Anytime Coalition Structure Generation on Scale-Free and Community Networks

Filippo Bistaffa[1], Alessandro Farinelli[1], Jesús Cerquides[2⋆], Juan A. Rodríguez-Aguilar[2⋆], and Sarvapali D. Ramchurn[3⋆⋆]

[1] University of Verona, Italy
[2] IIIA-CSIC, Spain
[3] University of Southampton, UK

Abstract. We consider the coalition structure generation (CSG) problem on synergy graphs, which arises in many practical applications where communication constraints, social or trust relationships must be taken into account when forming coalitions. We propose a novel representation of this problem based on the concept of *edge contraction*, and an innovative branch and bound approach (CFSS), which is particularly efficient when applied to a general class of characteristic functions. We evaluate CFSS on two particular synergy graph topologies, i.e., *scale-free networks* and *community networks*, which provide an efficient description of many interesting real-world scenarios, e.g., social networks, authorships, collective energy purchasing and carpooling. Our results show that, when the graphs are very sparse, CFSS is 4 orders of magnitude faster than DyCE, the best algorithm to date for CSG on synergy graphs.

1 Introduction

Coalition formation is one of the fundamental approaches for establishing collaborations among agents, each with individual capabilities and properties [13]. In particular, coalition structure generation (CSG) represents a key computational task in this scenario [12]. Now, in many real-world applications, sparse synergies between the agents may constrain the formation of some coalitions [14, 15]. These constraints may be due to communication infrastructures (e.g., non-overlapping communication loci or energy limitations for sending messages across a network), social or trust relationships (e.g., energy consumers who prefer to group with their friends and relatives in forming energy cooperatives), or physical constraints (e.g., emergency responders that have enough fuel to join only specific teams).

Myerson [10] and Demange [4] pioneered the study of *graph-restricted cooperative games*, in which a coalition is feasible if and only if its members represent the vertices of a connected subgraph of the considered graph. Subsequently, several approaches [15] recall this constraint representation as graphs in CSG problems, addressing the optimisation subproblem (named *CSG on synergy graphs*) and dropping any stability-related aspect. Similarly, this paper maintains the same nomenclature, focusing on the optimisation part and neglecting stability: thus, even the problem is referred with a slightly different terminology, the discussion is equivalent to [10, 4].

⋆ Cerquides and Rodríguez-Aguilar are funded by projects COR (TIN 2012-38876-C02-01), AT (CSD2007-0022), and the Generalitat of Catalunya grant 2009-SGR-1434.
⋆⋆ This work was supported by the EPSRC-Funded ORCHID Project EP/I011587/1.

In this context, the most prominent solution techniques for coalition structure generation are presented in [14, 15], and while they both represent significant contributions to the state of the art, there are some drawbacks that hinder their applicability. On the one hand, [14] requires that the valuation function fulfils the independence of disconnected members (IDM) property,[1] which may not hold in many real world applications, considering that the addition of every new agent to a coalition typically has a cost that depends on the specific application domain [13]. On the other hand, the memory requirements of [15] grow exponentially in the number of agents, hence limiting the scalability of such technique.

Against this background, in a recent paper [3] we have proposed the first anytime algorithm for coalition structure generation on synergy graphs, called CFSS (Coalition Formation with Sparse Synergies). Unlike previous approaches, CFSS provides anytime solutions with quality guarantees for large scale systems (i.e., more that 2700 agents). This tremendous speedup is possible because we focus on a general class of characteristic functions, the so called $m + a$ (monotonic-antimonotonic) functions, for which we can efficiently provide tight bounds. Since in [3] we tested our algorithm only considering *scale-free networks*, we now extend our contribution analysing and empirically evaluating CFSS on the *community network* topology. Community networks offer a realistic description of many interesting practical scenarios [9], e.g., social networks, authorships, collective energy purchasing and carpooling, hence their exploration is important for two main reasons: i) to reveal network organisation at a coarse level, which may help to formulate realistic mechanisms for its genesis and evolution; and ii) to uncover relationships between the nodes which are not apparent by inspecting the graph as a whole and which can typically be attributed to the function of the system.

