Algorithms for Graph-Constrained Coalition Formation in the Real World

FILIPPO BISTAFFA and ALESSANDRO FARINELLI, University of Verona
JESÚS CERQUIDES and JUAN RODRÍGUEZ-AGUILAR, IIIA-CSIC
SARVAPALI D. RAMCHURN, University of Southampton

Coalition formation typically involves the coming together of multiple, heterogeneous, agents to achieve both their individual and collective goals. In this paper, we focus on a special case of coalition formation known as Graph-Constrained Coalition Formation (GCCF) whereby a network connecting the agents constrains the formation of coalitions. We focus on this type of problem given that in many real-world applications, agents may be connected by a communication network or only trust certain peers in their social network. We propose a novel representation of this problem based on the concept of edge contraction, which allows us to model the search space induced by the GCCF problem as a rooted tree. Then, we propose an anytime solution algorithm (CFSS), which is particularly efficient when applied to a general class of characteristic functions called $m + a$ functions. Moreover, we show how CFSS can be efficiently parallelised to solve GCCF using a non-redundant partition of the search space. We benchmark CFSS on both synthetic and realistic scenarios, using a real-world dataset consisting of the energy consumption of a large number of households in the UK. Our results show that, in the best case, the serial version of CFSS is 4 orders of magnitude faster than the state of the art, while the parallel version is 9.44 times faster than the serial version on a 12-core machine. Moreover, CFSS is the first approach to provide anytime approximate solutions with quality guarantees for very large systems of agents (i.e., with more than 2700 agents).

Categories and Subject Descriptors: I.2 [**Computing Methodologies**]: Artificial Intelligence

General Terms: Algorithms

ACM Reference Format:
ACM Trans. Intell. Syst. Technol. 8, 4, Article 60 (February 2017), 23 pages.
DOI:http://dx.doi.org/10.1145/3040967

1. INTRODUCTION

Coalition Formation (CF) is one of the key approaches to establishing collaborations in multi-agent systems. It involves the coming together of multiple, possibly heterogeneous, agents in order to achieve either their individual or collective goals, whenever they cannot do so on their own. Building upon the seminal work of Shehory and Kraus [1998], Sandholm et al. [1999] identify the key computational tasks involved in the CF process: (i) coalitional value calculation: defining a *characteristic function* which, given a coalition as an argument, provides its coalitional value; (ii) coalition structure generation (CSG): finding a partition of the set of agents (into disjoint coalitions) that maximises the sum of the values of the chosen coalitions; and (iii) payment computation: finding the transfer or payment to each agent to ensure it is fairly rewarded for its contribution to its coalition.

Author's addresses: F. Bistaffa and A. Farinelli, Department of Computer Science, University of Verona, Verona, Italy; J. Cerquides and J. Rodríguez-Aguilar, IIIA-CSIC, Barcelona, Spain; S. D. Ramchurn, Electronics and Computer Science, University of Southampton, Southampton, United Kingdom.

On the one hand, typical CF approaches assume that the values of all the coalitions are stored in memory, allowing to read each value in constant time. However, this assumption makes the size of the input of the CSG and payment computation problems exponential, as the entire set of coalitions (whose size is 2^n for n agents) must be mapped to a value. On the other hand, CSG and payment computation are combinatorial in nature and most existing solutions do not scale well with the number of agents. In this paper, we focus on the CSG problem to provide solutions that can be applied to real-world problems, which usually involve hundreds or thousands of agents.

The computational complexity of the CSG problem is due to the size of its search space,[1] which contains every possible subset of agents as a potential coalition. However, in many real-world applications, there are constraints that may limit the formation of some coalitions [Rahwan et al. 2011]. Specifically, we focus on a specific type of constraints that encodes synergies or relationships among the agents and that can be expressed by a graph [Myerson 1977], where nodes represent agents and edges encode the relationships between the agents. In this setting, edges enable connected agents to form a coalition and a coalition is considered feasible only if its members represent the vertices of a connected subgraph. Such constraints are present in several real-world scenarios, such as social or trust constraints (e.g., energy consumers who prefer to group with their friends and relatives in forming energy cooperatives [Hampshire County Council 2014]), physical constraints (e.g., emergency responders may join specific teams in disaster scenarios where only certain routes are available), or communication constraints (e.g., non-overlapping communication loci or energy limitations for sending messages across a network from one agent to another). Hereafter, we shall refer to the CF problem where coalitions are encoded by means of graphs as *Graph-Constrained Coalition Formation* (GCCF). It is important to note that the addition of these constraints does not lower the complexity of the problem. In particular, Voice et al. [2012a] show that the GCCF problem remains NP-complete.

In this work, we are primarily interested in developing CSG solutions for GCCF that are deployable in real-world scenarios involving hundreds or thousands of agents, such as collective energy purchasing [Vinyals et al. 2012; Farinelli et al. 2013] and ridesharing [Bistaffa et al. 2015]. Notice that, since the computation of an optimal solution is often infeasible for large-scale systems, our CSG algorithm should be able to provide anytime approximate solutions with good quality guarantees. Moreover, the memory requirements should scale well with the number of agents.

In this context, the works by Voice et al. [2012a; 2012b] represent the state of the art for GCCF. However, there are some drawbacks that hinder their applicability. Voice et al. [2012a] make assumptions that do not hold in most real-world applications (see Section 2.1.4), whereas the memory requirements of the approach in [Voice et al. 2012b] grow exponentially in the number of agents, hence limiting the scalability.

To overcome these drawbacks, in this paper we propose CFSS (Coalition Formation for Sparse Synergies), the first approach for GCCF that computes anytime solutions with theoretical quality guarantees for large systems (i.e., more than 2700 agents). As recently noticed in a survey on CSG by Rahwan et al. [2015], previous approaches in the CF literature have been either applied to small-scale synthetic scenarios, or, in the case of heuristic approaches, cannot provide any theoretical guarantees on the quality of their solutions. Moreover, we provide P-CFSS, a parallelised version of CFSS that exploits multi-core CPUs. Finally, we identify a general class of closed-form functions, denoted as $m + a$, for which we provide upper bounds, allowing for coalitional values to be computed online (i.e., their storage can be avoided).

[1]A set of n agents can be partitioned in $\Omega((\frac{n}{\ln(n)})^n)$ ways, i.e. the n^{th} Bell number [Berend and Tassa 2010].

In more detail, this paper advances[2] the state of the art in the following ways:

(1) We provide a new representation for GCCF which, by using edge contractions on the graph, can efficiently build a search tree where each node is a feasible coalition structure, while avoiding redundancy (i.e., each solution appears only once).
(2) We identify a general class of characteristic functions, i.e., $m + a$ functions, which are expressive enough to represent a wide range of real-world GCCF problems.
(3) We propose CFSS, a branch and bound algorithm that, when applied to CF with $m + a$ functions, can solve the CSG problem for GCCF and can provide anytime approximate solutions with good quality guarantees.
(4) We propose P-CFSS, a parallel version of CFSS that is up to 9.44 times faster than the serial version on a 12-core machine.

The rest of the paper is organised as follows. Section 2 discusses the relationship between our work and the existing literature, and Section 3 formally defines GCCF. Section 4 explains how we generate our search space, and Section 6 details the domains used to benchmark CFSS, our branch and bound approach described in Section 5, and Section 7 discusses our empirical evaluation. Finally, Section 8 concludes the paper.

2. RELATED WORK

In this section we elaborate on related work in the areas of CF (Section 2.1), team formation (Section 2.2), graph theory (Section 2.3) and optimisation (Section 2.4).

2.1. Coalition Formation

2.1.1. Classic CSG algorithms. A number of algorithms have been developed to solve CSG for the general CF problem where all coalitions can be formed (i.e., non-GCCF). These range from mixed-integer programming to branch and bound techniques [Rahwan et al. 2009] through Dynamic Programming (DP) [Rahwan and Jennings 2008b]. In particular, Sandholm et al. [1999] and Dang and Jennings [2004] focused on providing anytime solutions with quality guarantees. However, their solutions do not scale (growing in $O(n^n)$) and, as discussed by Voice et al. [2012b], they cannot be employed to solve CSG for GCCF, since assigning artificially low values (such as $-\infty$) to infeasible coalitions would not be suitable for assessing valid bounds. Finally, Rahwan et al. [2008a; 2009; 2012] developed IDP-IP*, the state of the art algorithm for classic CSG. However, IDP-IP* is limited to tens of agents (30 at most) due to its memory requirements (i.e., $\Theta\left(2^n\right)$), as such approaches need to store all coalition values.

To overcome the intractability due to such memory requirements, a number of works [Ohta et al. 2009; Ueda et al. 2011; Tran-Thanh et al. 2013] have examined alternative function representations, which allow to reduce the computational complexity of the associated CF problems. Unfortunately, their models may not be able to capture the realistic nature of functions such as the collective energy purchasing one we consider here. On the one hand, this function cannot be concisely expressed as a MC network, as its MC network would require an exponential amount of memory with respect to the number of agents. On the other hand, the concepts of agent types/skills imply that it is possible to fully characterise the contribution of each agent on the basis of a small set of features, in order to achieve the conciseness of the representation. However, in our scenario each agent is associated to its own energy consumption profile, resulting in a number of types/skills equal to the number of agents. Hence, we do not compare against these works, since we are interested in developing techniques that can handle complex functions such as the collective energy purchasing function.

[2]This paper subsumes the work of Bistaffa et al. [2014b] and the non-archival work of Bistaffa et al. [2014a].

2.1.2. CSG algorithms based on heuristics. Very few heuristic solutions to the CSG problem have been developed over the last few years. For example, Sen and Dutta [2000] propose a solution based on genetic algorithms, Dos Santos and Bazzan [2012] propose an approach based on swarm intelligence (the bee clustering algorithm) for task allocation in the RoboCup Rescue domain, and Farinelli et al. [2013] propose an approach based on hierarchical clustering. Meta-heuristic approaches to CSG have also been investigated, for example Keinanen [2009] proposes a CSG algorithm based on Simulated Annealing, while Di Mauro et al. [2010] use a stochastic local search approach (GRASP) to iteratively build a coalition structure of high quality. Even if these approaches are not able to provide any guarantees on the solution quality, they can compute solutions for large numbers of agents. Hence, in Section 7.5 we compare CFSS against C-Link, since it is the most recent heuristic approach for CSG and it has been tested using the collective energy purchasing function, which we also consider.

