Artificial Intelligence 186 (2012) 95–122

Contents lists available at SciVerse ScienceDirect

Artificial Intelligence

www.elsevier.com/locate/artint

Anytime coalition structure generation in multi-agent systems with positive or negative externalities

Talal Rahwan [a,*,1], Tomasz Michalak [a,b,1], Michael Wooldridge [c], Nicholas R. Jennings [a,d]

[a] School of Electronics and Computer Science, University of Southampton, Southampton SO17 1BJ, UK
[b] Institute of Informatics, University of Warsaw, Warsaw 02-097, Poland
[c] Department of Computer Science, University of Liverpool, Liverpool L69 3BX, UK
[d] Department of Computing and Information Technology, King Abdulaziz University, Saudi Arabia

ARTICLE INFO

Article history:
Received 2 February 2011
Received in revised form 29 October 2011
Accepted 17 March 2012
Available online 29 March 2012

Keywords:
Mechanism design
Classification
Game theory
Approximation

ABSTRACT

Much of the literature on multi-agent coalition formation has focused on *Characteristic Function Games*, where the effectiveness of a coalition is not affected by how the other agents are arranged in the system. In contrast, very little attention has been given to the more general class of *Partition Function Games*, where the emphasis is on how the formation of one coalition could influence the performance of other co-existing coalitions in the system. However, these inter-coalitional dependencies, called *externalities from coalition formation*, play a crucial role in many real-world multi-agent applications where agents have either conflicting or overlapping goals.

Against this background, this paper is the first computational study of coalitional games with externalities in the multi-agent system context. We focus on the *Coalition Structure Generation* (*CSG*) problem which involves finding an exhaustive and disjoint division of the agents into coalitions such that the performance of the entire system is optimized. While this problem is already very challenging in the absence of externalities, due to the exponential size of the search space, taking externalities into consideration makes it even more challenging as the size of the input, given n agents, grows from $O(2^n)$ to $O(n^n)$.

Our main contribution is the development of the first CSG algorithm for coalitional games with either positive or negative externalities. Specifically, we prove that it is possible to compute upper and lower bounds on the values of any set of disjoint coalitions. Building upon this, we prove that in order to establish a worst-case guarantee on solution quality it is necessary to search a certain set of coalition structures (which we define). We also show how to progressively improve this guarantee with further search.

Since there are no previous CSG algorithms for games with externalities, we benchmark our algorithm against other state-of-the-art approaches in games where no externalities are present. Surprisingly, we find that, as far as worst-case guarantees are concerned, our algorithm outperforms the others by orders of magnitude. For instance, to reach a bound of 3 given 24 agents, the number of coalition structures that need to be searched by our algorithm is only 0.0007% of that needed by Sandholm et al. (1999) [1], and 0.5% of that needed by Dang and Jennings (2004) [2]. This is despite the fact that the other algorithms take advantage of the special properties of games with no externalities, while ours does not.

* Corresponding author.
E-mail address: tr@ecs.soton.ac.uk (T. Rahwan).
[1] Both Talal Rahwan and Tomasz Michalak are principle authors of this paper.

0004-3702/$ – see front matter
http://dx.doi.org/10.1016/j.artint.2012.03.007

1. Introduction

The ability to create effective coalitions is one of the key goals of multi-agent systems research. *Coalitional games* are models that capture opportunities for cooperation by explicitly modeling the ability of the agents to take joint actions as primitives [3]. In this context, one of the key challenges is to generate a *coalition structure*, i.e., an exhaustive and disjoint division of the set of agents, such that the performance of the entire system is optimized. This *Coalition Structure Generation* (*CSG*) problem has received much attention in the multi-agent system literature [4,1,5,2,6].

To date, work on the CSG problem in the AI community has focused primarily on a class of coalitional games known as *Characteristic Function Games* (*CFG*s), where the effectiveness, or *value*, of a coalition is not affected by the way non-members are partitioned. There are, however, many settings where this assumption does not hold. Such settings are known as *Partition Function Games*, or *PFG*s [7]. Specifically, in *PFG*s a coalition's value may be influenced by the formation of another coalition in the system. For instance, given two coalition structures: $CS = \{C_1, C_2, C_3\}$ and $CS' = \{C_1, C_2 \cup C_3\}$, the value of C_1 may be different in CS than in CS' due to the merger of C_2 with C_3. Such an effect is known as an *externality from coalition formation* and, in this example, it is induced upon C_1 by the merge of C_2 and C_3, which leads to the formation of coalition $C_2 \cup C_3$.[2]

Games with externalities have been widely studied in economics and other social sciences, where interdependencies between coalitions play an important role. Examples include collusion in oligopolies, where cooperating companies seek to undermine the competitive position of other firms in the market, as well as various forms of international (macro-economic/environmental) policy coordination between countries [8,9]. For instance, when high-tech companies decide to cooperate in order to develop a new technology standard, other companies lose some of their competitive position, i.e., they are subject to *negative externalities*. For instance, in the 1970's, Japanese firms established the Very Large Scale Integration (VLSI) consortium as well as the Fifth Generation Computer Systems (FGCS) project [10]. Another example is the decision by one group of countries to reduce pollution, which has a positive impact on other countries or regions, i.e., it induces *positive externalities* (see, e.g., Finus [11], Yi [10]).

The issue of externalities is also becoming increasingly important in domains in which multi-agent system techniques are applied. In e-commerce, for example, the British company, Aerogistics,[3] enables small- and medium-size aircraft component manufacturers and service providers to form online, *ad hoc* supply-chain coalitions to bid for manufacturing projects too large for any individual participant. Since all components must ultimately conform to the same standards, the cost of standarization procedures incurred by any coalition depends on the number and structure of other winning coalitions.

In many multi-agent systems, negative externalities between a coalition and non-members can be caused by sharing resources [12,1]. Thus, if agents, after forming a coalition, consume more resources than before, then this means that fewer resources are now available to the other coalitions. This is the case, for instance, in congestion games [13]. Negative externalities can also be caused by conflicting goals. Intuitively, by satisfying its own goals, a coalition may actually move the world further from the other coalitions' goals [14]. Conversely, overlapping or partially overlapping goals may cause positive externalities as some coalitions may satisfy goals of other coalitions [1].

In spite of so many important applications, very little attention has been given to the computational aspects of games with externalities. In particular, there exist no algorithms in the literature to solve the CSG problem in *PFG*s. To this end, it should be noted that the space of possible solutions to the CSG problem is the same for both *CFG*s and *PFG*s, with $O(n^n)$ solutions for n agents. The main difference, however, lies in the size of the input, which is $O(2^n)$ for *CFG*s, but $O(n^n)$ for *PFG*s. This is because, for every coalition, the input contains a single value — representing its performance — in the *CFG* case, while in the *PFG* case it contains as many values as there are possible partitions of the agents outside that coalition (see Section 2 for more details). This makes the CSG problem significantly more challenging compared to the *CFG* case, which is already NP-complete [1]. In fact, it can easily be shown that for the general case of *PFG*s it is not possible to solve the *CSG* problem without examining every possible coalition structure. Intuitively, even if all but one have been examined, this last one may be arbitrarily better than the rest.

Against this background, we focus on two important classes of *PFG*s:

- PFG^+ — coalitional games with weakly positive externalities. In this class, externalities are always non-negative, i.e., the merge of any two coalitions is never detrimental to other coalitions in the system.
- PFG^- — coalitional games with weakly negative externalities. Here, all externalities are non-positive, i.e., the merge of any two coalitions is never beneficial to other coalitions in the system.[4]

It should be stressed that nearly all the examples of coalitional games with externalities that are listed above belong to either one or the other class! Specifically, cartels, environmental policy coordination between countries, and multi-agent systems with partially overlapping goals are all games with positive externalities. That is, they belong to the class PFG^+. Conversely, collusion in oligopolies, exogenous coalition formation in e-market places, as well as multi-agent systems with shared resources and/or conflicting goals all belong to the PFG^- class.

[2] In the remainder of this paper we will refer to coalitional games as, simply, games, and to externalities from coalition formation as, simply, externalities.

[3] See `www.aerogistics.com` for more details.

[4] Throughout the paper, we will refer to weakly positive (negative) externalities as simply positive (negative) externalities.

Our aim in this paper is to develop *anytime* techniques to solve the CSG problem in PFG^+ and PFG^-, henceforth denoted PFG^+/PFG^-. To this end, an algorithm is deemed "anytime" if it can return a solution at any point in time during its execution, and the quality of its solution improves monotonically until termination. This is particularly desirable in the multi-agent system context since the agents might not always have sufficient time to run the algorithm to completion, especially when the search space of the problem at hand is of exponential size. Moreover, being anytime makes the algorithm more robust against failure; if the execution is stopped before the algorithm would have normally terminated, then it would still provide the agents with a solution that is better than the initial solution, or any other intermediate one.

To this end, one of the major obstacles when developing algorithms for PFG^+/PFG^- is how these algorithms can be tested/evaluated. This is significantly more challenging — compared to the case of games with no externalities — due to two main challenges:

- The first is how to generate random input instances that are guaranteed to satisfy the positive/negative externality condition. To this end, note that, in the general *CFG* case, there are no conditions that have to be considered when generating input instances. Thus, the characteristic function can be sampled from any distribution, e.g., Uniform, Normal, etc. On the other hand, when dealing with $PFG^+(PFG^-)$, one has to ensure that only positive (negative) externalities occur whenever two coalitions merge. This means, in particular, that for any two arbitrary coalition structures, CS and CS', such that CS' can be created from CS by a series of coalition merges, the value of any coalition not involved in these merges must not be smaller (greater) in CS' than in CS.
- The second challenge is how different input instances can be stored in memory. This is particularly needed to compare different algorithms or settings. As mentioned above, the size of the partition function quickly becomes prohibitive with a growing number of agents. For instance, given 20 agents, the partition function consists of more than 4×10^{14} values, and these require 394 terabytes of memory. What would be desirable, then, is to be able to store partition functions more concisely.

Taking all the above into account, our main contributions are as follows:

- We develop the first CSG algorithm for PFG^+/PFG^-, which we call $IP^{+/-}$. The building blocks of this algorithm can be summarized as follows:
  - (i) We prove that it is possible to compute upper and lower bounds on the values of any set of disjoint coalitions in PFG^+/PFG^-. These bounds can be used to identify unpromising coalition structures and, hence, improve the efficiency of CSG algorithms;
  - (ii) We prove that in order to establish a worst-case guarantee on solution quality in PFG^+/PFG^- it is necessary to search a certain set of coalition structures;
  - (iii) We identify subsets of coalition structures that need to be searched — in a certain order — so that the worst-case guarantee is proven to drop after each subset;
  - (iv) In the general *PFG* case, it is not possible to prune any partition of a subset of agents, even if the upper bound of that partition is lower than the lower bound of another partition of that subset. Yet, we show that in PFG^+/PFG^- such pruning is possible under specific conditions. Based on this result, we develop a pre-processing technique to prune unpromising coalition structures from the subsets that need to be searched;
  - (v) In order to search through promising subsets, we extend the depth-first search algorithm of Rahwan et al. [6] so that it is applicable to PFG^+/PFG^-.
- We provide an equation for generating random input instances, and prove that the resulting partition function is guaranteed to satisfy the conditions of PFG^+/PFG^-. We also prove that this function, which consists of $O(n^n)$ values, only requires storing $O(2^n)$ values in memory. This function can be used as a standard benchmark for evaluating any potential CSG algorithms that could be developed in the future for PFG^+/PFG^-.

Since there are no previous CSG algorithms for games with externalities, we benchmark our algorithm against the state-of-the-art algorithms in games where no externalities are present. We find that, as far as worst-case guarantees are concerned, our algorithm outperforms the other algorithms by orders of magnitude. For instance, to reach a bound of 3 given 24 agents, the number of coalition structures that need to be searched by our algorithm is only 0.0007% of that needed by Sandholm et al. [1], and 0.5% of that needed by Dang and Jennings [2]. This is despite the fact that the other algorithms take advantage of the special properties of games with no externalities, while ours does not.

The remainder of the paper is organized as follows. In Section 2, we introduce basic concepts related to coalitional games with and without externalities. In Section 3, we describe the algorithms that are currently available for solving the coalition structure generation problem, and discuss their relative advantages and limitations. In Section 4, we describe the CSG algorithm in PFG^+/PFG^-. Empirical evaluation are provided in Section 5. Section 6 concludes the paper and outlines future work. Finally, we provide in Appendix A a summary of the main notations used throughout this paper.

2. Preliminaries

In this section we formally introduce the notions of coalitions and coalition structures (Section 2.1), games with no externalities (Section 2.2), and games with externalities (Section 2.3). Finally, we formalize the CSG problem in coalitional games, and provide a list of the important notations used in this paper (Section 2.4).

2.1. Coalitions and coalition structures

Throughout the paper, we denote by n the number of agents, and by $A = \{a_1, a_2, \ldots, a_n\}$ the set of agents. We assume that every non-empty subset of A is a possible coalition, and denote by $\mathcal{C}$ the set of coalitions. More formally, $\mathcal{C} = \{C\colon\ C \subseteq A, C \neq \emptyset\}$. This implies that $|\mathcal{C}| = 2^n - 1$.

Example 1. There are 15 coalitions that can be created from $A = \{a_1, a_2, a_3, a_4\}$: $\{a_1\}$, $\{a_2\}$, $\{a_3\}$, $\{a_4\}$, $\{a_1, a_2\}$, $\{a_1, a_3\}$, $\{a_1, a_4\}$, $\{a_2, a_3\}$, $\{a_2, a_4\}$, $\{a_3, a_4\}$, $\{a_1, a_2, a_3\}$, $\{a_1, a_2, a_4\}$, $\{a_1, a_3, a_4\}$, $\{a_2, a_3, a_4\}$, and $\{a_1, a_2, a_3, a_4\}$.

A formal definition of a coalition structure is as follows:

Definition 1 *(Coalition structure).* A *coalition structure*, denoted $CS = \{C_1, C_2, \ldots, C_{|CS|}\}$, is an exhaustive partition of A into disjoint coalitions, i.e., it satisfies the following conditions: (a) $\forall i \in \{1, \ldots, |CS|\}, C_i \neq \emptyset$; (b) $\bigcup_{i=1}^{|CS|} C_i = A$; and (c) $\forall i, j \in \{1, \ldots, |CS|\}\colon\ i \neq j, C_i \cap C_j = \emptyset$.

The set of all coalition structures will be denoted as $\mathcal{P}^A$, and the set of coalition structures containing exactly s coalitions will be denoted as $\mathcal{P}^A_s$. The number of possible coalition structures — also known as the nth Bell number given n agents — is computed as: $|\mathcal{P}^A| = \sum_{s=1}^{n} |\mathcal{P}^A_s|$, where:

$$\left|\mathcal{P}^A_s\right| = \frac{1}{i!}\sum_{j=0}^{s-1}(-1)^j\binom{s}{j}(s-j)^n \tag{1}$$

Example 2. In total, there are 15 coalition structures that can be created from $A = \{a_1, a_2, a_3, a_4\}$[5]:

$$\begin{array}{lll}
\{\{a_1\}\{a_2\}\{a_3\}\{a_4\}\} & \{\{a_1\}\{a_2, a_4\}\{a_3\}\} & \{\{a_1\}\{a_2, a_3, a_4\}\} \\
\{\{a_1, a_2\}\{a_3\}\{a_4\}\} & \{\{a_1\}\{a_2\}\{a_3, a_4\}\} & \{\{a_1, a_2\}\{a_3, a_4\}\} \\
\{\{a_1, a_3\}\{a_2\}\{a_4\}\} & \{\{a_1, a_2, a_3\}\{a_4\}\} & \{\{a_1, a_3\}\{a_2, a_4\}\} \\
\{\{a_1, a_4\}\{a_2\}\{a_3\}\} & \{\{a_1, a_2, a_4\}\{a_3\}\} & \{\{a_1, a_4\}\{a_2, a_3\}\} \\
\{\{a_1\}\{a_2, a_3\}\{a_4\}\} & \{\{a_1, a_3, a_4\}\{a_2\}\} & \{\{a_1, a_2, a_3, a_4\}\}
\end{array}$$

Observe that the difference between $|\mathcal{C}|$ and $|\mathcal{P}^A|$ grows exponentially with the number of agents. An example is shown in the following table:

n	1	3	5	7	9	11	13	15
$\lvert\mathcal{C}\rvert$	1	7	31	127	511	2047	8195	32 783
$\lvert\mathcal{P}^A\rvert$	1	5	52	877	21 147	678 570	27 644 437	1 382 958 545

2.2. Characteristic function games

Conventionally, coalitional games with no externalities are modeled using the *characteristic function*, which takes, as an input, a coalition $C \in \mathcal{C}$ and outputs its value, $v(C) \in \mathbb{R}$, which reflects the performance of this coalition. Formally: $v : \mathcal{C} \to \mathbb{R}$. A tuple (A, v) is called a *Characteristic Function Game* (*CFG*). It is clear from the above definition that in this type of game the value of any coalition is independent of any other coalition. As standard in the literature (e.g., [1,2,6]) we assume that $v(C) \geqslant 0$ for all $C \in \mathcal{C}$.

