Coalitional Games via Network Flows

Talal Rahwan[1], **Tri–Dung Nguyen**[1], **Tomasz P. Michalak**[2,3], **Maria Polukarov**[1], **Madalina Croitoru**[4] **and Nicholas R. Jennings**[1]
[1] University of Southampton, UK
[2] University of Oxford, UK
[3] University of Warsaw, Poland
[4] University of Montpelier, France

Abstract

We introduce a new representation scheme for coalitional games, called *coalition-flow networks* (CF-NETs), where the formation of effective coalitions *in a task-based setting* is reduced to the problem of directing flow through a network. We show that our representation is intuitive, fully expressive, and captures certain patterns in a significantly more concise manner compared to the conventional approach. Furthermore, our representation has the flexibility to express various classes of games, such as characteristic function games, coalitional games with overlapping coalitions, and coalitional games with agent types. As such, to the best of our knowledge, CF-NETs is the first representation that allows for switching conveniently and efficiently between overlapping/non-overlapping coalitions, with/without agent types. We demonstrate the efficiency of our scheme on the *coalition structure generation* problem, where near-optimal solutions for large instances can be found in a matter of seconds.

1 Introduction

Cooperation is a central concept in artificial intelligence and at the heart of multi-agent systems design. Coalitional game theory is the standard framework to model cooperation; it provides theoretical constructs that allow agents to take joint actions as primitives [Ieong and Shoham, 2006]. One of the most important research questions in this domain is the *coalition structure generation problem* (CSG), i.e., how can the set of agents be partitioned into groups (coalitions) such that social welfare is maximized? This problem has attracted considerable attention in the recent AI literature [Bachrach and Rosenschein, 2008; Michalak *et al.*, 2010; Rahwan *et al.*, 2011], as it has important applications and inspires new theoretical and computational challenges. Existing work on coalition structure generation focuses on *characteristic function games* (CFGs), i.e., games in which the outcome obtained by a coalition is not influenced by agents outside the coalition.[1]

The prevailing assumption in CSG algorithms is that coalition structures are disjoint. However, there exist many realistic scenarios in which this assumption does not hold [Chalkiadakis *et al.*, 2010]. For example, in multi-sensor systems, where sensors coordinate their movements to monitor particular targets of interest, the performance can be improved by allowing the sensors to form overlapping coalitions [Dang *et al.*, 2006].

A second prevailing assumption in existing CSG algorithms is that every agent is unique, i.e., such works do not consider situations where some agents are identical. Again, the ability to differentiate between identical and non-identical agents can often improve performance. For example, in disaster response scenarios, one should distinguish between a fire brigade and an ambulance (as they are aimed for different activities), but may be indifferent between two ambulances with equal capabilities. The theory of coalition structure generation has been extended only recently to such domains [Aziz and de Keijzer, 2011; Ueda *et al.*, 2011].

The standard CFG representation explicitly lists the values of all the possible coalitions and requires space that is exponential in the number of agents. However, in many classes of games, there exists additional structure (such as identical agents) that can be exploited, and well-crafted representation schemes can significantly improve performance [Deng and Papadimitriou, 1994; Ieong and Shoham, 2006; Elkind *et al.*, 2009; Aadithya *et al.*, 2011]. However, the majority of works in this line of research focused on computing solution concepts, rather than solving the coalition structure generation problem.

To date, with very few exceptions (e.g., [Shehory and Kraus, 1998; Dang and Jennings, 2006]) the incorporation of tasks is absent from the research on coalition structure generation. The advantage of incorporating tasks is that it allows for capturing a much wider range of applications. In particular, many real-world applications involve a set of tasks to be achieved, where the agents need to form coalitions since (some of) those tasks cannot be performed by an individual agent, or can be achieved more efficiently by a group of agents (compared to an individual). However, the disadvantage of incorporating tasks is that it makes the optimization problems (such as coalition structure generation) significantly harder to solve. This is because the value of a coalition does

[1] Recently, a coalition structure generation algorithm has been proposed in games with *"externalities"* (i.e., possible influences between co-existing coalitions) [Rahwan *et al.*, 2012].

not depend solely on the identities of its members; it depends on the tasks being achieved by the coalition. As such, a coalition does not necessarily have a single value. Instead, it can have a different value for every task.

Our Contribution

We formulate a unified approach for modelling different types of coalitional games *in task-based settings*. In particular, we formulate the coalition structure generation problem as a network flow problem—a problem widely studied in the combinatorial optimization literature, with many important applications in the real world [Schrijver, 2003]. More specifically:

- We propose a new representation of coalitional games, namely *coalition-flow networks* (CF-NETs), where the coalition formation process is represented as the process of directing the flow through a network with edge-capacity constraints. We show that CF-NETs are an appropriate representation for several important classes of coalitional games, such as conventional (non-overlapping) coalitional games, overlapping coalitional games, and coalitional games with agent types. Importantly, to the best of our knowledge, this is the first representation with which one can easily and efficiently switch between cases with overlapping/non-overlapping coalitions, with/without agent types. In addition, we show how CF-NETs allow for the succinct storage of certain, potentially useful patterns.
- We show that under the CF-NET representation, the coalition structure generation problem in a task-based setting can be reformulated as a mathematical program related to the well-known production-transportation problem [Tuy *et al.*, 1996; Holmberg and Tuy, 1999; Hochbaum and Hong, 1996].
- We provide an anytime approximation technique for the CSG problem, which provides worst-case guarantees on the quality of the solutions produced. Using numerical simulations, we show that our algorithm scales to thousands of agents. For example, for $n = 5000$ agents, it finds solutions within 75-85% of the optimum in a matter of seconds.

Related Work

Researchers have attempted to circumvent the intractability of the characteristic function form by developing alternative representations, which are more compact and allow efficient computations. These representations can be divided into two main categories:

1. The representation is guaranteed to be succinct, but is not fully expressive (i.e., it cannot represent any arbitrary characteristic function game) [Deng and Papadimitriou, 1994; Wooldridge and Dunne, 2006].
2. The representation is fully expressive, but is only succinct for some problem instances [Ieong and Shoham, 2006; Elkind *et al.*, 2009; Conitzer and Sandholm, 2004b]). Our CF-NET representation scheme falls in this category.

