An Efficient Vector–Based Representation for Coalitional Games

Long Tran–Thanh, Tri–Dung Nguyen, Talal Rahwan,
Alex Rogers and Nicholas R. Jennings
University of Southampton
{ltt08r, tr, acr, nrj}@ecs.soton.ac.uk
t.d.nguyen@soton.ac.uk

Abstract

We propose a new representation for coalitional games, called the *coalitional skill vector* model, where there is a set of *skills* in the system, and each agent has a skill vector—a vector consisting of values that reflect the agents' level in different skills. Furthermore, there is a set of *goals*, each with requirements expressed in terms of the minimum skill level necessary to achieve the goal. Agents can form coalitions to aggregate their skills, and achieve goals otherwise unachievable. We show that this representation is fully expressive, that is, it can represent any characteristic function game. We also show that, for some interesting classes of games, our representation is significantly more compact than the classical representation, and facilitates the development of efficient algorithms to solve the coalition structure generation problem, as well as the problem of computing the core and/or the least core. We also demonstrate that by using the coalitional skill vector representation, our solver can handle up to 500 agents.

1 Introduction

Coalition formation is an important research topic in artificial intelligence, as it studies situations where autonomous agents can cooperate and coordinate their activities to improve their performance. It has a wide range of applications, including, among others, increasing the throughput of cognitive radio networks [Khan *et al.*, 2010], optimising energy consumption in smart grids [Robu *et al.*, 2012] and aggregating the demands of buyers to obtain price discounts [Li *et al.*, 2010].

A formal model of a coalition formation scenario is called a cooperative game. Here, given a set of agents, A, each subset $C \subseteq A$ is called a coalition, and each partition of A (into disjoint and exhaustive coalitions) is called a coalition structure. A class of cooperative games that has received considerable attention in the literature is called *characteristic function games (CFGs)*, where the assumption is that the worth (or *value*) of any coalition, $C \subseteq A$, can be *"represented"* using a single, real value. It is also assumed that this value depends solely on the identities of the agents in C. To be more precise, CFGs assume the existence of a *characteristic function*, $v : 2^A \rightarrow \mathbb{R}$, which, given a coalition, returns its value. Arguably, the most fundamental problems that have been studied in this class of games are:

1. **Coalition Structure Generation**. How to efficiently search through the many possible coalition structures, and find the one(s) that maximize the sum of coalition values. This problem has received significant attention in recent years [Larson and Sandholm, 2000; Rahwan, 2007; Keinänen, 2009; Mauro *et al.*, 2010].
2. **Payoff Distribution**. How should the gains from cooperation be distributed among the coalition members. Several solution concepts have been proposed to answer this question, with the most common ones in the literature being the *Shapley value* [Shapley, 1953] and the *core* [Gillies, 1959]. While the focus of the former values is on *fairness*, the core focuses on *stability*, *i.e.*, how to identify solutions from which no group of agents has an incentive to deviate.

In this paper, the two optimisation problems that we will be focusing on are the coalition structure generation problem, and the problem of identifying stable solutions (i.e., solutions that are in the core), as they represent two important problems within the coalition formation domain. While those two problems are very interesting to study, their use in real-world applications has been limited by their inherent combinatorial nature, which often makes them unsolvable in practice except for a small number of agents. To be more precise, the problem of determining whether there exists stable solutions is NP-complete [Conitzer and Sandholm, 2003]. Likewise, the coalition structure generation problem is NP-hard [Sandholm *et al.*, 1999], and while a number of *exact* algorithms (i.e., algorithms that provide an optimal solution) have been developed to solve this problem, none of them can handle more than 30 agents [Rahwan *et al.*, 2012; Michalak *et al.*, 2010].

By studying the reason behind this intractability, many researchers realised that a key part of the problem comes from using the classical characteristic function representation. Specifically, in this representation, no assumptions are made on how a coalition's value is calculated. Instead, it is often assumed that the values of all coalitions are stored in one big table (which provides instant access to every coalition's value). This leads to several major limitations. First, memory requirement grows exponentially with the number of agents. Second, algorithms that are based on this representation often cannot exploit any extra information that might be available

about the way a coalition's value is computed. Consider, for example, a scenario where every coalition's value depends solely on the coalition's size. Such information can significantly reduce the difficulty of solving the aforementioned optimisation problems. Often, however, an algorithm designed to deal with the classical representation cannot readily and/or efficiently exploit such information. While at first glance it might appear that the most widely-applicable algorithms are those that place no assumption on the value function (because they can be applied with any function), in practice it turns out such algorithms are in fact the least applicable. This is because, by being too general, they lose the ability to scale. Those arguments form the rationale behind a line of research that focuses on developing alternative representations that can efficiently capture situations where the characteristic function has some structure [Conitzer and Sandholm, 2004; Ieong and Shoham, 2006; Elkind *et al.*, 2009; Ohta *et al.*, 2009]. In this context, an alternative representation should ideally have the following properties:

- **Expressiveness:** The ability to represent any characteristic function game. A representation with such an ability is said to be *"complete"* or *"fully expressive"*.
- **Conciseness:** The ability to represent certain classes of games *"concisely"*, i.e., without the need to store all $2^{|A|}$ values.
- **Efficiency:** The ability to facilitate the development of efficient algorithms for solving the aforementioned optimisation problems.