[5] For notational convenience, commas are omitted between coalitions whenever there is no risk of confusion.

Example 3. Let a sample characteristic function for $A = \{a_1, a_2, a_3, a_4\}$ be defined as follows:

$$v(\{a_1\}) = 1 \quad v(\{a_1, a_2\}) = 3 \quad v(\{a_2, a_4\}) = 6 \quad v(\{a_1, a_3, a_4\}) = 6$$
$$v(\{a_2\}) = 2 \quad v(\{a_1, a_3\}) = 4 \quad v(\{a_3, a_4\}) = 4 \quad v(\{a_2, a_3, a_4\}) = 8$$
$$v(\{a_3\}) = 2 \quad v(\{a_1, a_4\}) = 1 \quad v(\{a_1, a_2, a_3\}) = 5 \quad v(\{a_1, a_2, a_3, a_4\}) = 7$$
$$v(\{a_4\}) = 1 \quad v(\{a_2, a_3\}) = 5 \quad v(\{a_1, a_2, a_4\}) = 5$$

2.3. Partition function games

In games with externalities, the value of a coalition depends on the way other agents are partitioned. A coalition C that is part of a coalition structure CS will be called an *embedded coalition* and denoted (C, CS). The set of all embedded coalitions for the set A will be denoted $\mathcal{EC}$. For any embedded coalition $(C, CS) \in \mathcal{EC}$, the *partition function* w outputs a value that reflects the performance of C in CS. Formally: $w : \mathcal{EC} \to \mathbb{R}$.

Example 4. Let a sample partition function for $A = \{a_1, a_2, a_3, a_4\}$ be:

$$w(\{a_1\}, \{\{a_1\}\{a_2\}\{a_3\}\{a_4\}\}) = 1 \quad w(\{a_4\}, \{\{a_1, a_2, a_3\}\{a_4\}\}) = 0$$
$$w(\{a_1\}, \{\{a_2, a_3\}\{a_1\}\{a_4\}\}) = 1 \quad w(\{a_1, a_2\}, \{\{a_1, a_2\}\{a_3\}\{a_4\}\}) = 3$$
$$w(\{a_1\}, \{\{a_2, a_4\}\{a_1\}\{a_3\}\}) = 0 \quad w(\{a_1, a_2\}, \{\{a_1, a_2\}\{a_3, a_4\}\}) = 2$$
$$w(\{a_1\}, \{\{a_3, a_4\}\{a_1\}\{a_2\}\}) = 0.5 \quad w(\{a_1, a_3\}, \{\{a_1, a_3\}\{a_2\}\{a_4\}\}) = 4$$
$$w(\{a_1\}, \{\{a_1\}\{a_2, a_3, a_4\}\}) = 0 \quad w(\{a_1, a_3\}, \{\{a_1, a_3\}\{a_2, a_4\}\}) = 3$$
$$w(\{a_2\}, \{\{a_1\}\{a_2\}\{a_3\}\{a_4\}\}) = 2 \quad w(\{a_1, a_4\}, \{\{a_1, a_4\}\{a_2\}\{a_3\}\}) = 1$$
$$w(\{a_2\}, \{\{a_3, a_4\}\{a_1\}\{a_2\}\}) = 1 \quad w(\{a_1, a_4\}, \{\{a_1, a_4\}\{a_2, a_3\}\}) = 1$$
$$w(\{a_2\}, \{\{a_1, a_3\}\{a_2\}\{a_4\}\}) = 2 \quad w(\{a_2, a_3\}, \{\{a_2, a_3\}\{a_1\}\{a_4\}\}) = 5$$
$$w(\{a_2\}, \{\{a_1, a_4\}\{a_2\}\{a_3\}\}) = 1 \quad w(\{a_2, a_3\}, \{\{a_2, a_3\}\{a_1, a_4\}\}) = 4$$
$$w(\{a_2\}, \{\{a_1, a_3, a_4\}\{a_2\}\}) = 0 \quad w(\{a_2, a_4\}, \{\{a_2, a_4\}\{a_1\}\{a_3\}\}) = 6$$
$$w(\{a_3\}, \{\{a_1\}\{a_2\}\{a_3\}\{a_4\}\}) = 2 \quad w(\{a_2, a_4\}, \{\{a_2, a_4\}\{a_1, a_3\}\}) = 4$$
$$w(\{a_3\}, \{\{a_1, a_2\}\{a_3\}\{a_4\}\}) = 1 \quad w(\{a_3, a_4\}, \{\{a_3, a_4\}\{a_1\}\{a_2\}\}) = 4$$
$$w(\{a_3\}, \{\{a_1, a_4\}\{a_2\}\{a_3\}\}) = 2 \quad w(\{a_3, a_4\}, \{\{a_3, a_4\}\{a_1, a_2\}\}) = 3$$
$$w(\{a_3\}, \{\{a_2, a_4\}\{a_1\}\{a_3\}\}) = 1 \quad w(\{a_1, a_2, a_3\}, \{\{a_1, a_2, a_3\}\{a_4\}\}) = 5$$
$$w(\{a_3\}, \{\{a_1, a_2, a_4\}\{a_3\}\}) = 0 \quad w(\{a_1, a_2, a_4\}, \{\{a_1, a_2, a_4\}\{a_3\}\}) = 5$$
$$w(\{a_4\}, \{\{a_1\}\{a_2\}\{a_3\}\{a_4\}\}) = 1 \quad w(\{a_1, a_3, a_4\}, \{\{a_1, a_3, a_4\}\{a_2\}\}) = 6$$
$$w(\{a_4\}, \{\{a_2, a_3\}\{a_1\}\{a_4\}\}) = 0.5 \quad w(\{a_2, a_3, a_4\}, \{\{a_2, a_3, a_4\}\{a_1\}\}) = 8$$
$$w(\{a_4\}, \{\{a_1, a_3\}\{a_2\}\{a_4\}\}) = 0.5 \quad w(\{a_1, a_2, a_3, a_4\}, \{\{a_1, a_2, a_3, a_4\}\}) = 7$$
$$w(\{a_4\}, \{\{a_1, a_2\}\{a_3\}\{a_4\}\}) = 1$$

A tuple (A, w) is called a *Partition Function Game* (*PFG*). Whenever convenient, we will use a concise notation to represent partition functions, where values of all coalitions embedded in a given coalition structure are represented as a vector. Formally, given a coalition structure $CS = \{C_1, \ldots, C_{|CS|}\}$, the values of $(C_1, CS), \ldots, (C_{|CS|}, CS)$ will be denoted as $[w(C_1, CS), \ldots, w(C_{|CS|}, CS)]$.

Example 5. A shorthand notation for the partition function from Example 4:

Coalition structure	Vector	Coalition structure	Vector
$\{\{a_1\}\{a_2\}\{a_3\}\{a_4\}\}$	$[1, 2, 2, 1]$	$\{\{a_1, a_3\}\{a_2, a_4\}\}$	$[3, 4]$
$\{\{a_1, a_2\}\{a_3\}\{a_4\}\}$	$[3, 1, 1]$	$\{\{a_1, a_4\}\{a_2, a_3\}\}$	$[1, 4]$
$\{\{a_1, a_3\}\{a_2\}\{a_4\}\}$	$[4, 2, 0.5]$	$\{\{a_1, a_2, a_3\}\{a_4\}\}$	$[5, 0]$
$\{\{a_1, a_4\}\{a_2\}\{a_3\}\}$	$[1, 1, 2]$	$\{\{a_1, a_2, a_4\}\{a_3\}\}$	$[5, 0]$
$\{\{a_1\}\{a_2, a_3\}\{a_4\}\}$	$[1, 5, 0.5]$	$\{\{a_1, a_3, a_4\}\{a_2\}\}$	$[6, 0]$
$\{\{a_1\}\{a_2, a_4\}\{a_3\}\}$	$[0, 6, 1]$	$\{\{a_1\}\{a_2, a_3, a_4\}\}$	$[0, 8]$
$\{\{a_1\}\{a_2\}\{a_3, a_4\}\}$	$[0.5, 1, 4]$	$\{\{a_1, a_2, a_3, a_4\}\}$	$[7]$
$\{\{a_1, a_2\}\{a_3, a_4\}\}$	$[2, 3]$		

The partition function takes into account any *externality from coalition formation*. By this we mean a change in a value of a given coalition caused by a merge of two other co-existing coalitions.[6] More formally:

[6] For a discussion of alternative notions of externalities in multi-agents systems see [15], and for a comprehensive economic study of externalities and related issues see [16].

Definition 2 *(Externalities).* Let $CS, CS' \in \mathcal{P}^A$ be two coalition structures such that $|CS'| \geqslant 3$ and there exist $C', C'' \in CS'$ such that $CS = \{C' \cup C''\} \cup CS' \setminus \{C', C''\}$. Then, the formation of $(C' \cup C'', CS)$ from (C', CS') and (C'', CS') imposes an externality on every other embedded coalition (C, CS) such that $C \neq C' \cup C''$, which is defined as:

$$\epsilon(C, CS, CS') = w(C, CS) - w(C, CS')$$

Thus, every externality is uniquely determined by a tuple (C, CS, CS'). A set of all such tuples in a game (A, w) will be denoted by $\mathcal{T}(w)$.

Now, we can formally define games with positive externalities (PFG^+) and games with negative externalities (PFG^-):

Definition 3 *(Game with positive (negative) externalities).* We say that partition function game (A, w) is characterized by positive (negative) externalities if and only if, for all $(C, CS, CS') \in \mathcal{T}(w)$, the following holds:

$$\epsilon(C, CS, CS') \quad \geqslant (\leqslant) \quad 0$$

Example 6. The game defined in Example 4 has only negative externalities. For example, the externalities affecting coalition $\{a_1\}$ are:

$$\begin{aligned}
&\epsilon\big(\{a_1\}, \big\{\{a_1\}\{\mathbf{a_2}, \mathbf{a_3}\}\{a_4\}\big\}, \big\{\{a_1\}\{\mathbf{a_2}\}\{\mathbf{a_3}\}\{a_4\}\big\}\big)\\
&\quad = w\big(\{a_1\}, \big\{\{a_1\}\{a_2, a_3\}\{a_4\}\big\}\big) - w\big(\{a_1\}, \big\{\{a_1\}\{a_2\}\{a_3\}\{a_4\}\big\}\big) = 0;\\
&\epsilon\big(\{a_1\}, \big\{\{a_1\}\{\mathbf{a_2}, \mathbf{a_4}\}\{a_3\}\big\}, \big\{\{a_1\}\{\mathbf{a_2}\}\{a_3\}\{\mathbf{a_4}\}\big\}\big)\\
&\quad = w\big(\{a_1\}, \big\{\{a_1\}\{a_2, a_4\}\{a_4\}\big\}\big) - w\big(\{a_1\}, \big\{\{a_1\}\{a_2\}\{a_3\}\{a_4\}\big\}\big) = -1;\\
&\epsilon\big(\{a_1\}, \big\{\{a_1\}\{a_2\}, \{\mathbf{a_3}, \mathbf{a_4}\}\big\}, \big\{\{a_1\}\{a_2\}\{\mathbf{a_3}\}\{\mathbf{a_4}\}\big\}\big)\\
&\quad = w\big(\{a_1\}, \big\{\{a_1\}\{a_2\}\{a_3, a_4\}\big\}\big) - w\big(\{a_1\}, \big\{\{a_1\}\{a_2\}\{a_3\}\{a_4\}\big\}\big) = -0.5;\\
&\epsilon\big(\{a_1\}, \big\{\{a_1\}\{\mathbf{a_2}, \mathbf{a_3}, \mathbf{a_4}\}\big\}, \big\{\{a_1\}\{\mathbf{a_2}, \mathbf{a_3}\}\{\mathbf{a_4}\}\big\}\big)\\
&\quad = w\big(\{a_1\}, \big\{\{a_1\}\{a_2, a_3, a_4\}\big\}\big) - w\big(\{a_1\}, \big\{\{a_1\}\{a_2, a_3\}\{a_4\}\big\}\big) = -1;\\
&\epsilon\big(\{a_1\}, \big\{\{a_1\}\{\mathbf{a_2}, \mathbf{a_3}, \mathbf{a_4}\}\big\}, \big\{\{a_1\}\{\mathbf{a_2}, \mathbf{a_4}\}\{\mathbf{a_3}\}\big\}\big)\\
&\quad = w\big(\{a_1\}, \big\{\{a_1\}\{a_2, a_3, a_4\}\big\}\big) - w\big(\{a_1\}, \big\{\{a_1\}\{a_2, a_4\}\{a_3\}\big\}\big) = 0;\\
&\epsilon\big(\{a_1\}, \big\{\{a_1\}\{\mathbf{a_2}, \mathbf{a_3}, \mathbf{a_4}\}\big\}, \big\{\{a_1\}\{\mathbf{a_2}\}\{\mathbf{a_3}, \mathbf{a_4}\}\big\}\big)\\
&\quad = w\big(\{a_1\}, \big\{\{a_1\}\{a_2, a_3, a_4\}\big\}\big) - w\big(\{a_1\}, \big\{\{a_1\}\{a_2\}\{a_3, a_4\}\big\}\big) = -0.5.
\end{aligned}$$

It is not difficult to check that externalities affecting other coalitions in the game are also non-positive.

Observe that *CFG*s are a special case of *PFG*s where every externality equals zero. They are also a special case of PFG^+/PFG^-. This means, in particular, that any available algorithm designed for games with no externalities cannot be directly applied to those with externalities. Conversely, the algorithms that are proposed in this paper for PFG^+/PFG^- are directly applicable to *CFG*s.

2.4. The CSG problem formalized

The CSG problem is to find an optimal coalition structure $CS^* \in \mathcal{P}^A$, which is formally defined for *CFG*s as follows:

$$CS^* = \arg\max_{CS \in \mathcal{P}^A} \sum_{C \in CS} v(C),$$

while, for *PFG*s, it is defined as follows:

$$CS^* = \arg\max_{CS \in \mathcal{P}^A} \sum_{C \in CS} w(C, CS).$$

Example 7. Referring to the game with no externalities defined in Example 3, we have $CS^* = \{\{a_1, a_3\}, \{a_2, a_4\}\}$. This is because $v(\{a_1, a_3\}) + v(\{a_2, a_4\}) = 10$ which is greater than the value of any other coalition structure in this game. As for the game with externalities defined in Example 4, $CS^* = \{\{a_1\}\{a_2, a_3, a_4\}\}$ since $w(\{a_1\}, CS^*) + w(\{a_2, a_3, a_4\}, CS^*) = 8$ is greater than the value of any other coalition structure in this game.

3. Related work

In this section we will briefly discuss the available CSG algorithms for games with no externalities (i.e., *CFG*s). Broadly, these can be divided into exact and non-exact algorithms [6]. Basically, the non-exact algorithms return "good" solutions relatively quickly, and scale up well with the increase in the number of agents involved (see, e.g., Shehory and Kraus [4], Sen and Dutta [5]). However, they provide no guarantees on the quality of their solutions. Exact algorithms, on the other hand, are guaranteed to find an optimal coalition structure. In this paper, we are interested in finding the optimal coalition structure and so we focus solely on exact algorithms. To this end, such algorithms are based on two main techniques:

- Anytime optimization — The basic idea is to generate an initial solution that is guaranteed to be within a finite bound from the optimal one. After that, more coalition structures are examined so as to improve the solution and bound quality, until an optimal solution is found (see, e.g., Sandholm et al. [1], Dang and Jennings [2], Rahwan et al. [17,6]). Although anytime CSG algorithms might, in the worst case, end up searching the entire space (i.e., they run in $O(n^n)$ time), they are robust against failure; if the execution is stopped before the algorithm would normally have terminated, then it can still return a solution that is better than the initial, or any other intermediate, one.
- Dynamic Programming — The basic idea of dynamic programming is to break the optimization problem into sub-problems that can be solved recursively, and then combine the results to solve the original problem, thereby avoiding the work of recomputing the answer every time the sub-problem is encountered (see, e.g., Yeh [18], Rothkopf et al. [19], Rahwan and Jennings [20]). The advantage of dynamic programming algorithms, compared to their anytime counterpart, is that they run in $O(3^n)$ time. However, the disadvantage is that they provide no interim solution before completion.

