Coalition Structure Generation in Multi-Agent Systems With Positive and Negative Externalities

Talal Rahwan[1], **Tomasz Michalak**[2], **Nicholas R. Jennings**[1], **Michael Wooldridge**[2], **Peter McBurney**[2]
[1]School of Electronics and Computer Science, University of Southampton, UK
[2]Department of Computer Science, University of Liverpool, UK

Abstract

Coalition structure generation has received considerable attention in recent research. Several algorithms have been proposed to solve this problem in *Characteristic Function Games (CFGs)*, where every coalition is assumed to perform equally well in any coalition structure containing it. In contrast, very little attention has been given to the more general *Partition Function Games (PFGs)*, where a coalition's effectiveness may change from one coalition structure to another. In this paper, we deal with PFGs with positive and negative externalities. In this context, we identify the minimum search that is required in order to establish a bound on the quality of the best coalition structure found. We then develop an anytime algorithm that improves this bound with further search, and show that it outperforms the existing state-of-the-art algorithms by orders of magnitude.

1 Introduction

One of the important issues in multi-agent system research is to generate a *coalition structure* , i.e. an exhaustive and disjoint division of the set of agents, such that the outcome of the entire system is maximised. Usually, this *coalition structure generation (CSG)* problem is dealt with in *Characteristic Function Games (CFGs)*, where the value of a coalition is not affected by the way non-members are partitioned (see, e.g., [Shehory and Kraus, 1998; Sandholm *et al.*, 1999; Dang and Jennings, 2004; Rahwan and Jennings, 2008; Rahwan *et al.*, 2009]). However, even under this strict assumption, the CSG problem has been shown to be NP-complete [Sandholm *et al.*, 1999]. Given this, a number of approaches have been proposed to help circumvent this complexity, including the use of anytime algorithms. Specifically, such an algorithm generates an initial solution that is guaranteed to be within a certain bound from the optimum, and then improves the quality of its solution as it searches more of the search space, and establishes progressively better bounds until an optimal solution is found. Although these algorithms might, in the worst case, end up searching the entire space (i.e. they run in $O(n^n)$), the anytime property has a number of advantages. First, an outcome is delivered relatively quickly and, second, the algorithm is more robust against failure due to premature termination. All anytime CSG algorithms take advantage of the property of *CFGs* that the value of a particular coalition is always the same in every possible coalition structure. Although this assumption is valid in many environments, there exist a number of cases in which the performance of one coalition may be directly influenced by the cooperative arrangements of the other agents in the system. For instance, in resource allocation problems, a coalition that increases resource consumption in one part of the system can be detrimental to the agents' effectiveness in another part. In such cases, *CFGs* are not appropriate and much more general *Partition Function Games (PFGs)* have to be applied. In *PFGs* a coalition's value may depend on the formation of another coalition in a system. More formally, given two coalition structures: CS and CS' such that $C_1, C_2, C_3 \in CS$ and $C_1, (C_2 \cup C_3) \in CS'$, the value of C_1 may be different in CS than in CS' due to the merge of C_2 with C_3. Such an effect is usually referred to as an *externality* induced upon C_1 by the merge of C_2 and C_3 and the formation of coalition $C_2 \cup C_3$. In this context, only one CSG algorithm has been proposed [Michalak *et al.*, 2008], which deals with two classes of PFGs, denoted PF^+_{sub} and PF^-_{sup}. Specifically, in PF^+_{sub} a merger of any two coalitions decreases their joint value (or keeps it constant), and increases the values of other coalitions in the structure (or keeps them constant). In other words, the coalition values are *weakly sub-additivite*[1] and the externalities are positive (or equal to zero). Conversely, PF^-_{sup} is the class in which the values are *weakly super-additive*, and the externalities are negative (or equal to zero).

Both PF^+_{sub} and PF^-_{sup} are very popular in economics and other social sciences where interdependencies between coalitions constitute, in many cases, the core of the analysed problem. Examples include Research & Development coalitions among pharmaceutical companies or various forms of international (macroeconomic/environmental) policy coordination. When two companies decide to jointly develop a new drug, their market share is likely to increase, whereas the mar-

[1]Recall that *weak sub-additivity* refers to the case where, given any two coalitions C_1, C_2, and given any coalition structure $CS \supseteq \{C_1, C_2\}$, we have: $V(CS) \leq V(CS')$, where $CS' = (CS \backslash \{C_1, C_2\}) \cup \{C_1 \cup C_2\}$. On the other hand, *weak super-additivity* refers to the case where $V(CS) \geq V(CS')$.

ket position of other companies is likely to decrease. Conversely, countries which decide to join international agreements and limit their emissions may hamper their economic growth (sub-additivity property) but, at the same time, they may have a positive influence on the rest of the world. Clearly, modeling the above phenomena with $CFGs$ would be inappropriate and $PFGs$ should be applied instead. Examples of systems with externalities can also be found in multi-agent system contexts. As more commercial activity moves to the Internet, we can expect online economies and societies to become increasingly sophisticated, as is happening, for instance, with real time electronic purchase of wholesale telecommunications bandwidth or computer processor resources. In such contexts, *ad hoc* coalition formation will need to allow for coalition externalities in as broad a sense as possible. In fact, even PF^+_{sub} and PF^-_{sup} classes are too strict to model a number of environments. For instance, a merger of a coalition in a multi-agent system may be profitable when coalitions are relatively small. However, if the size and extent of a created group results in communication problem, the cost of cooperation may exceed profits.[2] In other words, such systems are likely to meet neither super- nor sub-additivity properties.