Against this background, we develop an novel representation, called the *coalitional skill vector* (CSV) model, where there is a set of *skills* in the system, and each agent has a *skill vector*—a vector consisting of values that reflect its level at mastering different skills. Furthermore, there is a set of tasks, each with requirements expressed in terms of the minimum skill levels necessary to achieve the task. The set of skill vectors that satisfy these requirements are termed as *set of goals*. While an agent on its own might not possess the required skill level to achieve a particular task (i.e. to have its skill vector to be within the set of goals), the agents may form coalitions that are capable of achieving such tasks (because every coalition's skill vector is an aggregation of the skill vectors of its members). Put differently, by forming coalitions, the agents can jointly produce coalitions with skill vectors that are within the goal set. This is a generalization of *coalitional skill games* [Bachrach and Rosenschein, 2008], where an agent's possession of a skill, and a task's dependence on skill, are both represented using binary numbers. We argue that, by incorporating the level of mastering different skills, the model becomes more realistic. This generalization is discussed more formally in subsequent sections.

We analyse the skill vector representation, and show that:

- It is fully expressive, i.e., it can represent any characteristic function game;
- For certain classes of games, such as the coalitional skill games, it is significantly more concise than the classical representation;
- It facilitates the development of efficient algorithms to solve the coalition structure generation problem, as well as the problem of computing the core and/or the *least core* using a constraint generation linear programming technique;
- We empirically evaluate the efficiency of our representation and solver, and show that for certain classes of games (i.e., games with convex goal sets and piecewise–linear value functions), it can solve problems with 500 agents, while existing exact algorithms can only handle 30 agents or so.

The remainder of the paper is as follows. Section 2 provide the necessary background. Section 3 formally introduces our CSV model. Section 4 discusses the coalition structure generation issues in more detail, while Section 5 discusses the core-related issues. Section 6 concludes.

2 Background

In this section, we outline the main notation and definitions (Section 2.1), and then outline previous models that incorporated skills into coalitional games (Section 2.2).

2.1 Preliminaries

A *characteristic function game* (CFG) is a tuple CFG(A, v), where $A = \{1, \ldots, n\}$ is a set of agents and $v : 2^A \rightarrow \mathbb{R}$ is a *characteristic function* that maps each subset, or *coalition*, of agents to a real number. A *coalition structure*, is a set of coalitions, $CS \subseteq 2^A$ such that:

- $\bigcup_{C \in CS} C = A$, and
- $C \cap C' = \emptyset$ for any $C, C' \in CS : C \neq C'$.

The set of all coalition structures over A will be denoted $\mathcal{CS}^A$. An *outcome* of a game is a pair, $(CS, \mathbf{x})$, where $CS \in \mathcal{CS}^A$ is a coalition structure, and $\mathbf{x} = (x_1, \ldots, x_n)$ is a *payoff vector*, which specifies how the value of each coalition $C \in CS$ is distributed among its members, *i.e.*, x_i specifies the payoff of agent i in CS. An outcome must satisfy $x_i \geq 0$ for all $i = 1, \ldots, |CS|$, and satisfy $\sum_{i \in C} x_i = v(C)$ for all $C \in CS$.

Definition 1 *Given a characteristic function game, CFG(A, v), the* coalition structure generation (CSG) problem *is the problem of finding a coalition structure in:* $\arg\max_{CS \in \mathcal{CS}^A} \sum_{C \in CS} v(C)$

Definition 2 *The* core *of a characteristic function game, CFG(A, v), is the set of all outcomes, $(CS, \mathbf{x})$, that satisfy $\sum_{i \in C} x_i \geq v(C)$ for any $C \subseteq A$. Similarly, the* ε-core *is the one that satisfies $\sum_{i \in C} x_i \geq v(C) - \varepsilon$ for any $C \subseteq A$.*

An outcome in the core is said to be *"stable"* since no agent has an incentive to deviate from it. Unfortunately, there are cases where the core happens to be empty. As for the ε-core, it is always non-empty given a sufficiently large value of ε. Of course, in practice we are usually interested in finding the smallest value of ε such that the ε-core is non-empty. The corresponding ε-core is called the *least core*. More formally:

Definition 3 *Given a characteristic function game, CFG(A, v), if we define ε^* as follows:*

$$\varepsilon^* = \inf\{\varepsilon \mid \varepsilon\text{-core of } (A, v) \text{ is non-empty}\},$$

then the least core *of CFG(A, v) is defined as ε^*-core.*

2.2 Skills & Coalitional Games

In this subsection, we briefly outline previous models that incorporated skills into coalitional games. To this end, Ohta *et al.* [2006] proposed a model where there is a set of skills, S, and each agent $i \in A$ has a subset of those skills, $S^i \subseteq S$. Moreover, there is a skill-value function $u : 2^S \rightarrow \mathbb{R}$ which assigns a real value to every subset of skills. The value of a coalition, $C \subseteq A$, is then equal to: $u(\cup_{i \in C} S^i)$.

Bachrach and Rosenschein [2008] incorporated into this model a set of tasks, T. Each task, $t \in T$, has a skill requirement $S^t \subseteq S$. A coalition $C \subseteq A$ can achieve a task t if it has all the required skills, i.e., if $S^t \subseteq \cup_{a_i \in C} S^{a_i}$. A task-value function, $w : 2^T \rightarrow \mathbb{R}$, specifies the payoff that can be obtained by a coalition when it achieves a given subset of tasks. The value of a coalition C is then given by

$$v(C) = w(\{t \mid S^t \subseteq \cup_{a_i \in C} S^{a_i}\}).$$

Such games are known as *coalitional skill games* [Bachrach and Rosenschein, 2008]. Given special classes of these games (e.g., the class where $w(T') = |T'|$ for any $T' \subseteq T$), the authors studied the computational complexity of several problems, such as: (1) calculating the core, (2) testing whether the core is empty, and (3) computing the Shapley value.