Most previous work focuses on the computation of solution concepts, such as the Shapley value [Deng and Papadimitriou, 1994; Ieong and Shoham, 2006; Elkind *et al.*, 2009; Conitzer and Sandholm, 2004b], the core [Conitzer and Sandholm, 2004a; 2004b], the bargaining set, and the kernel [Greco *et al.*, 2009]. On the other hand, the coalition structure generation problem has recently attracted attention, with most of the studies focusing on characteristic function games [Rahwan *et al.*, 2009].

Among the very few works that consider overlapping coalitions, we mention the works by Shehory and Kraus [1996] and Dang *et al.* [2006]. The former only proposes a greedy algorithm for a subclass of these games, whereas the latter makes heavy use of several strong assumptions placed on the characteristic function. Recently, Chalkiadakis *et al.* [2010] introduced a formal model of overlapping coalition formation, in which an agent can participate in multiple coalitions simultaneously, by contributing a portion of its resources to each coalition containing it.

Our CF-NET representation is designed for the purpose of solving the coalition structure generation problem efficiently. The only other representation designed with this problem in mind is due to Ohta *et al.* [2009]. However, unlike our representation, theirs does not incorporate tasks, and does not consider games with agent types or overlapping coalitions.

The basic idea of applying network flows to model coalitional games has been examined before, for example by Kalai and Zemel [1982] and by Bachrach and Rosenschein [2009]. However, the fundamental difference is that, in those papers, the units of flow can be thought of as utility units (so the solution to a network flow problem influences the value of a coalition). In our representation, what flows in the network is "agents", so solving a flow problem returns a coalition structure. This is precisely what gives flexibility to our framework, allowing for overlapping coalitions, or identical agents, to be considered whenever needed.

2 Preliminaries

In this section, we provide the standard definitions for the classes of coalitional games studied in this work.

We define a task-based *characteristic function game*, TCFG, as a coalitional game given by a tuple $\langle A, K, v\rangle$, where

- $A = \{a_1, \ldots, a_n\}$ is a set of agents;
- $K = \{k_1, \ldots, k_q\}$ is a set of tasks;
- $v : 2^A \times K \to \mathbb{R}$ is a function that assigns a value to every pair (C, k_i), where C is a coalition and k_i is a task.

In this paper, we assume that every performed task is performed by exactly one coalition. The possibility of having a group of agents that does not perform any tasks can easily be incorporated by adding a single dummy task. On the other hand, the possibility of having a single coalition perform multiple tasks can be incorporated by allowing overlapping coalitions (in which case every performed task is performed by a single, not necessarily unique, coalition).

An implicit assumption is that every agent can participate in exactly one coalition. When an agent is allowed to belong to multiple coalitions simultaneously, the tuple $\langle A, v\rangle$

defines a task-based *coalitional game with overlapping coalitions*, denoted by TCFG^{o}. In such a game, each agent can be a member of up to 2^{n-1} coalitions. In reality, this number is reasonable only if coordination costs are low and the number of available resources is sufficiently large.

Thus, the definition can naturally be extended to a task-based *resource-constrained coalitional game with overlapping coalitions* (TCFG^{rco}). In such a game, each agent $a_i \in A$ has an upper bound on the number of coalitions it can join, which can be interpreted as a a limit on the resources that can be employed by an agent to perform tasks. In this setting, a game is formally defined by a tuple $\langle A, r, v\rangle$, where A and v are defined as above, and the function $r : A \to \{1, \ldots, 2^{n-1}\}$ assigns to every agent its resource constraints.[2] Note that the TCFG^{o} and TCFG representations are special cases of TCFG^{rco}, where the number of coalitions that an agent can join is maximal (2^{n-1}) and minimal (1), respectively.

Next, we formalize task-based coalitional games with agent types. Let A and v be defined as above and $T = \{t_1, \ldots, t_m : m \leq n\}$ a set of types, such that each agent in A is associated with one type in T. Let $t : A \to T$ be a function that returns the type of any given agent. An important situation to model in such games is that of two agents with the same type bringing the same marginal contribution to any coalition they belong to. That is, given a type $t_k \in T$, for any two agents a_i, a_j of type t_k and any coalition $C \subseteq A \backslash \{a_i, a_j\}$, the following holds: $v(C \cup \{a_i\}) - v(C) = v(C \cup \{a_j\}) - v(C)$.

Recall that the conventional TCFG representation can be used to represent (albeit not concisely) the agent-type setting. Clearly, however, the game can be represented more concisely by considering the values of coalitions as a function of the types (rather than the identities of the agents). To this end, we define coalitional games with agent types.

Definition 1 *A task-based* coalitional game with agent types (TCFG^{at}) *is a tuple $\langle A, T, t, \mathcal{T}, v^{at}\rangle$, where:*

- $A = \{a_1, \ldots, a_n\}$ *is a set of agents;*
- $T = \{t_1, \ldots, t_{m \leq n}\}$ *is a set of agent types;*
- $t : A \to T$ *is a function that returns the type of any agent, $a_i \in A$;*
- $\mathcal{T} = \mathcal{T}_1 \cup \cdots \cup \mathcal{T}_m$ *is a multi-set of agent types, such that $\mathcal{T}_k = \{t_k, \cdots, t_k\}$, where $|\mathcal{T}_k| = |\{a_i \in A : t(a_i) = t_k\}|$, $\forall k \in \{1, \cdots, m\}$;*
- $K = \{k_1, \ldots, k_q\}$ *is a set of tasks;*
- $v^{at} : 2^{\mathcal{T}} \times 2^K \to \mathbb{R}$ *is a characteristic function of agent types.*

3 The CF-NET Representation

Next, we introduce the framework of *coalition-flow networks* (CF-NETs) to represent the classes of coalitional games discussed above.

[2]The above definitions of overlapping coalitional games can be viewed as a special case of a highly theoretical construct of Chalkiadakis *et al.* [2010], where each agent can devote a unit of resource to an infinite number of (possibly identical) coalitions.