Against this background, in this paper, we deal with two significantly broader classes of *PFGs* created from PF^+_{sub} and PF^-_{sup} by removing the assumption of super- and sub-additivity. For these two classes, denoted PF^+ and PF^-, we develop a number of techniques to solve the CSG problem in an anytime fashion. In particular:

- The first challenge related to PF^+ and PF^- is caused by the fact that, in the presence of externalities, a coalition's value may differ from one structure to another. However, we show how to compute upper and lower bounds on the values of all feasible coalitions in the system (Section 3);
- We identify the minimum set of coalition structures that needs to be searched in order to establish a worst-case guarantee on solution quality (Section 4);
- We develop a novel algorithm for improving the worst-case guarantee with further search (Section 5);
- We develop a preprocessing algorithm that prunes parts of the search space (Section 6.1); and
- We propose a revised version of the state-of-the-art anytime CSG algorithm [Rahwan *et al.*, 2009], called IP$^{+/-}$, that can be applied in our PF^+/PF^- setting (Section 6.2).

Since there are no previous algorithms that improve worst-case guarantees in PF^+/PF^-, we benchmark our algorithm against the state-of-the-art CFG algorithms that were developed for this purpose. Note that CFGs are a special case of PF^+/PF^- and, therefore, the bounds provided by our algorithm hold even in a CFG setting. Surprisingly, we find that our algorithm outperforms the other algorithms by orders of magnitude. This is despite the fact that the other algorithms take advantage of the special properties of CFGs, while ours does not. Moreover, unlike Michalak *et al.* [2008] who could not test their algorithm for more than 10 agents (because of the inefficient way by which they generate random instances of their problem), we develop an efficient function for generating random instances of PF^+/PF^-, which enables us to test our IP$^{+/-}$ algorithm for up to 20 agents (Section 7.2).

[2] Such a situation occurs, for instance, when the number of users of a local network increases over its optimal capacity.

2 Basic Definitions

Given a coalition C and a coalition structure CS, we denote by $v(C, CS) \geq 0$ the value of C in CS, and denote by $V(CS) = \sum_{C \in CS} v(C, CS)$ the value of CS. Moreover, given the set of n agents, $A = \{a_1, a_2, ..., a_n\}$, and a coalition C, we denote by $\overline{C}$ the agents in A that do not belong to C (i.e. $\overline{C} = A \backslash C$).

We define a *partition of* C as a set containing disjoint coalitions of which the union equals C. The *value of a partition* is defined as the sum of the values of all the coalitions in that partition. We denote by $V(P, CS)$ the value of partition P in CS (i.e. $V(P, CS) = \sum_{p \in P} V(p, CS)$), and denote by $\mathcal{P}_C$ the set of all the possible partitions of C.[3]

We denote by $\mathcal{I}_s : s \in \mathbb{N}$ the set of possible integer partitions of s,[4] and denote by $S_I : I \in \mathcal{I}_n$ the sub-space containing all the coalition structures within which the coalition sizes match the parts of the integer partition (e.g. given 4 agents, $S_{[1,1,2]}$ contains every coalition structure in which two coalitions are of size 1, and one coalition is of size 2). Moreover, we denote by $G(\mathcal{I}_s)$ the *integer partition graph*[5] of s. Finally, given a set (or a multiset) Π, we denote by $|\Pi|$ the cardinality of Π.

3 Computing Upper and Lower Bounds

In this section, we show how to compute upper and lower bounds on the value of any partition or coalition. This is important since these bounds are going to be used in every step of the CSG process.

Theorem 1 *Consider a PF^- (PF^+) setting. Given a coalition $C \subseteq A$, a partition $P \in \mathcal{P}_C$, and a coalition structure $CS \supseteq P$, the following holds, where $\overline{C} = \{a_1, ..., a_{|\overline{C}|}\}$:*

$$V(P, \{\overline{C}\} \cup P\}) \leq (\geq)\ V(P, CS) \leq (\geq)\ V(P, \{\{a_1\}, ..., \{a_{|\overline{C}|}\}\} \cup P)$$

Proof. To simplify notation, let $CS' = \{\overline{C}\} \cup P$ and let $CS'' = \{\{a_1\}, ..., \{a_{|\overline{C}|}\}\} \cup P$. Also, without loss of generality, assume that $CS \neq CS'$ and $CS \neq CS''$. Then, given a PF^- (PF^+) setting, we first need to prove that:

$$V(P, CS') \ \leq (\geq)\ \ V(P, CS) \qquad (1)$$

Beginning with CS, one could reach CS' by performing multiple steps, each involving the merging of two coalitions into one. After each step, the coalitions in P remain unchanged

[3] Note that if $C = A$, then any partition of C would be a coalition structure, and $\mathcal{P}_A$ would be the set of all possible coalition structures.

[4] Recall that an integer partition of s is a multiset of positive integers (called *parts*) that add up to exactly s. For presentation clarity, we use square brackets throughout this paper (instead of curly ones) to represent integer partitions.

[5] For more details on this graph, see [Rahwan and Jennings, 2008].

and, due to negative (positive) externalities, their values can only decrease (increase). As a result, the inequality in (1) holds. Similarly, beginning with CS'', it can be proved that the following holds: $V(P, CS) \leq (\geq)\ V(P, CS'')$.