Recently, a number of algorithms have been developed that combine the above two techniques (see, e.g., Rahwan and Jennings [21], Service and Adams [22,23]).

In all the above algorithms, the focus was on settings where the coalition values are given explicitly, i.e., as a table of 2^n values. However, there is another line of research that focuses on solving the CSG problem given *concise representations* (see, e.g., Ohta et al. [24], Rahwan et al. [25], Ueda et al. [26]). However, this issue is beyond the scope of this paper.

Next, we will describe in detail some algorithms from the *CFG* literature (since we will use similar techniques later on in Section 4 to construct our *PFG* algorithm). To this end, observe, that dynamic programming techniques cannot be applied in *PFG*s. This is because they depend heavily on the *CFG* assumption that a coalition (or a group of disjoint coalitions) has the same value in any coalition structure containing it. This assumption makes it possible to re-use the same result in different coalition structures. In *PFG*s, however, this assumption does not hold since the externalities in one coalition structure can be different from that in another. Against this background, we will only describe the anytime algorithms that do not use dynamic programming techniques. Such algorithms can be divided into two categories, where the focus in the first is on the worst-case guarantee, and the focus in the second is on solution quality. The following two subsections describe these categories in more detail.

3.1. Algorithms that focus on worst-case guarantees

The main focus of such algorithm is on how to establish a worst-case ratio bound β on solution quality, i.e., how to identify a subset of coalition structures $\mathcal{P}' \subseteq \mathcal{P}^A$ such that:

$$\frac{\max_{CS \in \mathcal{P}^A} \sum_{C \in CS} w(C, CS)}{\max_{CS \in \mathcal{P}'} \sum_{C \in CS} w(C, CS)} \leqslant \beta$$

In more detail, these algorithms focus on dividing the search space into subsets, and identifying the sequence in which these subsets have to be searched so that β guaranteed to improve after each subset. The first anytime CSG algorithm was developed by Sandholm et al. [1]. Here, the space of coalition structures, i.e., $\mathcal{P}^A$, is represented as a *coalition structure graph*, where:

- Every node represents a coalition structure. All nodes are categorized into n levels, noted as $\mathcal{P}^A_1, \ldots, \mathcal{P}^A_n$ where level $\mathcal{P}^A_s$ is the set of coalition structures that contain exactly s coalitions;
- Every undirected edge connects two coalition structures, belonging to two consecutive levels, $\mathcal{P}^A_{s-1}$ and $\mathcal{P}^A_s$, such that the coalition structure in level $\mathcal{P}^A_{s-1}$ can be created from the one in level $\mathcal{P}^A_s$ by merging exactly two coalitions into one. In other words, an edge in the graph represents a merger of two coalitions when followed upwards, and a split of a coalition into two when followed downwards.

Fig. 1 shows an example of the coalition structure graph given 4 agents. Sandholm et al. [1] proved that there exists a minimum set of coalition structures that have to be searched in order to establish any bound on the quality of the solution. Specifically, this is achieved by searching through the first two levels of the coalition structure graph (which contain 2^{n-1} coalition structures in total) and the result of this search is within a ratio bound $\beta = n$ from the optimal.

If additional time is available after the first two levels have been searched, then it would be desirable to improve the bound with further search. To this end, Sandholm et al. proposed to search the remaining levels one by one, starting from

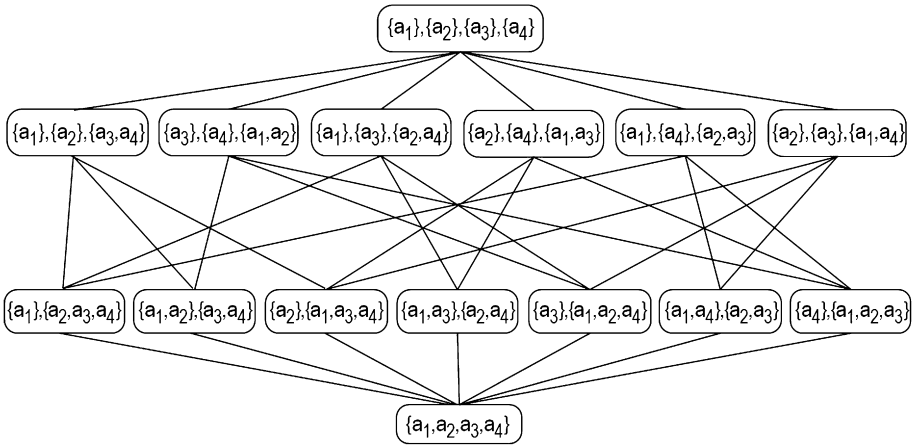

Fig. 1. The coalition structure graph for 4 agents.

the bottom level (i.e., $\mathcal{P}_n^A$), and moving upwards. They proved that the bound improves as the algorithm searches more levels. In more detail, assuming that the algorithm has just completed searching level $\mathcal{P}_s^A$, the bound becomes $\beta = \lfloor n/h \rfloor$, where $h = \lfloor (n-s)/2 \rfloor + 2$. The only exception is when $n \equiv h-1 \pmod h$ and $n \equiv s \pmod 2$, in which case the bound becomes $\beta = \lceil n/h \rceil$.

A different algorithm was proposed by Dang and Jennings [2]. This algorithm starts by searching the top two levels, as well as the bottom one (as Sandholm et al.'s algorithm does). After that, however, instead of searching through the remaining levels one by one (as Sandholm et al. do), the algorithm searches through certain subsets of all remaining levels. Specifically, it searches the coalition structures that have at least one coalition of which the size is not less than $\lceil n(d-1)/d \rceil$ (with d running from $\lfloor (n+1)/4 \rfloor$ down to 2). Dang and Jennings proved that, for any given value of d, the algorithm establishes a bound $\beta = 2d-1$.

As mentioned in the introduction, however, such algorithms only identify the coalition structures that need to be searched in order to further improve the ratio bound. They do not specify how the search process is carried out, and it is implicitly assumed that a simple brute-force procedure is applied.

3.2. Algorithms that focus on solution quality

Such algorithms focus on searching the space of coalition structures as quickly as possible, and not on reducing the worst-case ratio bound. The state-of-the-art algorithm in this category is the IP algorithm [17]. This algorithm is based on a novel representation of the search space that divides the coalition structures into subspaces based on the sizes of the coalitions they contain [27]. In more detail, a subspace is represented by an integer partition of n.[7] For example, given 4 agents, the possible integer partitions are [4], [1, 3], [2, 2], [1, 1, 2], [1, 1, 1, 1], and each of these represents a subspace containing all the coalition structures within which the coalition sizes match the parts of the integer partition. For example, [1, 1, 2] represents the subspace of all the coalition structures within which two coalitions are of size 1, and one coalition is of size 2. The *integer partition graph* is a graph where a node represents an integer partition, and an edge connects two integer partitions $I, I' \in \mathcal{I}^n$, where $|I| > |I'|$, if and only if there exist $i, j \in I$ such that $I' = (I \setminus \{i, j\}) \uplus \{i + j\}$ [21]. Fig. 2 shows an example of 4 agents.

What is significant about this representation is that, for every subspace, it is possible to compute upper and lower bounds on the value of the best solution that can be found in it. To this end, let Max_s and Avg_s be the maximum and the average values of the coalitions of size s respectively. Also, let $\mathcal{I}^n$ be the set of integer partitions of n, and $\mathcal{P}_I^A$ be the subspace that corresponds to the integer partition $I \in \mathcal{I}^n$. Then, for all $I \in \mathcal{I}^n$, it is possible to compute an upper bound UB_I^A on the value of the best solution in $\mathcal{P}_I^A$ as follows: $UB_I^A = \sum_{s \in I} Max_s$. Similarly, a lower bound LB_I^A on the value of the best solution in $\mathcal{P}_I^A$ can be computed as follows: $LB_I = \sum_{s \in I} Avg_s$.[8] These bounds are then used to establish worst-case guarantees on the quality of the best solution found so far, and to prune any subspace that has no potential to contain a solution better than the current best one. As for the remaining subspaces, IP searches them one at a time, unless a value is found that is higher than the upper bound of another subspace, in which case that subspace no longer needs to be searched. Searching a subspace is done using an efficient process that applies branch-and-bound techniques to avoid examining every solution in it. For more details on the IP algorithm, see [6].

[7] Recall that an integer partition of a positive integer number i consists of a set of positive integers (called "*parts*") that add up to exactly i [28]. For presentation clarity, we use square brackets throughout this paper (instead of curly ones) to represent integer partitions.

[8] Interestingly, Rahwan et al. [17] proved that this lower bound is actually the average value of all the solutions in $\mathcal{P}_I^A$.

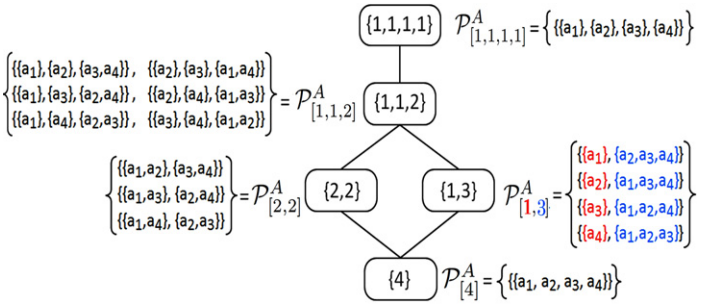

Fig. 2. The integer partition graph, and the subspaces represented by different nodes.

4. The $IP^{+/-}$ algorithm

In this section, we describe our CSG algorithm for PFG^+/PFG^-. The name $IP^{+/-}$ comes from the fact that the algorithm is based on the integer partition representation of the coalition structure space (see Section 3 for more details).

The remainder of this section is composed of five subsections that introduce the main building blocks of $IP^{+/-}$. Specifically, in Section 4.1, we show that it is possible to compute upper and lower bounds on the values of coalitions and sets of disjoint coalitions. In Section 4.2, we prove that there exists a certain set of coalition structures that has to be searched in order to establish a bound from the optimal solution. In Section 4.3, we present a procedure that identifies the subspaces of coalition structures that need to be searched in order to improve the bound. In Section 4.4 we propose a pre-processing procedure that uses upper and lower bounds of subspaces to prune unpromising ones. Finally, in order to search the promising subspaces, we extend in Section 4.5 the breadth-first search technique of Rahwan et al. [6] so that it is applicable to PFG^+/PFG^-.

4.1. Computing upper and lower bounds

We define the *value of a coalition structure* $CS \in \mathcal{P}^A$, denoted $W(CS)$, as the sum of the values of all the coalitions in CS, that is, $W(CS) = \sum_{C \in CS} w(C, CS)$. Moreover, for any coalition $C \in \mathcal{C}$, we denote by $\bar{C}$ the agents in A that do not belong to C (i.e., $\bar{C} = A \backslash C$). Furthermore, we define a *partition of* C as a set containing disjoint coalitions of which the union equals C, and denote the set of all such partitions as $\mathcal{P}^C$.[9] While every element of a partition $P \in \mathcal{P}^C$ is a coalition, where there is no risk of confusion, we will denote such an element by small letter p for notational convenience. Finally, for any coalition structure CS, the *value of a partition* $P \subseteq CS$, denoted $W(P, CS)$, is defined as the sum of the values of all the coalitions in that partition, i.e., $W(P, CS) = \sum_{p \in P} w(p, CS)$.

Now, we have the following theorem:

Theorem 1. *Consider a PFG^- (PFG^+) setting. Given a coalition $C \subseteq \mathcal{C}$, a partition $P \in \mathcal{P}^C$, and a coalition structure $CS \supseteq P$, the following holds, where the agents in $\bar{C}$ are denoted as $\bar{a}_1, \ldots, \bar{a}_{|\bar{C}|}$:*

$$W\left(P, \{\bar{C}\} \cup P\right) \quad \leqslant (\geqslant) \quad W(P, CS) \quad \leqslant (\geqslant) \quad W\left(P, \left\{\{\bar{a}_1\}, \ldots, \{\bar{a}_{|\bar{C}|}\}\right\} \cup P\right)$$

Proof. To simplify notation, let $CS' = \{\bar{C}\} \cup P$ and $CS'' = \{\{\bar{a}_1\}, \ldots, \{\bar{a}_{|\bar{C}|}\}\} \cup P$. Also, without loss of generality, assume that $CS \neq CS'$ and $CS \neq CS''$. Then, given a PFG^- (PFG^+) setting, we first need to prove that:

$$W\left(P, CS'\right) \quad \leqslant (\geqslant) \quad W(P, CS) \tag{2}$$

Beginning with CS, it is always possible to reach CS' by performing multiple steps, each involving the merging of two coalitions into one. In each step, the coalitions in P remain unchanged, and due to negative (positive) externalities, their values can only decrease (increase). As a result, the inequality in (2) holds. Similarly, beginning with CS'', it can be proved that the following holds: $W(P, CS) \leqslant (\geqslant)\ W(P, CS'')$. □

Theorem 1 bounds the value of *any given partition of* C. Specifically, for every partition $P \in \mathcal{P}^C$, the upper bound UB^P and lower bound LB^P can be computed in a PFG^- (PFG^+) setting as follows, where $\bar{C} = \{\bar{a}_1, \ldots, \bar{a}_{|\bar{C}|}\}$:

[9] Note that if $C = A$ then any partition of C would be a coalition structure.

$$LB^P\left(UB^P\right) = \sum_{p\in P} w\left(p, P\cup\{\bar{C}\}\right)$$

$$UB^P\left(LB^P\right) = \sum_{p\in P} w\left(p, P\cup\left\{\{\bar{a}_1\},\ldots,\{\bar{a}_{|\bar{C}|}\}\right\}\right)$$

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.

Now, let us denote by 2^C the power set of C, i.e., the set of all subsets of C. Moreover, let us denote by $\mathcal{I}^k$ the set of possible integer partitions of number $k\in\mathbb{N}$. Then, given a coalition $C\in\mathcal{C}$, and an integer partition $I\in\mathcal{I}^s$: $s\leqslant|C|$, we will define $\mathcal{P}_I^C$ as the set consisting of every possible $P\subseteq 2^C$ such that the sizes of the subsets in P match the parts in I. Consider the following example:

Example 8. For $C=\{a_1,a_2,a_3,a_4\}$ and $I=[1,2]$ we have:

$$\mathcal{P}_{[1,2]}^{\{a_1,a_2,a_3,a_4\}} = \left\{\begin{array}{l} \{\{a_1\},\{a_2,a_3\}\},\ \{\{a_1\},\{a_2,a_4\}\},\ \{\{a_1\},\{a_3,a_4\}\},\\ \{\{a_2\},\{a_1,a_3\}\},\ \{\{a_2\},\{a_1,a_4\}\},\ \{\{a_2\},\{a_3,a_4\}\},\\ \{\{a_3\},\{a_1,a_2\}\},\ \{\{a_3\},\{a_1,a_4\}\},\ \{\{a_3\},\{a_2,a_4\}\},\\ \{\{a_4\},\{a_1,a_2\}\},\ \{\{a_4\},\{a_1,a_3\}\},\ \{\{a_4\},\{a_2,a_3\}\} \end{array}\right\}$$

Now, given Theorem 1, we can compute upper and lower bounds (denoted UB_I^C and LB_I^C respectively) on the values of the elements of $\mathcal{P}_I^C$ as follows:

$$\forall C\in\mathcal{C},\ \forall s\leqslant|C|,\ \forall I\in\mathcal{I}^s,\quad UB_I^C=\max_{P\in\mathcal{P}_I^C} UB^P,\quad\text{and}\quad LB_I^C=\min_{P\in\mathcal{P}_I^C} LB^P$$

Observe that $\mathcal{P}_I^C$ is a subset of $\mathcal{P}^C$ if and only if $I\in\mathcal{I}^{|C|}$. For example, $\mathcal{P}_{[1,1,2]}^{\{a_1,a_2,a_3,a_4\}}$ is a subset of $\mathcal{P}^{\{a_1,a_2,a_3,a_4\}}$, while $\mathcal{P}_{[1,2]}^{\{a_1,a_2,a_3,a_4\}}$ is not.