2.1.3. Constrained CF. The works discussed above focus on unconstrained CF and cannot be directly used in contexts where constraints of various types may limit the formation of some coalitions. In this respect, Shehory and Kraus [1998] first introduced the idea, arising in many realistic scenarios, of restricting the maximum cardinality k of the coalitions in CSG, highlighting that, even though this constraint lowers the number of coalitions from exponential, i.e., 2^n, to polynomial, i.e., $O\left(n^k\right)$, the problem remains NP-hard. Therefore, the authors propose an approximate algorithm with quality guarantees, which, however, can be used if all $O\left(n^k\right)$ coalitions are valid.

On the other hand, Rahwan et al. [2011] developed a model of Constrained Coalition Formation (CCF), differing from standard CF due to the presence of constraints that forbid the formation of certain coalitions. However, authors provide an algorithm for optimal CSG only for *Basic* CCF (BCCF) games, which cannot be used to represent every GCCF problem, as shown in Section A.1 of the Appendix.

Finally, in a recent work, Iwasaki et al. [2015] proposed an approach to check the non-emptiness of the core when the grand coalition does not form, hence effectively addressing a CSG problem. Notice that, even though such an approach is tested on 1000 agents, the authors assume that the number of feasible coalitions is less than 10000. This assumption is not reasonable for large-scale scenarios we are interested to solve. For the sake of comparison, the number of feasible coalitions with 50 agents and $m = 1$ (i.e., the simplest network topology we consider in our tests) is ~ 150 billions, thus severely limiting the scalability of such an approach on large-scale scenarios due to its memory requirements.

2.1.4. State of the art algorithms for GCCF. Voice et al. [2012a; 2012b] were the first to propose algorithms for the GCCF problem. However, there are some drawbacks that hinder their applicability. First, [Voice et al. 2012a] can only be applied to characteristic functions fulfilling the independence of disconnected members (IDM) property. The IDM property requires that, given two disconnected agents i and j, the presence of agent i does not affect the marginal contribution of agent j to a coalition. This assumption is rather strong for real-world applications. As noticed by Shehory and Kraus [1998] considering task allocation, the addition of a new agent to a coalition could result in intra-coalition coordination and communication costs, which increase with the size of the coalition. Hence, realistic functions capturing such costs (such as the ones in Section 6.1) do not satisfy the IDM property, hence this approach cannot be applied. Second, the DyCE algorithm [Voice et al. 2012b] uses DP to find the optimal coalition structure by progressively splitting the current solution into its best partition. DyCE is not an anytime algorithm and requires an exponential amount of memory in the number of agents (i.e., $\Theta\left(2^n\right)$). Hence, the scalability of this approach is limited to systems consisting of tens of agents (around 30).

2.2. Team formation

The problem of forming groups of agents has also been widely studied in the context of Team Formation, in which several formal definitions of such problem have been proposed. As an example, Gaston and desJardins [2005] devise a heuristic to modify the graph connecting the agents based on local autonomous reasoning, without considering any concept of global optimal solution. The problem studied by Lappas et al. [2009] focuses on finding a single group of agents who possess a given set of skills, so as to minimise the communication cost within such a group. Marcolino et al. [2013] focus on forming a single group of agents that has the maximum strength in the set of world states. Finally, Liemhetcharat and Veloso [2014] are interested in modelling the values of the characteristic function, based on observations of the agents. In this paper, we address the specific group formation problem in which groups must form a partition (into disjoint coalitions) of a given set of agents, with the objective of maximising the sum of the coalitional values. Such problem is equivalent to the *complete set partitioning* problem [Yun Yeh 1986], i.e., the standard definition adopted in the CF literature.

2.3. Graph theory techniques

Our approach enumerates all the feasible partitions of the set of agents by means of the edge contraction operation, a graph theoretic technique known for its application in the algorithm to solve the Min-Cut problem [Karger 1993]. Edge contraction has never been employed in CF [Rahwan et al. 2015], hence we aim at investigating its use in this paper. In this context, the problem of enumerating all the connected subgraphs (corresponding to feasible coalitions in GCCF scenarios) of a given graph has been studied in a number of works [Voice et al. 2012b; Skibski et al. 2014]. Nonetheless, such algorithms can only be used to enumerate feasible coalitions, and cannot be applied to enumerate feasible coalition *structures* (as CFSS does), which are *sets* of disjoint feasible coalitions that collectively cover the entire set of agents.

2.4. Submodular-supermodular decomposition

Submodular functions have been widely studied in the optimisation literature [Schrijver 2003] in virtue of their natural *diminishing returns* property, which makes them suitable for many applications [Nemhauser et al. 1978; Narayanan 1997]. Moreover, Shekhovtsov et al. [2006; 2008] focused on general functions that can be decomposed as the sum of supermodular and submodular components, exploiting such a property to achieve better results in the solution of several optimisation problems.

While this approach is similar to the decomposition we propose in Section 6.1, our result holds for superadditive and subadditive functions (cf. Definition 4), which are *weaker* (i.e., more general) properties with respect to supermodularity and submodularity. In fact, it is easy to show that supermodularity (resp. submodularity) implies superadditivity (resp. subadditivity), but the converse is not true [Schrijver 2003].

3. GCCF PROBLEM DEFINITION

The Coalition Structure Generation (CSG) problem [Sandholm et al. 1999; Shehory and Kraus 1998] takes as input a finite set of n agents $\mathcal{A}$ and a characteristic function $v : 2^{\mathcal{A}} \to \mathbb{R}$, that maps each coalition $C \in 2^{\mathcal{A}}$ to its value, describing how much collective payoff a set of players can gain by forming a coalition. A coalition structure CS is a partition of the set of agents into disjoint coalitions. The set of all coalition structures is $\Pi(\mathcal{A})$. The value of a coalition structure CS is assessed as the sum of the values of its composing coalitions, i.e.,

$$V(CS) = \sum_{C \in CS} v(C). \tag{1}$$

CSG aims at identifying CS^*, the most valuable coalition structure, i.e., $CS^* = \arg\max_{CS \in \Pi(\mathcal{A})} V(CS)$. Graphs have been used in different scenarios to encode synergies, coordination among players, possible collaborations or cooperation structures [Myerson 1977; Voice et al. 2012b; Meir et al. 2012]. Myerson [1977] and Demange [2004] pioneered the study of graphs to model cooperation structures. Given an undirected graph $G = (\mathcal{A}, \mathcal{E})$, where $\mathcal{E} \subseteq \mathcal{A} \times \mathcal{A}$ is a set of edges between agents, representing the relationships between them, Myerson considers a coalition C to be feasible if all of their members are connected in the subgraph of G induced by C. That is, for each pair of players from $a, b \in C$ there is a path in G that connects them without going out of C. Thus, given a graph G the set of feasible coalitions is

$$\mathcal{FC}(G) = \{C \subseteq \mathcal{A} \mid \text{The subgraph induced by } C \text{ on } G \text{ is connected}\}.$$

A Graph-Constrained Coalition Formation (GCCF) problem is a CSG problem together with a graph G, in which a coalition C is considered feasible if $C \in \mathcal{FC}(G)$. Moreover, a coalition structure CS is considered feasible if each of its coalitions is feasible, i.e.,

$$\mathcal{CS}(G) = \{CS \in \Pi(\mathcal{A}) \mid CS \subseteq \mathcal{FC}(G)\}.$$

A GCCF problem aims at identifying the most valuable coalition structure, defined as $CS^* = \arg\max_{CS \in \mathcal{CS}(G)} V(CS)$.

In the next section, we propose a novel representation of the GCCF problem based on the concept of edge contraction.

4. A GENERAL ALGORITHM FOR GCCF

We now present a general algorithm to solve GCCF by showing that all feasible coalition structures induced by G can be modelled as the nodes of a search tree in which each feasible coalition structure is represented only once. Specifically, we first detail how we use edge contractions to represent the GCCF problem and then we provide a depth-first approach to build and traverse the search tree to find the optimal solution.

4.1. Generating feasible coalition structures via edge contractions

In this section we show that each $CS \in \mathcal{CS}(G)$ can be represented by a corresponding graph $G_{CS} = (\mathcal{V}, \mathcal{F})$, where $\mathcal{V} \subseteq 2^{\mathcal{A}}$ and $\mathcal{F} \subseteq \mathcal{V} \times \mathcal{V}$, i.e., each node $u \in \mathcal{V}$ represents a particular coalition. Notice that in the initial graph $G = (\mathcal{A}, \mathcal{E})$ each vertex $u \in \mathcal{A}$ represents a single agent, and hence, G can be seen as the representation of the feasible coalition structure formed by all the singletons.
In what follows, we will show that, for each $CS \in \mathcal{CS}(G)$, the corresponding G_{CS} can be obtained as the contraction of a set of edges of G, and that each contraction of a set of edges of G represents a feasible coalition structure $CS \in \mathcal{CS}(G)$. In more detail, let us define an *edge contraction* as follows.

DEFINITION 1. *Given a graph $G = (\mathcal{V}, \mathcal{F})$, where $\mathcal{V} \subseteq 2^{\mathcal{A}}$ and $\mathcal{F} \subseteq \mathcal{V} \times \mathcal{V}$, and an edge $e = (u, v) \in \mathcal{F}$, the result of the contraction of e is a graph G' obtained by removing e and the corresponding vertices u and v, and adding a new vertex $w = u \cup v$. Moreover, each edge incident to either u or v in G will become incident to w in G', merging the parallel edges (i.e., the edges that are incident to the same two vertices) that may result.*

Intuitively, one edge contraction represents the merging of the coalitions associated to the incident vertices. Figure 1 shows the contraction of the edge $(\{A\}, \{C\})$, which results in a new vertex $\{A, C\}$ connected to vertex $\{B\}$. Notice that edge contraction is a commutative operation (i.e., first contracting e and then e' results in the same graph as first contracting e' and then e). Hence, we can define the contraction of a set of edges as the result of contracting each of the edges of the set in any given order.

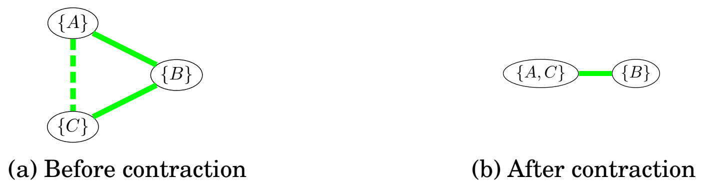


Fig. 1: Example of an edge contraction (the dashed edge is contracted).

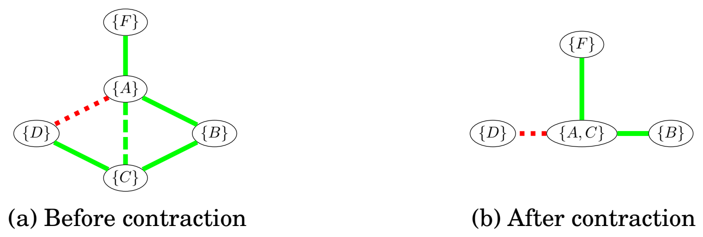


Fig. 2: Example of a 2-coloured edge contraction (the dashed edge is contracted).