Definition 2 *A* coalition-flow network *(CF-NET) is a tuple $\langle N, E, \mathcal{X}, \mathcal{Y}, \mathcal{Z}\rangle$, where (1) (N, E) is an acyclic digraph with a set of nodes N and a set of directed edges E, and (2) $\mathcal{X}$, $\mathcal{Y}$ and $\mathcal{Z}$ are sets of constraints. In particular:*

1. ***The set of nodes**, N, is the union of the following disjoint sets:*
   - *$\{S, G\}$ contains the* source *node S from which the "flow" is pushed into the network and the* sink *node G towards which the flow needs to be directed.*
   - *N^a is the set of* agent *nodes: each agent (or type) is represented by exactly one such node. As such, $|N^a| = n$.*
   - *N^k is the set of* task *nodes: each task is represented by exactly one such node. Thus, $|N^k| = q$.*
2. ***The set of edges** is $E = (\{S\} \times N^a) \cup E' \cup (N^k \times \{G\})$, where $E' \subseteq \{(n_i, n_j) : n_i \in N^a, n_j \in N^k\}$.*
3. ***The set of constraints** is $\mathcal{X} \cup \mathcal{Y} \cup \mathcal{Z}$, where $\mathcal{X} = \{X_i : i = 1 \ldots n\}$, $\mathcal{Y} = \{Y_{ij} : i = 1 \ldots n, j = 1 \ldots q\}$, and $\mathcal{Z} = \{Z_j : j = 1 \ldots q\}$, which restrict the possible values of the flow through the edges in $\{S\} \times N^a$, E' and $N^k \times \{G\}$, respectively. In particular:*
   - *X_i represents the permitted multiplicities of agent a_i (or the permitted numbers of agents whose type is t_i) in the game.*
   - *Y_{ij} represents the permitted multiplicities of agent a_i (or permitted numbers of agents of type t_i) that will perform task k_j.*
   - *Z_j represents the sizes of the coalition that will perform task k_j.*

An illustration of the CF-NET representation can be found Figure 1). Now, based on the network structure (N, E) defined above, and the constraints $\mathcal{X} \cup \mathcal{Y} \cup \mathcal{Z}$ imposed on the flow through the edges, we introduce next the notion of *CF-flow*. Intuitively, the CF-flow depicts the coalition formation process. Analogously to the flow in the network, the process of directing the flow from the source node to the sink node (through the agent/type nodes and the task nodes) can be interpreted as the process of determining which coalition should perform each task.

Definition 3 *A* coalition formation flow *(CF-flow) in a CF-NET $\langle N, E, \mathcal{X}, \mathcal{Y}, \mathcal{Z}\rangle$ is a function $f : E \to \mathbb{R}$ with the following properties:*

1. $f(S, n_i) \in \mathcal{X}$, $\forall n_i \in N^a$.
2. $f(n_i, n_j) \in \mathcal{Y}$, $\forall (n_i, n_j) \in E'$.
3. $f(n_j, G) \in \mathcal{Z}$, $\forall n_j \in N^k$.
4. $\sum_{(n_i, n_j) \in E} f(n_i, n_j) = \sum_{(n_j, n_k) \in E} f(n_j, n_k)$, $\forall n_j \in N^a \cup N^k$.

Let (x, y, z) be an *instantiation* of a CF-flow f in a CF-NET $\langle N, E, \mathcal{X}, \mathcal{Y}, \mathcal{Z}\rangle$, such that $x_i = f(S, n_i)$ for $n_i \in N^a$, $y_{ij} = f(n_i, n_j)$ for $(n_i, n_j) \in E'$, and $z_j = f(n_j, G)$ for $n_j \in N^k$. Then, from Definition 3, we have:

$$x_i = \sum_{j=1}^{m} y_{ij},\ \forall\, i = 1, \ldots, n;\quad z_j = \sum_{i=1}^{n} y_{ij},\ \forall\, j = 1, \ldots, m; \quad (1)$$

$$x_i \in \mathcal{X}_i,\ y_{ij} \in Y_{ij},\ z_j \in Z_j,\ \forall\, i = 1, \ldots, n, \forall\, j = 1, \ldots, m.$$

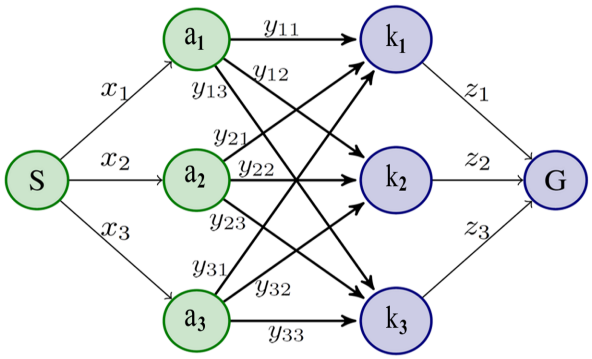

Figure 1: Sample CF-NET representation

Given a CF-flow, we define next the values of the coalitions constructed by it. To this end, we equip a CF-NET with three valuation functions as follows:

Definition 4 *Given a CF-NET $\langle N, E, \mathcal{X}, \mathcal{Y}, \mathcal{Z}\rangle$, let c, d, g be the* valuation functions *defined, respectively, on $\{S\} \times N^a$, E' and $N^k \times \{G\}$, such that:*

- *For all $(S, n_i) \in \{S\} \times N^a$, $c_i = c(S, n_i)$ represents the value of a* singleton *coalition of agent a_i (or an agent of type t_i).*
- *For all $(n_i, n_j) \in E'$, $d_{ij} = d(n_i, n_j)$ is the* contribution *of agent a_i (or an agent of type t_i) to task k_j.*
- *For all $(n_j, G) \in N^k \times \{G\}$, $g_j(z_j) = g(n_j, G)$ is a* synergy *function, where $z_j \in Z_j$ is the size of the coalition performing task k_j.*[3]

