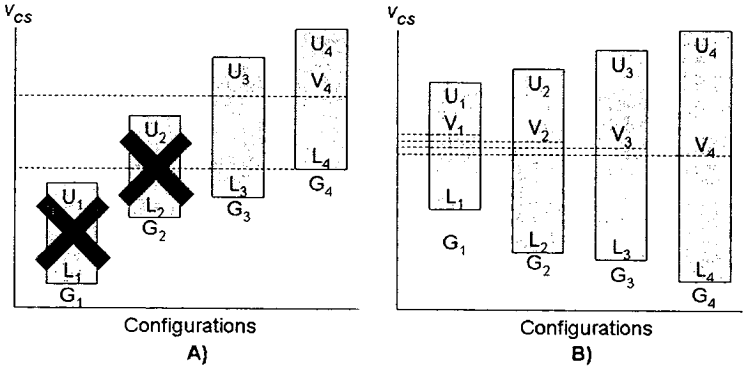

Figure 2.3: **Configuration Bounds**
In general Rahwan et al.'s algorithm can prune portions of the search space (A) when a present solution is better than that of remaining configurations. However, it may fail to do so due to misleading upper bounds(B).

space into smaller sections, referred to as "pattern". Rahwan, Ramchurn, Dang and Jennings [16] follow this idea by proposing a near optimal coalition structures algorithm. To begin with, all coalitions in the first two levels will be examined. Along with this, the upper bound and the mean value for each configuration are established. The configuration to be searched first is either *i*) one whose ratio between the upper bound and the mean value is the lowest, or *ii*) one that is likely for the algorithm to prune the largest search space in its configuration. As the algorithm proceeds, the upper bound of each configuration will be updated and the configuration will be eliminated if its upper bound is lower than the best solution found so far. Given enough time, the algorithm terminates when there is no configuration left to be searched and the solution is the optimal. Based on this work, Rahwan, Ramchurn, Dang, Giovannucci and Jennings [15, 17] develop an optimal algorithm that, to our best knowledge, is *the state of the art of anytime optimal coalition structure algorithm.* They apply pruning over configurations and within coalitions to cut the search space massively and generate optimal solutions rapidly under a number of data distributions [7]. However, this algorithm can be misled by miscalculation of the bound. As shown in figure 2.3, the search will take place in all configurations whose bounds are higher than the actual values.

Yun Yeh [29] proposes a deterministic algorithm to compute optimal coalition structure based on the integer programming technique. His solution is based on the bi-partitioning principle. A whole set of agents will be bi-partitioned into ${}^nC_2$ pairs. Each coalition in each pair will be recursively partitioned downward to the singleton level where all coalitions are of

cardinality 1. The algorithm then works upwards for the optimal value of each coalition. At each level, the pair whose combined value is the highest will be the optimal value of the coalition. This algorithm guarantees optimal results with 3^n time and space complexities. However, this algorithm is not appropriate for a multi-agent systems environment because the problem becomes intractable for even a small number of agents. On the other hand, Sen and Dutta [19] use an order-based genetic algorithm (an evolution algorithm, which deploys the biological evolution concept, i.e., inheritance, mutation, selection, and crossover, to constitute the search for approximate answer in large combinatorial problems) as a stochastic search process to identify the optimal coalition structure. Although their algorithm has no performance guarantees, they claim that it is found to dominate the deterministic algorithm in a significant number of problem settings. Due to the nature of genetic algorithms, an additional advantage of their algorithm is its scalability to larger problem sizes and to problems where performance of a coalition depends on other coalitions in the environment. Larson and Sandholm [7] present experimental results for three anytime algorithms that search the space of coalition structures. They show that, in the average case, all three algorithms do much better than the recently established theoretical worst case results in [18]. They also show that no one algorithm is dominant. Each algorithm's performance is influenced by the particular instance distribution, with each algorithm outperforming the others for different instances. They present a possible explanation for the behavior of the algorithms and support their hypothesis with data collected from a controlled experimental run.

Rahwan and Jennings [13, 14] develop an algorithm for computing coalition values in a distributed manner. The task of computing coalition values is distributed evenly among cooperative agents, who seem to be involved in computing CS^*, with respect to communication and computation redundancy. They claim to have massively reduced the number of of messages sent among agents and memory usage. However, they have tested their algorithm against 25 agents only, which is practically small for multi-agent environments. As the number of agents increases linearly, the size of the problem, 2^n, increases exponentially. Even though the algorithm can divide the task among agents evenly and efficiently, the workload of each agent becomes unmanageable for even a small number of agents. Even with 40, 60 and 80 agents, the size of the task is approximately 2.45×10^{10}, 1.92×10^{16}and1.51×10^{22}, respectively, let alone realistic environments where the number of agents is much more than this.