□

As opposed to Michalak et al. [2008] who bound the value of any given coalition C, Theorem 1 bounds the value of *any given partition of* C. Specifically, for every partition $P \in \mathcal{P}_C$, the upper and lower bounds can be computed in a PF^- (PF^+) setting as follows, where $\overline{C} = \{a_1, ..., a_{|\overline{C}|}\}$:

$$LB_P\ (UB_P) = \sum_{p \in P} v(p, P \cup \{\overline{C}\})$$

$$UB_P\ (LB_P) = \sum_{p \in P} v(p, P \cup \{\{a_1\}, ..., \{a_{|\overline{C}|}\}\})$$

Note that, by assuming that $P = \{C\}$, it is possible, using the above equations, to compute an upper bound UB_C and a lower bound LB_C on the value of any coalition C. Moreover, given an integer partition $I \in \mathcal{I}_s : s \leq n$, if we denote by S_I the set of partitions within which the coalition sizes match the parts in I, then we can compute upper and lower bounds (denoted UB_I and LB_I respectively) on the values of the partitions in S_I:[6]

$$\forall I \in \mathcal{I}_s, UB_I = \max_{P \in S_I} UB_P, \text{ and } LB_I = \min_{P \in S_I} LB_P$$

4 Establishing a Worst-Case Bound β

Having computed upper and lower bounds for coalition values, we now show how these can be used to identify the minimum search required to establish a worst-case *ratio bound* β *from the optimum.* In order to do so, we will use the following theorem:

Theorem 2 *Let X be a set of elements, and let Y_s be a set containing subsets of X such that: $\forall y \in Y_s, |y| \leq s$. Moreover, for all $x \in X, y \in Y_s$, let us define $v(x, y) \geq 0$ as the value of x in y, and let us define $V(y) = \sum_{x \in y} v(x, y)$ as the value of y. Then, for any $Y'_s \subseteq Y_s$, if:*

$$\forall x \in X, \exists y' \in Y'_s : x \in y' \text{ and } v(x, y') = \max_{y \in Y_s} v(x, y) \quad (2)$$

then the following holds:

$$\max_{y \in Y_s} V(y) \ \leq\ s \times \max_{y \in Y'_s} V(y)$$

In other words, if every element in x appears with its maximum value in one, or more, of the subsets in Y'_s, then the best subset in Y'_s is within a ratio bound $\beta = s$ from the best subset in Y_s.

Proof. Let y^*, y^{**} be the best subset in Y_s, Y'_s respectively, i.e. $y^* = \arg\max_{y \in Y_s} V(y)$ and $y^{**} = \arg\max_{y \in Y'_s} V(y)$. Then, assuming that (2) holds, we need to prove the following:

$$V(y^*) \leq s \times V(y^{**})$$

To this end, let x^* be defined as: $x^* = \arg\max_{x \in y^*} v(x, y^*)$. Since y^* contains at most s elements, then the following holds:

$$V(y^*) \leq s \times v(x^*, y^*) \leq s \times \max_{y \in Y_s} v(x^*, y) \quad (3)$$

Moreover, from (2), we know that there exist $y' \in Y'_s$ such that $x^* \in y'$ and:

$$v(x^*, y') = \max_{y \in Y_s} v(x^*, y) \quad (4)$$

Since the value of y' is the sum of the values of the coalitions in y', and since $x^* \in y'$, then:

$$v(x^*, y') \leq V(y') \quad (5)$$

Finally, from the definition of y^{**}, we know that:

$$V(y') \leq V(y^{**}) \quad (6)$$

From (3), (4), (5) and (6), we find that:

$$V(y^*) \leq s \times \max_{y \in Y_s} v(x^*, y) \leq s \times V(y') \leq s \times V(y^{**})$$

□

Having proved Theorem 2, we now show how it can be used while proving the main theorem for establishing a ratio bound β:

Theorem 3 *To establish a bound β on the value of a coalition structure given a PF^+ setting, every sub-space $S_I : I \in \mathcal{I}_n : |I| \leq 2$ must be searched. With this search, the number of coalition structures searched is 2^{n-1}, and the bound $\beta = n$. On the other hand, given a PF^- setting, every sub-space $S_I : I \in \{[n], [n-1, 1], [n-2, 1, 1], \cdots, [1, 1, \cdots, 1]\}$ must be searched. With this search, the number of coalition structures searched is $2^n - n + 1$, and $\beta = \lceil \frac{n}{2} \rceil$*

Proof. To establish a bound, the *maximum possible value* of each coalition C has to be observed (in some coalition structure). Given a PF^+ setting, the only coalition structure in which C is guaranteed to have its maximum value is $\{C, A \backslash C\}$. Based on this, the sub-spaces $S_I : I \in \mathcal{I}_n : |I| \leq 2$ must be searched, and these contain 2^{n-1} coalition structures.

To prove that $\beta = n$, we use Theorem 2, and that is by considering X to be the set of coalitions, Y_n to be the set of coalition structures[7], and Y'_n to be the union of the sub-spaces $S_I : I \in \mathcal{I}_n : |I| \leq 2$. In more detail, since every element in X appears *with its maximum value* in one, or more, of the coalition structures in Y'_s, then the best coalition structure in Y'_s is within a ratio bound $\beta = n$ from the best coalition structure in Y_n (see Theorem 2).