In a follow-up publication, Bachrach *et al.* [2010] studied the coalition structure generation problem in coalitional skill games. They showed that the problem can be solved in time polynomial in the number of agents and skills, but exponential in the number of tasks and in the treewidth of a certain, graphical representation of the agents' skills. This implies that the CSG problem is still computationally expensive within the formulation of coalitional skill games.

Our model is a generalization of coalitional skill games; it relaxes the assumption that the dependence between skills and tasks is a binary relation between S and T. Instead, this dependence is represented in our model as a mapping from $S \times T$ to $\mathbb{R}$. This is formalised in the following section.

3 The Coalitional Skill Vector Game Model

In this section, we introduce our model—the coalitional skill vector games—and demonstrate that it is fully expressive, concise, and efficient.

Let CSV$(A, m, \{\boldsymbol{r}_i\}, f)$ denote a coalitional skill vector model as follows. Let $1, 2, \ldots, n \in A$ denote the agents within the system, where n is the number of agents. We assign an m dimensional *skill vector* $\boldsymbol{r}_i = (r_{i1}, \ldots, r_{im})$ to agent i, where the value $r_{ij} \in \mathbb{R}$ represents agent i's level at mastering skill j. Here, m denotes the number of skills we take into account within our model. Furthermore, for any agents i and k, if $r_{ij} < r_{kj}$ then we say that agent k is better than agent i at skill j. Let $\boldsymbol{R} \in \mathbb{R}^{m \times n}$ be the skill matrix with columns $\boldsymbol{r}_i$ and row $\boldsymbol{r}_j$.

Let $C \subseteq \{1, 2, \ldots, n\}$ denote a coalition of agents. Then the skill vector of C is the sum of skill vectors of agents from C (i.e., the superposition of the corresponding vectors). That is, we have $\boldsymbol{r}(C) = \sum_{i \in C} \boldsymbol{r}_i$, where $\boldsymbol{r}(C)$ is the skill vector of coalition C. In addition, let $d(\boldsymbol{r}_1, \boldsymbol{r}_2)$ define the distance between skill vectors $\boldsymbol{r}_1$ and $\boldsymbol{r}_2$ in some norm L.

The agents' goal is to achieve a number of given tasks $T' \subseteq T$. To do so, the agents have to meet certain skill level requirements, that can be expressed as a set of skill vectors $\boldsymbol{G}$. Put differently, let $\boldsymbol{G} \subseteq \mathbb{R}^m$ denote the set of skill vectors that are suitable for successfully achieving the aforementioned tasks. For the sake of simplicity, hereafter we refer to $\boldsymbol{G}$ as the set of *goals*. In other words, the agents' *"goal"* is to (collectively) reach a certain level of skills that is sufficient to achieve certain tasks. With the use of $\boldsymbol{G}$, we define the value of coalition C's skill vector as follows:

$$v\left(\boldsymbol{r}(C)\right) = f\left(d\left(\boldsymbol{r}(C), \boldsymbol{G}\right)\right), \tag{1}$$

where $d\left(\boldsymbol{r}(C), \boldsymbol{G}\right) = \min_{\boldsymbol{g} \in \boldsymbol{G}} d(\boldsymbol{r}(C), \boldsymbol{g})$ is the distance of $\boldsymbol{r}(C)$ from $\boldsymbol{G}$, and $f : \mathbb{R}^+ \cup \{0\} \rightarrow \mathbb{R}$ is a value function of d. The intuition behind the CSV model is that each agent is represented by its own skill vector, which reflects the skill level of the particular agent. By forming coalitions, these skill values are added together. A coalition formation is then motivated by the desire of agents to get into the goal set. The coalition's value is defined as a function over the distance of the coalition's skill vector and $\boldsymbol{G}$.

Given the described model, we show that the CSV game representation is fully expressive. That is, it is equivalent to that of the class of CFG (i.e., characteristic function games). Thus, for any instance of CFG defined in Section 2, there exists a skill vector model that is identical to the CFG. This is guaranteed by the following theorem:

Theorem 1 *For any instance of CFG, there exists an equivalent skill vector game CSV on the same set of agents A, where for any feasible coalition $C \subseteq CFG$, $v_{CSV}(\boldsymbol{r}(C))$ is equal to the value of S within CFG (i.e., $v_{CFG}(C)$).*

Proof sketch: Consider a skill vector model M where for any pair of agents i, j, the corresponding skill vectors are orthogonal, that is, $\boldsymbol{r}_i \boldsymbol{r}_j = 0$. Given this, for any coalition C, $\boldsymbol{r}(C)$ is unique. This implies that there exists a vector $\boldsymbol{g} \in \mathbb{R}^m$ such that $d(\boldsymbol{r}(C), \boldsymbol{g})$ is unique for each coalition C. We set the set of goals $\boldsymbol{G}$ to be $\{\boldsymbol{g}\}$. Now, we define f as follows: for each C, we have $f(d(\boldsymbol{r}(C), \boldsymbol{g})) = v(C)$, where $v(C)$ is the value of coalition C given in the coalition formation game CFG. □

We now turn to the discussion of the model's conciseness. Arguably, in many realistic situations, the set of goals $\boldsymbol{G}$ and the value function f naturally happen to be concise (i.e., can be expressed in a closed form). In particular, in many cases, the set of goals is typically convex (e.g., each skill has to satisfy some thresholds), and thus, can be expressed in a closed form (e.g., with a set of constraints that have to be satisfied). In addition, the value function is typically piece–wise linear, polynomial, or exponential (or a combination of these) with a finite number of "pieces" (for more details, see, e.g., Khan *et al.* 2010, Robu *et al.* 2012, or Li *et al.* 2010). Given this, within these situations, we only need the set of each agent's skill vector and the closed formulae of the value function and set of goals to describe the CSV model. This type of representation is indeed significantly more concise, compared to the original CSF model, as it does not need all the $2^{|A|}$ coalition values for description.