Remark 4.1. Given a graph G, the graph G' resulting from the contraction of any set of edges of G represents a feasible coalition structure, where coalitions correspond to the vertices of G'.

Remark 4.2. Given a graph G, any feasible coalition structure CS can be generated by contracting a set of edges of G.

Thus, a possible way of listing all feasible coalition structures is to list the contraction of every subset of edges of the initial graph. However, notice that the number of subsets of edges is larger than the number of feasible coalition structures over the graph. For example, in the triangle graph in Figure 1a, the number of subsets of edges is $2^{|\mathcal{E}|} = 2^3 = 8$, but the number of feasible coalition structures is 5 (i.e., $\{A\}\{B\}\{C\}$, $\{A,B\}\{C\}$, $\{A,C\}\{B\}$, $\{A\}\{B,C\}$ and $\{A,B,C\}$). This redundancy is due to the fact that the contraction of any two or three edges leads to the same coalition structure, i.e., the grand coalition $\mathcal{A} = \{A,B,C\}$. Thus, we need a way to avoid listing feasible coalition structures more than once. To avoid such redundancies, we mark each edge of the graph to keep track of the edges that have been contracted so far. Notice that there are only two different alternative actions for each edge: either we contract it, or we do not. If we decide to contract an edge, it will be removed from the graph in all the subtree rooted in the current node, but if we decide not to contract it, we have to mark such edge to make sure that we do not contract it in the future steps of the algorithm. To represent such marking, we will use the notion of *2-coloured graph*.

DEFINITION 2. *A 2-coloured graph* $G_c = (\mathcal{V}, \mathcal{F}, c)$ *is composed of a set of vertices* $\mathcal{V} \subseteq 2^{\mathcal{A}}$ *and a set of edges* $\mathcal{F} \subseteq \mathcal{V} \times \mathcal{V}$*, as well as a function* $c : \mathcal{F} \rightarrow \{red, green\}$ *that assigns a colour* $(red$ *or* $green)$ *to each edge of the graph.*

In our case, a red edge means that a previous decision not to contract that edge was made. On the one hand, green edges can be still contracted. Figure 2a shows an example of a 2-colour graph in which edge $(\{A\}, \{D\})$ is coloured in red (dotted line). Hence, in any subsequent step of the algorithm it is impossible to contract it. On the other hand, all other edges in such graph can still be contracted. In a 2-coloured graph, we define a *green edge contraction* (e.g., dashed line in Figure 2a) as follows.

Algorithm 1 SolveGCCF(G_c)

```
1: $best \leftarrow G_c$, $F \leftarrow \emptyset$          ▷ Initialise solution with singletons and search frontier $F$ with empty stack
2: $F$.PUSH($G_c$)                                          ▷ Push $G_c$ as the first node to visit
3: while $F \neq \emptyset$ do                              ▷ Search loop
4:     $node \leftarrow F$.POP()                            ▷ Get current node
5:     if $V(node) > V(best)$ then                          ▷ Check function value
6:         $best \leftarrow node$                           ▷ Update current best solution
7:     $F$.PUSH(CHILDREN$(node)$)                          ▷ Update frontier $F$
8: return $best$                                            ▷ Return optimal solution
```

Algorithm 2 CHILDREN(G_c)

```
1: $G' \leftarrow G_c$, $Ch \leftarrow \emptyset$           ▷ Initialise graph $G'$ with $G_c$ and empty set of children
2: for all $e \in G_c : c(e) = green$ do                    ▷ For all green edges
3:     $Ch \leftarrow Ch \cup \{$GreenEdgeContraction$(G', e)\}$
4:     Mark edge $e$ with colour $red$ in $G'$
5: return $Ch$                                              ▷ Return the set of children
```

Definition 3. *Given a 2-coloured graph $G = (\mathcal{V}, \mathcal{F}, c)$ and a green edge $e \in \mathcal{F}$, the result of the contraction of e is a new graph G' obtained by performing the contraction of e on G. Whenever two parallel edges are merged into a single one, the resulting edge is coloured in red if at least one of them is red-coloured, and it is green-coloured otherwise.*

The rationale behind marking parallel edges in this way is that, whenever we mark an edge $e = (u, v)$ to be red, we want the agents in u and v to be in separate coalitions, hence whenever we merge some edges with e we must mark the new edge as red to be sure that future edge contractions will not generate a coalition that contains both the agents corresponding to nodes u and v. For example, note that in Figure 2 the red edge $(\{A\}, \{D\})$ (dotted in the figure) and the green edge $(\{D\}, \{C\})$ are merged as a consequence of the contraction of edge $(\{A\}, \{C\})$, resulting in an edge $(\{D\}, \{A, C\})$ marked in red. In this way, we enforce that any possible contraction in the new graph will keep agents A and D in separate coalitions.

Having defined how we can use the edge contraction operation to generate feasible coalition structures, we now provide a way to generate the whole search space of feasible coalition structures.

4.2. Generating the entire search space

Given the green edge contraction operation defined above, we can generate each feasible coalition structure only once. In more detail, at each point of the generation process, each red edge indicates that it has been discarded for contraction from that point onwards, and hence its vertices cannot be joined. Observe that the way we defined green edge contraction guarantees that the information in red edges is always preserved. Thus, given a 2-coloured graph, its children can be readily assessed as follows: for each edge in the graph, we generate the graph that results from contracting that edge. Moreover, we colour the selected edge in red so that it cannot be contracted again in subsequent edge contractions. Algorithm 1 implements the depth-first[3] generation and traversal of our search tree, in which each feasible coalition structure is evaluated by means of the characteristic function and compared with the best (i.e., the one with the highest value) coalition structure so far, hence computing the optimal solution.

[3]The DFS strategy allows us to traverse the entire tree with polynomial memory requirements, since at each stage of the search we only need to store the ancestors of the current node.

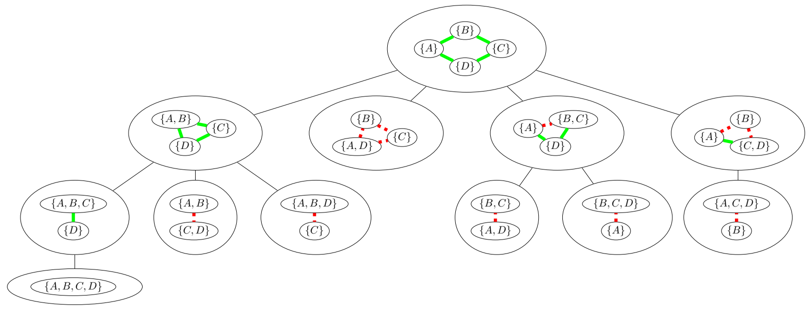


Fig. 3: Search tree for a square graph.

As an example, Figure 3 shows the search tree generated starting from a square graph, highlighting each generation step with labels on the edges. We now prove that Algorithm 1 visits all feasible coalition structures and each of them is visited only once.

PROPOSITION 4.3. *Given G_c, the tree generated by Algorithm 1 rooted at G_c contains all the coalition structures compatible with G_c, each appearing only once.*

SKETCH OF PROOF. By induction on the number of green edges. Full proof is provided in the Online Appendix.

PROPOSITION 4.4. *The complexity of Algorithm 1 is $O(|\mathcal{CS}(G)| \cdot |\mathcal{E}|)$.*

PROOF. There is a bijection between $\mathcal{CS}(G)$ and the nodes visited by Algorithm 1, by direct application of Proposition 4.3 to G with all green edges. The creation of each new node yields a GREENEDGECONTRACTION(G, e) operation, whose complexity is $O(|\mathcal{E}|)$ (Definition 3). Hence, the complexity of creating the search tree is $O(|\mathcal{CS}(G)| \cdot |\mathcal{E}|)$.[4] □

Notice that, even for sparse graphs, the number of feasible coalition structures can be very large, as, in general, the GCCF problem is NP-complete [Voice et al. 2012a]. Hence, in the next section we propose a branch and bound technique that helps prune significant parts of the search space, allowing us to compute the optimal solution for any GCCF problem based on an $m + a$ function by generating only a minimal portion of the solution space (i.e., less than 0.32% in our experiments in Section 7.2).

In addition, such a bounding technique is employed in the approximate version of our approach, which can compute solutions with quality guarantees for large-scale systems. It is important to note that, in contrast with the optimal version, our approximate approach is not characterised by the above discussed exponential complexity, as the search for the solution is executed only for a given time budget (see Section 5.2).

5. A GENERAL BRANCH AND BOUND ALGORITHM FOR $m + a$ FUNCTIONS

We now describe CFSS (Coalition Formation for Sparse Synergies), our branch and bound approach to GCCF when applied to the family of $m + a$ characteristic functions.

DEFINITION 4. *Given a graph G, a function $v : \mathcal{FC}(G) \to \mathbb{R}$ is superadditive (resp. subadditive) if the value of the union of disjoint coalitions is no less (resp. no greater) than the sum of the coalitions' separate values, i.e., $v(S \cup T) \geq$ (resp. $\leq$) $v(S) + v(T)$ for all $S, T \subseteq \mathcal{A}$ such that $S \cap T = \emptyset$.*

[4]Notice that, since Coalition Structure Generation (CSG) is a particular case of GCCF (i.e., CSG is a GCCF problem with a complete graph), $|\mathcal{CS}(G)|$ can be, in the worst case, equivalent to the n^{th} Bell number, i.e., $\Omega((\frac{n}{\ln(n)})^n)$ [Berend and Tassa 2010], where n is the number of agents. Nonetheless, in the problems we consider G is sparse and, hence, $\mathcal{CS}(G)$ contains a lower number of feasible coalition structures.

We also define such properties for the function $V : \mathcal{CS}(G) \to \mathbb{R}$ defined in Equation 1.

DEFINITION 5. *Given a graph G, a function $V : \mathcal{CS}(G) \to \mathbb{R}$ defined according to Equation 1 is superadditive (resp. subadditive) if the underlying function $v : \mathcal{FC}(G) \to \mathbb{R}$ is superadditive (resp. subadditive).*

DEFINITION 6. *Given a graph G, $V : \mathcal{CS}(G) \to \mathbb{R}$ is an $m{+}a$ function if it is the sum of a superadditive (i.e., monotonic increasing) function $V^+ : \mathcal{CS}(G) \to \mathbb{R}$ and a subadditive (i.e., antimonotonic) function $V^- : \mathcal{CS}(G) \to \mathbb{R}$.*

This family is interesting because it allows us to provide an upper bound that underlies our branch and bound strategy, so as to prune significant portions of the search space and have a computationally affordable solution algorithm. We provide a technique to compute an upper bound for the value assumed by the characteristic function in every coalition structure of the subtree $ST(CS_i)$ rooted at a given coalition structure CS_i. In order to explain how to compute such an upper bound, we first define the element $\overline{CS_i}$.