That is, the value *of a coalition C performing task k_j is given as:*

$$v(C, k_j) = \sum_{a_i \in C} d_{ij} y_{ij} + g_j(|C|). \quad (2)$$

The first advantage of CF-NETs is their ability to represent any of the four classes of games discussed above, (i.e. TCFG, TCFG$^{\text{o}}$, TCFG$^{\text{rco}}$, and TCFG$^{\text{at}}$). This can be done simply by setting the appropriate $\mathcal{X}$, $\mathcal{Y}$, and $\mathcal{Z}$. In more detail,

- In TCFG, $\forall a_i \in A, X_i = \{1\}$, $\forall a_i \in A, \forall k_j \in K, Y_{ij} = \{0, 1\}$, and $\forall k_j \in K, Z_j = \{0, \ldots, n\}$;
- In TCFG$^{\text{o}}$, $\forall a_i \in A, X_i = \{1, \ldots, 2^{n-1}\}$, $\forall a_i \in A, \forall k_j \in K, Y_{ij} = \{0, 1\}$, and $\forall k_j \in K, Z_j = \{0, \ldots, n\}$;
- In TCFG$^{\text{rco}}$, $\forall a_i \in A, X_i = \{1, \ldots, r(a_i)\}$, $\forall a_i \in A, \forall k_j \in K, Y_{ij} = \{0, 1\}$, and $\forall k_j \in K, Z_j = \{\{0, \ldots, n\}\}$;
- In TCFG$^{\text{at}}$, $\forall a_i \in A, X_i = \{|\mathcal{T}_i|\}$, $\forall a_i \in A, \forall k_j \in K, Y_{ij} = \{0, \ldots, |\mathcal{T}_i|\}$, and $\forall k_j \in K, Z_j = \{0, \ldots, m\}$.

Next, we discuss expressiveness.

Proposition 1 *Every coalitional game that can be modelled as a TCFG, TCFG$^{\text{o}}$, TCFG$^{\text{rco}}$ and/or TCFG$^{\text{at}}$, can also be represented as a CF-NET.*

[3]Throughout the paper this is assumed that $g_j(0) = 0$.

Proof : We demonstrate that, for any arbitrary coalitional game under consideration, there exists a CF-NET representation that uniquely defines this game. Specifically:

1. TCFG: Our aim is to construct CF-NET representing an arbitrary game $\langle A, K, v\rangle$. First, we create a one-to-one function $I : 2^A \times K \to \{1, \ldots, 2^n \times q\}$. Now, for every coalition $C \in 2^A$ and task $k_j \in K$, we create a hypothetical task $w_{I(C,k_j)}$ which is connected to all the agents in C and none of the agents in $A \backslash C$. Furthermore, for every $a_i \in C$, we set $X_i = \{1\}$, $Y_{i,I(C,k_j)} = \{0, 1\}$, and $d_{i,I(C,k_j)} = 0$. We also set $Z_{I(C,k_j)} = \{0, |C|\}$, and set $g_{I(C,k_j)}(|C|) = v(C, k_j)$ and $g_{I(C,k_j)}(0) = 0$.
2. TCFG$^{\text{o}}$/TCFG$^{\text{rco}}$: This case is similar to the previous one. The difference is *only* in the definition of X_i, which is now given by $X_i = \{1, \ldots, 2^{n-1}\}$ in the case of TCFG$^{\text{o}}$, or by $X_i = \{1, \ldots, r(a_i)\}$ in the case of TCFG$^{\text{rco}}$.
3. TCFG$^{\text{at}}$: Here, agent nodes depict available types of agents. That is, every node $n_i \in N^a$ represents a type t_i. This case is similar to TCFG case, except that we now set $X_i = \{|\mathcal{T}_i|\}$ and $Y_{i,I(C,k_j)} = \{0, |C \cap \mathcal{T}_i|\}$.

□

The constructs in the proof of Proposition 1 imply that CF-NETs are no less concise than the corresponding TCFG, TCFG$^{\text{o}}$/TCFG$^{\text{rco}}$ and TCFG$^{\text{at}}$ representations. Indeed, the edges do not need to be explicitly represented; for every hypothetical task w_x it is possible to identify to it simply by using the inverse of function I (which in turn can be concisely represented).

Observe that, for certain patterns often encountered in coalitional games, CF-NETs provide much more concise representations. For instance, suppose there exist additional requirements for coalitions to be formed, such as certain agents being incompatible with each other, or constraints on the coalition sizes. In these cases, one can simply exclude "infeasible" coalitions from the set of coalition nodes. This only decreases the size of the representation, which for certain game classes makes CF-NETs exponentially more concise than the corresponding characteristic function representations.

Next we prove our main technical result, demonstrating the computational power of the CF-NET representation in the coalition structure generation problem.

4 Coalition Structure Generation in CF-NETs

In this section, we formally define the coalition structure generation (CSG) problem and propose an approximation method for solving it on CF-NETs. Our technique utilizes the advantages of the CF-NET representation to produce anytime solutions and estimate their quality.

The CSG Problem

First, we make explicit the notion of a coalition structure for each of the classes TCFG, TCFG$^{\text{o}}$, TCFG$^{\text{rco}}$ and TCFG$^{\text{at}}$.

1. TCFG: A coalition structure, π, is a *partition* of the agents:

$$\pi = \left\{ C : C \subseteq A, \bigcup_{C \in \pi} C = A, \forall C, C' \in \pi : C \cap C' = \emptyset \right\}$$

2. TCFG$^{\text{o}}$: Similar to TCFG, except that coalitions can now overlap. A coalition structure, π^o, is defined as:

$$\pi^o = \left\{ C : C \subseteq A, \bigcup_{C \in \pi} C = A \right\}$$

3. TCFG$^{\text{rco}}$: In this case, each agent $a_i \in A$ can belong to at most $r(a_i)$ coalitions simultaneously:

$$\pi^{rco} = \left\{ C : C \subseteq A, \bigcup_{C \in \pi} C = A, \forall a_i \in A : |\{C \in \pi : a_i \in C\}| \leq r(a_i) \right\}$$

4. TCFG$^{\text{at}}$: Coalitions are multi-sets of types, rather than sets of agents as in previous cases. Let $m_y(x)$ denote the multiplicity of element x in multi-set y; then:

$$\pi^{at} = \left\{ C : C \subseteq \mathcal{T}, \bigcup_{C \in \pi} C = \mathcal{T}, \forall t_i \in \mathcal{T} : \sum_{C \in \pi} m_C(t_i) = m_{\mathcal{T}}(t_i) \right\}$$