Overall, our work makes the following contributions to the state-of-the-art:

- We provide a new representation for CSG on synergy graphs proving that, by performing a series of edge contraction operations, it is possible to efficiently build a search tree where each node corresponds to a feasible coalition structure, while avoiding redundancy (i.e., a coalition structure will appear only once in such tree).
- We propose a branch and bound strategy that, when applied to $m + a$ functions, can efficiently solve the CSG problem: in particular, we focus on a widely adopted scenario previously proposed in literature, i.e., the *collective energy purchasing* [5].
- We compare CFSS with DyCE, the state-of-the-art algorithm for CSG on synergy graphs, using two particular test topologies as synergy graphs, i.e., *scale-free networks* and *community networks*. Our results show that CFSS outperforms the counterpart approach in these scenarios and, for very sparse graphs, our algorithm can solve to optimality problems with 60 agents, which is far beyond the limit of DyCE.
- We evaluate the performance and optimality guarantees that our algorithm yields when scaling to very large systems (i.e., more than 2700 agents). Our results show that CFSS provides a tight approximation ratio,[2] which is always lower than 1.12, thus producing solutions that are, in the worst case, at least 88% of the optimal one.

[1] 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.

[2] The ratio between the upper bound on the optimal solution and the value of the solution returned by our approach.

The rest of the paper is organised as follows: Section 2 illustrates the background on CSG on synergy graphs, on community networks and on the DyCE algorithm. Section 3 gives a method to generate our search space while Section 4 details our branch and bound approach. Section 5 discusses our empirical evaluation and Section 6 concludes the paper.

2 Problem and background

The purpose of this section is manifold. First, in Section 2.1 we define the CSG problem on synergy graphs based on the definition introduced in [15]. Second, in Section 2.2 we detail the concept of community network, a synergy graph topology adopted in our experimental evaluation. Finally, in Section 2.3 we provide the necessary background on the DyCE algorithm.

2.1 Problem definition

Consider a set of agents $\mathcal{A} = \{a_1, \ldots, a_N\}$, where N is the total number of agents. Consider also a connected[3] graph $G = (\mathcal{A}, E)$, where $E \subseteq \mathcal{A} \times \mathcal{A}$ is a set of edges between agents, representing the relationships between them. We say that a coalition of agents $C \subseteq \mathcal{A}$ is *feasible* if and only if there exists a connected subgraph $G' = (C, E')$ whose vertices are the agents in C, and whose edges are a subset of those in G, namely $E' \subseteq E$. Thus, the set of feasible coalitions is *restricted* by the relationships represented by the graph G. Henceforth, we refer to the set of all feasible coalitions in G as $\mathcal{FC}(G)$. To value a feasible coalition $C \in \mathcal{FC}(G)$, we assume that there is a function $v : \mathcal{FC}(G) \rightarrow \Re$. Although function v may be arbitrarily defined, we will assume that the value of a coalition is independent of any other coalitions that may exist. For example, in the *collective energy purchasing* scenario, the value of a coalition is the total cost to the agents if they buy energy together, and this cost does not depend on which coalitions other agents will form. Given the coalitional values, the CSG problem on synergy graphs involves finding the optimal *coalition structure* (i.e., a partition of the set of agents) represented by the graph. Hence, the optimal solution is represented by the best set of disjoint feasible coalitions that collectively cover all agents. To find the optimal coalition structure, we must consider the set of all feasible coalition structures, $\mathcal{FCS}(G)$, namely the set of coalition structures that only contain feasible coalitions. Then, solving the CSG problem on a synergy graph G amounts to finding the coalition structure CS^*:

$$CS^* = \underset{CS \in \mathcal{FCS}(G)}{\arg\max} \underbrace{\sum_{C \in CS} v(C)}_{f(CS)} \tag{1}$$

where the function $f : \mathcal{FCS}(G) \rightarrow \Re$ is obtained by adding the coalitional values of the single coalitions. Given this objective, we next detail the concept of *community network*, a synergy graph topology adopted in our empirical evaluation (see Section 5).