4 Coalition Structure Generation in CSVs

Given the CSV model above, we now turn to the discussion of the computational efficiency of the model. To do so, we fo-

cus in this section on the coalition structure generation (CSG) problem in CSVs. In particular, we show that the CSG problem with the CSV model naturally lends itself to a mixed integer linear programming (MILP) problem. However, this MILP model is still computationally expensive, as it has to deal with an exponential number of constraints. To overcome this issue, we propose our constraint generation based method in Section 4.1, and demonstrate its computational efficiency in Section 4.2.

Note that there are 2^n possible coalitions in a characteristic function game with n agents. Let these coalitions be $\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_{2^n}$ and let $A = \{1, \ldots, n\}$ be the grand coalition. Let $\boldsymbol{A} \in \mathbb{R}^{n \times 2^n}$ be a matrix with element a_{ij} in row i, column j, that indicates whether agent i is in coalition $\mathcal{C}_j$. Let $\boldsymbol{a}_j = (a_{1j}, \ldots, a_{nj})^t$ be the j column of $\boldsymbol{A}$. For convenience in notation, we denote $v_j \equiv v(\boldsymbol{a}_j) \equiv v(\mathcal{C}_j)$ as the payoff of coalition $\mathcal{C}_j$.

In the coalition structure generation (CSG) problem, we want to divide the agents into a disjoint set of coalitions. Thus, each agent will appear in at most one coalition, such that the total value the coalitions is maximized. Let $\boldsymbol{x} = (x_1, x_2, \ldots, x_{2^n})$ be a vector of binary variables that indicate whether coalitions $\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_{2^n}$ are in the coalition structure. Then the set of feasible coalition structures is:

$$\mathcal{X} = \left\{ \boldsymbol{x} \mid \boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{e},\ \boldsymbol{x} \in \{0,1\}^{2^n} \right\},$$

where $\boldsymbol{e}$ is a column vector with all elements equal to one. The total value of a coalition structure $\boldsymbol{x}$ is $\boldsymbol{v}^T\boldsymbol{x} = \sum_{j=1}^{2^n} x_j v_j$. The CSG problem can be formulated as a MILP problem as follows:

$$\mathbf{CSG} := \max_{\boldsymbol{x} \in \mathcal{X}} \quad \boldsymbol{v}^T\boldsymbol{x}.$$

This MILP is very difficult to solve as it has an exponentially large number of binary variables. Instead, we aim to solve its linear relaxation version, i.e., allowing x_j to be fractional. Being able to efficiently solve the relaxed version, we can then use standard branch–and–bound techniques to find the exact optimal solution of the original LP. The LP relaxation problem can be described as follows:

$$\begin{aligned} \mathbf{CSG_r} := \max_{\boldsymbol{x}} \quad & \boldsymbol{v}^T\boldsymbol{x}, \\ s.t. \quad & \boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{e},\ \boldsymbol{x} \geq 0, \end{aligned}$$

where the constraint $\boldsymbol{x} \leq 1$ can be ignored because the constraint $\boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{e}$ has already enforced this. Notice, however, that the LP problem is not easy to solve in general because of the exponentially large number of decision variables. We will, however, show how to exploit the special structure of the CSV game to solve the LP relaxation problem efficiently for reasonably large games.

4.1 A Constraint Generation Method for $\mathbf{CSG_r}$

We formalise the dual of $\mathbf{CSG_r}$ as follows:

$$\begin{aligned} \mathbf{CSG_{rd}} := \min_{\boldsymbol{y} \geq 0} \quad & \boldsymbol{e}^t\boldsymbol{y}, \\ s.t. \quad & \boldsymbol{a}_j^t\boldsymbol{y} \geq v_j,\ \forall j \in 1, .., 2^n. \end{aligned} \tag{2}$$

The dual problem contains a decision variable $\boldsymbol{y} \in \mathbb{R}^n$ and an exponentially large number of constraints which make high computational cost. However, it is interesting to observe that typically only a small number of constraints are tight at the optimal solution [Desaulniers *et al.,* 2005]. This means the remaining non-binding constraints can be removed without changing the optimal solution. This leads to the very popular *constraint generation* method (see, for example, [Desaulniers *et al.*, 2005]) in the operations research literature, where we start with a relaxed problem of $\mathbf{CSG_{rd}}$ with a small set of constraints and then keep introducing violating constraints to the relaxed problem until all the constraints are satisfied. At that point, the optimal solution of the relaxed problem is also the optimal solution of the original problem. Formally, the constraint generation method for solving $\mathbf{CSG_{rd}}$ is:

Initialization step: Starting with any initial weight vector $\boldsymbol{y}^{(0)} \geq 0$, set the initial relaxed constraint set $\mathcal{J}^{(0)} \equiv \boldsymbol{I}$ as the identity matrix of size $(n \times n)$ and set $k = 0$.