DEFINITION 7. *Given a feasible coalition structure CS_i represented by a 2-coloured graph G_c, we define $\overline{CS_i}$ as the coalition structure obtained by removing all red edges from G_c and then contracting all the remaining green edges. Intuitively, $\overline{CS_i}$ represents the connected components in the graph after the removal of all red edges.*

THEOREM 5.1. *Given an $m{+}a$ function $V : \mathcal{CS}(G) \to \mathbb{R}$, then $M(CS_i) = V^-(CS_i) + V^+\left(\overline{CS_i}\right)$ is an upper bound for the value assumed by such function in every coalition structure of the subtree $ST(CS_i)$ rooted at CS_i, i.e.,*

$$M(CS_i) = V^-(CS_i) + V^+\left(\overline{CS_i}\right) \geq \max\{V(CS_j) \mid CS_j \in ST(CS_i)\}. \tag{2}$$

SKETCH OF PROOF. $V^-(CS_i)$ (resp. $V^+\left(\overline{CS_i}\right)$) is an upper bound for the subadditive (resp. superadditive) component, hence $M(CS_i)$ is an upper bound for the characteristic function. Full proof is provided in the Online Appendix.

Remark 5.2. Given CS_i represented by a 2-coloured graph $G_c = (\mathcal{V}, \mathcal{F}, c)$, it is possible to compute a more precise upper bound for the edge sum with coordination cost function (see Section 6.1.2) by replacing $V^+\left(\overline{CS_i}\right)$ with $\sum_{e \in \mathcal{F} : c(e) = green} w^+(e)$.

Building upon Theorem 5.1, we can efficiently assess an upper bound for the value of the characteristic function in any subtree and prune it, if such a value is smaller than the value of the best solution found so far. Algorithm 3 implements CFSS, our branch and bound approach to solve the GCCF problem.

We remark that Algorithm 3 is correct and complete, i.e., it computes the optimal solution regardless of the order in which the children of the current node are visited, namely the operation of the CHILDREN function. However, such an order has a strong influence on the performance of CFSS (as shown in Section 7.3), since it can be used

Algorithm 3 CFSS(G_c)

```
1: $best \leftarrow G_c$, $F \leftarrow \emptyset$      ▷ Initialise solution with singletons and search frontier $F$ with empty stack
2: $F$.PUSH($G_c$)                                  ▷ Push $G_c$ as the first node to visit
3: while $F \neq \emptyset$ do                      ▷ Branch and bound loop
4:     $node \leftarrow F$.POP()                    ▷ Get current node
5:     if $M(node) > V(best)$ then                  ▷ Check bound value
6:         if $V(node) > V(best)$ then $best \leftarrow node$     ▷ Update current best solution
7:         $F$.PUSH(CHILDREN($node$))               ▷ Update frontier $F$
8: return $best$                                    ▷ Return optimal solution
```

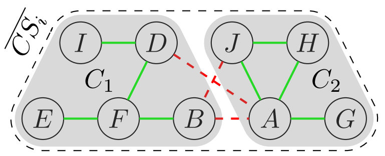


Fig. 4. Example of a partition with a cut-set of 3 edges.

to compute an upper bound that better resembles the characteristic function (hence improving the effectiveness of the branch and bound pruning).

5.1. Edge ordering heuristic

In this section we propose a heuristic to define a total ordering among the edges of a graph G, in order to guide the traversal of the search tree. This results in a significant speed-up of the algorithm, by means of an improvement of the upper bound. In particular, we notice that the value of $M\left(CS_i\right) = V^-\left(CS_i\right) + V^+\left(\overline{CS_i}\right)$ is heavily influenced by the value of $V^+\left(\overline{CS_i}\right)$. In fact, it is possible that $\overline{CS_i} = \{\mathcal{A}\}$ (i.e., the grand coalition), when CS_i contains enough green edges to connect all the nodes of the graph G. This results in a poor bound, since V^+ is a superadditive function and it reaches its maximum value for $\mathcal{A}$.

On the other hand, if red edges form a cut-set for the 2-coloured graph, the procedure in Definition 7 results in a coalition structure $\overline{CS_i} = \{C_1, C_2\}$, as Figure 4 shows. In this case, our bounding technique produces a *lower* upper bound $M\left(CS_i\right) = V^-\left(CS_i\right) + v^+\left(C_1\right) + v^+\left(C_2\right)$, since $v^+\left(\cdot\right)$ is superadditive and, therefore, $v^+\left(C_1\right) + v^+\left(C_2\right) \leq v^+\left(\mathcal{A}\right)$. Notice that, having an upper bound that provides a lower overestimation of the characteristic function is crucial for the performance of CFSS, as the condition at line 5 in Algorithm 3 would be verified less often, hence allowing us to prune bigger portions of the search space. Also, it easy to see that when the value of the characteristic function increases in a non-linear way with respect to the size of the coalitions (such as the functions we consider in this paper), the more C_1 and C_2 are closer to a *bisection* of $\mathcal{A}$ (i.e., the more $|C_1|$ and $|C_2|$ are close to $|\mathcal{A}|/2$), the more pronounced such improvement is.

Following this observation, it is preferable to visit the edges that produce a cut of the graph in the first steps of the algorithm, since they will result in the above-explained improvement once such edges are marked in red. Henceforth, we define a total ordering among the edges of G, producing an *ordered* graph G_o by means of Algorithm 4. Intuitively, such algorithm computes small[5] cut-sets by means of the routine $\textsc{Cut}(G)$, which outputs the subgraphs $G_1 = (\mathcal{V}_1, \mathcal{F}_1)$ and $G_2 = (\mathcal{V}_2, \mathcal{F}_2)$ resulting from the cut, and the cut-set $\mathcal{F}'$. Once the cut-set has been found, we label its edges as the first ones in the ordered graph, recursively applying such procedure for all the subsequent subgraphs resulting at each partitioning, until every edge has been ordered.

Remark 5.3. In the worst-case, Algorithm 4 makes $|\mathcal{E}|$ calls to $\textsc{Cut}$, whose complexity is $O(|\mathcal{E}|)$ [Karypis and Kumar 1998]. Hence, its worst-case complexity is $O(|\mathcal{E}|^2)$.

In addition to this edge ordering heuristic, our bounding technique can be employed to provide anytime approximate solutions, as shown in the next section.

[5]To traverse the minimum number of edges necessary to partition the graph, we need the *smallest* cut-set. Unfortunately, such a problem (known as the Minimum Bisection problem) is a well known NP-complete problem [Garey and Johnson 1990]. However, our heuristic does not need an optimal solution, since if a suboptimal cut-set (i.e., bigger than the optimal one) is used, our algorithm will still partition the graph in a higher number of steps, resulting in a slightly smaller improvement. Therefore, we adopt an approximate algorithm implemented with the METIS graph partitioning library [Karypis and Kumar 1998].

Algorithm 4 ORDER(G)

```
1:  $i \leftarrow 1$, $G_o \leftarrow G$, $Q \leftarrow \emptyset$        ▷ Initialise edge counter, ordered graph, and empty queue
2:  $Q$.PUSH($G$)                                                           ▷ Push $G$ as the first graph to partition
3:  while $Q \neq \emptyset$ do                                             ▷ Partitioning loop
4:      $\langle G_1, G_2, \mathcal{F}' \rangle \leftarrow$ CUT ($Q$.POP())   ▷ Partition current graph
5:      Label in $G_o$ each edge $\in \mathcal{F}'$ from $i$ to $i + |\mathcal{F}'| - 1$
6:      $i \leftarrow i + |\mathcal{F}'|$                                   ▷ Increase edge counter
7:      if $|\mathcal{V}_1| > 1$ then                                       ▷ If the first subgraph has at least 2 nodes...
8:          $Q$.PUSH($G_1$)                                                 ▷ ... enqueue it
9:      if $|\mathcal{V}_2| > 1$ then                                       ▷ If the second subgraph has at least 2 nodes...
10:         $Q$.PUSH($G_2$)                                                 ▷ ... enqueue it
11: return $G_o$                                                            ▷ Return ordered graph
```

5.2. Anytime approximate properties

Theorem 5.1 can be directly applied to compute an overall bound of an $m + a$ function, with anytime properties. More precisely, let us consider frontier F in Algorithm 3. When we expand frontier F (Line 9) we keep track of the highest value of $V(\cdot)$ in the visited nodes. Hence, given a frontier F, the bound $B(F)$ is defined as

$$B(F) = \max\{V\,(best)\,, \max_{CS \in F} M(CS)\} \tag{3}$$

Thus, $B(F)$ is the maximum between the values assumed by $V(\cdot)$ inside the frontier (i.e., $V\,(best)$) and an estimated upper bound outside of it (i.e., $\max_{CS \in F} M\,(CS)$). Notice that since each $M\,(CS)$ is an overestimation of the value of $V(\cdot)$ in the corresponding subtree, such a maximisation provides a valid upper bound for $V(\cdot)$ in the portion of search space not visited yet. Furthermore, the quality of $B(F)$ can only be improved by expanding frontier F. More formally, if F' is such an expansion, then

$$B\,(F) \geq B\,(F') \geq \max\{V(CS) \mid CS \in \mathcal{CS}\,(G)\}. \tag{4}$$

This can be easily verified using the definition of $M(\cdot)$. In fact, each bound resulting from the children of a substituted node $u \in F$ must be less or equal to $M(u)$ and, hence, Inequality 4 holds. Intuitively, the larger the search space explored, the better is the bound provided. Finally, notice that the fastest way to compute a bound for $V(\cdot)$ is to consider a frontier formed exclusively by the root (i.e., the coalition structure formed by all singletons). Assessing this bound has the same time complexity of computing M, i.e., $O(|\mathcal{E}|)$, and its quality can be satisfactory, as shown in Section 7.4.

After the discussion of our branch and bound algorithm for $m + a$ functions, in the next section we discuss some scenarios in which GCCF can be applied, and, in particular, we present three $m + a$ functions that will be used to evaluate our approach.

6. APPLICATIONS FOR GCCF

As previously discussed, GCCF is a well known model in cooperative game theory that can be applied to several realistic scenarios. In what follows, we focus on two real-world scenarios, namely social ridesharing and collective energy purchasing, that can be modelled as GCCF problems.