[3] If the graph is not connected we decompose the problem into smaller independent subproblems, each with a connected graph.

2.2 Community networks

Many realistic scenarios in which CSG can be successfully applied exhibit an underlying structure determined by sparse synergies which naturally results from interactions between agents. These fields of application include social networks, authorships, collective energy purchasing and carpooling. One fundamental feature of these networks of agents is represented by their mesoscopic structure, characterised by the presence of groups of nodes, called *communities* or *modules* (as shown in Figure 8(a)), with a high density of links between nodes of the same group and a comparatively low density of links between nodes of different groups. This compartmental organisation of networks is very common in systems of diverse origin, hence analysing the performance of our approach with the adoption of this particular synergy graph topology is very important.

In particular, in our experiments *community networks* are provided by the Block Two-Level Erdős-Rényi (BTER) model[4] [7]. Such model is based on the idea of a graph comprising communities in the form of dense ER subgraphs (i.e., generated with the Erdős-Rényi model), hence matching well real-world graphs. The generation of community networks with the BTER model is divided in two phases: in the first one, BTER builds a collection of ER subgraphs in such a way that the specified degree distribution is respected. The BTER model allows one to construct a graph with any degree distribution. Real-world degree distributions might be idealised as power laws, but it is by no means a completely accurate description. When the degree distribution is heavy tailed, then the BTER graph naturally has scale-free ER subgraphs. The internal connectivity of the ER graphs is specified by the user and can be tuned to match observed data. These communities are then interconnected in the second phase, in which a Chung-Lu model [1] is used over the excess degrees to form the edges that connect communities.

Community networks, together with scale-free networks, will be considered in our benchmark evaluation against the DyCE algorithm, the current best solution approach, which will be described in the next section.

2.3 The DyCE algorithm

DyCE [15] uses dynamic programming to find the optimal coalition structure by progressively splitting the current solution into its best partition. This modus operandi is similar to IDP [11], with the fundamental addition of an initial preprocessing phase that enumerates all feasible coalitions. Then, DyCE considers only such coalitions significantly reducing the run time of the overall approach. However, DyCE requires an exponential amount of memory in the number of agents (i.e., $O\left(2^n\right)$) and is not an anytime approach. Hence, the scalability of such approach is limited to systems consisting of tens of agents (e.g., around 30). Moreover, DyCE is a serial algorithm and cannot be efficiently parallelised due to the exponential memory requirements in the number of agents. To overcome these limitations, we propose a new way of representing the CSG problem, and in particular its search space, by means of edge contractions, as detailed in the next section.

[4] The FEASTPACK v1.1 MATLAB implementation [8] of the BTER model has been used.

(a) Before contraction (b) After contraction

Fig. 1: Example of an edge contraction in a triangle graph (the dashed edge is contracted to give the graph on the right)

3 Representing the search space

In this section we show that all feasible coalition structures induced by G can be easily modelled as a search tree in which each feasible coalition structure is represented only once. Specifically, we first detail how we can use edge contractions to represent the CSG problem and then we provide an algorithm to build the search space.

3.1 Representing CSG via edge contractions

The main idea to generate our search space is that each feasible coalition structure can be represented 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. In more detail, let us define an *edge contraction* as follows:

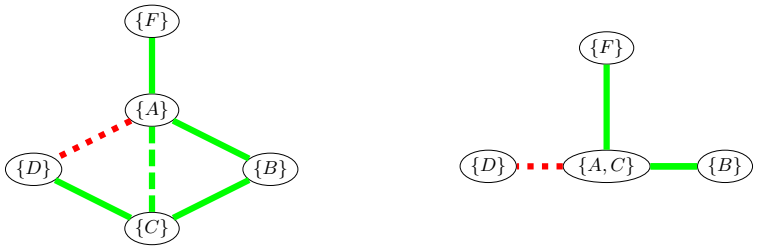
Definition 1 *Given a graph $G = (V, E)$ and an edge $e = (u, v)$, where $e \in E$ and $u, v \in V$, the result of the contraction of edge $e = (u, v)$ is a graph G' obtained by removing edge e and adding a new vertex w obtained by merging u and v (i.e., labelling w with the union of their labels). Moreover, each edge incident to either u or v in G will become incident on w in G', merging the parallel edges*[5] *that may result from the aforementioned operations.*