Let Π^x denote the sets of all possible coalition structures in TCFG^x, where index x is either empty or stands for "o", "rco" or "at". The CSG problem in TCFG$^{\text{x}}$ is to find an optimal coalition structure, π^x_{OPT}, that maximizes the sum of coalition values:

$$\pi^x_{OPT} \in \arg \max_{\pi = \{C_1, \ldots, C_{|\pi|}\} \in \Pi^x, K^\pi = \{k_1, \ldots, k_{|\pi|}\} \subseteq K} \sum_{C_i \in \pi} v^x(C_i, k_i),$$

where $v^x = v^{at}$ if TCFG$^{\text{at}}$, and $v^i = v$, otherwise.

Solving the CSG Problem in CF-NETs

We now formalize the CSG problem in terms of CF-NETs. Here, since the value of a coalition performing a task is defined by (2), the value of a coalition structure is:

$$\underbrace{\sum_{i=1}^{n} \sum_{j=1}^{q} d_{ij} y_{ij}}_{\text{individual marginal contributions}} + \underbrace{\sum_{j=1}^{q} g_j(z_j)}_{\text{synergy values}} .$$

For ease of exposition, we focus next on the non-overlapping model. However, we emphasize the fact that our method can be easily extended to games with overlapping coalitions or with agent types (by defining X_i, Y_{ij} and Z_j accordingly).

For all i, j, we have $X_i = \{1\}$, $Y_{ij} = \{0, 1\}$ and $Z_j = \{0, \ldots, n\}$. The coalition structure generation problem can then be formulated as a mathematical program:

$$\begin{aligned} \mathit{CSG} \quad := \quad & \max_{\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}} \sum_{i=1}^{n} \sum_{j=1}^{q} d_{ij} y_{ij} + \sum_{j=1}^{q} g_j(z_j) \\ s.t. \quad & z_j = \sum_{i=1}^{n} y_{ij}, \qquad \forall\, j = 1, \ldots, q, \qquad (3) \\ & x_i = \sum_{j=1}^{q} y_{ij}, \qquad \forall\, i = 1, \ldots, n, \qquad (4) \\ & \boldsymbol{x} = \{1\}^n, \boldsymbol{y} \in \{0, 1\}^{n \times q}, \boldsymbol{z} \in \{0, \ldots, n\}^q. \qquad (5) \end{aligned}$$

Note that if the synergy functions g_j are linear, the problem is equivalent to the classical maximum network flow problem and can be solved efficiently [Cormen *et al.*, 2001]. However, for non-linear synergies, CSG becomes a non-linear integer programming problem, which is generally NP-hard [Sandholm *et al.*, 1999]. In this paper, we solve/approximate CSG for general synergy functions.

First, observe that our problem is related to the *production-transportation problem* from Operations Research [Tuy *et al.*, 1996; Holmberg and Tuy, 1999], which also has a network flow interpretation. Here, the non-linear terms $g_j(z_j)$ can be viewed as the production part that the decision makers have to decide upon. Once the production $\boldsymbol{z}$ has been fixed, the problem becomes a standard transportation problem – of finding $(\boldsymbol{x}, \boldsymbol{y})$ – and can be solved efficiently. However, solving for optimal $(\boldsymbol{x}, \boldsymbol{y})$ and $\boldsymbol{z}$ simultaneously is non-trivial [Hochbaum and Hong, 1996].

Therefore, in this work we develop an approximation technique for CSG. Our method has two advantages: it gives an anytime algorithm and it provides upper and lower bounds on the optimal value; as a result, one can quantify how far the solution provided is from the optimal solution. Specifically, we use constraint relaxation and duality to modify the mathematical program as follows.

Relax the constraint (3) $z_i = \sum_{i=1}^{n} y_{ij}$ in CSG and consider the corresponding dual problem:

$$\begin{aligned} \min_{\boldsymbol{\lambda}} \quad & \max_{\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}} \sum_{i=1}^{n} \sum_{j=1}^{q} d_{ij} y_{ij} + \sum_{j=1}^{q} g_j(z_j) \\ & - \sum_{j=1}^{q} \lambda_j (z_j - \sum_{i=1}^{n} y_{ij}) \\ s.t. \quad & (4), (5) \end{aligned}$$

Now, for each fixed $\boldsymbol{\lambda}$, consider the inner problem:

$$\begin{aligned} h(\boldsymbol{\lambda}) \quad = & \max_{\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}} \sum_{i=1}^{n} \sum_{j=1}^{q} (d_{ij} + \lambda_j) y_{ij} \\ & + \sum_{j=1}^{q} (g_j(z_j) - \lambda_j z_j) \\ s.t. \quad & (4), (5) \end{aligned}$$

By replacing every x_i with $\sum_{j=1}^{q} y_{ij}$ and noticing that $h(\boldsymbol{\lambda}))$ is now separable in $\boldsymbol{y}$ and $\boldsymbol{z}$, we can show that $h(\boldsymbol{\lambda}) = \sum_{i=1}^{n} h_{1i}(\boldsymbol{\lambda}) + \sum_{j=1}^{q} h_{2j}(\lambda_j)$, where:

$$\begin{aligned} h_{1i}(\boldsymbol{\lambda}) \quad = \quad & \max_{\boldsymbol{y}_{i.}} \sum_{i=1}^{n} \sum_{j=1}^{q} (d_{ij} + \lambda_j) y_{ij} \\ & s.t. \quad \sum_{j=1}^{q} y_{ij} \leq 1, \\ & \boldsymbol{y}_{i.} \in \{0, 1\}^q, \quad \text{and} \\ h_{2j}(\lambda_j) \quad = \quad & \max_{z_j \in \{0, \ldots, n\}} g_j(z_j) - \lambda_j z_j. \end{aligned}$$

Finally, by simplifying $h_{1i}(\boldsymbol{\lambda})$ to $h_{1i}(\boldsymbol{\lambda}) = \max\{0, \max_j (d_{ij} + \lambda_j)\}$, we get a *one variable* integer programming problem, which can be easily solved.