Iterative steps:

1. Solve the **constraint generation problem**:
$$\boldsymbol{w} = \operatorname*{argmin}_{\boldsymbol{z} \in \{0,1\}^n} \left[\boldsymbol{z}^t\boldsymbol{y}^{(k)} - v(\boldsymbol{z}) \right].$$
2. If $\boldsymbol{w}^t\boldsymbol{y}^{(k)} - v(\boldsymbol{w}) \geq 0$, terminate the algorithm (because we have already found the optimal solution $\boldsymbol{y}^{(k)}$). Otherwise, add $\boldsymbol{w}$ to the relaxed constraint set, i.e., $\mathcal{J}^{(k+1)} = \{\mathcal{J}^{(k)}, \boldsymbol{w}\}$.
3. Set $k = k + 1$ and solve the **relaxed problem**:
$$\boldsymbol{y}^{(k)} = \operatorname*{argmin}_{\boldsymbol{y} \geq 0, \boldsymbol{z}^t\boldsymbol{y} \geq v(\boldsymbol{z}), \forall \boldsymbol{z} \in \mathcal{J}} \boldsymbol{e}^t\boldsymbol{y}.$$
4. Go back to step 1.

In step 1, we find a coalition that violates constraint (2) mostly for the given proposal $\boldsymbol{y}^{(k)}$. Here, a coalition is characterised by a binary indicator vector $\boldsymbol{z}$ where $z_i = 1$ if agent i is in the coalition and $z_i = 0$ otherwise. In step 2 we check the optimality condition. If the worst coalition does not violate (2) then all other coalitions satisfy this constraint and hence $\boldsymbol{y}^{(k)}$ is an optimal solution of $\mathbf{CSG_{rd}}$. Otherwise, we introduce the newly generated constraint $\boldsymbol{w}^t\boldsymbol{y} - v(\boldsymbol{w}) \geq 0$ to the relaxed problem. In step 3, we solve the updated relaxed problem to obtain a new proposal $\boldsymbol{y}^{(k)}$ before going back to step 1.

Notice that the relaxed problem is a LP with smaller size and is easy to solve. The key, and often difficult, part for a successful constraint generation algorithm is the ability to generate violating constraints efficiently. We will show that this is indeed the case in the CSV game with various forms of the value functions as follows:

$$v(\boldsymbol{z}) = K - \min_{\boldsymbol{g} \in \boldsymbol{G}} \ ||\boldsymbol{R}\boldsymbol{z} - \boldsymbol{g}||,$$

where $\boldsymbol{R}\boldsymbol{z} = \sum_i z_i \boldsymbol{r}_i \equiv \boldsymbol{r}(C)$ is the aggregated skill of coalition C and K is some appropriate constant such that $v(\emptyset) = 0$. The distance $d(\boldsymbol{r}(C), \boldsymbol{G})$ can be measured in various norms such as L_1, L_2, L_Q and L_∞. In this case, the constraint generation problem $min_{\boldsymbol{z} \in \{0,1\}^n} \quad \left[\boldsymbol{z}^t\boldsymbol{y}^{(k)} - v(\boldsymbol{z}) \right]$ can be reformulated as:

$$\min_{\boldsymbol{z} \in \{0,1\}^n} \boldsymbol{z}^t\boldsymbol{y}^{(k)} + \left[\min_{\boldsymbol{g} \in \boldsymbol{G}} \ ||\boldsymbol{R}\boldsymbol{z} - \boldsymbol{g}|| - K \right],$$

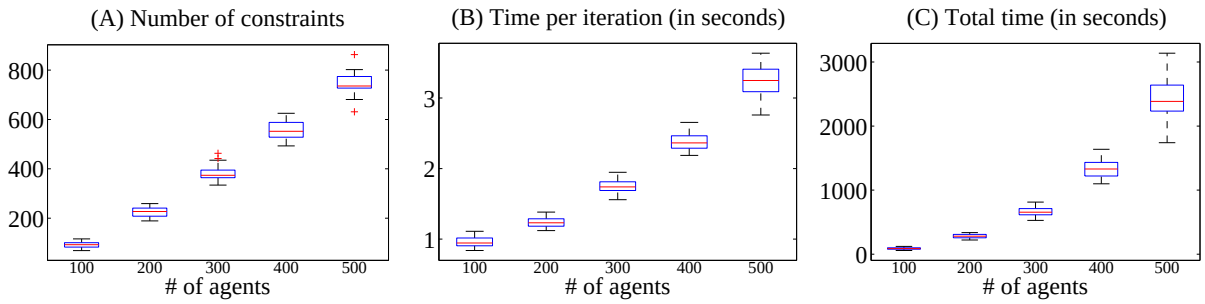

Figure 1: Numerical results for finding optimal fractional coalition structure generation of the CSV games.

which is equivalent to

$$\mathbf{CG} := \min_{\boldsymbol{z}\in\{0,1\}^n, \boldsymbol{g}\in\boldsymbol{G}} \boldsymbol{z}^t \boldsymbol{y}^{(k)} + ||\boldsymbol{R}\boldsymbol{z} - \boldsymbol{g}||,$$

by combining the two minimisation operators. Notice that the objective function is convex on $(\boldsymbol{z}, \boldsymbol{g})$. In fact, if the distance $d(\boldsymbol{r}(C), \boldsymbol{G})$ is measured in L_1 or L_∞ norms, then the problem can be reformulated as a mixed integer linear programming problem. This also applies to any piece-wise linear functions of the following form:

$$v(\boldsymbol{z}) = K - \max_{l\in 1,..,L} \left(a_l \min_{\boldsymbol{g}\in\boldsymbol{G}} \ ||\boldsymbol{R}\boldsymbol{z} - \boldsymbol{g})|| + b_l \right).$$