Intuitively, when we contract an edge we merge the coalitions associated to its 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 and assert the following propositions (proofs can be found in [3]):

Proposition 1. *The graph resulting from the contraction of any set of edges represents a feasible coalition structure, where coalitions correspond to the labels of the vertices of the resulting graph.*

Proposition 2. *Any feasible coalition structure CS can be generated by contracting a set of edges of G.*

[5] Two edges are parallel if they are incident on the same two vertices.



(a) Before contraction

(b) After contraction

Fig. 2: Example of a green edge contraction in a 2-coloured graph (the dashed edge is contracted to give the graph on the right)

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 1(a), contracting any two edges leads to 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 so 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, such edge 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 operations of the search space generation. To represent such marking, we will use the notion of *2-coloured graph*:

Definition 2 *A 2-coloured graph $G^c = (V, E, c)$ is composed of a set of vertices V and a set of edges E, as well as a function $c : E \to \{red, green\}$ 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 other hand, green edges can be still contracted. Figure 2(a) shows an example of a 2-colour graph in which edge $(\{A\}, \{D\})$ is coloured in red: 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* as follows:

Definition 3 *Given a 2-coloured graph $G = (V, E, c)$ and a green edge $e = (u, v)$, where $e \in E$ and $u, v \in V$, the result of the contraction of edge $e = (u, v)$ is a graph G' obtained by performing the contraction of the edge e in the graph 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 nodes of that edge 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 $(\{D\},\{A\})$ (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 represent coalition structures, we now provide a way to model the search space of the CSG problem on synergy graphs.

3.2 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 shows the procedure that builds our search tree, and it is possible to show that such algorithm visits all feasible coalition structures and each of them is visited only once. Nonetheless, it is possible that, even for sparse graphs, the number of feasible coalition structures is very large, making their visit not affordable. Hence, in the next section we propose a branch and bound technique that prunes significant parts of the search space.

Algorithm 1 VISITALLCOALITIONSTRUCTURES(G^c)

```
1: VISIT(G^c)
2: for all G ∈ CHILDREN(G^c) do
3:     VISITALLCOALITIONSTRUCTURES(G)
4:
5: function CHILDREN(G^c)
6:     G' ← G^c                                  ▷ Initialise graph G' with G^c
7:     Ch ← ∅                                    ▷ Initialise the set of children
8:     for all e ∈ G^c : c(e) = green do         ▷ For all green edges
9:         Ch ← Ch ∪ {GREENEDGECONTRACTION(G', e)}
10:        Mark edge e with colour red in G'
11:    return Ch                                 ▷ Return the set of children
```

4 The CFSS algorithm

In this section we describe CFSS, our branch and bound approach to CSG on synergy graphs. In particular, we will focus on a general class of characteristic functions (called $m+a$ functions). Next, we present a specific function, previously used in the literature (that can be proven to be $m+a$), detailing how we can provide bounds for such function so to drive the branch and bound strategy.