In the ridesharing domain, Ma et al. [2013] adopted an heuristic approach in order to increase the potential passenger coverage of a fleet of taxis, while decreasing the total travel mileage of the system. Later on, Bistaffa et al. [2015] tackled the optimisation problem of arranging one-time shared rides among a set of commuters connected through a social network, with the objective of minimising the overall travel cost. Unlike Ma et al. [2013], Bistaffa et al. [2015] explicitly consider coalitions, showing that such a scenario can be modelled as a GCCF problem where the set of feasible coalitions is restricted by the social network. Intuitively, each group of agents that travel in the same car is mapped to a feasible coalition, whose coalitional value is defined as the total travel cost of that particular car, i.e., the cost of driving through its passengers' pick-up and destination points. Bistaffa et al. [2015] show that the adoption of the GCCF model in this scenario leads to a cost reduction of up to -36.22% when applied to realistic datasets for both spatial and social data.

In the *collective energy purchasing* scenario [Vinyals et al. 2012], each agent is characterised by an energy consumption profile that represents its energy consumption throughout a day. A profile records the energy consumption of a household at fixed intervals (every half hour in our case). The characteristic function of a coalition of agents is the total cost that the group would incur if they bought energy as a collective in two different markets: the spot market, a short term market (e.g., half hourly, hourly) intended for small amounts of energy; and the forward market, a long term one in which larger amounts of energy (spanning weeks and months) can be bought at cheaper prices [Vinyals et al. 2012]. In the *edge sum with coordination cost* scenario, every edge is associated to a value that represents how well (or bad) those agents perform together, or the cost of completing a coordination task in a robotic environment [Dasgupta et al. 2012]. In the *coalition size with distance cost* scenario, the formation of coalitions favours bigger groups and maximises the similarity of the opinion among their members. Such application could be employed to cluster public opinion, or to detect the presence of "virtual coalitions" among members of a parliament based on their recorded votes (e.g., the votes by the Democratic and the Republican parties).

In addition to such practical motivations, these three scenarios are particularly interesting as they are modelled by characteristic functions part of a large family of functions, i.e., $m+a$ functions. In what follows, we discuss the properties of such functions, showing how they can be exploited to significantly speed-up the solution of the associated GCCF problem (see Section 5).

6.1. Benchmark $m+a$ functions

We now present three benchmark functions for GCCF, namely the *collective energy purchasing* function, the *edge sum with coordination cost* function and the *coalition size with distance cost* function. In particular, we are interested in their characterisation as $m+a$ functions, showing that they can be seen as the sums of the superadditive and the subadditive parts [Owen 1995]. Such characteristic functions are particularly interesting as they enable an efficient bounding technique to prune part of the search space during the execution of our branch and bound algorithm, presented in Section 5.

6.1.1. Collective energy purchasing.

In the collective energy purchasing scenario, Farinelli et al. [2013] proposed the characteristic function

$$v(C) = \underbrace{\sum_{t=1}^{T} q_S^t(C) \cdot p_S + T \cdot q_F(C) \cdot p_F}_{energy(C)} + \kappa(C),$$

where $T = 48$ is the number of energy measurements in each profile, $p_S \in \mathbb{R}^-$ and $p_F \in \mathbb{R}^-$ represent the unit prices of energy in the spot and forward market respectively, $q_F : \mathcal{FC}(G) \to \mathbb{R}^-$ stands for the time unit amount of electricity to buy in the forward market and $q_S^t : \mathcal{FC}(G) \to \mathbb{R}^-$ for the amount to buy in the spot market at time slot t.[6] $energy : \mathcal{FC}(G) \to \mathbb{R}^-$ represents the total energy cost.

[6]Unit prices (whose values are reported in Section 7) are negative numbers, i.e., they belong to the set $\mathbb{R}^- = \{i \in \mathbb{R} \mid i \leq 0\}$, to reflect the direction of payments. Thus, the values of the characteristic function are negative as well, hence they represent costs that, maintaining the maximisation task, we aim to minimise.

Finally, $\kappa : \mathcal{FC}(G) \to \mathbb{R}^-$ stands for a coalition management cost that depends on the size of the coalition and captures the intuition that larger coalitions are harder to manage. The definition of this cost depends on several low level issues (e.g., the capacity of the power networks connecting the customers in the groups, legal fees, and other costs associated to group contracts, etc.), hence a precise definition of this term goes beyond the scope of this paper. Following Farinelli et al. [2013] we use $\kappa(C) = -|C|^\gamma$ with $\gamma > 1$ to introduce a non-linear element that penalises the formation of larger coalitions. Hence, the *collective energy purchasing* function is defined as

$$V(CS) = \underbrace{\sum\nolimits_{C \in CS} \left[\sum\nolimits_{t=1}^{T} q_S^t (C) \cdot p_S + T \cdot q_F (C) \cdot p_F \right]}_{V^+(CS)} + \underbrace{\sum\nolimits_{C \in CS} \kappa (C)}_{V^-(CS)}.$$

PROPOSITION 6.1. *The collective energy purchasing function is* $m + a$.

SKETCH OF PROOF. The cost of the energy necessary to fulfil the aggregated consumption profiles of the coalitions, i.e., $V^+(CS)$, is clearly superadditive, while the sum of the coalition management costs, i.e., $V^-(CS)$, is subadditive, as they increase when coalition sizes increase. Full proof is provided in the Online Appendix.

6.1.2. Edge sum with coordination cost. In the *edge sum with coordination cost* function every edge of G is mapped to a real value by a function $w : \mathcal{E} \to \mathbb{R}$ [Deng and Papadimitriou 1994]. Each coalitional value is the sum of the weights of the edges among its members. In order to have a better description of the management and communication costs in larger coalitions, we also introduce a penalising factor $\kappa(C)$,[7] with the same definition given in the previous section. Hence, we define this function as

$$v(C) = \sum\nolimits_{e \in edges(C)} w(e) + \kappa (C), \tag{5}$$

where the function $edges : \mathcal{FC}(G) \to 2^{\mathcal{E}}$ provides the set of all the edges connecting any two members of a given coalition C, i.e., $edges(C) = \{(v_1, v_2) \in \mathcal{E} \mid v_1 \in C \text{ and } v_2 \in C\}$. In order to characterise this scenario with an $m + a$ function, we rewrite Equation 5 as

$$v(C) = \sum\nolimits_{e \in edges(C)} \left[w^+(e) + w^-(e) \right] + \kappa (C),$$

$$\text{where} \qquad w^+(e) = \begin{cases} w(e), & \text{if } w(e) \geq 0, \\ 0, & \text{otherwise,} \end{cases} \quad w^-(e) = \begin{cases} w(e), & \text{if } w(e) < 0, \\ 0, & \text{otherwise.} \end{cases}$$

In other words, $\sum_{e \in edges(C)} w^+(e)$ represents the sum of all the positive weights of the edges in $edges(C)$, while $\sum_{e \in edges(C)} w^-(e)$ represents the sum of the negative ones. The *edge sum with coordination cost* function is then defined as

$$V(CS) = \underbrace{\sum\nolimits_{C \in CS} \left[\sum\nolimits_{e \in edges(C)} w^+(e) \right]}_{V^+(CS)} + \underbrace{\sum\nolimits_{C \in CS} \left[\sum\nolimits_{e \in edges(C)} w^-(e) + \kappa (C) \right]}_{V^-(CS)}.$$

PROPOSITION 6.2. *The edge sum with coordination cost function is* $m + a$.

[7]Such penalising factor makes the edge sum with coordination cost function to violate the IDM property (cf. Section 2.1.4), therefore the approach proposed by Voice et al. [2012a] cannot be used.

SKETCH OF PROOF. It is easy to verify that $V^+(CS)$, i.e., the sum of all positive edges, is superadditive, while the sum of the negative ones, i.e., $V^-(CS)$, is subadditive. Full proof is provided in the Online Appendix.

6.1.3. Coalition size with distance cost. The *coalition size with distance cost* can be modelled evaluating each coalition C with the function

$$v(C) = |C|^\alpha - \sum\nolimits_{(i,j)\in C\times C} d(i,j), \tag{6}$$

where $\alpha \geq 1$, and $d : \mathcal{A} \times \mathcal{A} \to \mathbb{R}^+$ is a function that measures the distance between the opinions of agent i and agent j. From Equation 6 it follows that the input of our problem has size N^2, where N is the total number of agents, since we must know the distances between each pair or agents. The *coalition size with distance cost* function of a coalition structure CS is then defined as

$$V(CS) = \underbrace{\sum\nolimits_{C\in CS} |C|^\alpha}_{V^+(CS)} + \underbrace{\sum\nolimits_{C\in CS} \left[-\sum\nolimits_{(i,j)\in C\times C} d(i,j)\right]}_{V^-(CS)}.$$

PROPOSITION 6.3. *The coalition size with distance cost function is* $m + a$.

PROOF. On the one hand, it is easy to verify that $v^+(C) = |C|^\alpha$ is a superadditive function, assuming that $\alpha \geq 1$. On the other hand, $v^-(C) = -\sum_{(i,j)\in C\times C} d(i,j)$ is subadditive, since $v^-(C_1 \cup C_2) = v^-(C_1) + v^-(C_2) - \sum_{i\in C_1, j\in C_2} d(i,j) \leq v^-(C_1) + v^-(C_2)$. □

These functions will be used in our experimental evaluation in the next section.

7. EMPIRICAL EVALUATION

The main goals of our empirical evaluation of CFSS are:

(1) To evaluate its runtime performance with respect to DyCE considering a variety of graphs, both realistic (i.e., subgraphs of the Twitter network) and synthetic (i.e., scale-free networks). Additional experiments on community networks and a detailed discussion on these network topologies are in the Online Appendix.
(2) To evaluate the effectiveness of our bounding technique.
(3) To evaluate the anytime performance and guarantees that our approach can provide when scaling to very large numbers of agents (i.e., more than 2700).
(4) To compare the quality of our approximate solutions with the ones computed by C-Link [Farinelli et al. 2013] on large-scale instances.
(5) To evaluate the speed-up that can be obtained by using multi-core machines.
(6) To evaluate the speed-up produced by our edge ordering heuristic.

Following Voice et al. [2012b], we consider scale-free networks generated with the Barabási-Albert model with $m \in \{1, 2, 3\}$. This parameter determines the sparsity of the graph, as every newly added node is connected, on average, to m existing nodes. It is easy to verify that the average degree of a scale-free network is $\sim 2 \cdot m$. We compare our approach with DyCE in our three reference domains, measuring the runtime in seconds. In our characteristic functions we use the following parameters:

— Following Farinelli et al. [2013], in the *collective energy purchasing* function we set $p_S{=}{-80}$ and $p_F{=}{-70}$. The consumption data is provided by a realistic dataset, comprising the measurements collected over a month from 2732 households in the UK.
— In the *edge sum with coordination cost* function we assigned a uniformly distributed random weight within $[-10, 10]$ to each edge.
— Following Farinelli et al. [2013], in both the above scenarios we considered $\gamma = 1.3$.