We will demonstrate these results with the L_1 distance measure.[1] To this end, in the L_1 norm case, we have $v(\boldsymbol{z}) = K - \min_{\boldsymbol{g}\in\boldsymbol{G}} \ \sum_{j=1}^m |\boldsymbol{r}_{j.}\boldsymbol{z} - g_j|$ with $\boldsymbol{r}_{j.}$ is the j-row of the skill matrix $\boldsymbol{R}$. The constraint generation problem becomes:

$$\min_{\boldsymbol{z}\in\{0,1\}^n, \boldsymbol{g}\in\boldsymbol{G}} \qquad \boldsymbol{z}^t \boldsymbol{y}^{(k)} + \sum_{j=1}^m |\boldsymbol{r}_{j.}\boldsymbol{z} - g_j|.$$

Let $\delta_j = |\boldsymbol{r}_{j.}\boldsymbol{z} - g_j|$, the problem can be reformulated as:

$$\begin{aligned} \min_{\boldsymbol{z},\boldsymbol{\delta},\boldsymbol{g}} \quad & \boldsymbol{z}^t \boldsymbol{y}^{(k)} + \sum_{j=1}^m \delta_j, \\ s.t. \quad & \delta_j \geq \boldsymbol{r}_{j.}\boldsymbol{z} - g_j, \ \forall j \in 1,..,m \\ & \delta_j \geq -\boldsymbol{r}_{j.}\boldsymbol{z} + g_j, \ \forall j \in 1,..,m \\ & \boldsymbol{z} \in \{0,1\}^n, \boldsymbol{g} \in \boldsymbol{G}. \end{aligned}$$

This is a MILP and thus, is typically NP-hard. However, in the following subsection we will show numerically that CPLEX—a state-of-the-art MILP solver—can solve the CSV game very efficiently for many instances with up to 500 agents. e an approximated solution is sufficient.

4.2 Numerical Results

We perform numerical tests on the algorithm for various settings using the L_2 measure.[2] In this case, problem **CG** is equivalent to a mixed integer second order cone problem [Drewes and Ulbrich, 2009]. We vary the number of agents n between 100 and 500 while fixing the skill dimension m at 5 (other values of dimension show similar results). In each test, we generate the skill vectors uniformly, i.e., $r_{ij} \sim U[0,1]$. For each fixed n, we generate K random instances of the problem so that we can test the robustness of the algorithm when varying the data.[3]. The goal set $\boldsymbol{G}$ is fixed as $\boldsymbol{G} = \{\boldsymbol{g} \ : \ g_j \geq 1\}$ [4]. This type of goal set, we argue, is natural in many real–world applications, as it represents the requirement that in order to achieve a goal, an agent (or a coalition of agents) has to satisfy a set of minimal skill levels. Note that we do not compare our algorithm against existing general coalition structure generation exact algorithms, as the latter are (by definition) designed to solve games with no underlying structure of the characteristic function, and thus, they can only solve up to 30 agents or so.

Figure 1 shows the performance of the constraint generation algorithm. In particular, the red lines depict the median values, the boxes represent the $75-25$ percentiles, and the red crosses are the outlier values. Sub-figure (A) shows the number of iterations the algorithm takes. This is also the number of constraints the algorithm generates in addition to n initial constraints. For a problem with 500 agents, the number of additional constraints generated is around 750. This means instead of solving a large LP with 2^{500} constraints, we only need to solve 750 smaller LPs, each with the number of constraints varying between 500 and 1250. It is interesting to note that the number of constraints generated, and hence the number of small LPs involved, grows linearly with the number of agents. Sub-figure (B) shows the time taken in each iteration for solving both the relaxed LP and for solving the constraint generation problem **CG**. Overall, there is a linear trend in the computational time as the number of agents increases. Sub-figure (C) shows the total computational time taken by the algorithm. This is equal to the product of the number of iterations and the computational time for each iteration. There is a quadratic trend in the total time as the number of agents increases. The algorithm took less than an hour to solve the largest instances with 500 agents[5].

[1] We can easily extend our results to models with other norms such as L_2 or L_∞ and for piece-wise linear functions.

[2] We have also used other norms, such as L_1 and L_∞, and the results are similar from a broad view.

[3] We set $K = 50$ for $n = \{100, 200, 300\}$, and $K = 10$ for $n = \{400, 500\}$. We run the Matlab function *rand('state',k)* with the random seed k varying between 1 and K for the K corresponding random instances before generating the random skill vectors. These fixed seeds are used for convenient replication and testing of these instances in the future.

[4] It is easy to extend the calculations to problem instances with other convex goal sets, such as cones, or hyperplanes.