4.1 A general branch and bound algorithm for $m + a$ functions

As shown in Equation 1, our objective is to maximise a function whose domain is the set of feasible coalition structures $\mathcal{FCS}(G)$. On the other hand, a strategy to prune significant portions of the search space is needed to have a computationally affordable solution technique. In what follows, we detail some properties of our reference domain that our branch and bound approach exploits:

Property 1 *$\mathcal{FCS}(G)$ is a lattice, i.e., a partially ordered set, in which every two elements CS_i and CS_j have a supremum ($CS_i \vee CS_j$) and an infimum ($CS_i \wedge CS_j$).*

In particular, we define the following partial order over partitions:

Definition 4 *Given any two partitions CS_i and CS_j, we say that $CS_i \leq CS_j$ if every element of CS_i is a subset of some element of CS_j.*

As an example, $\{A, B\}\{C\} \leq \{A, B, C\}$, but the order between $\{A, B\}\{C\}$ and $\{A\}\{B, C\}$ is not defined. It is well known that with this partial order the set of partitions forms a complete lattice (see Section V.4 in [6]), called the partition lattice or equivalence lattice. It is easy to see that our domain of interest is the sublattice generated by the set of feasible coalition structures and thus it is a lattice. Furthermore, in our scenario, the grand coalition represents a supremum of any two elements, while the coalition structure of all singletons represents an infimum.

Property 2 *The elements of $\mathcal{FCS}(G)$ can be arranged in an order-preserving tree: whenever CS_j is a descendant of CS_i in the tree, then $CS_j \geq CS_i$. Thus, CS_i is the infimum of the subtree rooted at CS_i:*

$$CS_i = \bigwedge ST(CS_i) = \inf ST(CS_i)$$

In the search tree defined in Section 3 each child is the result of contracting an edge in the parent. As a consequence of the contraction, two of the coalitions in the parent are merged, making the child coalition coarser than that of the parent. Hence, by direct application of Property 1, the above statement holds.

Property 3 *Given a node CS_i, there is a computationally efficient procedure to assess an element $\overline{CS_i}$ that is bigger than any of the elements of the subtree:*

$$\overline{CS_i} \geq \bigvee ST(CS_i) = \sup ST(CS_i)$$

It is easy to see that $\overline{CS_i}$ can be found by removing all red edges from the 2-coloured graph representing CS_i and then contracting all the remaining green edges (which is equivalent to find the connected components in the graph after the removal of all red edges). What we are doing can be interpreted as finding the coarsest partition forgetting that we decided not to contract some edges. Clearly, any partition in the subtree will be at most as coarse as this one. This procedure can be implemented in time $O(N + E)$, visiting all the vertices that can be reached from any starting node by means of a breadth-first search, and iterating this procedure starting from any node that could not be reached, until all the vertices have been visited.

Property 4 *The function f is an $m + a$ function ($f = f^+ + f^-$), i.e., it is the sum of a monotonic function f^+ and an anti-monotonic function f^-.*

Functions that satisfy this property are interesting because they allow us to efficiently provide a tight bound that can drive the branch and bound strategy.

Proposition 3. *If a domain satisfies the aforementioned properties, an efficient technique can be used to compute a tight bounding function. Given a node CS_i:*

$$M\left(CS_i\right) = f^-\left(CS_i\right) + f^+\left(\overline{CS_i}\right)$$

is an admissible bound for the subtree rooted at CS_i:

$$M\left(CS_i\right) \geq \max\{f(CS_i) | CS_j \in ST(CS_i)\}$$

Building on Property 3, we can efficiently assess a bound for each subtree and prune it if the value of such bound is smaller than the value of the best solution found so far (as shown in Algorithm 2):

Algorithm 2 CFSS(G^c)

```
1:  best ← G^c                                  ▷ Initialise current best solution with singletons
2:  F ← ∅                                       ▷ Initialise frontier F with an empty stack
3:  F.PUSH(G^c)                                 ▷ Push G^c as the first node to visit
4:  while F ≠ ∅ do                              ▷ Branch and bound loop
5:      node ← F.POP()                          ▷ Get current node
6:      if M(node) > f(best) then               ▷ Check bound value
7:          if f(node) > f(best) then           ▷ Check function value
8:              best ← node                     ▷ Update current best solution
9:          F.PUSH(CHILDREN(node))              ▷ Update frontier F
10: return best                                 ▷ Return optimal solution
```