4.2. Establishing a worst-case bound

Having computed upper and lower bounds for coalition and partition values in the previous section, 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 general theorem:

Theorem 2. *Let us define $\mathcal{X}$, $\mathcal{Y}$, and $\mathcal{Z}$ as follows*:

- *$\mathcal{X}$ is a set of elements*;
- *$\mathcal{Y}$ is a set containing subsets of $\mathcal{X}$.*
- *$\mathcal{Z}$ is a set containing subsets of $\mathcal{X}$ such that every $y\in\mathcal{Y}$ can be partitioned using subsets from $\mathcal{Z}$.*

Furthermore, let us define ν and V as follows:

- *ν is a function*; *$\nu:\mathcal{X}\times\mathcal{Y}\to\mathbb{R}^+\cup\{0\}$. We call $\nu(x,y)$ the value of x in y, where $\nu(x,y)=0$ for all y and all $x\notin y$.*
- *V is a function defined for every $y\in\mathcal{Y}$ as follows*: *$V(y)=\sum_{x\in y}\nu(x,y)$. We call $V(y)$ the value of y.*

Then, for any $\mathcal{Y}'\subseteq\mathcal{Y}$, if the following holds:

$$\forall z\in\mathcal{Z},\ \exists y'\in\mathcal{Y}'\colon\quad z\subseteq y'\ \wedge\ \sum_{x\in z}\nu\left(x,y'\right)=\max_{y\in\mathcal{Y}}\sum_{x\in z}\nu(x,y) \tag{3}$$

We have:

$$\max_{y\in\mathcal{Y}} V(y)\leqslant\max_{y\in\mathcal{Y}}\|y\|^{\mathcal{Z}}*\max_{y'\in\mathcal{Y}'} V\left(y'\right)$$

where $\|y\|^{\mathcal{Z}}$ is defined as follows:

$$\|y\|^{\mathcal{Z}}=\min_{\mathcal{Z}'\subseteq\mathcal{Z}\colon\bigcup\mathcal{Z}'=y\wedge\forall z,z'\in\mathcal{Z}'\colon z\cap z'=\emptyset}\left|\mathcal{Z}'\right|$$

In other words, let us define $\|y\|^{\mathcal{Z}}$ as *the size of y with respect to $\mathcal{Z}$*, which is basically the minimum number of subsets from $\mathcal{Z}$ that partition the subset. For example, if $\mathcal{Z}$ contains the following four subsets: $\{x_1\}$, $\{x_2\}$, $\{x_4\}$ and $\{x_1,x_3\}$, then,

given $y = \{x_1, x_3, x_4\}$, we have $\|y\|^{\mathcal{Z}} = 2$. In this case, we say that the size of y with respect to $\mathcal{Z}$ is 2. Now, if every subset in $\mathcal{Z}$ appears *with its maximum value* in at least one of the subsets in $\mathcal{Y}'$, then the best subset in $\mathcal{Y}'$ is within a ratio bound from the best subset in $\mathcal{Y}$. This bound equals *the biggest subset in $\mathcal{Y}$ with respect to $\mathcal{Z}$*.

Proof. Let y^* be the best subset in $\mathcal{Y}$, i.e., $y^* = \arg\max_{y\in\mathcal{Y}} V(y)$. Moreover, let $\mathcal{Z}^*$ be the smallest partition of y^* in $\mathcal{Z}$. That is:

$$\mathcal{Z}^* = \{z_1^*, \ldots, z_{|\mathcal{Z}^*|}^*\} \subseteq \mathcal{Z}: \quad \bigcup \mathcal{Z}^* = y^*, z_i^* \cap z_{j\neq i}^* = \emptyset, \qquad |\mathcal{Z}^*| = \|y^*\|^{\mathcal{Z}}$$

Now, since $\mathcal{Z}^*$ is a partition of y^*, then we can write $V(y^*)$ as follows:

$$V(y^*) = \sum_{x\in z_1^*} \nu(x, y^*) + \cdots + \sum_{x\in z_{|\mathcal{Z}^*|}^*} \nu(x, y^*)$$

This, in turn, implies that:

$$V(y^*) \leqslant |\mathcal{Z}^*| * \max_{z_i^*\in\mathcal{Z}^*} \sum_{x\in z_i^*} \nu(x, y^*)$$

Now since $|\mathcal{Z}^*| = \|y^*\|^{\mathcal{Z}}$, and since $\|y^*\|^{\mathcal{Z}} \leqslant \max_{y\in\mathcal{Y}} \|y\|^{\mathcal{Z}}$, we find that:

$$V(y^*) \leqslant \max_{y\in\mathcal{Y}} \|y\|^{\mathcal{Z}} * \max_{z_i^*\in\mathcal{Z}^*} \sum_{x\in z_i^*} \nu(x, y^*) \tag{4}$$

Furthermore, assuming that (3) holds, we have:

$$\forall z_i^* \in \mathcal{Z}^*, \exists y' \in \mathcal{Y}': \quad z_i^* \subseteq y' \wedge \sum_{x\in z_i^*} \nu(x, y') = \max_{y\in\mathcal{Y}} \sum_{x\in z_i^*} \nu(x, y) \tag{5}$$

Now since the following holds for every $z_i^* \in \mathcal{Z}^*$:

$$\sum_{x\in z_i^*} \nu(x, y^*) \leqslant \max_{y\in\mathcal{Y}} \sum_{x\in z_i^*} \nu(x, y) \tag{6}$$

Then, from (5) and (6), we find that:

$$\forall z_i^* \in \mathcal{Z}^*, \exists y' \in \mathcal{Y}': \quad z_i^* \subseteq y' \wedge \sum_{x\in z_i^*} \nu(x, y') \geqslant \sum_{x\in z_i^*} \nu(x, y^*) \tag{7}$$

Moreover, we know that:

$$\forall y' \in \mathcal{Y}', \ \forall z_i^* \subseteq y': \quad \sum_{x\in y'} \nu(x, y') \geqslant \sum_{x\in z_i^*} \nu(x, y') \tag{8}$$

Based on (8), as well as the fact that $V(y') = \sum_{x\in y'} \nu(x, y')$, we find that:

$$\forall y' \in \mathcal{Y}', \forall z_i^* \subseteq y': \quad V(y') \geqslant \sum_{x\in z_i^*} \nu(x, y') \tag{9}$$

From (7) and (9), we find that:

$$\forall z_i^* \in \mathcal{Z}^*, \exists y' \in \mathcal{Y}': \quad z_i^* \subseteq y' \wedge V(y') \geqslant \sum_{x\in z_i^*} \nu(x, y^*)$$

This, in turn, implies that:

$$\exists y' \in \mathcal{Y}': \quad V(y') \geqslant \max_{z_i^*\in\mathcal{Z}^*} \sum_{x\in z_i^*} \nu(x, y^*) \tag{10}$$

Finally, from (4) and (10), we find that:

$$\exists y' \in \mathcal{Y}': \quad V(y^*) \leqslant \max_{y\in\mathcal{Y}} \|y\|^{\mathcal{Z}} * V(y') \qquad \square$$

Having proved Theorem 2, we now show how it can be used while proving the main theorem for establishing a ratio bound β in PFG^+/PFG^-.

Theorem 3. *To establish a bound β on the value of a coalition structure given a PFG$^+$ setting, every subspace $\mathcal{P}_I^A$: $I \in \mathcal{I}^n$: $|I| \leqslant 2$ 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 PFG$^-$ setting, every subspace $\mathcal{P}_I^A$: $I \in \{[n], [n-1,1], [n-2,1,1], \ldots, [1,1,\ldots,1]\}$ 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 *PFG*$^+$ setting, the only coalition structure in which C is guaranteed to have its maximum value is $\{C, A \backslash C\}$. Based on this, the subspaces $\mathcal{P}_I^A$: $I \in \mathcal{I}^n$: $|I| \leqslant 2$ must be searched, and these contain 2^{n-1} coalition structures.

To prove that $\beta = n$, we use Theorem 2 as follows:

- We consider $\mathcal{X}$ to be the set of coalitions. That is, $\mathcal{X} = \mathcal{C}$.
- We consider $\mathcal{Y}$ to be the set of coalition structures, and $\mathcal{Y}'$ to be a subset of $\mathcal{Y}$ containing the coalition structures of size 1 or 2. That is, $\mathcal{Y} = \mathcal{P}^A$ and $\mathcal{Y}' = \mathcal{P}_I^A$: $I \in \mathcal{I}^n$: $|I| \leqslant 2$.
- We consider $\mathcal{Z} = \{\{C\}$: $C \in \mathcal{C}\}$.

Now since every subset in $\mathcal{Z}$ appears *with its maximum value* in one, or more, of the coalition structures in $\mathcal{Y}'$, then the best coalition structure in $\mathcal{Y}'$ is within a ratio bound β from the best coalition structure in $\mathcal{Y}$, where $\beta = \max_{y \in \mathcal{Y}} \|y\|^{\mathcal{Z}}$. This implies that $\beta = n$ since $\|\{\{a_1\}, \ldots, \{a_n\}\}\|^{\mathcal{Z}} = n$.

On the other hand, given a *PFG*$^-$ setting, the only coalition structure in which C is guaranteed to have its maximum value is: $\{C, \{\bar{a}_1\}, \ldots, \{\bar{a}_{|\bar{C}|}\}\}$, where $\{\bar{a}_1\} \cup \cdots \cup \{\bar{a}_{|\bar{C}|}\} = \bar{C}$. Based on this, the following subspaces have to be searched: $\mathcal{P}_I^A$: $I \in \{[n], [n-1,1], [n-2,1,1], \ldots, [1,1,\ldots,1]\}$. With this search, the number of searched coalition structures would be $2^n - n + 1$ 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 *PFG*$^+$ case, we use Theorem 2 to prove that $\beta = \lceil \frac{n}{2} \rceil$. Here:

- We consider $\mathcal{X}$ to be the set of coalitions.
- We consider $\mathcal{Y}$ to be the set of coalition structures (i.e., $\mathcal{Y} = \mathcal{P}^A$), and consider: $\mathcal{Y}' = \mathcal{P}_{[n]}^A \cup \mathcal{P}_{[n-1,1]}^A \cup \mathcal{P}_{[n-2,1,1]}^A \cup \cdots \cup \mathcal{P}_{[1,\ldots,1]}^A$.
- We consider $\mathcal{Z} = \{\{C\}$: $C \in \mathcal{C}\} \cup \{\{C$: $C \in \mathcal{C}, |C| = 1\}\}$, i.e., every subset in $\mathcal{Z}$ contains either a single coalition, or a combination of singletons. Note that the *maximum possible value* of every such combination has been observed in $\mathcal{P}_{[1,\ldots,1]}^A$ (see Theorem 1).

The above implies that the best coalition structure in $\mathcal{Y}'$ is within a ratio bound β from the best coalition structure in $\mathcal{Y}$ since every possible subset in $\mathcal{Z}$ appears *with its maximum value* in $\mathcal{Y}'$. This bound equals the size of the biggest coalition structure with respect to $\mathcal{Z}$ (see Theorem 2). Importantly, since every combination of singletons belongs to $\mathcal{Z}$ then, for any two coalition structures CS and CS' such that CS' contains more than one singleton, and CS' is derived from CS by grouping all singletons into one coalition, we have $\|CS\|^{\mathcal{Z}} = \|CS'\|^{\mathcal{Z}}$. Based on this, it can easily be shown that the biggest coalition structures with respect to $\mathcal{Z}$ are those belonging to $\mathcal{P}_{[2,2,\ldots,2,1]}^A$ when the number of agents is odd, and to $\mathcal{P}_{[2,2,\ldots,2]}^A$ when the number of agents is even. In either case, we have: $\max_{y \in \mathcal{Y}} \|y\|^{\mathcal{Z}} = \lceil \frac{n}{2} \rceil$. □

Interestingly, in *CFG*s, it is *sufficient* to search the first and second levels of the coalition structure graph in order to bound β [1]. 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 *PFG*$^-$ setting, it is *necessary* to search $\mathcal{P}_I^A$: $I \in \{[n], [n-1,1], [n-2,1,1], \ldots, [1,1,\ldots,1]\}$ and, given a *PFG*$^+$ setting, it is *necessary* to search $\mathcal{P}_I^A$: $I \in \mathcal{I}^n$: $|I| \leqslant 2$ (see Theorem 3).

4.3. Improving the worst-case bound

In this section, we present our procedure for reducing the ratio bound with further search. This procedure is based on Theorem 2, where $\mathcal{X}$ is considered to be the set of coalitions, and $\mathcal{Y}$ is considered to be the set of coalition structures, and the basic idea is to select $\mathcal{Y}'$ and $\mathcal{Z}$ such that the desired bound is obtained. That is, we first select $\mathcal{Y}'$ and $\mathcal{Z}$ so as to obtain the initial bound β identified in Theorem 3. After that, we add certain elements to $\mathcal{Y}'$ and $\mathcal{Z}$ so as to drop the bound to $\beta - 1$, and then repeat the same process to drop it to $\beta - 2$, and so on.

For presentation clarity, we will first discuss an example of how the procedure works, and then present the pseudo code of this procedure. In particular, an example of 10 agents with positive externalities is shown in Fig. 3, where integers are used to represent coalition sizes. For instance, $\{\{C\}$: $C \in \mathcal{C}, |C| = 2\}$ is represented by [2]. Similarly, $\{\{C, C'\}$: $C, C' \in \mathcal{C}, |C| = 4, |C'| = 3\}$ is represented by [4, 3]. In more detail:

- **Fig. 3(A)**: $\mathcal{Z}$ initially contains every possible subset of $\mathcal{X}$ that is of size 1, while $\mathcal{Y}'$ initially contains the coalition structures in $\mathcal{P}_I^A$: $I \in \mathcal{I}^{10}$: $|I| \leqslant 2$ (see how $\mathcal{Z}$ contains the integers $1, 2, \ldots, 10$ in the figure, while $\mathcal{Y}'$ contains the integer

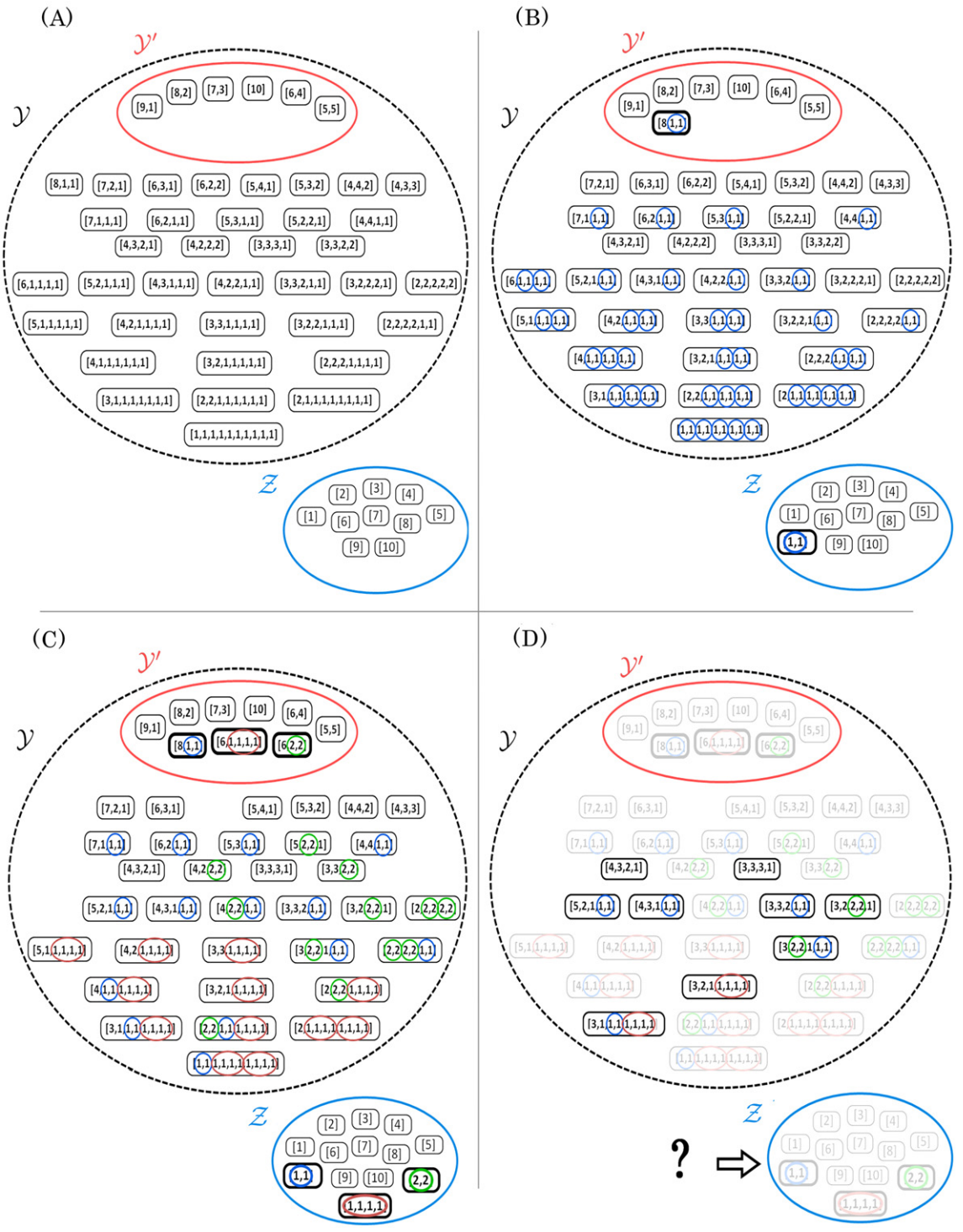

Fig. 3. Example of our procedure for reducing the ratio bound algorithm with further search given 10 agents and positive externalities. The circles that surround the same combination of integers have the same color.

partitions of 10 that contains exactly two integers each). Since every subset in $\mathcal{Z}$ appears in $\mathcal{Y}'$ with its maximum value (see Theorem 1), then the best coalition structure in $\mathcal{Y}'$ is within a ratio bound β from the best one in $\mathcal{Y}$. This bound is equal to the size of the biggest subset of $\mathcal{Y}$ with respect to $\mathcal{Z}$. Here, the biggest subset happens to be the one represented by $[1, \ldots, 1]$ (i.e., it is $\{\{a_1\}, \ldots, \{a_n\}\}$), which is of size 10.