[5] All the numerical tests appearing in this paper are performed on

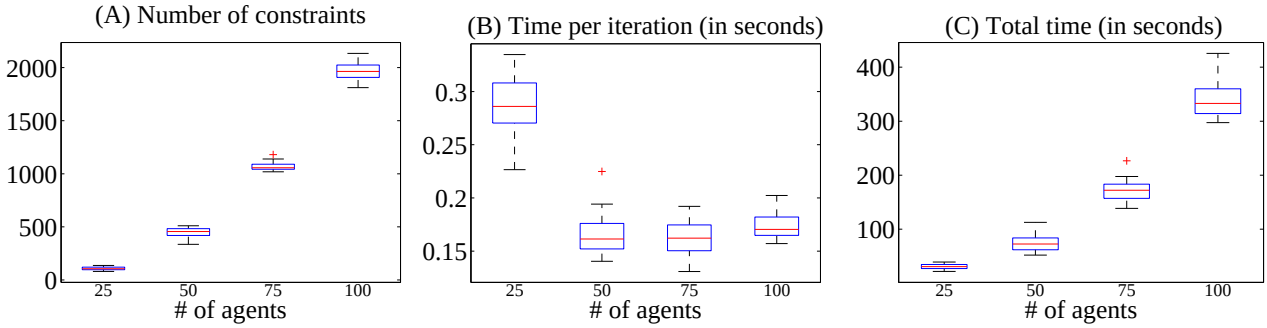

Figure 2: Numerical results for finding the least core of the CSV games.

5 Stable Payoff Distribution in CSVs

In this section, we present the computational method for finding the core and the least core in CSV games. To this end, we denote the core of a games as:

$$\text{Core} = \left\{ \boldsymbol{y} \mid \boldsymbol{a}_j^t \boldsymbol{y} \geq v(\boldsymbol{a}_j),\ \forall j \in 1,..,2^n, \boldsymbol{e}^t \boldsymbol{y} = v(A) \right\}.$$

A solution in the core can be found by finding a feasible solution in polyhedron Core. We notice that this constraint set is very similar to the feasible set of the CSG_{rd} problem defined in section 4, i.e., it also includes the same exponential set of constraints $\boldsymbol{a}_j^t \boldsymbol{y} \geq v(\boldsymbol{a}_j),\ \forall j \in 1,..,2^n$. Therefore, we can apply the same constraint generation technique to find a feasible solution (by optimising any arbitrary linear function). Note, however, that the core contains an additional constraint, $\boldsymbol{e}^t \boldsymbol{y} = v(A)$, that requires the total share of all the agents to be equal to the payoff that the grand coalition can achieve. This additional restriction might result in the core being empty. Hence, we focus on finding a solution within the least core, which can be formalised as follows:

$$\begin{aligned} \mathbf{LC} := \min_{\boldsymbol{y},\varepsilon} \quad & \varepsilon, \\ s.t. \quad & \boldsymbol{a}_j^t \boldsymbol{y} + \varepsilon \geq v(\boldsymbol{a}_j),\ \forall j \in 1,..,2^n, \\ & \boldsymbol{e}^t \boldsymbol{y} = v(A). \end{aligned}$$

The least core is always non-empty because we can choose large enough ε such that the constraint set is non-empty. To solve **LC**, we can apply the same constraint generation method for solving $\mathbf{CSG_{rd}}$ as follows:

Initialization step: Start with any initial weight vector $(\boldsymbol{y}^{(0)}, \varepsilon^{(0)})$ such that $\boldsymbol{e}^t \boldsymbol{y} = v(A)$, set the relaxed constraint set $\mathcal{J}^{(0)} = \boldsymbol{I}$ and set $k = 0$.

Iterative steps:

1. Solve the **constraint generation problem**:
$$\boldsymbol{w} = \underset{\boldsymbol{z} \in \{0,1\}^n}{\text{argmin}} \left[\boldsymbol{z}^t \boldsymbol{y}^{(k)} - v(\boldsymbol{z}) \right].$$
2. If $\boldsymbol{w}^t \boldsymbol{y}^{(k)} + \varepsilon^{(k)} - v(\boldsymbol{w}) \geq 0$, terminate the algorithm, we have already found the optimal solution $(\boldsymbol{y}^{(k)}, \varepsilon^{(k)})$. Otherwise, add $\boldsymbol{w}$ to the relaxed constraint set, i.e., $\mathcal{J}^{(k+1)} = \{\mathcal{J}^{(k)}, \boldsymbol{w}\}$.
3. Set $k = k + 1$ and solve the **relaxed problem**:
$$(\boldsymbol{y}^{(k)}, \varepsilon^{(k)}) = \underset{(\boldsymbol{y},\varepsilon)\ :\ \boldsymbol{e}^t \boldsymbol{y} = v(A), \boldsymbol{z}^t \boldsymbol{y} + \varepsilon \geq v(\boldsymbol{z}), \forall \boldsymbol{z} \in \mathcal{J}}{\text{argmin}} \varepsilon$$
4. Go back to step 1.

We perform numerical tests on finding the core of the CSV game for a number of instances with the number of agents ranging between 25 and 100. Figure 2 shows the performance of the constraint generation algorithm with sub-figures (A-C) showing the number of iterations, the computational time taken in each iteration and the total time the algorithm takes, respectively. The boxplots show the variation of these statistics over 20 random instances for each fixed number of agents. Overall, the computational time for each iteration is about the same as that for solving $\mathbf{CSG_{rd}}$ and this increases linearly with the number of agents. However, the time for computing the core is higher than that for solving $\mathbf{CSG_{rd}}$ due to a larger number of iterations required. Nevertheless, the total computational time follows a quadratic trend as the number of agents increase. The algorithm took less than 8 minutes to find the core for the case of 100 agents.