On the other hand, given a PF^- setting, the only coalition structure in which C is guaranteed to have its maximum value is: $\{C, \{a_1\}, ..., \{a_{|\overline{C}|}\}\}$, where $\{a_1\} \cup ... \cup \{a_{|\overline{C}|}\} = \overline{C}$. Based on this, $S_I :\in \{[n], [n-1, 1], [n-2, 1, 1], ..., [1, 1, ..., 1]\}$ must be searched. With this search, the number of searched coalition structures would be $2^n -$

[6] Note that $S_I : I \in \mathcal{I}_n$ refers to the set of *coalition structures* that match I, while $S_I : I \in \mathcal{I}_s : s \leq n$ refers to the set of *partitions* that match I.

[7] This is possible since every coalition structure is a subset of X containing at most n of the elements (i.e. coalitions) in X.

$n+1$ since every possible coalition appears in a unique coalition structure, except for the singleton ones (which all appear in a single coalition structure).

As in the PF^+ case, we use Theorem 2 to prove that $\beta = \lceil \frac{n}{2} \rceil$. Here, however:

- We consider X to contain all the possible coalitions, as well as the possible partitions that contain singletons. For example, given 3 agents, X contains all the possible coalitions of size 3, as well as $\{\{a_1\},\{a_2\}\}$, $\{\{a_1\},\{a_3\}\}$, $\{\{a_2\},\{a_3\}\}$ and $\{\{a_1\},\{a_2\},\{a_3\}\}$. Note that the *maximum possible value* of each one of these partitions has been observed in $S_{[1,\cdots,1]}$ (see Theorem 1).
- We consider the singletons in every coalition structure to be grouped together (e.g. $\{\{a_1\},\{a_2\},\{a_3,a_4\},\{a_5,a_6\}\}$ becomes $\{\{\{a_1\},\{a_2\}\},\{a_3,a_4\},\{a_5,a_6\}\}$). Based on this, we can consider $Y_{\lceil n/2 \rceil}$ to be the set of coalition structures (because any coalition structure now contains at most $\lceil n/2 \rceil$ elements from X).
- We consider: $Y'_{\lceil n/2 \rceil} = S_{[n]} \cup S_{[n-1,1]} \cup \cdots \cup S_{[1,\cdots,1]}$.

The above three points imply that the best coalition structure in $Y'_{\lceil n/2 \rceil}$ is within a ratio bound $\beta = \lceil n/2 \rceil$ from the best coalition structure in $Y_{\lceil n/2 \rceil}$, and that is because every possible element in X appears *with its maximum value* in $Y'_{\lceil n/2 \rceil}$.

□

Note that, in CFGs, it is *sufficient* to search levels 1 and 2 of the coalition structure graph in order to bound β [Sandholm *et al.*, 1999]. However, it is also possible to bound β by searching any other set of coalition structures as long as every coalition appears at least once in this set. On the other hand, given a PF^- setting, it is *necessary* to search $S_I : I \in \{[n],[n-1,1],[n-2,1,1],\cdots,[1,1,\cdots,1]\}$ and, given a PF^+ setting, it is *necessary* to search $S_I : I \in \mathcal{I}_n : |I| \leq 2$ (see Theorem 3).

5 Improving the Worst-Case Bound β

In this section, we present a novel anytime algorithm that reduces the ratio bound β with further search. This algorithm is based on the following observation (which is a direct result of Theorem 2): If we consider X to be the set of all possible coalitions, and Y_n to be the set of all possible coalition structures, and if $Y'_n \subseteq Y_n$ is a set of coalition structures in which the maximum value of every element in X appears at least once, then by searching through Y'_n, we end up with a solution that is within a ratio bound $\beta = n$ from the optimal coalition structure.

Based on the above observation, the ratio bound β can be reduced by using the following two main steps:

1. Add certain partitions to X and, in every coalition structure, group the coalitions that match any of those partitions. This is done such that the maximum number of elements from X that can appear in a single coalition structure drops to $m < n$. Given 3 agents, for example, the possible structures are: $\{\{a_1,a_2,a_3\}\}$, $\{\{a_1\},\{a_2,a_3\}\}$, $\{\{a_2\},\{a_1,a_3\}\}$, $\{\{a_3\},\{a_1,a_2\}\}$ and $\{\{a_1\},\{a_2\},\{a_3\}\}$. Now, suppose that X contains every possible coalition. Then, if we add the partition $\{\{a_1\},\{a_2\}\}$ to X, and we group the coalitions that match this partition (i.e. $\{\{a_1\},\{a_2\},\{a_3\}\}$ becomes $\{\{\{a_1\},\{a_2\}\},\{a_3\}\}$), then the maximum number of elements from X that can appear in a single coalition structure drops from 3 to 2.
2. For every partition P that was added to X, examine the coalition structure in which P appears with its maximum value. More specifically, given a PF^- setting, examine the coalition structure $CS = \{\{a_1\},\cdots,\{a_{|\overline{C}|}\}\} \cup P$ and, given a PF^+ setting, examine $CS = \{\overline{C}\} \cup P$ (see Theorem 1). In so doing, we would have searched through a set of coalition structures in which the maximum value of every element in X appears at least once. This, in turn, reduces the ratio bound to m (see Theorem 2).