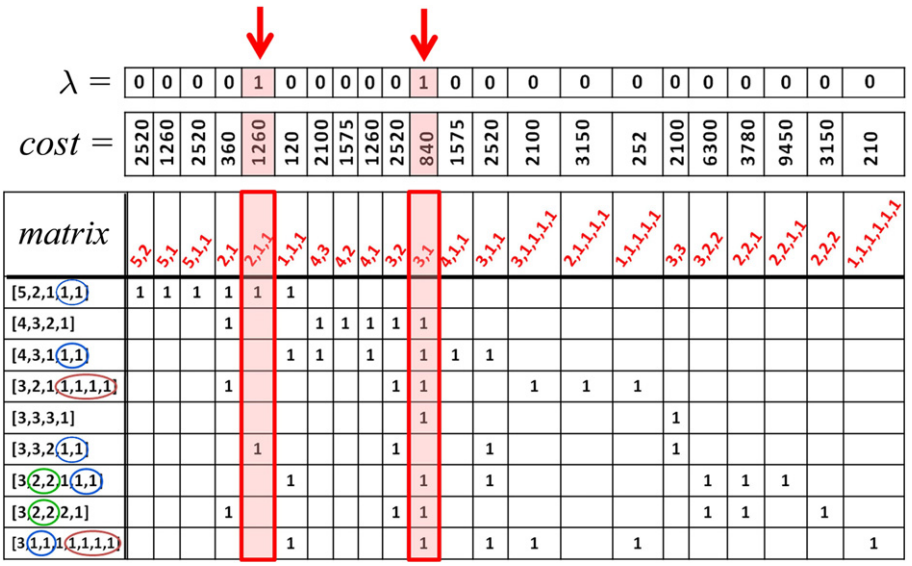

Fig. 4. Example of how the integer model finds the subspaces to be searched in order to improve the ratio bound.

- **Fig. 3(B)**: Here, we add to $\mathcal{Z}$ every combination of two singletons (see how $\mathcal{Z}$ now contains $[1,1]$ in the figure). To ensure that, after this addition, every subset in $\mathcal{Z}$ still appears in $\mathcal{Y}'$ with its maximum value, we need to add to $\mathcal{Y}'$ the coalition structures that correspond to $[8,1,1]$ (see Theorem 1). Importantly, the modification to $\mathcal{Z}$ reduces the size of the biggest subset in $\mathcal{Y}$ with respect to $\mathcal{Z}$. In particular, when computing the size of $\{\{a_1\},\ldots,\{a_n\}\}$ with respect to $\mathcal{Z}$, we can see that it drops from 10 to 5.
- **Fig. 3(C)**: Here, we add to $\mathcal{Z}$ all the combinations of four singletons and all the combinations of two coalitions of size 2 each. We also add to $\mathcal{Y}'$ the coalition structures where those combinations appear with their maximum values (see the elements that were added to $\mathcal{Z}$ and $\mathcal{Y}'$ in the figure). As a result of those additions to $\mathcal{Z}$ and $\mathcal{Y}'$, we can see that the size of the biggest subset in $\mathcal{Y}$ with respect to $\mathcal{Z}$ has now dropped from 5 to 4.
- **Fig. 3(D)**: The key question, then, at every stage is the following: which elements should be added to $\mathcal{Z}$ and $\mathcal{Y}'$ in order to drop the size of the biggest subset in $\mathcal{Y}$ with respect to $\mathcal{Z}$? In our example, the biggest subsets in $\mathcal{Y}$ are now of size 4 each, and they belong to the subspaces that are highlighted in the figure. As can be seen, it is not trivial to determine which elements to add to $\mathcal{Z}$ and $\mathcal{Y}'$. We solve this problem by using an Integer Programming solver. The basic idea is to represent the problem as a boolean matrix, where every row corresponds to one of the subspaces that need to be dealt with (i.e., a subspace containing subsets of which the size, with respect to $\mathcal{Z}$, equals β), and every column corresponds to a unique combination that could be added to $\mathcal{Z}$ (see Fig. 4). The value of the matrix at any given row r and column c equals 1 if, by adding to $\mathcal{Z}$ the combination that corresponds to column c, we drop the size of the subspace corresponding to row r. Otherwise, the value is 0.[10] In Fig. 4, for example, the value at row 2 and column 4 is 1. This is because the size of any subset in $\mathcal{P}^A_{[4,3,2,1]}$ drops from 4 to 3 by adding to $\mathcal{Z}$ all the combinations of two coalitions where one is of size 2 and the other is of size 1. What we need, then, is to find a set of columns such that, for every row, the value in the matrix equals 1 in one or more of those columns. In Fig. 4, for example, the highlighted columns represent a possible solution. The process of finding the required set of columns is done using an integer programming solver. The integer formulation takes into consideration, for every column, the size of the subspace that needs to be added to $\mathcal{Y}'$ (see the *cost* vector in Fig. 4). For example, adding to $\mathcal{Z}$ the combinations of coalitions of sizes $2,1,1$ has a cost of 1260. This is because we need to add $\mathcal{P}^A_{[6,2,1,1]}$ to $\mathcal{Y}'$, and this contains 1260 coalition structures. The integer solver finds a feasible set of columns of which the cost is minimal.

Having presented a detailed example of our procedure for reducing the ratio bound, we now present the pseudo code of this procedure (see Algorithm 1). In more detail, we initially search $\mathcal{P}^A_{[n]}, \mathcal{P}^A_{[n-1,1]},\ldots,\mathcal{P}^A_{[1,1,\ldots,1]}$ in the *PFG*$^-$ case, and search $\mathcal{P}^A_{[n-2,1,1]} \cup \mathcal{P}^A_I$: $I \in \mathcal{I}^n, |I| \leqslant 2$ in the *PFG*$^+$ (steps 1 to 5). This can be interpreted as putting those subspaces in $\mathcal{Y}'$, and putting in $\mathcal{Z}$ the subsets of $\mathcal{X}$ that appear with their maximum value in those subspace (i.e., $\mathcal{Z} = \{\{C\}\colon C \in \mathcal{C}\} \cup \{\mathcal{C}'\colon \mathcal{C}' \subseteq \mathcal{C} \wedge \forall C \in \mathcal{C}'\colon |C| = 1\}$ in *PFG*$^-$, and $\mathcal{Z} = \{\{C\}\colon C \in \mathcal{C}\} \cup \{\{C, C'\}\colon C, C' \in \mathcal{C}, |C| = 1, |C'| = 1\}$ in *PFG*$^+$). It can easily be shown that the bound in this case would be $\beta = \lceil \frac{n}{2} \rceil$ (see Theorem 2). After that, for every subspace $\mathcal{P}^A_I$ that has not yet been searched, we compute the smallest partition of I in $\mathcal{Z}$, denoted $partition_I$ (steps 6 to 8). This partition simply groups all singletons together in the *PFG*$^-$ case, and every pair of singletons in the *PFG*$^+$ case (see how $partition_I$ is initialized in

[10] The values that are equal to 0 are omitted from the matrix in Fig. 4 for presentation clarity.

Algorithm 1 Lowering the ratio bound with further search.

if PFG^- **then**
  Search $\mathcal{P}^A_{[n]}, \mathcal{P}^A_{[n-1,1]}, \mathcal{P}^A_{[n-2,1,1]}, \ldots, \mathcal{P}^A_{[1,1,\ldots,1]}$.
else
  Search $\mathcal{P}^A_I$: $|I| \leqslant 2$ and then search $\mathcal{P}^A_{[n-2,1,1]}$.
end if
for $I \in \mathcal{I}^n$ **such that** $searched(\mathcal{P}^A_I) = 0$ **do**
  $initialize(partition_I)$. {see Algorithm 1.1}
end for
for $\beta = \lceil \frac{n}{2} \rceil$ **down to** 2 {main loop} **do**
  $r \leftarrow 1$; $c \leftarrow 1$.
  for $I \in \mathcal{I}^n$ **such that** $searched(\mathcal{P}^A_I) = 0$ **and** $|partition_I| = \beta$ **do**
    $rows[r] \leftarrow partition_I$; $r \leftarrow r + 1$.
    for $I', I'' \in partition_I$ **do**
      if $I' \uplus I'' \notin columns$ **then**
        $columns[c] \leftarrow I' \uplus I''$; $c \leftarrow c + 1$.
      end if
    end for
  end for
  for $r = 1$ **to** $|rows|$ **do**
    for $c = 1$ **to** $|columns|$ **do**
      if $\exists I', I'' \in rows[r] : I' \uplus I'' = columns[c]$ **then**
        $matrix[r][c] \leftarrow 1$.
      else
        $matrix[r][c] \leftarrow 0$.
      end if
    end for
  end for
  for $c = 1$ **to** $|columns|$ **do**
    if PFG^- **then**
      $subspace[c] \leftarrow \mathcal{P}^A_{columns[c] \uplus I'} : I' \in \mathcal{I}^{n-|columns[c]|}, I' = [1, \ldots, 1]$.
    else
      $subspace[c] \leftarrow \mathcal{P}^A_{columns[c] \uplus I'} : I' \in \mathcal{I}^{n-|columns[c]|}, I' = [n - |columns[c]|]$.
    end if
  end for
  for $c = 1$ **to** $|columns|$ **do**
    if $searched(\mathcal{P}^A_{subspace[c]}) = 1$ **then**
      $cost[c] \leftarrow 0$
    else
      $cost[c] \leftarrow |subspace[c]|$
    end if
  end for
  for $c = 1$ **to** $|columns| - 1$ **do**
    for $c' = c + 1$ **to** $|columns|$ **do**
      if $subspace[c] = subspace[c']$ **then**
        for $r = 1$ **to** $|rows|$ **do**
          if $matrix[r][c'] = 1$ **then**
            $matrix[r][c'] \leftarrow 0$; $matrix[r][c] \leftarrow 1$
          end if
        end for
      end if
    end for
  end for
  $\lambda \leftarrow integerSolver(matrix, cost)$ {call the integer solver}
  for $c = 1$ **to** $|columns|$ **do**
    if $\lambda[c] = 1$ **then**
      Search through $subspace[c]$.
    end if
  end for
  for $I \in \mathcal{I}^n$ **such that** $searched(\mathcal{P}^A_I) = 0$ **do**
    for $c = 1$ **to** $|columns|$ **do**
      if $\lambda[c] = 1$ **and** $\exists I', I'' \in partition_I$ **such that** $I' \uplus I'' = columns[c]$ **then**
        $partition_I \leftarrow partition_I \setminus \{I', I''\} \cup \{I' \uplus I''\}$
      end if
    end for
  end for
end for
Search through the remaining coalition structures. {This is to drop β from 2 to 1.}

Algorithm 1.1 *initilize*($partition_I$); part of Algorithm 1.

1: $partition_I \leftarrow \emptyset$
2: **for** $i \in I, i > 1$ **do**
3: $\quad partition_I \leftarrow partition_I \cup \{[i]\}$
4: **end for**
5: **if** PFG^- **then**
6: $\quad I' \leftarrow [1, \ldots, 1] : |I'| = multiplicity(1, I)$
7: $\quad partition_I \leftarrow partition_I \cup I'$
8: **else**
9: $\quad I' \leftarrow [1, 1]$
10: $\quad$ **for** $k = 1$ **to** $\lfloor multiplicity(1, I)/2 \rfloor$ **do**
11: $\quad\quad partition_I \leftarrow partition_I \cup I'$
12: $\quad$ **end for**
13: $\quad$ **if** $\lfloor multiplicity(1, I)/2 \rfloor \neq \lceil multiplicity(1, I)/2 \rceil$ **then**
14: $\quad\quad partition_I \leftarrow partition_I \cup [1]$
15: $\quad$ **end if**
16: **end if**

Algorithm 1.1). For example, given $I = [1, 1, 1, 3]$, we have $partition_I = \{[1, 1, 1], [3]\}$ in PFG^-, and $partition_I = \{[1], [1, 1], [3]\}$ in PFG^+. As described earlier, the size of the biggest such partition(s) is what determines the ratio bound β, and what we need is to drop this size in order to improve the bound. This is done in the main loop (steps 9 to 66), where the size (i.e., the bound β) drops by 1 after every iteration. More specifically, we build a matrix such that: (i) every row corresponds to one of the biggest partitions (steps 10 to 12), and (ii) every column corresponds to a combination that reduces the size of at least one of those partitions (steps 13 to 17), and (iii) the matrix at row r and column c is assigned a value of "1" if the combination at column c reduces the size of the partition at row r, otherwise it is assigned a value "0" (steps 19 to 27). An example of such a matrix can be seen in Fig. 4. After that, for every column c, we store in $subspace[c]$ the subspace that needs to be searched (i.e., needs to be added to $\mathcal{Y}'$) such that the combination corresponding to column c is observed with its maximum value (steps 28 to 34). For example, given 10 agents, and given the combination $[3, 4]$, the subspace that needs to be searched would be $[1, 1, 1, 3, 4]$ in PFG^-, and $[3, 3, 4]$ in PFG^+ (see Theorem 1). An integer solver is then used to find a set of columns such that, for every row, there exists at least one column in which the value is "1". This solver takes into consideration the "*cost*" of every column, which is computed in steps 35 to 41. In more detail, the cost of a given column is the size of the corresponding subspace, unless that subspace has already been searched, in which case the cost is 0. The integer formulation is as follows:

$$Minimize \sum_{c=1}^{|columns|} \lambda[c] \times cost[c]$$

$$\text{s.t.} \quad \forall r \in \{1, \ldots, |rows|\}, \sum_{c=1}^{|columns|} \lambda[c] \times matrix[r][c] \geqslant 1$$

$$\forall c \in \{1, \ldots, |columns|\}, \lambda[c] \in \{0, 1\}$$

An important point to note, here, is that different columns may have the same corresponding subspace. For example, given a PFG^+ setting with 10 agents, a column corresponding to the combination $[5, 1]$, and another corresponding to $[4, 1]$, both require searching $\mathcal{P}^A_{[5,4,1]}$. Similarly, the columns corresponding to $[7, 1]$, $[7, 1, 1]$ and $[1, 1, 1]$ all require searching $\mathcal{P}^A_{[7,1,1,1]}$. This means that if we "pay the cost" of adding one of these combinations to $\mathcal{Z}$ (i.e., if we search the subspace in which this combination appears with its maximum value), then we can add the other combinations to $\mathcal{Z}$ "at no extra cost". To take this into account, all such columns need to be combined into one. More specifically, given a set of columns $c^1_I, c^2_I, \ldots, c^\delta_I$ that require searching $\mathcal{P}^A_I$, we combine them all into one column, say c^δ_I, as follows: For every row r, we set $matrix[r][c^\delta_I] = 1$ if there exists at least one column c^k_I such that $matrix[r][c^k_I] = 1$, otherwise we set $matrix[r][c^\delta_I] = 0$. As for the remaining columns, i.e., c^j_I: $j \neq 1$, we set $matrix[r][c^j_I] = 0$. In Fig. 4, for example, since columns 2 and 9 require searching one subspace, namely $\mathcal{P}^A_{[5,4,1]}$, we can combine them into one column, say column 9, by setting $matrix[1][9] = 1$, $matrix[2][9] = 1$, and $marix[3][9] = 1$, and setting $matrix[r][2] = 0$ for every row r. Note that the cost of both columns remains unchanged. Once the integer solver finds the required columns (step 53), we search the subspace that corresponds to each of these columns (steps 54 to 58). Finally, for every subspace $\mathcal{P}^A_I$ that has not yet been searched, we update $partition_I$ to take into consideration the combinations that have just been observed with their maximum values (steps 59 to 65). For example, given $partition_{[2,2,3,3]} = \{[2], [2], [3, 3]\}$, and given that we searched the subspace that corresponds to the combination $[2, 2]$, then this can be interpreted as adding $[2, 2]$ to $\mathcal{Z}$. As a result, we update $partition_{[2,2,3,3]}$ by setting it to $\{[2, 2], [3, 3]\}$.