— In the *coalition size with distance cost* function we assigned a uniformly distributed random value within $[0, 100]$ to each distance between a pair of different agents (with $d(i, i) = 0$), and we considered $\alpha = 2.2$, motivated by the remarks in Section 7.4.

We conducted an additional set of experiments in which the graph G is a subgraph of a large crawl of the Twitter social graph. Specifically, such dataset is a graph with 41.6 million nodes and 1.4 billion edges published as part of the work by Kwak et al. [2010]. We obtain G by means of a standard algorithm [Russell 2013] to extract a subgraph from a larger graph, i.e., a breadth-first traversal starting from a random node of the whole graph, adding each node and the corresponding arcs to G, until the desired number of nodes is reached.

Moreover, we implemented a multi-threaded version of CFSS, namely P-CFSS (i.e., Parallel CFSS), and we analysed the speed-up of P-CFSS using Amdahl's law [Amdahl 1967], as it provides the maximum theoretical speed-up that can be achieved. All our results refer to the average value over 20 repetitions for each experiment. CFSS[8] and C-Link are implemented in C, while we used the DyCE implementation provided by its authors. We run our tests on a machine with a 3.40GHz CPU and 32 GB of memory.

7.1. DyCE vs CFSS: runtime comparison

In our experiments using scale-free networks, CFSS outperforms DyCE when coalition values are shaped by the above-described benchmark functions (as shown in Figures 6a, 6b and 6c). Specifically, for the *edge sum with coordination cost* function, CFSS outperforms DyCE by 4 orders of magnitude on networks with average connectivity (i.e., for $m = 2$), and by 3 orders of magnitude on networks with higher connectivity (i.e., for $m = 3$). Most probably this is due to the fact that the upper bound we adopt in this case closely resembles the function, allowing us to prune significant portions of the search space (see Section 7.2 for a more detailed discussion). In the *collective energy purchasing* scenario with 30 agents and $m = 2$, CFSS is 4.7 times faster than DyCE, and it is at least 2 orders of magnitude faster for $m = 1$. However, DyCE is significantly faster (44 times) than CFSS for $m = 3$. The adoption of the *coalition size with distance cost* function produces a similar behaviour, with a performance improvement for our method. In fact, CFSS is 17 times faster than DyCE for $m = 2$, and only 3 times slower for $m = 3$. On the other hand, the runtime of DyCE equals the previous case, since this approach is not sensitive to the values of the characteristic function. In our tests using subgraphs of the Twitter network, CFSS is at least four orders of magnitude faster than DyCE when solving instances with 30 agents (the biggest instances that DyCE can solve), and it can scale up to 45 agents. These results confirm the very good performance of CFSS when considering sparse networks. In fact, the average degree of these subgraphs is comparable with the one of a scale-free network with $1 < m < 2$. In all our tests, we increased the number of agents until the execution time reached 10^5 seconds. Notice that, in general, DyCE cannot scale over 30 agents (due to its exponential memory requirements), while CFSS does not have such limitation, hence it is possible to reach instances with thousands of agents, as shown in Section 7.4.

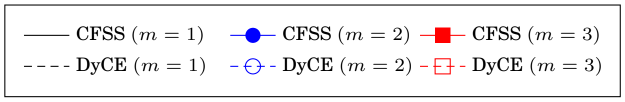
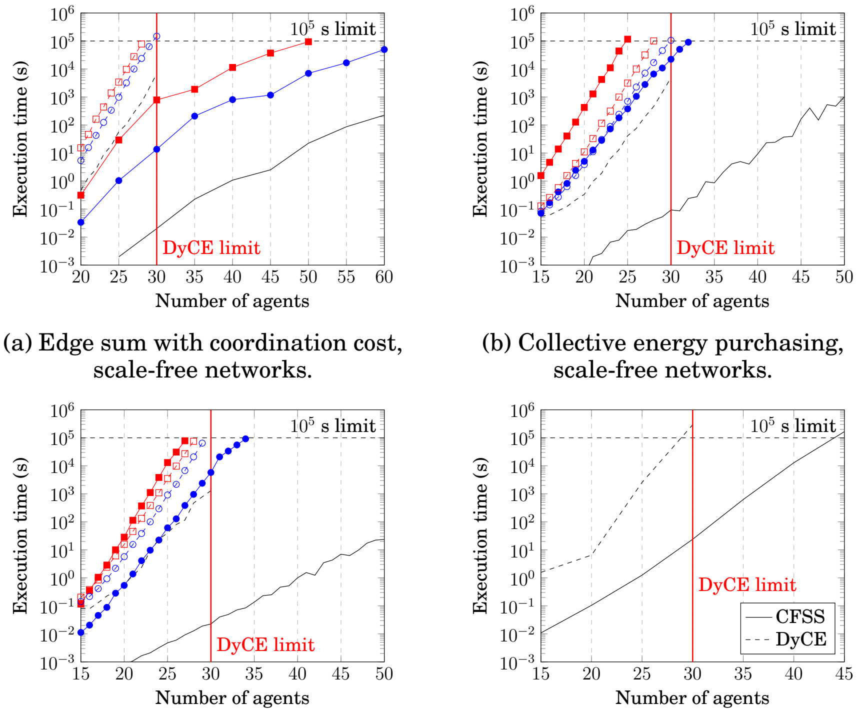


Fig. 5: Legend for scale-free networks.

[8]Our implementation of CFSS is publicly available at https://github.com/filippobistaffa/CFSS.



(a) Edge sum with coordination cost, scale-free networks.

(b) Collective energy purchasing, scale-free networks.

(c) Coalition size with distance cost, scale-free networks.

(d) Collective energy purchasing, Twitter subgraphs.

Fig. 6: Runtime to compute the optimal solution.

7.2. Bounding technique effectiveness

Here we compare the number of configurations explored by CFSS w.r.t. the entire search space, i.e., the one explored by Algorithm 1, to measure of the number of search nodes pruned by our bounding technique. We consider $n = 30$, adopting scale-free networks with $m = 2$. When the coalitional values are provided by the *collective energy purchasing* function, CFSS can compute the optimal solution exploring a number of configurations which is, on average, 0.32% of the entire search space. We measured a similar value in the *coalition size with distance cost* scenario (i.e., 0.28%). In the *edge sum with coordination cost* scenario (which allows a more precise upper bound, as explained in Remark 5.2), only 0.0045% of the entire search space is explored.

7.3. Edge ordering heuristic

The above table shows the speed-up obtained by using the ordering heuristic described in Section 5.1 and considering the *collective energy purchasing* and the *coalition size with distance cost* functions. Even though our heuristic is applicable also in the *edge sum with coordination cost* scenario, such function has not been included in this analysis since, as stated in Remark 5.2, it allows an ad-hoc bounding method that is more effective than the general one. Our experiments show a clear benefit in the adoption of such a heuristic, producing a maximum performance gain of 843% in the first scenario and 338% in the second one. Across all experimental scenarios, such a heuristic allows an average speed-up of 295% considering both domains.

Characteristic function	**Minimum**	**Average**	**Maximum**
Collective energy purchasing	176%	367%	843%
Coalition size with distance cost	136%	222%	338%

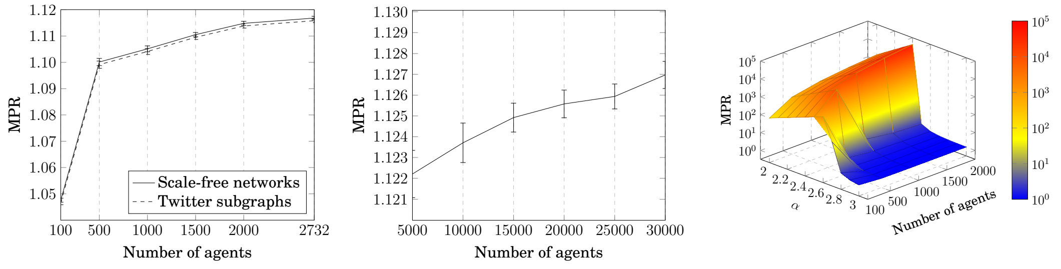
7.4. Anytime approximate performance

We evaluate the performance of the approximate version of CFSS on instances with thousands of agents considering the *Performance Ratio* (PR) [Ausiello et al. 2012], a standard measure to evaluate approximate algorithms defined as the ratio between the approximate solution and the optimal one on a given instance I. As computing the optimal solution for such large instances is not possible, we define the *Maximum Performance Ratio* (MPR) as the ratio between the approximate solution and the upper bound on the optimal solution defined in Equation 3.

DEFINITION 8. *Given an instance I, an approximate solution $Approx(I)$ and an upper bound on the optimal solution as $Bound(I)$, we define the Maximum Performance Ratio $MPR(I) = \max\left(\frac{Approx(I)}{Bound(I)}, \frac{Bound(I)}{Approx(I)}\right)$.*

$MPR(I)$ represents an upper bound of the PR on the instance I. The MPR provides an important quality guarantee on the approximate solution $Approx(I)$, since $Approx(I)$ cannot be worse than by a factor of $MPR(I)$ w.r.t. the optimal solution.

7.4.1. Collective energy purchasing. Figure 7a shows the value of the MPR in the *collective energy purchasing* scenario, using $n \in \{100, 500, 1000, 1500, 2000, 2732\}$, adopting scale-free networks with $m = 4$ and Twitter subgraphs as network topologies, and considering a time budget of 100 seconds. Other values for m show a similar behaviour (not reported here). We plot the average and the standard error of the mean over 20 repetitions. It is clear that the network topology does not impact the quality guarantees of our approach, hence we only adopt scale-free networks in the following experiments. In contrast, the MPR is heavily influenced by the nature of the characteristic function, as clarified later in this section. In addition, the results show that, for 100 agents, the provided bound is only 4.7% higher than the solution found within the time limit, reaching a maximum of +11.65% when the entire dataset is considered, i.e., with 2732 agents. Such small decrease is due to the fact that, for bigger instances, it is possible to explore a smaller part of the search space in the considered time budget, leaving a bigger portion to the estimation of the bound. Nonetheless, in this experiment CFSS provides a MPR of at most 1.12 and thus solutions that are at least 88% of the optimal. This confirms the effectiveness of this bounding technique when applied to the energy domain, which allows us to provide solutions and quality guarantees for problems involving a very large number of agents. In our tests, the bound is assessed at the root, without any frontier expansion, so it can be computed almost instantly, thus devoting all the available runtime to the search for a solution. This choice is further motivated by the fact that, in this scenario, the bound improves of a negligible value in the first levels of the search tree, due to the particular definition of the characteristic function. More precisely, if we consider a frontier formed by the children of the root, in each of them the bound of $V^-(\cdot)$ will improve by a factor of $2^\gamma - 2 \approx 1.5$ (i.e., the difference between the coalition management cost of the new coalition and the ones of the two merged singletons). On the other hand, the bound of $V^+(\cdot)$ will remain constant: in fact, since we are taking the maximum (i.e., the worst) bound at the frontier (as shown in Equation 3), the result of this maximisation will still be equal to $v^+(\mathcal{A})$, because in at least one of the children nodes the computation of $\overline{CS}$ will result in joining all the agents together. In this case, it is not worth to expand the frontier from the root, since the gain would be insignificant w.r.t. the additional computational cost.