6 Searching the Sub-Spaces

We now present two algorithms for searching the sub-spaces. The first is a preprocessing algorithm that prunes some of the sub-spaces by comparing their upper and lower bounds (Section 6.1), while the second is a revised version of Rahwan et al.'s [2009] IP algorithm, which we call IP$^{+/-}$, that uses the preprocessing algorithm, and solves the CSG problem in PF^+/PF^- settings (Section 6.2).

6.1 Preprocessing

Before detailing the preprocessing algorithm, we first discuss its underlying theoretical background. In particular, the main theoretical results upon which we build are as follows:

Lemma 1 *Given the integer partition graph of s (i.e. $G(\mathcal{I}_s)$), let $G(\mathcal{I}_s)^{s'}$ denote the part of $G(\mathcal{I}_s)$ in which every node (i.e. integer partition) contains at least s' integers that are equal to 1 (where $s' < s$). Then, if we remove those s' parts from every node in $G(\mathcal{I}_s)^{s'}$, then $G(\mathcal{I}_s)^{s'}$ becomes identical to $G(\mathcal{I}_{s-s'})$.*

An example is shown in Figure 1. What is interesting is that, by removing $[1,1]$ from every node in $G(\mathcal{I}_6)^2$ for example, then we will not only get all the integer partitions in $\mathcal{I}_4$, but these integer partitions will also be *ordered and connected* in exactly the same way as they are in $G(\mathcal{I}_4)$. This particular observation will be used in our preprocessing algorithm as we will see later in this section. Now, we give the main theorem:

Theorem 4 *Consider a PF^- (PF^+) setting. Then, given a coalition $C \subseteq A$ and a partition $P \in \mathcal{P}_C$, any coalition structure containing P can be pruned from the search space if there exists another partition $P' \in \mathcal{P}_C$ such that:*

$$\forall p' \in P', \exists p \in P : p' \subseteq (\supseteq)\, p \quad \textit{and} \quad UB_P \leq LB_{P'}$$

Proof. Given a PF^- (PF^+) setting, and given two partitions $P, P' \in \mathcal{P}_C$ such that: $\forall p' \in P', \exists p \in P : p' \subseteq (\supseteq)\, p$ and $UB_P \leq LB_{P'}$, we will prove that, for any coalition structure $CS \supseteq P$, there exists another coalition structure CS' such that $V(CS) \leq V(CS')$.

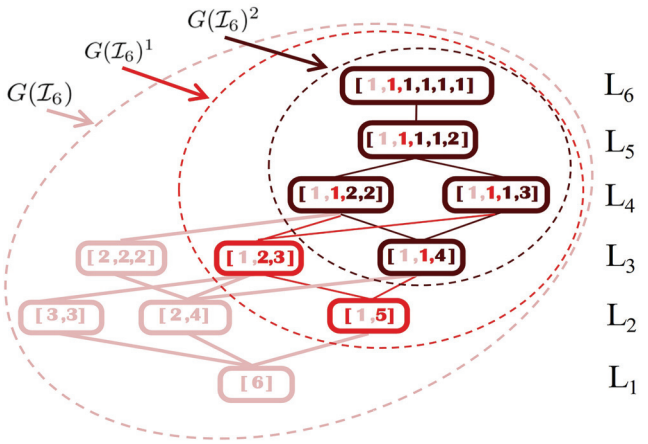

Figure 1: The figure shows how the integer partitions of 4 and 5 appear in the integer partition of 6.

Since P is a partition of C, then $CS \backslash P$ must be a partition of $\overline{C}$. In particular, let $\overline{P} = CS \backslash P$, then of course, $CS = P \cup \overline{P}$. Now, by replacing P with P', we end up with a different coalition structure, denoted CS', such that: $CS' = P' \cup \overline{P}$. In this case, we have:

$$V(CS) = V(P, CS) + V(\overline{P}, CS) \text{ and } V(CS') = V(P', CS') + V(\overline{P}, CS') \quad (7)$$

Since we have: $\forall p' \in P', \exists p \in P : p' \subseteq (\supseteq)\ p$, then every coalition in P' (P) is a subset of some coalition in P (P'). Based on this, as well as the fact that P and P' are partitions of the same coalition, we find that P (P') can be reached from P' (P) by performing multiple steps, each involving the merging of two coalitions from P' (P). This, in turn, implies that CS (CS') can be reached from CS' (CS) by performing merging steps that do not involve any of the coalitions in $\overline{P}$. As a result, and due to negative (positive) externalities, we have:

$$V(\overline{P}, CS) \leq V(\overline{P}, CS') \quad (8)$$

On the other hand, since $UB_P \leq LB_{P'}$, then we have:

$$V(P, CS) \leq V(P', CS) \quad (9)$$

From (7), (8) and (9), we find that $V(CS) \leq V(CS')$. This, in turn, implies that CS can be pruned from the search space. □

From Theorem 4, we can see that the following lemma holds:

Lemma 2 *Consider a PF^- (PF^+) setting. Then, given an integer partition $I \in \mathcal{I}_s : s \leq n$, any sub-space represented by an integer partition $G \in \mathcal{I}_n : I \subseteq G$ can be pruned from the search space if there exists another integer partition $I' \in \mathcal{I}_s$ such that:*

$$\forall i \in I\ (I'), \exists J \subseteq I'(I) : \sum_{j \in J} = i \quad \textit{and} \quad UB_I \leq LB_{I'}$$

The condition $\forall i \in I\ (I'), \exists J \subseteq I'(I) : \sum_{j \in J} = i$ in Lemma 2 implies that the number of parts in I is smaller (greater) than the number of parts in I', and that I and I' are connected in the integer partition graph of s via a series of nodes belonging to consequent levels of the graph. In this case, we use the notation: $I \rightarrow I'$ ($I' \rightarrow I$). In Figure 1, for example, we have: $[1, 5] \rightarrow [1, 1, 1, 1, 2]$.