Thus, for each fixed $\boldsymbol{\lambda}$, we can compute $h(\boldsymbol{\lambda})$ very efficiently. Notice that $\min_{\boldsymbol{\lambda}} h(\boldsymbol{\lambda})$ provides an upper bound on CSG (duality theory) and hence any choice of $\boldsymbol{\lambda}$ gives

an upper bound (but we are interested in the smallest one, $\min_{\boldsymbol{\lambda}} h(\boldsymbol{\lambda})$). Notice also that $h(\boldsymbol{\lambda})$ is a piece-wise linear function on $\boldsymbol{\lambda}$ with possible jumps. We then can solve the problem $\min_{\boldsymbol{\lambda}} h(\boldsymbol{\lambda})$ using a sub-gradient based method for updating $\boldsymbol{\lambda}$, i.e., we reduce λ_j if $(z_j - \sum_{i=1}^{n} y_{ij}) < 0$ and increase λ_j otherwise. Although we might not be able to solve $\min_{\boldsymbol{\lambda}} h(\boldsymbol{\lambda})$ to optimality, a sub-optimal solution still provides us with an upper bound on CSG.

Finally, lower bounds are obtained. First, for each $\boldsymbol{\lambda}$, the inner problem can be solved to find its optimal solution on $(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$. This solution is a feasible solution to CSG and hence its objective value provides a lower bound. Another method for producing a lower bound is to fix the optimal $\boldsymbol{z}$ found in the inner problem and then solve the transportation problem to find the corresponding optimal $(\boldsymbol{x}, \boldsymbol{y})$. This will give another feasible solution and hence, a new lower bound.[4]

5 Performance Evaluation

We perform numerical tests on the algorithm for various settings with the number of agents n varying between 100 and 5000 and the number of tasks q varying between 50 and 200. For each combination of (n, q), we generate 100 random samples using random seeds between 1 and 100. In total, we have tested the algorithm with 2000 random instances. On each instance, the parameters $\boldsymbol{c}$ and $\boldsymbol{d}$ are generated uniformly, i.e., $d_{ij} \sim U[0, 1]$. The synergy function $g_j(z_j)$ is also a random discrete function of the following form:

$$g_j(k) \;=\; \epsilon_{j1} + \epsilon_{j2} + \ldots + \epsilon_{jk},\; \forall k = 1 \ldots n,\; \forall j = 1 \ldots q,$$

where $\epsilon_{ji} \sim U[0, 1]$ are uniform random variables. This means the synergy function g_j is the sum of uniform random variables and the coalition value increase by ϵ_{js} when the coalition size increases from $s - 1$ to s. By creating 100 random instances, we can test the robustness of the algorithm when input data varies[5].

Figure 2 shows the performance of the algorithm when the number of agents varies between 100 and 5000, while the number of tasks is fixed at 100. Sub-figure (A) shows the total computational time, sub-figure (B) shows the number of iteration, while sub-figure (C) shows the optimality bound between the feasible coalition structure found and the worst upper bound. We can see a linear trend in the computational time from sub-figure (A) with less than three minutes[6] to solve the largest and the worst instance (among 600 random instances for this case). The linear trend in the computational time can be explained by the fact that the number of arithmetic operations in each iteration grows linearly, while the number of iterations (shown in sub-figure (B)) does not change much. Sub-figure (C) shows the guaranteed bound between the feasible coalition structure found and the optimality. Notice that these bounds are guaranteed despite the fact that we don't know the optimal coalition structures thanks to the availability of the upper bounds derived by the algorithm. This also means that the actual optimality bounds could be higher than the average optimality bounds between 75-81% that appear in sub-figure (C).

Figure 3 shows the performance of the algorithm when the number of agents varies is fixed at $n = 2000$, while the number of tasks varies between 50 and 200. We can see a very similar linear trend in the computational time in sub-figure (A) and the optimality bounds between 75-81% in sub-figure (C). The total computational time for the largest instance is less than 35 seconds.

Figures 4, 5 show the same set of statistics as in Figures 2, 3 except that the CFG$^{\text{rco}}$ games now allow each players to join up to 5 coalitions. The guaranteed optimality bounds vary between 76-85% on these instances. We can also see a linear trend in the computational time as the number of players and the number of tasks increase and the algorithm takes less than 90 seconds for the worst instance.

6 Conclusions

We introduced CF-NETs, a representation scheme for coalitional games in task-based settings, which is inspired by network flows. We examined its qualities with respect to conventional coalitional games with non-overlapping coalitions, (resource-constrained) overlapping coalitional games, and coalitional games with agent types. We utilized the advantages of this representation to develop an approximation technique for coalition structure generation, which applies to all these game classes and allows to effectively solve large instances of the problem.

Our work can be extended in several ways. It would be interesting to extend the CF-NET framework with components that would allow to capture game patterns other than those considered in this paper. Furthermore, we are keen on testing the properties of the CF-NET representation with respect to computing different solution concepts such as the Shapley value and the core.

Acknowledgements

We would like to thank the anonymous IJCAI reviewers for valuable feedback. Tomasz Michalak was supported by the European Research Council under Advanced Grant 291528 ("RACE"). Talal Rahwan and Nicholas R. Jennings were supported by the ORCHID Project, funded by EPSRC (Engineering and Physical Research Council) under the grant EP/I011587/1.

[4] With the obtained bounds, it is possible to extend the algorithm in the future by incorporating branch-and-bound techniques to improve solution quality.

[5] For each pair of (n, q), we will present box plots that show the statistics among 100 random instances generated with the middle red horizontal lines showing the medians, the boxes showing the 25 and 75 percentiles, and the red crosses showing the outliers.

[6] All the numerical tests appear in this manuscript are performed on 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.