4.4. Preprocessing

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

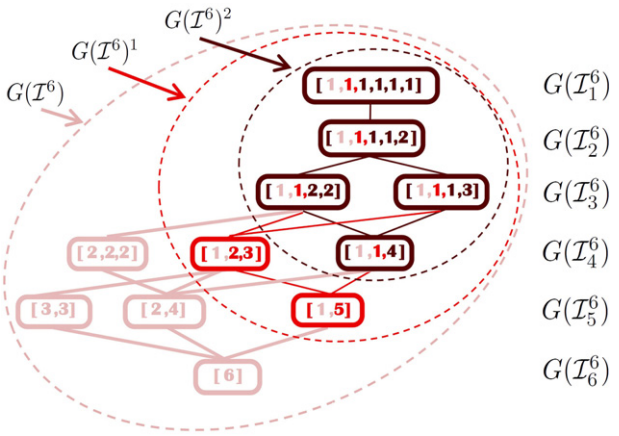

Fig. 5. The figure shows how the integer partitions of 4 and 5 appear in the integer partition of 6.

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 Fig. 5. 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 PFG$^-$ (PFG$^+$) setting. Then, given a coalition $C \subseteq \mathcal{C}$ 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*:

$$\forall p' \in P',\ \exists p \in P: \quad p' \subseteq (\supseteq) p \quad \text{and} \quad UB^P \leqslant LB^{P'}$$

Proof. Given a PFG$^-$ (PFG$^+$) 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 $UB^P \leqslant LB^{P'}$, we will prove that, for any coalition structure $CS \supseteq P$, there exists another coalition structure $CS' \supseteq P'$ such that $W(CS) \leqslant W(CS')$.

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:

$$W(CS) = W(P, CS) + W(\overline{P}, CS) \quad \text{and} \quad W(CS') = W(P', CS') + W(\overline{P}, CS') \tag{11}$$

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:

$$V(\overline{P}, CS) \leqslant V(\overline{P}, CS') \tag{12}$$

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

$$V(P, CS) \leqslant V(P', CS) \tag{13}$$

From (11), (12), and (13), we find that $V(CS) \leqslant V(CS')$. 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 PFG$^-$ (PFG$^+$) setting. Then, given an integer partition $I \in \mathcal{I}^s$: $s \leqslant n$, any subspace 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*:

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

Algorithm 2 Prune subspaces based on Lemma 2.

```
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 ( PFG⁻ and ∃I′ ∈ 𝓘̂ : I → I′, UB_I ⩽ LB_I′ )
         or (PFG⁺ 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
```

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: $I \to I'$ $(I' \to I)$. In Fig. 5, for example, we have: $[1, 5] \to [1, 1, 1, 1, 2]$ ($[1, 1, 1, 1, 2] \to [1, 5]$).

We can now show how both Lemmas 1 and 2 are used in our preprocessing algorithm to prune subspaces (see Algorithm 2). Basically, 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 1. To generate the integer partitions of s that do not contain any 1s, we use `getIntParts(`s`, 2)`. This function can be implemented using any algorithm that generates *doubly-restricted*[11] integer partitions, e.g., the *Parta* algorithm [29].

4.5. Searching a subspace

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

As mentioned earlier in Section 3, the IP algorithm computes, for all $I \in \mathcal{I}^n$, upper and lower bounds on the value of the best coalition structure in $\mathcal{P}_I^A$ as follows: $UB_I = \sum_{s\in I} Max_s$, $LB_I = \sum_{s\in I} Avg_s$. These bounds are used to identify, and consequently prune, unpromising subspaces. As for the promising ones, IP searches them one at a time, unless a value is found that is higher than the upper bound of another subspace, in which case, that subspace no longer needs to be searched. The order in which the algorithm searches through these subspaces 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 subspace $\mathcal{P}_I^A$: $I = [i_1, i_2, \ldots, i_{|I|}]$ 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 $C_1, C_2, \ldots, C_{|I|-1}$. For every such coalition, denoted $C_{|I|}$, we would have a coalition structure $CS = \{C_1, \ldots, C_{|I|}\} \in \mathcal{P}_I^A$. This process is repeated in such a way that is guaranteed to go through every coalition structure in $\mathcal{P}_I^A$ 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 found so far, then, before the algorithm goes through the coalitions of size i_x that do not overlap with $C_1, \ldots, C_{x-1}$, it checks whether the following holds[12]:

$$v(C_1) + \cdots + v(C_{x-1}) + Max_{i_x} + \cdots + Max_{i_{|I|}} \leqslant V\left(CS^{**}\right)$$

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 $C_1, \ldots, C_{x-1}$. This is because any coalition structure containing $C_1, \ldots, C_{x-1}$ cannot possibly be better than CS^{**}.

The main difference in our PFG^-/PFG^+ setting (compared to *CFG*s) is that, instead of having one value for every coalition C, we have a maximum value UB^C and a minimum value LB^C (see Section 4.1 for more details). Based on this, we modify IP as follows:

- We add a preprocessing stage, which consists of Algorithm 1 (to prune unpromising subspaces) and Algorithm 2 (to identify the order by which the remaining subspaces need to be searched).

[11] 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 1s.

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

- We compute Max_s and Avg_s as follows:

$$\forall s \leqslant n, \quad Max_s = \max_{C \subseteq A:\ |C|=s} UB_C \quad \text{and} \quad Avg_s = \text{avg}_{C \subseteq A:\ |C|=s}\, LB_C.$$

- The order in which we search through subspaces is based on our anytime algorithm for reducing the ratio bound β (i.e., we always search the subspaces that are necessary to drop the current ratio bound).
- While searching a subspace $\mathcal{P}_I^A$: $I = [i_1, i_2, \ldots, i_{|I|}]$, and before going through the coalitions of size i_x that do not overlap with $C_1, \ldots, C_{x-1}$, we check whether the following holds, where $UB_{\{C_1,\ldots,C_{x-1}\}}$ is computed as in Section 4.1:

$$UB_{\{C_1,\ldots,C_{x-1}\}} + Max_{i_x} + \cdots + Max_{i_{|I|}} < V\left(CS^{**}\right) \tag{14}$$

5. Performance evaluation

In this section, we propose an equation for generating random input instances to be used for evaluating CSG algorithms in PFG^+/PFG^- (Section 5.1). This equation is then used while evaluating our $IP^{+/-}$ algorithm (Section 5.2).

5.1. Input generation

In order to evaluate our $IP^{+/-}$ algorithm, we need to be able to generate random input instances of $PFG^+(PFG^-)$ such that only positive (negative) externalities occur whenever two coalitions merge. This means, in particular, that for any two arbitrary coalition structures, CS and CS', such that CS' can be created from CS by a series of coalition mergers, the value of any coalition not involved in these mergers must not be smaller (greater) in CS' than in CS. Next, we provide an equation for generating random input instances, and prove that the resulting partition function is guaranteed to satisfy the conditions of PFG^+/PFG^-.

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[13] agent in each coalition. Now, let $l(a_i, C)$ be the location of agent a_i in C (e.g., $l(a_6, \{a_2, a_5, a_6, a_9\}) = 3$), and let $l(a_i, P)$ be 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 PFG^- (PFG^+) condition is met. At first, for every coalition C, we generate the following non-negative random values: v_C and e_C and $e_{C,1}, e_{C,2}, \ldots, e_{C,|\overline{C}|}$ 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:

$$w(C, CS) = v_C - (+) \sum_{a_i \in \overline{C}} e_{C,l(a_i,\overline{C})} * \left(1 - \frac{l(a_i, CS\backslash\{C\}) - 1}{|\overline{C}|}\right) \tag{15}$$

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 (15) 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: $\forall a_i \in \overline{C}$: $l(a_i, CS\backslash\{C\}) \geqslant 1$, which implies that:

$$\begin{aligned}
\sum_{a_i \in \overline{C}} e_{C,l(a_i,\overline{C})} * \left(1 - \frac{l(a_i, CS\backslash\{C\}) - 1}{|\overline{C}|}\right) &\leqslant \sum_{a_i \in \overline{C}} e_{C,l(a_i,\overline{C})} * \left(1 - \frac{1-1}{|\overline{C}|}\right) \\
&\leqslant \sum_{a_i \in \overline{C}} e_{C,l(a_i,\overline{C})} \\
&\leqslant e_C
\end{aligned}$$

Theorem 5. *By setting the value of each coalition as per Eq.* (15)*, we obtain a PFG^- (PFG^+) setting.*

Proof. Given a partition P and a coalition $p \in P$, let $s(p)$ denote the smallest agent in p, and let $s(p, P)$ be defined as follows:

$$s(p, P) = \left\{s(\tilde{p}):\ \tilde{p} \in P,\ s(\tilde{p}) < s(p)\right\} \tag{16}$$

In other words, $s(p, P)$ contains the smallest agent from every coalition that appears before p in P, e.g., $s(\{a_4, a_9\}, \{\{a_1, a_7\}, \{a_3, a_5, a_6\}, \{a_4, a_9\}\}) = \{a_1, a_3\}$.

Next, given a coalition C, and given any two partitions $P, P' \in \mathcal{P}^{\overline{C}}$ such that $\forall p' \in P', \exists p \in P$: $p' \subseteq p$, we will prove that the following holds:

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

$$v\big(C, \{C\} \cup P\big) \quad \leqslant (\geqslant) \quad v\big(C, \{C\} \cup P'\big) \tag{17}$$

From (15), we find that the inequality in (17) holds if, and only if:

$$\sum_{a_i \in \overline{C}} e_{C,l(a_i,\overline{C})} * \left(1 - \frac{l(a_i, P) - 1}{|\overline{C}|}\right) \geqslant \sum_{a_i \in \overline{C}} e_{C,l(a_i,\overline{C})} * \left(1 - \frac{l(a_i, P') - 1}{|\overline{C}|}\right)$$

By removing $e_{C,l(a_i,\overline{C})}$, we find that:

$$\sum_{a_i \in \overline{C}} \left(1 - \frac{l(a_i, P) - 1}{|\overline{C}|}\right) \geqslant \sum_{a_i \in \overline{C}} \left(1 - \frac{l(a_i, P') - 1}{|\overline{C}|}\right)$$

$$\sum_{a_i \in \overline{C}} \frac{l(a_i, P) - 1}{|\overline{C}|} \leqslant \sum_{a_i \in \overline{C}} \frac{l(a_i, P') - 1}{|\overline{C}|}$$

$$\sum_{a_i \in \overline{C}} l(a_i, P) \leqslant \sum_{a_i \in \overline{C}} l\big(a_i, P'\big)$$

Then, to prove that (17) holds, it is sufficient to prove that:

$$\forall a_i \in \overline{C}, \quad l(a_i, P) \leqslant l\big(a_i, P'\big) \tag{18}$$

To this end, for any agent $a_i \in \overline{C}$, let p_i, p_i' be the coalitions that contain a_i in P, P' respectively. Then, to prove that (18) holds, we need to prove that the number of coalitions that appear before p_i in P is smaller than, or equal to, the number of coalitions that appear before p_i' in P'. In other words, we need to prove that: $|s(p_i, P)| \leqslant |s(p_i', P')|$. This can be done by proving that:

$$s(p_i, P) \subseteq s\big(p_i', P'\big)$$

In order to do so, we will first prove that all the agents in $s(p_i, P)$ belong to $s(p_i', P')$. After that, we will prove that there could be agents in $s(p_i', P')$ that do not belong to $s(p_i, P)$. To this end, note that:

(a) *Every agent in* $s(p_i, P)$ *appears in a different coalition in* P'. In other words, no coalition in P' contains more than one agent from $s(p_i, P)$. Otherwise, if two, or more, agents from $s(p_i, P)$ appear in the same coalition in P', then this coalition would not be a subset of a coalition in P, which contradicts with the following assumption: $\forall p' \in P', \exists p \in P\colon\ p' \subseteq p$.
(b) $s(p_i)$ *is smaller than, or equal to,* $s(p_i')$. This is because p_i' is a subset of p_i, which comes from the fact that $a_i \in p_i$ and $a_i \in p_i'$ and $\forall p' \in P', \exists p \in P\colon\ p' \subseteq p$.

From the above two observations, as well as (16), we find that every agent in $s(p_i, P)$ belongs to $s(p_i', P')$. What is left is to show that there could be agents in $s(p_i', P')$ that do not belong to $s(p_i, P)$. We show this through the following example. Assume that $i = 6$, and that:

$$P = \big\{\{a_1, a_8\}, \{a_3, a_4, a_9\}, \{a_5, a_6, a_7\}\big\}$$

$$P' = \big\{\{a_1, a_8\}, \{a_3, a_9\}, \{a_4\}, \{a_5, a_7\}, \{a_6\}\big\}$$

In this case: $p_6 = \{a_5, a_6, a_7\}$ and $p_6' = \{a_6\}$. As can be seen: $s(p_6, P) = \{a_1, a_3\}$, while $s(p_6', P') = \{a_1, a_3, a_4, a_5\}$. □

Importantly, Eq. (15) places no assumptions/restrictions on v_C and e_C. In other words, these can be sampled from any distribution, e.g., Gaussian, Uniform, etc. Also note that, for any coalition C and coalition structure $CS \ni C$, the total externality imposed upon C is smaller than, or equal to, e_C. This gives the ability to place an upper bound on the externality. For example, in order to simulate a setting where a coalition's value can only increase (decrease) by at most 30% due to externalities, we simply set $e_C = 0.3 * v_C$.

Next, we analyze the memory required to store different input instances. In particular, while the resulting partition function consists of $O(n^n)$ values, we will prove that it only requires storing $O(n2^n)$ values.

Theorem 6. *An input instance generated as per Eq.* (15) *can be stored in memory by maintaining* $O(n2^n)$ *values.*

Proof. From Eq. (15), we can see that for every coalition C the equation only requires storing v_C and $e_{C,1}, e_{C,2}, \ldots, e_{C,|\overline{C}|}$.[14] Moreover, for all $C \in \mathcal{C}$, we have: $|\overline{C}| \leqslant (n-1)$. This means the number of values that need to be stored for any coalition $C \in \mathcal{C}$ is less than, or equal to, n. The total number of values is therefore less than $n * 2^n$. □

[14] Note that we do not need to store e_C as it can be computed by simply summing the values: $e_{C,1}, e_{C,2}, \ldots, e_{C,|\overline{C}|}$.

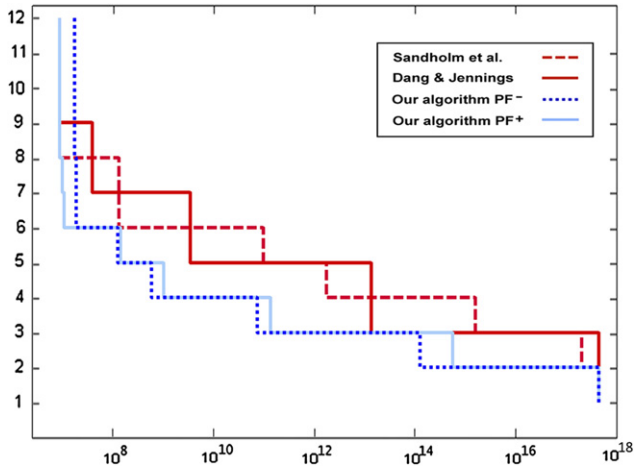

Fig. 6. Given 24 agents, the figure shows the ratio bound as a function of the number of searched coalition structures (*log scale*).