We can now show how both Lemma 1 and Lemma 2 are used in our preprocessing algorithm to prune sub-spaces (see Algorithm 1). In short, the algorithm tries to find the integer partitions in $\mathcal{I}_2$ that can be pruned using Lemma 2, and then moves to $\mathcal{I}_3$, and then $\mathcal{I}_4$, and so on until it reaches $\mathcal{I}_n$. The way it moves from $\mathcal{I}_{s-1}$ to $\mathcal{I}_s$ is done using the observation from Lemma 1. More specifically, the algorithm adds $[1]$ to every integer partition in $\mathcal{I}_{s-1}$, and then combines the resulting integer partitions with those in $\mathcal{I}_s$ that do not contain any 1. To generate the integer partitions of s that do not contain any 1, we use `getIntParts`$(s, 2)$. This function is implemented using the *parta* algorithm [Riha and James, 1974], which is the state-of-the-art for generating *doubly-restricted*[8] integer partitions.

```
1: Î ← {[1]}  {Initialization}
2: for s = 2 to n do
3:    for I ∈ Î  {Add [1] to every element in Î}  do
4:       I ← I ∪ [1]
5:    end for
6:    Î ← Î ∪ getIntParts(s,2)
7:    for I ∈ Î do
8:       if ( PF⁻ and ∃I' ∈ Î : I → I', UB_I ≤ LB_I')
         or (PF⁺ and ∃I' ∈ Î : I' → I, UB_I ≤ LB_I') then
9:          Î ← Î \ I  {remove I from Î}
10:      end if
11:   end for
12: end for
13: return Î
```

Algorithm 1: Prunes sub-spaces based on Lemma 2.

6.2 The IP$^{+/-}$ algorithm

In this section, we briefly describe the original IP algorithm, and then show how it can be revised for the PF^+/PF^- case.

IP: Let Max_s and Avg_s be the maximum and average value of the coalitions of size s respectively. Then, for all $I \in \mathcal{I}_n$, the IP algorithm computes upper and lower bounds on the value of the best coalition structure in S_I as follows: $UB_I = \sum_{s \in I} Max_s$, $LB_I = \sum_{s \in I} Avg_s$.[9] These bounds are then used to identify any sub-spaces that have no potential of containing a solution better than the current best one (in which case they are pruned from the search space). As for the remaining sub-spaces, IP searches them one at a time, unless a value is found that is higher than the upper bound of another sub-space, in which case, that sub-space no longer needs to be searched. The order in which the algorithm searches through these sub-spaces is based on their upper bounds (i.e. it starts with the one with the highest upper bound, and then the second-highest and so on). Searching a

[8] Unlike *restricted* integer partitions, where the parts are only bounded by a maximum value, *doubly-restricted* integer partitions consist of parts that are also bounded by a minimum value. `getIntParts`$(s, 2)$ sets the minimum value to 2 so that the resulting integer partitions do not contain any 1.

[9] Interestingly enough, Rahwan *et al.* [2009] proved that this lower bound is actually the average value of all the solutions in S_I.

sub-space $S_I : I = [i_1, i_2, ..., i_{|I|}]$ is carried out using *depth-first search* combined with *branch-and-bound*. Basically, for every coalition of size i_1, denoted C_1, the algorithm finds the coalitions of size i_2 that do not overlap with C_1, and for every such coalition, denoted C_2, the algorithm finds the coalitions of size i_3 that do not overlap with C_1 and C_2, and so on. This is repeated until we reach the coalitions of size $i_{|I|}$ that do not overlap with $C_1, C_2, ..., C_{|I|-1}$. For every such coalition, denoted $C_{|I|}$, we would have a coalition structure $CS = \{C_1, ..., C_{|I|}\} \in S_I$. This process is repeated in such a way that is guaranteed to go through every coalition structure in S_I exactly once. To speed up the search, IP applies a branch-and-bound technique as follows. If we denote by CS^{**} the best coalition structure bound so far, then, before the algorithm goes through the coalitions of size i_x that do not overlap with $C_1, ..., C_{x-1}$, it checks whether the following holds:[10]

$$v(C_1) + ... + v(C_{x-1}) + Max_{i_x} + ... + Max_{i_{|I|}} < V(CS^{**})$$

If the above holds, then there is no need to go through any of the coalitions of size i_x that do not overlap with $C_1, ..., C_{x-1}$. This is because any coalition structure containing $C_1, ..., C_{x-1}$ cannot possibly be better than CS^{**}. This simple branch-and-bound technique has proved to be very effective in CFGs (see [Rahwan *et al.*, 2009]).