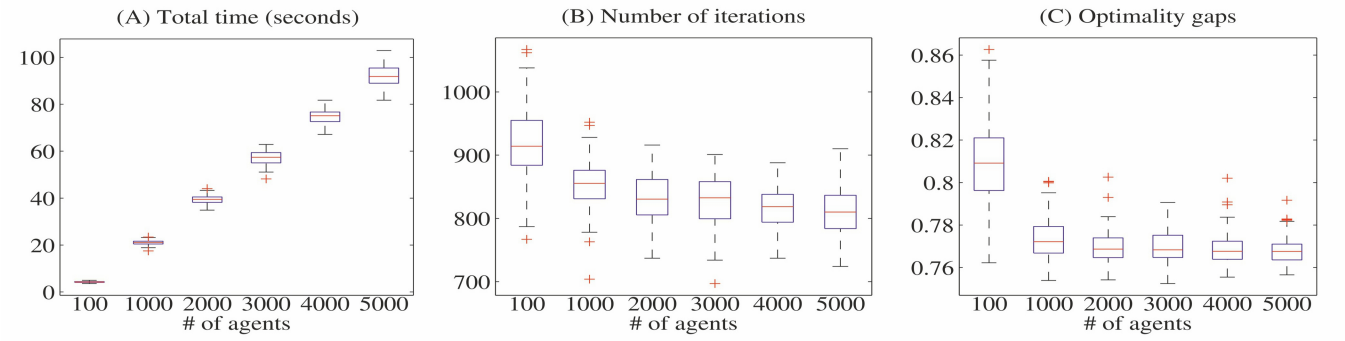

Figure 2: Finding near-optimal coalition structures in CFG games, given different numbers of *agents*.

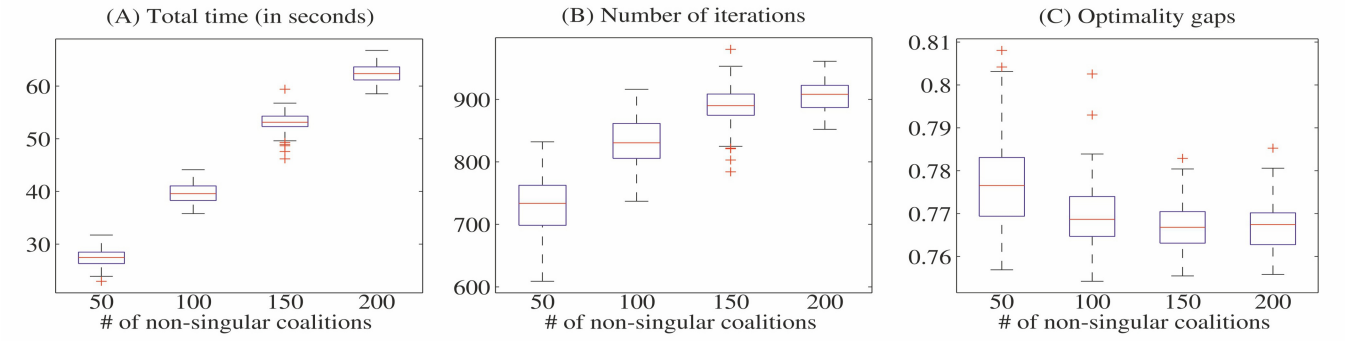

Figure 3: Finding near-optimal coalition structures in CFG games, given different numbers of *tasks*.

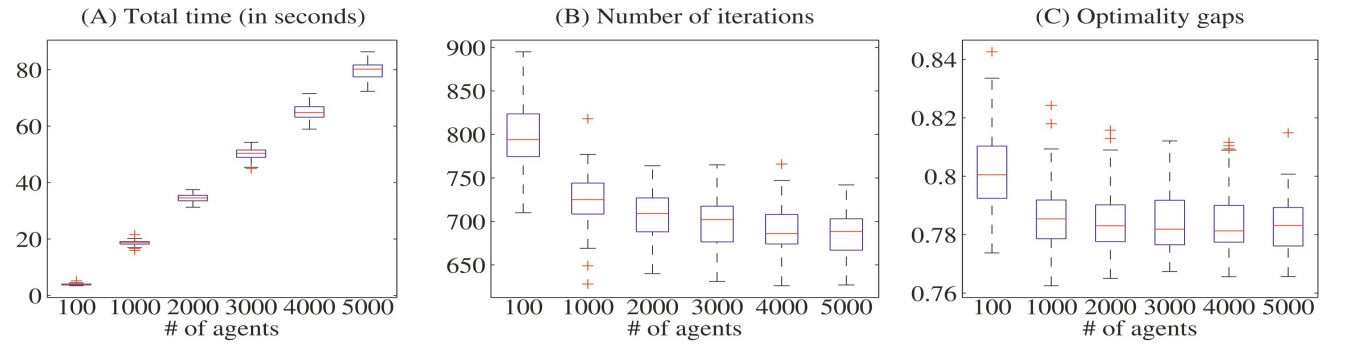

Figure 4: Finding near-optimal coalition structures in CFG^{rco} games, given different numbers of *agents*.

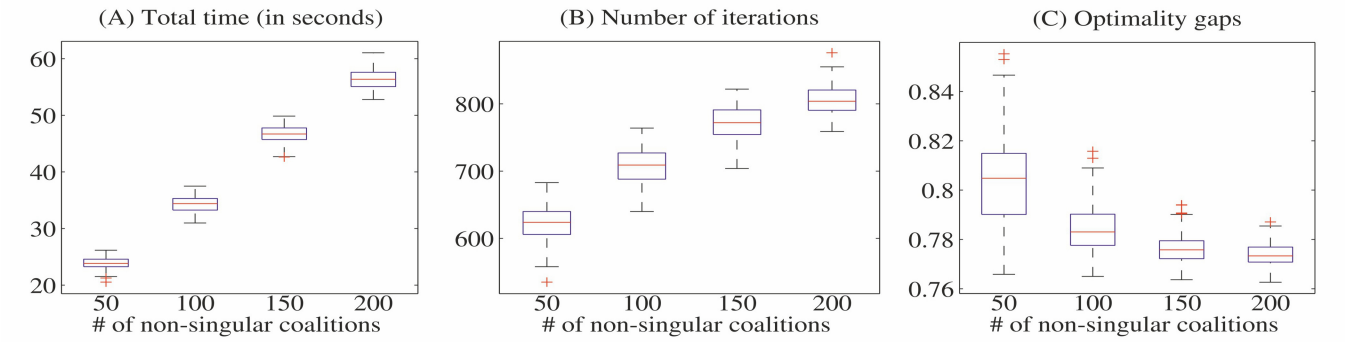

Figure 5: Finding near-optimal coalition structures in CFG^{rco} games, given different numbers of *tasks*.