5.2. Performance evaluation

As mentioned earlier in the introduction, to establish progressively better ratio bounds in a *PFG* setting, we propose (i) a way of dividing the search space into subspaces, and (ii) a sequence in which these subspaces must be searched, so that the bound is improved after each subspace. While there are two algorithms that do exactly that, namely Sandholm et al.'s [1] and Dang and Jennings's [2], these are designed for *CFG*s, and so cannot be applied in a *PFG* setting. Nevertheless, since ours is the first such algorithm for *PFG*s, and since *CFG*s are a special case of *PFG*s, then we will benchmark ours against those two algorithms (in a *CFG* setting). Intuitively, one could expect our algorithm to perform worse than the others, especially since they exploit the special properties of *CFG*s, while ours does not. In more detail, the *CFG* algorithms take advantage of the assumption that every coalition has the same value regardless of the coalition structure in which it is embedded. Obviously, we do not take any such advantage since our algorithm is originally designed for *PFG*s.

Given 24 agents, Fig. 6 illustrates (using a log scale) the ratio bound as a function of the number of coalition structures searched.[15] As can be seen, given a PFG^+ setting, our algorithm is significantly faster than the existing *CFG* ones. For example, the number of coalition structures required by our algorithm to establish a ratio bound of 3 is only 0.001% of that required by Sandholm et al. [1], and only 1% of that required by Dang and Jennings [2]. On the other hand, given a PFG^- setting, our 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 this bound, which only requires searching 4×10^{-11} of the space, it becomes significantly faster, compared to the others. For example, the number of coalition structures to establish a ratio bound of 3 is only 0.0007% and 0.5% compared to Sandholm et al.'s and Dang and Jennings's, respectively.

The main reason behind this gain is that our steps are defined in a much more informed manner compared to the other algorithms. More specifically, every step in Sandholm et al.'s algorithm searches all coalition structures of certain size, and every step in Dang and Jennings's algorithm searches all coalition structures where the biggest coalition is of a certain size. Our results show that these are not the best steps that one can take to drop the bound. By tailoring our steps to suite our objective, which is to rapidly reduce the bound, we manage to outperform those algorithms.

So far, we have evaluated the way $IP^{+/-}$ divides the search space, and the sequence in which the subspaces are searched. Next, we evaluate the algorithm's ability to find progressively better solutions in various test settings. In this context, we tested our algorithm given:

- different numbers of agents, ranging from 10 to 20[16];
- different types of externalities (i.e., positive and negative);
- different percentages of externalities ranging from 0% to 100%. Here, when reporting a result given x% externality, we mean that the total externality affecting any given coalition is *at most* x% of its value. More specifically, for any coalition C, the value e_C is sampled from the uniform distribution $U(0, v(C) * x/100)$;
- different value distributions. Specifically, we use the following standard distributions in the CSG literature [30,6]:
  (a) Normal: $v(C) \sim |C| \times N(\mu, \sigma^2)$ where $\mu = 1$ and $\sigma = 0.1$.
  (b) Uniform: $v(C) \sim |C| \times U(a, b)$ where $a = 0$ and $b = 1$.
  (c) NDCS: $v(C) \sim N(\mu, \sigma^2)$, where $\mu = |C|$ and $\sigma = \sqrt{|C|}$.

[15] Similar patterns were observed given different numbers of agents.
[16] The results are reported as an average of 500 runs given $n \in \{10, 11, 12\}$, 300 runs given $n \in \{13, 14, 15\}$, and 100 runs given $n \in \{16, \ldots, 20\}$.

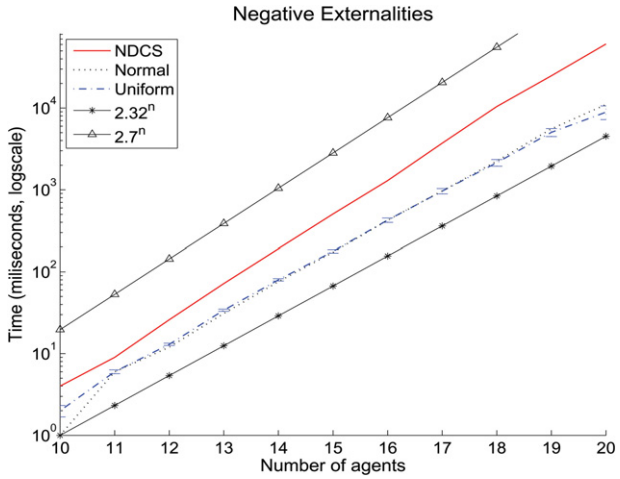

Fig. 7. Given negative externalities, the figure shows *on a log scale* the total run time of $IP^{+/-}$ given different numbers of agents.

Every combination of the above represents a different setting. For example, a possible setting would be: 18 agents, with positive externalities, with percentage of externalities equal to 10%, and with Uniform distribution. Hence, given the large number of possible settings, we only report the results for some of them, and that is whenever similar patterns are observed for the remaining ones (e.g., if we report a certain result for only negative externalities, then this implicitly means that similar results where observed for positive externalities).

In all our experiments, the process of calculating the sequence of subspaces took insignificant time and space, e.g., less than 1 second and less than 15 kb of memory for each of the aforementioned settings. This is because this process deals with the space of integer partitions, not the space of coalition structures (e.g., given 20 agents, there are only 627 integer partitions, as opposed to more than 5×10^{13} coalition structures). Furthermore, the process of calculating the sequence of subspaces needs to be done once for any given number of agents; the sequence can then be stored and used for different problem instances (unlike the process of *searching* the subspaces, which needs to be repeated for each problem instance).

Given negative externalities, Fig. 7 shows *on a log scale* the total run time of $IP^{+/-}$ given different numbers of agents, where time is measured as the clock time (in milliseconds) on a PC with 8 Intel 2.67 GHz processors and 12 GB of RAM. As can be seen, the growth rate of the run-time is similar to $O(2.7^n)$ given the NDCS distribution, and similar to $O(2.32^n)$ given Uniform and Normal distributions. This is significantly smaller than $O(n^n)$ — the growth rate of the size of the search space. This difference comes from the algorithm's ability to identify, and thus prune, unpromising subspaces and coalition structures.

Next, we test our algorithm given different percentages of externalities. In particular, Fig. 8 shows how a change in this percentage affects both the total run time and the number of expansions made by the algorithm.[17] As expected, there is a direct correlation between the two results. The figure also shows that the number of expansions is always slightly more in PFG^- than in PFG^+. This is because the effectiveness of the branch-and-bound technique, which is directly reflected in the number of expansions made, is slightly less in PFG^-. To better understand the reason behind this, we need to analyze the effect that different externalities have on Eq. (14) — the main equation in the branch-and-bound technique. Intuitively, while both sides of the equation increase in PFG^+ and decrease in PFG^-, the ratio between the two sides becomes different in PFG^+ than in PFG^-, resulting in a change in the effectiveness of branch-and-bound. In more detail:

- the term $V(CS^{**})$ in Eq. (14) increases in PFG^+ and decreases in PFG^-.
- the terms "$Max_{i_x} + \cdots + Max_{i_{|I|}}$" increases in PFG^+ but remain unchanged in PFG^-.
- the term "$UB_{\{C_1,\ldots,C_{x-1}\}}$" increases in PFG^+ and decreases in PFG^-, except for when $x = 0$, in which case it remains unchanged in PFG^-.

The above means that the increase in the left-hand side of Eq. (14) in PFG^+ is greater than the decrease in PFG^-. This, in turn, implies that the number of prunings is greater in PFG^+ than in PFG^-.

Next, we analyze how the solution quality and bound quality improve during run time. Observe that the bound that is evaluated here is different from the one that was evaluated earlier in Fig. 6: here, the bound is calculated based on the coalition values of the problem instance at hand, while the previous bound is calculated based on the sequence of subspaces (i.e., it is independent of the coalition values).

[17] Recall that each subspace is searched using depth-first search. The number of expansions is then the total number of nodes that were searched by the algorithm in different subspaces.

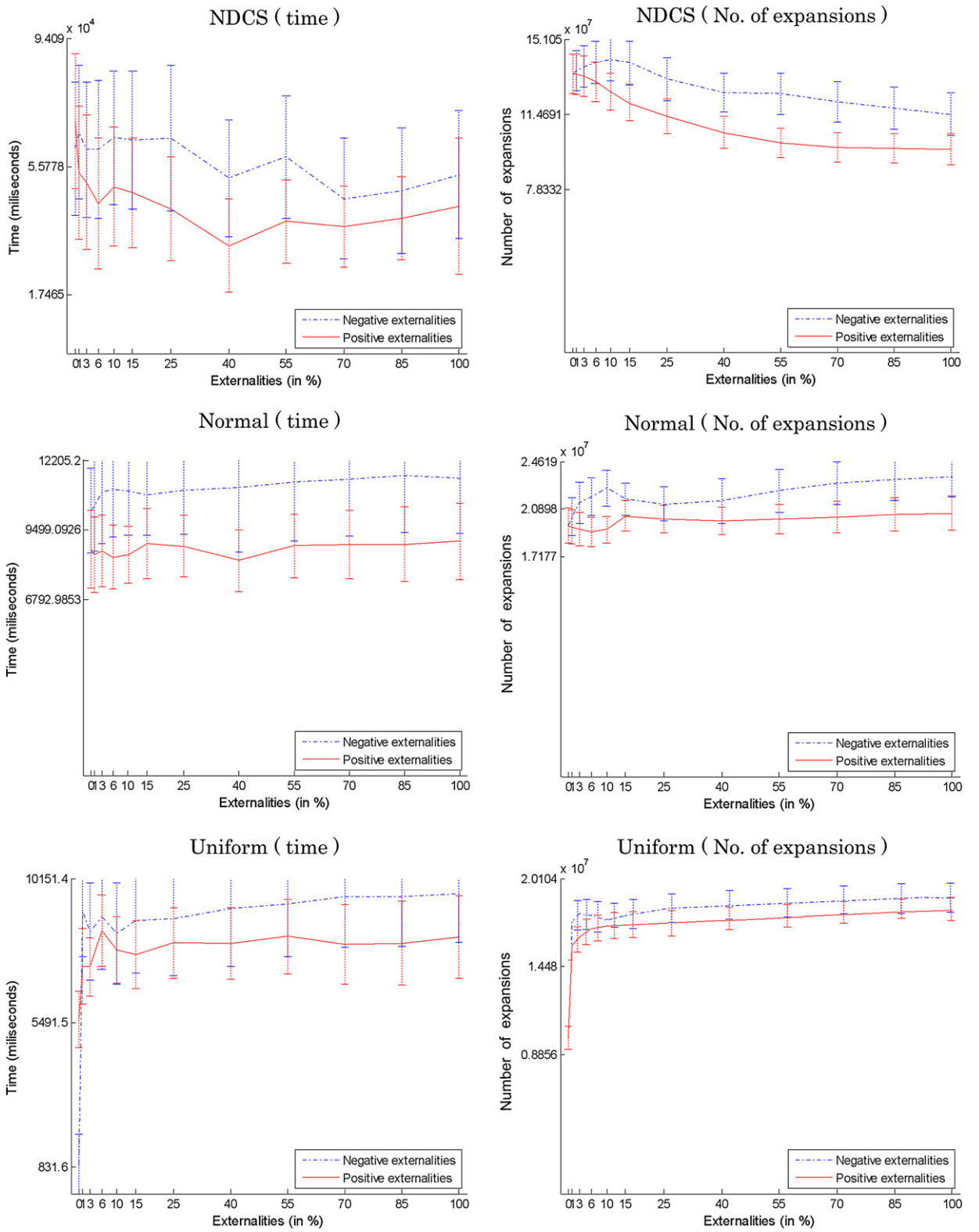

Fig. 8. Given 20 agents, the figure shows the total run time of $IP^{+/-}$ (left-hand side) and the total number of expansions made by $IP^{+/-}$ in the search tree (right-hand side) given different percentages of externalities.

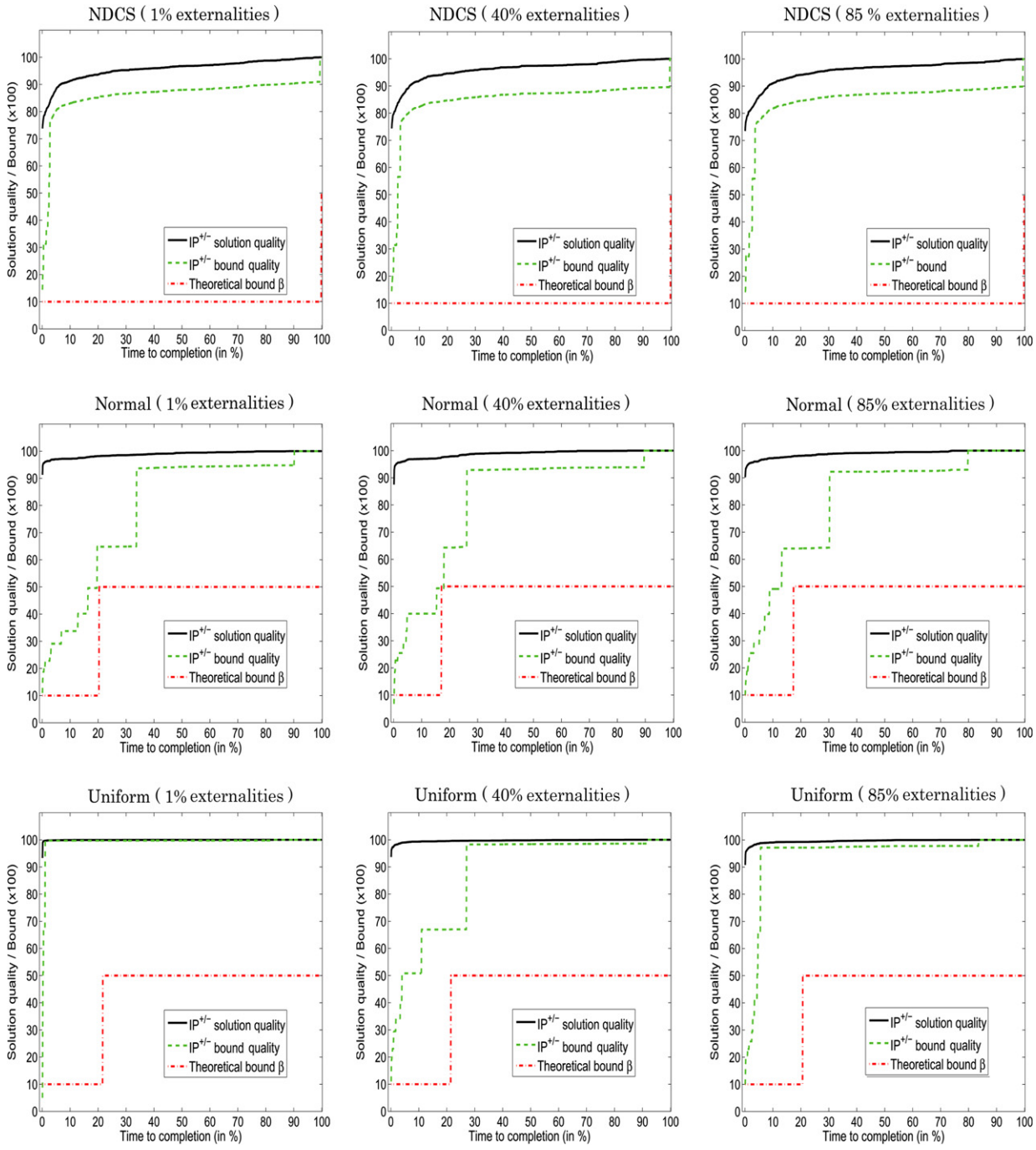

Fig. 9. Given 20 agents and negative externalities, the figure shows how the solution quality and bound quality improve during the run time of $IP^{+/-}$.

Fig. 9 shows how the solution quality and bound quality improve during run time, and that is given 20 agents and negative externalities.[18] In more detail, the X-axis in the figure corresponds to the percentage of time that has elapsed, with 0% being the time at which the algorithm starts, and 100% being the time at which it terminates. For every percentage of time t%, we report the following:

[18] Observer that this bound is calculated using information that was extracted from the problem instance at hand (unlike the bound that was calculated earlier based on the sequence of subspaces, which is independent of the problem instance). Thus, it is different from the bound that was calculated earlier based on the sequence of subspaces (which is independent of the problem instance).