IP$^{+/-}$: The main difference in our PF^-/PF^+ setting (compared to a CFG setting) is that, instead of having one value for every coalition C, we now have a maximum value UB_C and a minimum value LB_C (see Section 3 for more details). Based on this, IP$^{+/-}$ differs from IP in the following ways:

- It uses the preprocessing algorithm from Section 6.1.
- It computes Max_s and Avg_s as follows: $\forall s \leq n, Max_s = \max_{C \subseteq A:|C|=s} UB_C$ and $Avg_s = \text{avg}_{C \subseteq A:|C|=s} LB_C$.
- The order in which it searches through sub-spaces is based on our anytime algorithm for reducing the ratio bound β (i.e., it always searches the sub-spaces that are necessary to drop the current ratio bound).
- While searching a sub-space $S_I : I = [i_1, i_2, ..., i_{|I|}]$, and before going through the coalitions of size i_x that do not overlap with $C_1, ..., C_{x-1}$, the algorithm checks whether the following holds, where $UB_{\{C_1,...,C_{x-1}\}}$ is computed as in Section 3:

$$UB_{\{C_1,...,C_{x-1}\}} + Max_{i_x} + ... + Max_{i_{|I|}} < V(CS^{**})$$

7 Performance Evaluation

In this section, we evaluate our anytime algorithm for improving the ratio bound, as well as our IP$^{+/-}$ algorithm.

[10]Here, $v(C)$ is used (instead of $v(C, CS)$) to refer to the value of C in CS. This is because IP deals with CFGs where every coalition has one value, regardless of the coalition structure to which it belongs.

7.1 Evaluating the Anytime Ratio Bound Algorithm

Figure 2 shows *on a log scale* the ratio bound β as a function of the number of searched coalition structures, and that is for 24 agents. As can be seen, given a PF^+ setting, our algorithm is significantly faster than the existing CFG algorithms. For example, the number of coalition structures required by our algorithm to establish a ratio bound $\beta = 3$ is only 0.00003% of that required by Sandholm *et al.* [1999], and only 0.9% of that required by Dang and Jennings [2004]. On the other hand, given a PF^- setting, the algorithm requires searching more coalition structures (compared to other algorithms) in order to establish its first ratio bound (see Theorem 3). However, once it establishes its first bound, which requires searching less than 4×10^{-11} of the space, it becomes significantly faster than the other algorithms. For example, the number of coalition structures to establish a ratio bound $\beta = 3$ is only 0.00002% and 0.5% compared to Sandholm *et al.*'s and Dang and Jennings's, respectively.

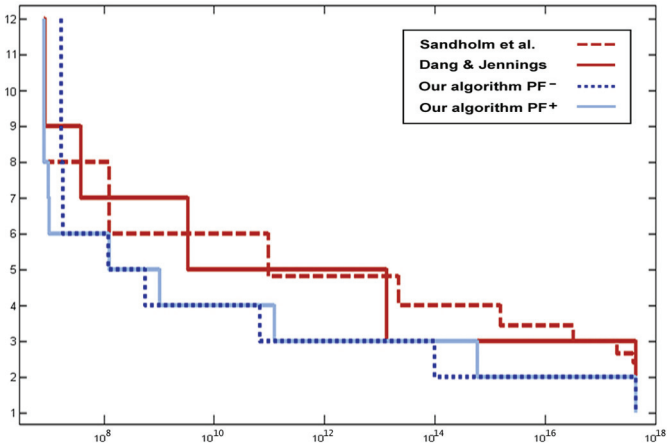

Figure 2: The ratio bound β as a function of the number of searched coalition structures (given 24 agents).

7.2 Evaluating IP$^{+/-}$

In order to evaluate our IP$^{+/-}$ algorithm, we need to be able to efficiently generate random instances of PF^+/PF^-. To this end, let us assume, without loss of generality, that the agents in every coalition are ordered ascendingly, and that the coalitions in any partition are also ordered ascendingly based on the smallest[11] agent in each coalition. Now, let $l(a_i, C)$ denote the location of agent a_i in C (e.g., $l(a_6, \{a_2, a_5, a_6, a_9\}) = 3$), and let $l(a_i, P)$ denote the location of the coalition in P that contains a_i (e.g., $l(a_6, \{\{a_1, a_7\}, \{a_3, a_5, a_6\}, \{a_4, a_9\}\}) = 2$). With these definitions in place, we now show how to generate the coalition values such that the PF^- (PF^+) condition is met. At first, for every coalition C, we generate the following non-negative random values: v_C and e_C and

[11]For any two agents $a_i, a_j \in A$, we say that a_i is smaller than a_j if $i < j$.

$e_{C,1}, e_{C,2}, \cdots, e_{C,|\overline{C}|}$ such that $\sum_{j=1}^{|\overline{C}|} e_{C,j} = e_C$. After that, for any coalition structure $CS \ni C$, we set the value of C as follows:

$$v(C, CS) = v_C -(+) \sum_{a_i \in \overline{C}} e_{C,l(a_i,\overline{C})} * (1 - \frac{l(a_i, CS \backslash \{C\}) - 1}{|\overline{C}|}) \quad (10)$$