(a) Collective energy purchasing.

(b) Edge sum with coordination cost.

(c) Coalition size with distance cost.

Fig. 7: Maximum Performance Ratio (MPR) in the considered domains.

7.4.2. Edge sum with coordination cost. We further evaluate the scalability of our approach by considering Twitter subgraphs as network topologies, and the *edge sum with coordination cost* function, which allows to generate coalitional values for instances with any number of agents. Such a function can be either positive or negative (in contrast with the *collective energy purchasing* one, which is always negative to represent its nature of cost). Hence, it is possible that $Approx(I)$ is negative and $Bound(I)$ is positive, resulting in a *negative* MPR. In order to avoid this unreasonable behaviour, here we consider $MPR(I) = \frac{Bound(I)-LB(I)}{Approx(I)-LB(I)}$, where $LB(I)$ is a lower bound on the characteristic function considering the instance I. Notice that it is always possible to compute $LB(I)$ for the *edge sum with coordination cost* function as $LB(I) = V^{-}(\mathcal{A})$.

Figure 7b shows that, on our machine, CFSS can scale up to instances with 30000 agents, providing solutions with a MPR of 1.127 (at least 89% of the optimal).

7.4.3. Coalition size with distance cost. The MPR exhibits a different behaviour when considering the *coalition size with distance cost* function, being heavily influenced by the value of the α exponent. Figure 7c shows how the MPR varies significantly with respect to $\alpha \in [2, 3]$, growing up to 41825.6 for $\alpha = 2.4$ and then falling down to 1.13 for $\alpha = 2.7$, with a tendency to 1 when increasing this exponent. This behaviour can be explained by reasoning about the structure of the characteristic function. Up to $\alpha = 2.4$, the subadditive component (i.e., $-\sum_{C \in CS} \sum_{(i,j) \in C \times C} d(i, j)$) dominates the superadditive one (i.e., $\sum_{C \in CS} |C|^{\alpha}$), hence the search for a solution is not able to find any coalition structure better than the initial one (i.e., the coalition structure with all singletons, which is probably the optimal one). Nonetheless, the MPR keeps growing when we increase α, since it equals $\frac{N^{\alpha}}{N} = N^{\alpha-1}$, i.e., the bound computed at the root (i.e., $V^{+}(\mathcal{A}) = N^{\alpha}$) divided by the value of the initial solution (i.e., N). On the other hand, when α is sufficiently large (i.e., for $\alpha = 2.5$), this behaviour is inverted, because $V^{+}(\cdot)$ has a greater impact and the entire characteristic function tends to become superadditive. Thus, coalition structures closer to the grand coalition represent good solutions, which explains why the MPR tends to 1 when we increase α. These remarks motivate us to study the impact of α also on the optimal algorithm. Figure 8 displays the runtime needed to find the optimal solution on random instances with 25 agents on scale-free networks with $m = 2$, showing that the performance of CFSS decreases when we increase α from 2 to 3. The value of the bound provided by Equation 2 is larger when α grows, hence its quality decreases, producing a less effective bounding technique and, thus, a higher runtime. To summarise, the adoption of a bigger α in the *coalition size with distance cost* function negatively impacts the performance of our approach when computing optimal solutions, while improving approximate solutions as α grows. This

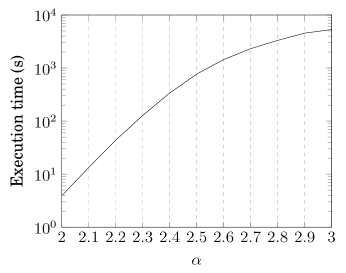


Fig. 8: Runtime w.r.t. α.

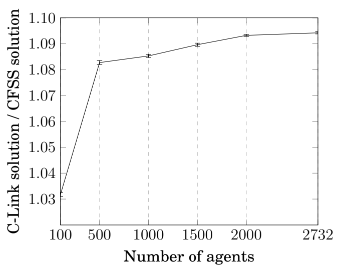


Fig. 9: C-Link vs. CFSS.

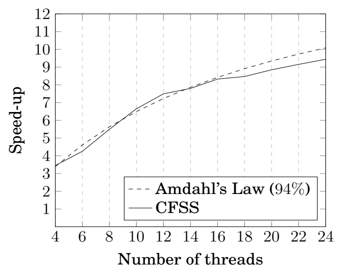


Fig. 10: Parallel speed-up.

motivates our choice of defining $\alpha = 2.2$ in the previous experiments, as it represents a good value to benchmark CFSS. In fact, it is big enough to avoid excessively low runtimes in the optimal version, but it does not exceed the 2.4 boundary, beyond which the quality guarantees it provides are extremely good (i.e., the MPR tends to 1).

7.5. CFSS vs C-Link: solution quality comparison

We further evaluate the approximate performance of CFSS by comparing it against C-Link [Farinelli et al. 2013], an heuristic approach to solve CSG based on hierarchical clustering. We chose C-Link among the other approaches discussed in Section 2.1.2 because it is the most recent one and it has also been tested using the *collective energy purchasing* function by its authors. Here we adopt the same experimental setting discussed in the previous section, i.e., we consider scale-free networks with $n \in \{100, 500, 1000, 1500, 2000, 2732\}$ and $m{=}4$ (generating 20 random repetitions of each experiment), and we adopt the *collective energy purchasing* characteristic function. We solve each instance with C-Link (adopting the best heuristic proposed by Farinelli et al. [2013], i.e., Gain-Link) and then we run CFSS on the same instance with a time budget equal to C-Link's runtime. Figure 9 shows the average and the standard error of the mean of the ratio between the value of the solution computed by C-Link and the one computed by CFSS. Since we consider solutions with negative values, when such ratio is > 1 the solution computed by C-Link is better (i.e., has a lower cost) than the one computed by CFSS. Our results show that, even though C-Link can compute better solutions, the quality of our solutions is worse only by 3% for 100 agents. When we consider the entire dataset (i.e., with 2732 agents) the quality of our solutions is still within the 9% w.r.t. the counterpart. Notice that C-Link slightly outperforms CFSS. This comes as no surprise since the fundamental difference between C-Link and CFSS is that C-Link does a backtrack-free visit of the search graph adopting a greedy heuristic to determine the choice at each step. In other words, C-Link explores only one path of the search graph. On the other hand, CFSS does not employ any heuristic as it is designed to execute a systematic visit of the search graph with backtracking. Notice that we can easily include the C-Link's greedy heuristic into CFSS to guide the visit of the children nodes in the search. With C-Link's heuristic, CFSS first explores the same path explored by C-Link, and then, if given more time, continues the visit of the rest of the search space by backtracking. Since we provide CFSS with a time budget equal to C-Link's runtime, if we employ C-Link's heuristic then CFSS effectively becomes the same algorithm as C-Link, and hence returns solutions of the same quality.

7.6. P-CFSS

Here we detail the parallelisation approach of the multi-threaded version of CFSS, analysing the speed-up with respect to its serial version. Following Bader et al. [2005], parallelisation is achieved by having different threads searching different branches of the search tree. The only required synchronisation point is the computation of the current best solution that must be read and updated by every thread. In particular, the distribution of the computational burden among the t_a available threads is done by considering the first i subtrees rooted in every node of the first generation (starting from the left) and assigning each of them to t_j threads ($1 \leq j \leq i$). The remaining rightmost subtrees are computed by a team of $t_a - \sum_{j=1}^{i} t_j$ threads using a dynamic schedule.[9] Parameters i and t_j are arbitrarily set, since it is assumed (and verified by an empirical analysis) that the distribution of the nodes over the search tree does not significantly vary among different instances. More advanced techniques, such as estimating the number of nodes in the search tree as suggested by Lelis et al. [2013], will be considered in the future. We run P-CFSS on random instances with 27 agents on scale-free networks with $m = 2$, using a machine with 2 Intel® Xeon® E5-2420 processors. The speed-up measured during these tests has been compared with the maximum theoretical one provided by Amdahl's Law, considering an estimated non-parallelisable part of 6%, due to memory allocation and thread initialisation.

As can be seen in Figure 10, the actual speed-up follows the theoretical one up to 12 threads, the number of physical cores. After that, hyper-threading still provides some improvement, reaching a final speed-up of 9.44 with all 24 threads active.

8. CONCLUSIONS

In this paper we considered the GCCF problem and proposed a branch and bound solution (the CFSS algorithm) that can be applied to a general class of functions (i.e., $m+a$ functions). Our empirical evaluation shows that CFSS outperforms DyCE, the state of the art algorithm, when applied to three characteristic functions. Specifically, CFSS is at least 3 orders of magnitude faster than DyCE in the first scenario, while solving bigger instances for the remaining two. Moreover, the adoption of our edge ordering heuristic provides a further speed-up of 296%. P-CFSS, the parallel version of CFSS, achieves a speed-up of 944% on a 12-core machine, close to the maximum theoretical speed-up. Finally, our algorithm provides approximate solutions with good quality guarantees (i.e., with a MPR of 1.12 in the worst case) for systems of unprecedented scale (i.e., more than 2700 agents). Overall, our work is the first to show how coalition formation techniques can start coping with real-world scenarios, opening the possibility of employing coalition formation on practical applications, rather than purely synthetic, small-scale environments.

Future work will look at applying our approach to other realistic scenarios (e.g., the formation of team of experts connected by a social network [Lappas et al. 2009]) and focusing on different multi-threading models (e.g., GPUs).

ELECTRONIC APPENDIX

The electronic appendix for this article can be accessed in the ACM Digital Library.

ACKNOWLEDGMENTS

COR (TIN 2012-38876-C02-01), Collectiveware TIN 2015-66863-C2-1-R (MINECO/FEDER), and the Generalitat of Catalunya 2014-SGR-118 funded Cerquides and Rodríguez-Aguilar. This work was also supported by the EPSRC-Funded ORCHID Project EP/I011587/1.