4.2 Anytime properties

Notice that such branch and bound procedure can be directly applied to compute an overall bound of an $m + a$ function, with anytime properties. More precisely, let us consider the frontier F reported in Algorithm 2. When we expand the frontier F (Line 9) we can keep track of the highest value of the function f evaluated in all visited nodes. Hence, given a frontier F on the search tree, the bound $B(F)$ is defined as:

$$B(F) = \max\left\{f\left(best\right), \max_{CS \in F} M\left(CS\right)\right\} \tag{2}$$

In other words, $B(F)$ is the maximum between the values assumed by f inside the frontier (i.e., $f\left(best\right)$, see line 7 of Algorithm 2) and an estimated bound outside of it. Intuitively, the more search space is explored, the better the bound provided. The fastest way to compute a bound for f is considering a frontier formed exclusively by the root: assessing this bound has the same time complexity of computing the function M, and its quality can be satisfactory, as shown in Section 5.4.

Even though Algorithm 2 could be applied to any characteristic function, the result in Section 4.1 can be used to prune significant portions of the search space as long as the characteristic function is the sum of a monotonic and an anti-monotonic function. Next, we provide an example of a function that fulfils such property.

4.3 Collective energy purchasing function

In what follows, we focus on a particular benchmark function for the CSG problem on synergy graphs, namely the *collective energy purchasing* function. This function is typical of realistic coalition formation problems as shown in [5], and, as proven in [3], it can be characterised as an $m + a$ function, i.e., as the sum of a monotonic and an antimonotonic part, thus enabling the aforementioned bounding techniques to prune part of the search space during the execution of the CFSS algorithm.

In the *collective energy purchasing* scenario, each agent is characterised by an energy consumption profile that represents its energy consumption throughout a day [5]. In more detail, a profile records the energy consumption of a household at fixed intervals (every half hour in our case). Hence each profile is a vector of T elements (where $T = 48$ in our case), whose values represent the actual measurements collected over a month from 2732 households in UK. The characteristic function of a coalition of agents is the total cost that the group would incur if they buy energy as a collective on two different markets: the spot market, a short term market intended for small amounts of energy, and the forward market, a long term one in which bigger portions of energy can be bought at cheaper prices. In particular, following [5] the characteristic function is defined as:

$$v\left(C\right) = \sum\nolimits_{t=1}^{T} q_S^t\left(C\right) \cdot p_S + T \cdot q_F\left(C\right) \cdot p_F + \kappa\left(C\right) \tag{3}$$

where p_S and p_F represent the unit price of energy in the spot and forward market respectively,[6] $q_F\left(C\right)$ stands for the time unit amount of electricity to buy in the forward market and $q_S^t\left(C\right)$ for the amount to buy in the spot market at time slot t. These quantities are the ones that optimise the buying strategy of the group while satisfying the group electricity demand (see [5] for further details). Finally, $\kappa\left(C\right)$ 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 power network capacity of 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 [5], we use $\kappa\left(C\right) = -|C|^{\gamma}$ to introduce a non-linear element that penalises the formation of big coalitions, so that the grand coalition is not always the best coalition structure. Thus, the *collective energy purchasing* function is defined as:

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

[6] Following [5], in our experiments we fixed $p_S = -80$ and $p_F = -70$, using negative unit prices to reflect the direction of payments.

5 Empirical evaluation

Having described and analysed our branch and bound approach for the CSG problem on synergy graphs, we now present the empirical evaluation. In what follows, we first discuss the methodology we use for comparison and then present the results obtained in the domain described in Section 4.3.

5.1 Evaluation methodology

The main goal of our empirical analysis is twofold. First, we aim to show the performance of our approach using a standard experimental setting, where, following [15], we consider scale-free networks generated with the Barabási-Albert model [2] with the parameter $m \in \{1, 2, 3\}$. Second, we extend our study with an additional test case, in which community networks generated with the BTER model[7] [7, 8] are taken into account. Our aim is to investigate how the synergy graph topology impacts on the algorithm performance. As a reference, we compare our approach with DyCE, measuring the runtime in seconds. In all our experiments, we use the *collective energy purchasing* characteristic function, increasing the number of agents until the execution time of the algorithm does not exceed 10^5 seconds (which represents a reasonable time limit for a single experiment). The results are obtained as the average of 50 instances in which scale-free networks and community networks are generated for every configuration of the parameters determining the number of nodes and the connectivity (i.e., m and average degree). Finally, our approach is implemented in C and executed on a machine with a 3.40GHz processor and 16 GB of memory. For DyCE we used the implementation provided by the authors of [15].