In more detail, v_C represents the value of coalition C in the absence of any externalities, while the remainder of the left hand side of (10) represents the externality induced upon C in CS. Note that this externality is always smaller than, or equal to, e_C. This comes from the fact that: $\forall a_i \in \overline{C}$: $l(a_i, CS \backslash \{C\}) \geq 1$, which implies that:

$$\begin{aligned} \sum_{a_i \in \overline{C}} e_{C,l(a_i,\overline{C})} * (1 - \tfrac{l(a_i,CS\backslash\{C\})-1}{|\overline{C}|}) &\leq \sum_{a_i \in \overline{C}} e_{C,l(a_i,\overline{C})} * (1 - \tfrac{1-1}{|\overline{C}|}) \\ &\leq \sum_{a_i \in \overline{C}} e_{C,l(a_i,\overline{C})} \\ &\leq e_c \end{aligned}$$

Theorem 5 *By setting the value of each coalition as per equation (10), we end up with a PF^- (PF^+) setting.*[12]

Next, we analyze the memory required to store the input. As opposed to [Michalak *et al.*, 2008], where a value must be stored for every pair $(C, CS) : C \in CS$, using equation (10) only requires storing v_c and $e_{C,1}, e_{C,2}, \cdots, e_{C,|\overline{C}|}$ for every coalition C. Thus, given n agents, the total number of values to be stored is computed as follows, where $n!$ is the factorial of n:

$$2^n + \sum_{s=1}^{n} \frac{n!}{(s-1)! \times (n-s)!}$$

This is significantly smaller, compared to the number of values stored by Michalak *et al.* (e.g. 0.000000007% given 25 agents).

In our experiments, v_C is generated using a normal distribution, while e_C is generated using a uniform distribution as follows: $e \sim U(a, b)$, where $a = 0$ and $b = v_C$.

Given 20 agents, Figure 3 shows *on a log scale* how the quality of our solution improves over the running time.[13] As can be seen, the solution quality grows very rapidly (e.g. it reaches 94% within only one second). Moreover, an optimal solution is reached within only 5% of the total running time, while the rest of the time is spent trying to verify that the found solution is indeed optimal.

8 Conclusions

The coalition structure generation problem is usually studied in the context of characteristic function games where no externalities are allowed. However, in many cases externalities do exist and need to be accounted for. In this paper, we presented a number of important results for such partition function games: we showed how to bound coalition values in games with positive or negative externalities, identified the minimum amount of search required to establish a worst-case bound from the optimal solution, developed a novel algorithm for improving that bound with further search, and showed that it outperforms existing approaches by orders of magnitude. By so doing, we have laid the foundation for further work on this important setting. In particular, we are keen to develop an efficient approach to systems that posses mixed characteristics and cannot be neatly categorized to only one class.

[12]The proof of Theorem 5 is omitted from the paper due to space limitations.

[13]The PC on which we ran our experiments had 4 processors (each is an Intel(R) Xeon(R) CPU @ 2.66 GHz), with 3GB of RAM.

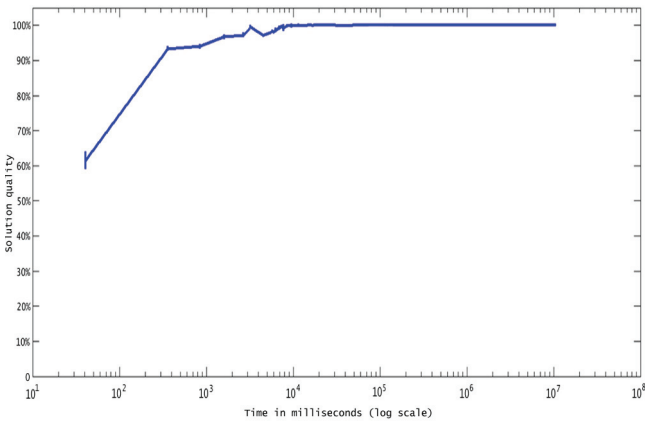

Figure 3: Solution quality over running time (given 20 agents).

References

[Dang and Jennings, 2004] V. D. Dang and N. R. Jennings. Generating coalition structures with finite bound from the optimal guarantees. In *Proceedings of the Third International Conference on Autonomous Agents and Multi Agent Systems (AAMAS-04)*, pages 564–571, 2004.

[Michalak *et al.*, 2008] T. Michalak, A. Dowell, P. McBurney, and M. Wooldridge. Optimal coalition structure generation in partition function games. In *Proceedings of the 18th European Conference on Artificial Intelligence (ECAI-08)*, pages 388–392, 2008.

[Rahwan and Jennings, 2008] T. Rahwan and N. R. Jennings. Coalition structure generation: Dynamic programming meets anytime optimisation. In *Proceedings of the 23rd Conference on Artificial Intelligence (AAAI-08)*, pages 156–161, 2008.

[Rahwan *et al.*, 2009] T. Rahwan, S. D. Ramchurn, A. Giovannucci, and N. R. Jennings. An anytime algorithm for optimal coalition structure generation. *Journal of Artificial Intelligence Research (JAIR)*, 34:521–567, 2009.

[Riha and James, 1974] W. Riha and K. R. James. Algorithm 29 efficient algorithms for doubly and multiply restricted partitions. *Journal of Computing*, 16:163–168, 1974.

[Sandholm *et al.*, 1999] T. W. Sandholm, K. Larson, M. Andersson, O. Shehory, and F. Tohme. Coalition structure generation with worst case guarantees. *Artificial Intelligence (AIJ)*, 111(1–2):209–238, 1999.

[Shehory and Kraus, 1998] O. Shehory and S. Kraus. Methods for task allocation via agent coalition formation. *Artificial Intelligence (AIJ)*, 101(1–2):165–200, 1998.