References

[Aadithya *et al.*, 2011] K. V. Aadithya, T. P. Michalak, and N. R. Jennings. Representation of coalitional games with algebraic decision diagrams. In *Proceedings of AAMAS*, pages 1121–1122, 2011.

[Aziz and de Keijzer, 2011] Haris Aziz and Bart de Keijzer. Complexity of coalition structure generation. In *Proceedings of AAMAS*, pages 191–198, 2011.

[Bachrach and Rosenschein, 2008] Y. Bachrach and J. S. Rosenschein. Coalitional skill games. In *Proceedings of AAMAS*, pages 1023–1030, 2008.

[Bachrach and Rosenschein, 2009] Y. Bachrach and J.S. Rosenschein. Power in threshold network flow games. *AAMAS*, 18(1):106–132, 2009.

[Chalkiadakis *et al.*, 2010] G. Chalkiadakis, E. Elkind, E. Markakis, M. Polukarov, and N. R. Jennings. Cooperative games with overlapping coalitions. *J. of Artif. Int. Res. (JAIR)*, 39(1):179–216, 2010.

[Conitzer and Sandholm, 2004a] V. Conitzer and T. Sandholm. Complexity of Determining Nonemptiness in The Core. In *Proceedings of IJCAI*, pages 219–225, 2004.

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

[Cormen *et al.*, 2001] Thomas H. Cormen, Clifford Stein, Ronald L. Rivest, and Charles E. Leiserson. *Introduction to Algorithms: 26. Maximum Flows*. McGraw-Hill Higher Education, 2nd edition, 2001.

[Dang and Jennings, 2006] Viet Dung Dang and Nicholas R. Jennings. Coalition structure generation in task-based settings. In *ECAI*, pages 210–214, 2006.

[Dang *et al.*, 2006] V. D. Dang, R. K. Dash, A. Rogers, and N. R. Jennings. Overlapping coalition formation for efficient data fusion in multi-sensor networks. In *Proceedings of AAAI*, pages 635–640, 2006.

[Deng and Papadimitriou, 1994] X. Deng and C. Papadimitriou. On the complexity of cooperative solution concepts. *Mathematical OR*, (19):257–266, 1994.

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

[Greco *et al.*, 2009] G. Greco, E. Malizia, L. Palopoli, and F. Scarcello. On the complexity of compact coalitional games. In *Proceedings of IJCAI*, pages 147–152, 2009.

[Hochbaum and Hong, 1996] D.S. Hochbaum and S.P. Hong. On the complexity of the production-transportation problem. *SIAM J. on Optimization*, 6(1):250–264, 1996.

[Holmberg and Tuy, 1999] K. Holmberg and H. Tuy. A production-transportation problem with stochastic demand and concave production costs. *Mathematical programming*, 85(1):157–179, 1999.

[Ieong and Shoham, 2006] S. Ieong and Y. Shoham. Marginal contribution nets: A complact representation scheme for coalitional games. In *Proceedings of ACM EC*, pages 170–179, 2006.

[Kalai and ZemelL, 1982] E. Kalai and E. ZemelL. On Totally Balanced Games and Games of Flow. *In Mathematics of Operations Research*, 7(3):476–478, 1982.

[Michalak *et al.*, 2010] Tomasz Michalak, Jacek Sroka, Talal Rahwan, Michael Wooldridge, Peter McBurney, and Nicholas R. Jennings. A Distributed Algorithm for Anytime Coalition Structure Generation. In *Proceedings of AAMAS*, pages 1007–1014, 2010.

[Ohta *et al.*, 2009] N. Ohta, V. Conitzer, R. Ichimura, Y. Sakurai, A. Iwasaki, and M. Yokoo. Coalition structure generation utilizing compact characteristic function representations. In *Proceedings of CP (to appear)*, 2009.

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

[Rahwan *et al.*, 2011] Talal Rahwan, Tomasz P. Michalak, and Nicholas R. Jennings. Minimum search to establish worst-case guarantees in coalition structure generation. In *Proceedings of IJCAI*, pages 338–343, 2011.

[Rahwan *et al.*, 2012] Talal Rahwan, Tomasz P. Michalak, Michael Wooldridge, and Nicholas R. Jennings. Anytime coalition structure generation in multi-agent systems with positive or negative externalities. *Artificial Intelligence (AIJ)*, 186:95–122, 2012.

[Sandholm *et al.*, 1999] Tuomas Sandholm, Kate Larson, Martin Andersson, Onn Shehory, and Fernando Tohmé. Coalition structure generation with worst case guarantees. *Artif. Intell.*, 111(1-2):209–238, July 1999.

[Schrijver, 2003] A. Schrijver. *Combinatorial Optimization: Polyhedra and Efficiency*. Springer, 2003.

[Shehory and Kraus, 1996] O. Shehory and S. Kraus. Formation of overlapping coalitions for precedence-ordered task-execution among autonomous agents. In *Proceedings of ICMAS*, pages 330–337, 1996.

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

[Tuy *et al.*, 1996] H. Tuy, S. Ghannadan, A. Migdalas, and P. Värbrand. A strongly polynomial algorithm for a concave production-transportation problem with a fixed number of nonlinear variables. *Mathematical Programming*, 72(3):229–258, 1996.

[Ueda *et al.*, 2011] Suguru Ueda, Makoto Kitaki, Atsushi Iwasaki, and Makoto Yokoo. Concise characteristic function representations in coalitional games based on agent types. In *Proceedings of IJCAI*, pages 393–399, 2011.

[Wooldridge and Dunne, 2006] M. Wooldridge and P. Dunne. On the computational complexity of coalitional resource games. *Artificial Intelligence (AIJ)*, (170):835–871, 2006.