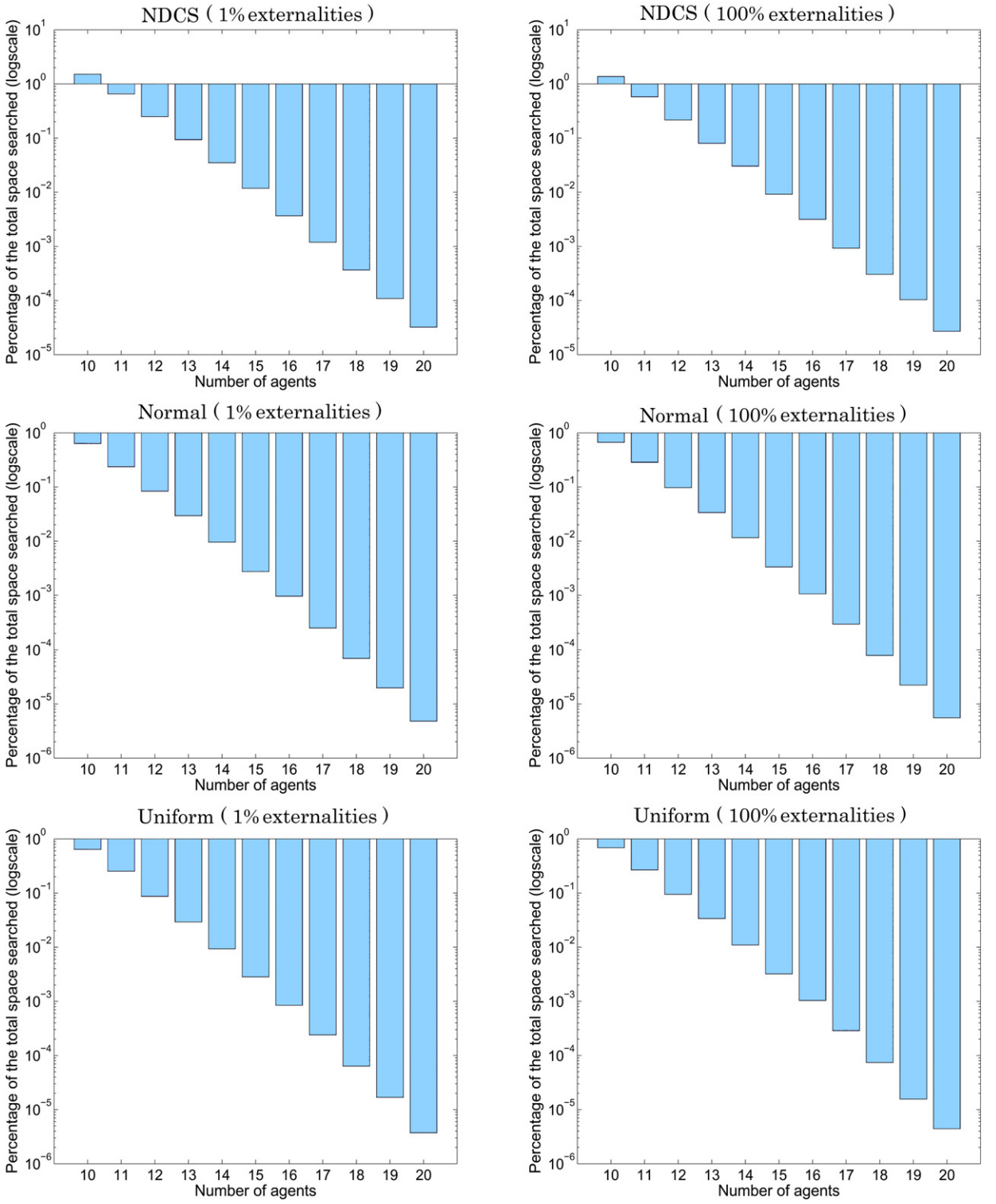

Fig. 10. Given negative externalities, the figure shows *on a log scale* the percentages of space that has been searched by $IP^{+/-}$ given different numbers of agents.

- $IP^{+/-}$ *solution quality*: This is computed as the ratio between the value of the best coalition structure (found at time t%) and the value of the optimal coalition structure (found at time 100%).
- $IP^{+/-}$ *bound quality*: This is computed as the ratio between the value of the best coalition structure found and the maximum upper bound of all remaining subspaces (i.e., those that were neither searched nor pruned).
- *Theoretical bound* β: This is computed as shown in Section 4.3.

As can be seen, the solution quality exceeds 90% in less than 1% of the run-time given the Normal and Uniform distributions, and less than 10% given the NDCS distribution. Moreover, the bound quality is significantly better than the theoretical, worst-case, bound.

Finally, given negative externalities, Fig. 10 shows *on a log scale* the percentage of space that has been searched by $IP^{+/-}$ given different numbers of agents. This is computed as the ratio between the number of expansions made by the algorithm and the total number of expansions in the search space. As can be seen, the algorithm only searches an extremely small fraction of the space (e.g., less than 0.00001% given 20 agents). Furthermore, this fraction gets consistently smaller as the number of agents increases.

6. Conclusions

The Coalition Structure Generation (CSG) problem within the AI community has been studied almost exclusively in the context of characteristic function games where no externalities are allowed. This is despite the fact that, in many real-world settings, externalities do exist and need to be accounted for. Taking externalities into consideration makes the CSG problem significantly more challenging as the input size grows from $O(2^n)$ to $O(n^n)$.

Against this background, we provide in this paper the first computational study on CSG in partition function games. More specifically, we develop a novel, anytime algorithm that solves the CSG problem in two important classes of these games, namely PFG^+ and PFG^-, where externalities are weakly positive in PFG^+ and weakly negative in PFG^-. Our algorithm capitalizes upon the following theoretical results. Firstly, we prove that it is possible to bound the values of any group of coalitions in PFG^+/PFG^-. Secondly, we identify the set of coalition structures that has to be searched in order to establish a worst-case bound on solution quality. Thirdly, we prove that, by traversing the space of possible solutions in a particular manner, we are guaranteed to improve this worst-case bound. Fourthly, we show that, by satisfying certain conditions, it is possible to prune parts of the search space.

To test our algorithm, we propose an equation for generating random input instances, and prove that the resulting partition function is guaranteed to satisfy the conditions of PFG^+/PFG^-. We also prove that this function, which consists of $O(n^n)$ values, only requires storing $O(2^n)$ values in memory. This function can be used as a standard benchmark for evaluating any potential CSG algorithms that could be developed in the future for PFG^+/PFG^-.

Since there are no previous CSG algorithms for games with externalities, we benchmark our algorithm against other state-of-the-art approaches in games where no externalities are present. Surprisingly, we find that, as far as worst-case guarantees are concerned, our algorithm outperforms the others by orders of magnitude despite the fact that they take advantage of the special properties of games with no externalities, while our algorithm does not.

With this paper, we have laid the foundations for further work on the CSG problem in partition functions games. In particular, we are keen to develop efficient CSG algorithm for classes of *PFG*s other than those considered in this paper. Indeed, building upon our theoretical results, the first such attempt was recently proposed by Banerjee and Landon [31], where externalities are influenced by the agents' types. However, algorithms for more general classes are yet to be developed. Another interesting direction is to extend our idea of generating random instances of *PFG*s using only $O(n2^n)$ space, and to use this idea as a building block for developing concise representations of various classes of *PFG*s.

Acknowledgements

This article is a significantly revised and extended version of [32] and [15]. Specifically, we present in this paper a more comprehensive review of the CSG literature. Moreover, while our procedure for reducing the ratio bound was only briefly described in [32], here we provide the pseudo code of this procedure, along with detailed explanations and extensive examples. We also provide a mathematical proof for a function that generates random instances of PFG^+/PFG^- game classes, and discuss its memory requirements. Furthermore, the evaluation section is substantially extended. In particular, we analyze the effect that different amounts of externalities have on the run-time of the algorithm, and on the number of expansions made by the algorithm. We also show how the solution quality and worst-case guarantee improve during run-time. Finally, we analyze the percentage of space that is searched by our algorithm given different numbers of agents and different value distributions.

Tomasz Michalak and Michael Wooldridge acknowledge support from the EPSRC under the project Market Based Control of Complex Computational Systems (GR/T10657/01). Tomasz Michalak acknowledges support from the EPSRC under the project ALADDIN (Autonomous Learning Agents for Decentralised Data and Information Systems) project jointly funded by a BAE Systems and EPSRC strategic partnership.

Appendix A. Summary of Notation

A	The set of agents.
a_i	Agent in A.
n	The number of agents in A.
C	A coalition.
$\overline{C}$	Agents in A that do not belong to C, i.e., $\overline{C} = A \backslash C$.
$\mathcal{C}$	The set of all coalitions over A.
$\mathcal{C}_s$	The set of all coalitions of size s.
CS	A coalition structure.
CS^*	An optimal coalition structure.
$\mathcal{P}^A$	The set of all coalition structures.
$\mathcal{P}^A_s$	The set of all coalition structures of size s, i.e., level s in the coalition structure graph.
$\mathcal{P}^A_I$	The set of all coalition structures in which coalition sizes match the parts in integer partition I.
$\mathcal{P}^C$	The set of all partitions of coalition C.
$\mathcal{P}^C_I$	The set of all partitions of C in which coalition sizes match the parts in integer partition I.
P	Partition in $\mathcal{P}^C$.
p	Element of a partition P.
(C, CS)	Coalition C embedded in coalition structure CS.
$w(C, CS)$	The value of C in CS, i.e., the value of embedded coalition (C, CS).
$W(P, CS)$	The value of P in CS.
$W(CS)$	The value of coalition structure CS.
$v(C)$	The value of coalition C.
UB^A_I	Upper bound on the value of the best CS in $\mathcal{P}^A_I$.
UB^C_I	Upper bound on the values of the partitions in $\mathcal{P}^C_I$.
UB^C	Upper bound on the value of coalition C.
UB^P	Upper bound on the value of partition P.
LB^A_I	Lower bound on the value of the best CS in $\mathcal{P}^A_I$.
LB^C_I	Lower bound on the values of the partitions in $\mathcal{P}^C_I$.
LB^C	Lower bound on the value of coalition C.
LB^P	Lower bound on the value of partition P.
$\mathcal{I}^k$	The set of all possible integer partition of number k.
$G(\mathcal{I}^k)$	The integer partition graph of k.
$G(\mathcal{I}^k_s)$	Level s in the integer partition graph of k.

References

[1] T.W. Sandholm, K. Larson, M. Andersson, O. Shehory, F. Tohme, Coalition structure generation with worst case guarantees, Artificial Intelligence (AIJ) 111 (1–2) (1999) 209–238.
[2] V.D. Dang, N.R. Jennings, Generating coalition structures with finite bound from the optimal guarantees, in: Proceedings of the Third International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS), 2004, pp. 564–571.
[3] S. Ieong, Y. Shoham, Marginal contribution nets: a compact representation scheme for coalitional games, in: EC-05, ACM, 2006, pp. 170–179.
[4] O. Shehory, S. Kraus, Methods for task allocation via agent coalition formation, Artificial Intelligence (AIJ) 101 (1–2) (1998) 165–200.
[5] S. Sen, P. Dutta, Searching for optimal coalition structures, in: Proceedings of the Sixth International Conference on Multi-Agent Systems (ICMAS-00), 2000, pp. 286–292.
[6] T. Rahwan, S.D. Ramchurn, A. Giovannucci, N.R. Jennings, An anytime algorithm for optimal coalition structure generation, Journal of Artificial Intelligence Research (JAIR) 34 (2009) 521–567.
[7] W. Lucas, R. Thrall, n-Person games in partition function form, Naval Research Logistics Quarterly X (1963) 281–298.
[8] E. Catilina, R. Feinberg, Market power and incentives to form research consortia, Review of Industrial Organization 28 (2) (2006) 129–144.
[9] J. Plasmans, J. Engwerda, B. van Aarle, G.D. Bartolomeo, T. Michalak, Dynamic Modelling of Monetary and Fiscal Cooperation Among Nations, Springer, New York, USA, 2006.
[10] S.-S. Yi, Endogenous formation of economic coalitions: a survey on the partition function approach, in: The Endogenous Formation of Economic Coalitions, Edward Elgar, London, UK, 2003, pp. 80–127.
[11] M. Finus, New developments in coalition theory: an application to the case of global pollution control, in: Environmental Policy in an International Perspective, Kluwer Academic Publishers, Netherlands, 2003, pp. 19–49.
[12] P. Dunne, Multiagent resource allocation in the presence of externalities, in: CEEMAS, 2005, pp. 408–417.
[13] J. Oh, Multiagent social learning in large repeated games, Ph.D. thesis, School of Computer Science, Carnegie Mellon University, 2009.
[14] J.S. Rosenschein, G. Zlotkin, Rules of Encounter: Designing Conventions for Automated Negotiation among Computers, MIT Press, MA, USA, 1994.
[15] T.P. Michalak, T. Rahwan, J. Sroka, A. Dowell, M.J. Wooldridge, P.J. McBurney, N.R. Jennings, On representing coalitional games with externalities, in: J. Chuang, L. Fortnow, P. Pu (Eds.), ACM Conference on Electronic Commerce (ACM-EC), 2009, pp. 11–20.
[16] R. Cornes, T. Sandler, The Theory of Externalities, Public Goods, and Club Goods, Cambridge University Press, 1996.
[17] T. Rahwan, S.D. Ramchurn, A. Giovannucci, V.D. Dang, N.R. Jennings, Anytime optimal coalition structure generation, in: Proceedings of the Twenty Second Conference on Artificial Intelligence (AAAI), 2007, pp. 1184–1190.
[18] D.Y. Yeh, A dynamic programming approach to the complete set partitioning problem, BIT Numerical Mathematics 26 (4) (1986) 467–474.
[19] M.H. Rothkopf, A. Pekec, R.M. Harstad, Computationally manageable combinatorial auctions, Management Science 44 (8) (1995) 1131–1147.

[20] T. Rahwan, N.R. Jennings, An improved dynamic programming algorithm for coalition structure generation, in: Proceedings of the Seventh International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS), 2008, pp. 1417–1420.
[21] T. Rahwan, N.R. Jennings, Coalition structure generation: dynamic programming meets anytime optimisation, in: Proceedings of the Twenty Third Conference on Artificial Intelligence (AAAI), 2008, pp. 156–161.
[22] T.C. Service, J.A. Adams, Approximate coalition structure generation, in: Proceedings of the Twenty Fourth Conference on Artificial Intelligence (AAAI), 2010.
[23] T.C. Service, J.A. Adams, Constant factor approximation algorithms for coalition structure generation, Autonomous Agents and Multi-Agent Systems 23 (1) (2011) 1–17.
[24] N. Ohta, V. Conitzer, R. Ichimura, Y. Sakurai, A. Iwasaki, M. Yokoo, Coalition structure generation utilizing compact characteristic function representations, in: I.P. Gent (Ed.), CP'09, in: Lecture Notes in Computer Science, vol. 5732, Springer, ISBN 978-3-642-04243-0, 2009, pp. 623–638.
[25] T. Rahwan, T.P. Michalak, E. Elkind, P. Faliszewski, J. Sroka, M. Wooldridge, N.R. Jennings, Constrained coalition formation, in: Twenty Fifth AAAI Conference on Artificial Intelligence (AAAI), 2011, pp. 719–725.
[26] S. Ueda, M. Kitaki, A. Iwasaki, M. Yokoo, Concise characteristic function representations in coalitional games based on agent types, in: IJCAI'11: Twenty Second International Joint Conference on Artificial Intelligence, 2011, pp. 393–399.
[27] T. Rahwan, S.D. Ramchurn, V.D. Dang, N.R. Jennings, Near-optimal anytime coalition structure generation, in: Proceedings of the Twentieth International Joint Conference on Artificial Intelligence (IJCAI), 2007, pp. 2365–2371.
[28] S.S. Skiena, The Algorithm Design Manual, Springer-Verlag, New York, USA, 1998.
[29] W. Riha, K.R. James, Algorithm 29 efficient algorithms for doubly and multiply restricted partitions, Journal of Computing 16 (1974) 163–168.
[30] K. Larson, T. Sandholm, Anytime coalition structure generation: an average case study, Journal of Experimental and Theoretical Artificial Intelligence 12 (1) (2000) 23–42.
[31] B. Banerjee, K. Landon, Coalition structure generation in multi-agent systems with mixed externalities, in: Proceedings of the Ninth International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS), 2010, pp. 175–182.
[32] T. Rahwan, T. Michalak, N.R. Jennings, M. Wooldridge, P. McBurney, Coalition structure generation in multi-agent systems with positive and negative externalities, in: Proceedings of the Twenty First International Joint Conference on Artificial Intelligence (IJCAI), Pasadena, USA, 2009.