6 Conclusions

In this paper we introduced a new vector–based representation, called the CSV model, for CFGs. In more detail, this model assigns a skill vector to each agent, and a coalition's skill level can be expressed with the aggregation of the skill vector of agents from the coalition. The value of a coalition is then measured as a function of the distance between the corresponding coalition's skill vector and a set of goals. We showed that the CSV model is fully expressive, that is, it can represent any CFG. In addition, we demonstrated that the model is concise, if the set of goals and the value function can be expressed within close forms, as is the case in many real–world applications. We also proposed an efficient method to calculate the optimal solution for the CSG and the stable payoff distribution problems with low computational cost. In particular, we demonstrated that the proposed method can provide tractability even within problem instances with up to 500 agents. This result significantly outperforms other existing methods which can only deal with up to 30 agents.

As future work, we aim to extend our solver to problem instances with non–convex goal sets and more complex value functions, such as exponential or polynomials with higher ranks. Since our method relies on the convexity and the (piece–wise) linearity of the value function, this extension is far from obvious.

a personal computer, Intel® Xeon® CPU W3520 @2.67GHz with 12GB RAM and under Windows 7 operation system. The code was written and tested on Matlab R2012a.

References

Y. Bachrach and J. S. Rosenschein. Coalitional skill games. In *Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems - Volume 2*, pages 1023–1030, Richland, SC, 2008.

Y. Bachrach, R. Meir, K. Jung, and P. Kohli. Coalitional structure generation in skill games. In *AAAI*, 2010.

V. Conitzer and T. Sandholm. Complexity of determining nonemptiness of the core. In *IJCAI*, pages 613–618, 2003.

V. Conitzer and T. Sandholm. Computing shapley values, manipulating value division schemes and checking core membership in multi-issue domains. In *In Proceedings of AAAI*, pages 42–47, 2004.

G. Desaulniers, J. Desrosiers, and M.M. Solomon. *Column generation*, volume 5. Springer, 2005.

S. Drewes and S. Ulbrich. *Mixed integer second order cone programming*. Verlag Dr. Hut, 2009.

E. Elkind, L. A. Goldberg, P. Goldberg, and M. Wooldridge. A tractable and expressive class of marginal contribution nets and its applications. *Mathematical Logic Quarterly*, 55(4):362–376, 2009.

D. B. Gillies. Solutions to general non-zero-sum games. In H. W. Kuhn, A. W. Tucker, and L. D. Luce, editors, *Contributions to the Theory of Games, volume IV*, pages 47–85. Princeton University Press, 1959.

S. Ieong and Y. Shoham. Marginal contribution nets: A compact representation scheme for coalitional games. *ACM EC-06*, pages 170–179, 2006.

H. Keinänen. Simulated annealing for multi-agent coalition formation. In *Proceedings of the Third KES International Symposium on Agent and Multi-Agent Systems: Technologies and Applications*, KES-AMSTA '09, pages 30–39, Berlin, Heidelberg, 2009. Springer-Verlag.

Z. Khan, J. Lehtomaki, and M. Latva-Aho L. A. DaSilva. On selfish and altruistic coalition formation in cognitive radio networks. In *Fifth International Conference on Cognitive Radio Oriented Wireless Networks and Communications (CROWNCOM-10)*, Cannes, France, 2010.

K. Larson and T. Sandholm. Anytime coalition structure generation: an average case study. *Journal of Experimental and Theoretical Artificial Intelligence*, 12(1):23–42, 2000.

C. Li, K. Sycara, and A. Scheller-Wolf. Combinatorial coalition formation for multi-item group-buying with heterogeneous customers. *Decis. Support Syst.*, 49(1):1–13, April 2010.

N. Di Mauro, T. M. A. Basile, S. Ferilli, and F. Esposito. Coalition structure generation with GRASP. In *AIMSA'10: Fourteenth International Conference on Artificial Intelligence: Methodology, Systems, and Applications*, pages 111–120, 2010.

T. Michalak, J. Sroka, T. Rahwan, M. Wooldridge, P. McBurney, and N. R. Jennings. A Distributed Algorithm for Anytime Coalition Structure Generation. In *AAMAS '10: Proceedings of the Ninth International Joint Conference on Autonomous Agents and Multi-Agent Systems*, pages 1007–1014, 2010.

N. Ohta, A. Iwasaki, M. Yokoo, K. Maruono, V. Conitzer, and T. Sandholm. A compact representation scheme for coalitional games in open anonymous environments. In *Proceedings of the 21st national conference on Artificial intelligence - Volume 1*, AAAI'06, pages 697–702. AAAI Press, 2006.

N. Ohta, V. Conitzer, R. Ichimura, Y. Sakurai, A. Iwasaki, and M. Yokoo. Coalition structure generation utilizing compact characteristic function representations. In *CP'09: 15th International Conference on Principles and Practice of Constraint Programming*, pages 623–638, 2009.

T. Rahwan, T. Michalak, and N. R. Jennings. A hybrid algorithm for coalition structure generation. In *Twenty Sixth Conference on Artificial Intelligence (AAAI-12)*, Toronto, Canada, 2012.

T. Rahwan. *Algorithms for Coalition Formation in Multi-Agent Systems*. PhD thesis, University of Southampton, 2007.

V. Robu, R. Kota, G. Chalkiadakis, A. Rogers, and N. R. Jennings. Cooperative virtual power plant formation using scoring rules. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems - Volume 3*, pages 1165–1166, 2012.

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

L. S. Shapley. A value for n-person games. In H. W. Kuhn and A. W. Tucker, editors, *Contributions to the Theory of Games, volume II*, pages 307–317. Princeton University Press, 1953.