REFERENCES

Gene M. Amdahl. 1967. Validity of the single processor approach to achieving large scale computing capabilities. In *American Federation of Information Processing Societies*. 483–485.

[9]Once a thread has completed the computation of one subtree, it starts with one of the remaining ones.

Giorgio Ausiello, Pierluigi Crescenzi, Giorgio Gambosi, Viggo Kann, Alberto Marchetti-Spaccamela, and Marco Protasi. 2012. *Complexity and approximation: Combinatorial optimization problems and their approximability properties*. Springer.

David A. Bader, William E. Hart, and Cynthia A. Phillips. 2005. Parallel Algorithm Design for Branch and Bound. In *Emerging Methodologies and Applications in Operations Research*. Springer, 5–44.

Daniel Berend and Tamir Tassa. 2010. Improved bounds on Bell numbers and on moments of sums of random variables. *Probability and Mathematical Statistics* 30 (2010), 185–205.

Filippo Bistaffa, Alessandro Farinelli, Jesús Cerquides, Juan Rodríguez-Aguilar, and Sarvapali D. Ramchurn. 2014a. Anytime Coalition Structure Generation on Scale-Free and Community Networks. In *International Joint Workshop on Optimisation in Multi-Agent Systems and Distributed Constraint Reasoning*.

Filippo Bistaffa, Alessandro Farinelli, Jesús Cerquides, Juan Rodríguez-Aguilar, and Sarvapali D. Ramchurn. 2014b. Anytime Coalition Structure Generation on Synergy Graphs. In *International Conference on Autonomous Agents and Multi-Agent Systems*. 13–20.

Filippo Bistaffa, Alessandro Farinelli, and Sarvapali D. Ramchurn. 2015. Sharing Rides with Friends: a Coalition Formation Algorithm for Ridesharing. In *AAAI Conference on Artificial Intelligence*. 608–614.

Viet Dung Dang and Nicholas R. Jennings. 2004. Generating coalition structures with finite bound from the optimal guarantees. In *International Conference on Autonomous Agents and Multi-Agent Systems*. 564–571.

Prithviraj Dasgupta, Vladimir Ufimtsev, Carl Nelson, and S. G. M. Hossain. 2012. Dynamic reconfiguration in modular robots using graph partitioning-based coalitions. In *International Conference on Autonomous Agents and Multi-Agent Systems*. 121–128.

Gabrielle Demange. 2004. On Group Stability in Hierarchies and Networks. *Political Economy* 112 (2004), 754–778.

Xiaotie Deng and Christos H. Papadimitriou. 1994. On the Complexity of Cooperative Solution Concepts. *Mathematics of Operations Research* 19 (1994), 257–266.

Nicola Di Mauro, Teresa M A. Basile, Stefano Ferilli, and Floriana Esposito. 2010. Coalition Structure Generation with GRASP. In *International Conference on Artificial Intelligence: Methodology, Systems, Applications*. 111–120.

Daniela Dos Santos and Ana Bazzan. 2012. Distributed clustering for group formation and task allocation in Multi-Agent systems: A swarm intelligence approach. *Applied Soft Computing* 12 (2012), 2123–2131.

Alessandro Farinelli, Manuele Bicego, Sarvapali Ramchurn, and Mauro Zucchelli. 2013. C-link: A Hierarchical Clustering Approach to Large-scale Near-optimal Coalition Formation. In *International Joint Conference on Artificial Intelligence*. 106–112.

Michael R. Garey and David S. Johnson. 1990. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co.

Matthew E. Gaston and Marie desJardins. 2005. Agent-organized Networks for Dynamic Team Formation. In *International Conference on Autonomous Agents and Multi-Agent Systems*. 230–237.

Hampshire County Council. 2014. Switch Hampshire. (2014). http://www3.hants.gov.uk/switch

Atsushi Iwasaki, Suguru Ueda, Naoyuki Hashimoto, and Makoto Yokoo. 2015. Finding core for coalition structure utilizing dual solution. *Artificial Intelligence* 222 (2015), 49–66.

David R. Karger. 1993. Global Min-cuts in RNC, and Other Ramifications of a Simple Min-out Algorithm. In *ACM-SIAM Symposium on Discrete Algorithms*. 21–30.

George Karypis and Vipin Kumar. 1998. A Fast and High Quality Multilevel Scheme for Partitioning Irregular Graphs. *SIAM Journal on Scientific Computing* 20 (1998), 359–392.

Helena Keinanen. 2009. Simulated Annealing for Multi-Agent Coalition Formation. In *Agent and Multi-Agent Systems: Technologies and Applications*. 30–39.

Haewoon Kwak, Changhyun Lee, Hosung Park, and Sue Moon. 2010. What is Twitter, a Social Network or a News Media?. In *World Wide Web*. 591–600.

Theodoros Lappas, Kun Liu, and Evimaria Terzi. 2009. Finding a team of experts in social networks. In *ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 467–476.

Levi H. S. Lelis, Lars Otten, and Rina Dechter. 2013. Predicting the Size of Depth-First Branch and Bound Search Trees. In *International Joint Conference on Artificial Intelligence*. 594–600.

Somchaya Liemhetcharat and Manuela Veloso. 2014. Weighted synergy graphs for effective team formation with heterogeneous ad hoc agents. *Artificial Intelligence* 208 (2014), 41–65.

Shuo Ma, Yu Zheng, and Ouri Wolfson. 2013. T-share: A large-scale dynamic taxi ridesharing service. In *International Conference on Data Engineering*. 410–421.

Leandro Soriano Marcolino, Albert Xin Jiang, and Milind Tambe. 2013. Multi-agent Team Formation: Diversity Beats Strength?. In *International Joint Conference on Artificial Intelligence*. 279–285.

Reshef Meir, Yair Zick, and Jeffrey S Rosenschein. 2012. Optimization and stability in games with restricted interactions. In *Workshop on Cooperative Games in Multi-Agent Systems*.

Roger B. Myerson. 1977. Graphs and Cooperation in Games. *Mathematics of Operations Research* 2 (1977), 225–229.

Hariharan Narayanan. 1997. *Submodular functions and electrical networks*. Elsevier.

George L Nemhauser, Laurence A Wolsey, and Marshall L Fisher. 1978. An analysis of approximations for maximizing submodular set functions. *Mathematical Programming* 14 (1978), 265–294.

Naoki Ohta, Vincent Conitzer, Ryo Ichimura, Yuko Sakurai, Atsushi Iwasaki, and Makoto Yokoo. 2009. Coalition structure generation utilizing compact characteristic function representations. In *Principles and Practice of Constraint Programming*. 623–638.

Guillermo Owen. 1995. *Game Theory*. Academic Press.

Talal Rahwan and Nicholas. R. Jennings. 2008a. Coalition Structure Generation: Dynamic Programming Meets Anytime Optimisation. In *AAAI Conference on Artificial Intelligence*. 156–161.

Talal Rahwan and Nicholas R. Jennings. 2008b. An improved dynamic programming algorithm for coalition structure generation. In *International Conference on Autonomous Agents and Multi-Agent Systems*. 1417–1420.

Talal Rahwan, Tomasz P. Michalak, Edith Elkind, Piotr Faliszewski, Jacek Sroka, Michael Wooldridge, and Nicholas R Jennings. 2011. Constrained Coalition Formation. In *AAAI Conference on Artificial Intelligence*. 719–725.

Talal Rahwan, Tomasz P. Michalak, and Nicholas R. Jennings. 2012. A hybrid algorithm for coalition structure generation. In *AAAI Conference on Artificial Intelligence*. 1443–1449.

Talal Rahwan, Tomasz P Michalak, Michael Wooldridge, and Nicholas R Jennings. 2015. Coalition structure generation: A survey. *Artificial Intelligence* 229 (2015), 139–174.

Talal Rahwan, Sarvapali Ramchurn, Nicholas Jennings, and Andrea Giovannucci. 2009. An anytime algorithm for optimal coalition structure generation. *Journal of Artificial Intelligence Research* 34 (2009), 521–567.

Matthew A. Russell. 2013. *Mining the Social Web*. O'Reilly Media.

Tuomas Sandholm, Kate Larson, Martin Andersson, Onn Shehory, and Fernando Tohmé. 1999. Coalition structure generation with worst case guarantees. *Artificial Intelligence* 111 (1999), 209–238.

Alexander Schrijver. 2003. *Combinatorial optimization: polyhedra and efficiency*. Springer.

Sandip Sen and Partha S. Dutta. 2000. Searching for optimal coalition structures. In *International Conference on Autonomous Agents and Multi-Agent Systems*. 287–292.

Onn Shehory and Sarit Kraus. 1998. Methods for task allocation via agent coalition formation. *Artificial Intelligence* 101 (1998), 165–200.

Alexander Shekhovtsov. 2006. Supermodular decomposition of structural labeling problem. *Control systems and computers* 1 (2006), 20.

Alexander Shekhovtsov, Vladimir Kolmogorov, Pushmeet Kohli, Václav Hlavác, Carsten Rother, and Philip Torr. 2008. *LP-relaxation of binarized energy minimization*. Technical Report. Czech Tech. University.

Oskar Skibski, Tomasz P. Michalak, Talal Rahwan, and Michael Wooldridge. 2014. Algorithms for the Shapley and Myerson Values in Graph-restricted Games. In *International Conference on Autonomous Agents and Multi-Agent Systems*. 197–204.

Long Tran-Thanh, Tri-Dung Nguyen, Talal Rahwan, Alex Rogers, and Nicholas R Jennings. 2013. An efficient vector-based representation for coalitional games. In *International Joint Conference on Artificial Intelligence*. 383–389.

Suguru Ueda, Makoto Kitaki, Atsushi Iwasaki, and Makoto Yokoo. 2011. Concise characteristic function representations in coalitional games based on agent types. In *International Conference on Autonomous Agents and Multi-Agent Systems*. 1271–1272.

Meritxell Vinyals, Filippo Bistaffa, Alessandro Farinelli, and Alex Rogers. 2012. Coalitional energy purchasing in the smart grid. In *IEEE International Energy Conference*. 848–853.

Thomas Voice, Maria Polukarov, and Nicholas R. Jennings. 2012a. Coalition structure generation over graphs. *Journal of Artificial Intelligence Research* 45 (2012), 165–196.

Thomas Voice, Sarvapali D. Ramchurn, and Nicholas R. Jennings. 2012b. On coalition formation with sparse synergies. In *International Conference on Autonomous Agents and Multi-Agent Systems*. 223–230.

D. Yun Yeh. 1986. A dynamic programming approach to the complete set partitioning problem. *BIT Numerical Mathematics* 26 (1986), 467–474.