5.2 Scale-free networks

CFSS outperforms DyCE in our experiments with sparse scale-free networks (Figure 3). Specifically, with $m = 2$, CFSS is 4.7 times faster than DyCE for 30 agents, and with $m = 1$ it is at least 2 orders of magnitude faster. However, DyCE is significantly faster than CFSS with $m = 3$ (44 times). Nonetheless, it is important to note 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 provide approximate solutions for instances with thousands of agents (as shown in Section 5.4).

5.3 Community networks

When we move to the community network topology, CFSS still achieves better performances than DyCE on sparse graphs, which is faster when the connectivity of the synergy graph increases, following the behaviour of the previous test case. In particular, with an average degree of 2 (i.e., with a number of edges equivalent to a scale-free network with $m \simeq 1$), CFSS is at least 2 orders of magnitude faster (Figure 4). On the other hand, DyCE is 3.2 times faster on networks with an average degree of 6.

[7] Networks are generated with an average degree $\in \{2, 4, 6\}$.

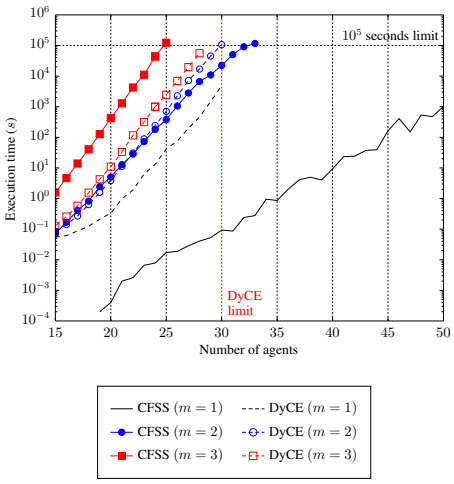


Fig. 3: Scale-free networks

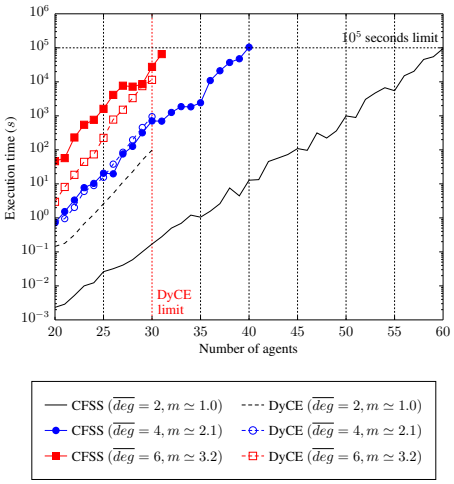


Fig. 4: Community networks

Our results show that both algorithms are significantly faster on community networks, with an improvement of at least one order of magnitude w.r.t. the previous scenario. This provides a clear indication that the graph topology has a strong impact on the runtime needed to find the optimal solution. As an example, with 25 agents and an average degree of 6, the runtime of CFSS is two orders of magnitude lower than a comparable experiment with scale-free networks (i.e., generated with $m \simeq 3.2$, resulting in a equivalent number of edges). This speedup is a direct consequence of the reduced size of the search space $\mathcal{FCS}(G)$ where G is a community network. Figure 5 shows the number of feasible coalition structures of several randomly generated networks using $n \in \{20, 21, 22, 23, 24, 25\}$ (more specifically, for each configuration we plot the average over 50 repetitions). Scale-free networks are generated with $m = 2$, while corresponding community networks are obtained with an average degree of 3.76, resulting in an equivalent number of edges.

It is easy to see that, considering a community network topology, the dimension of the search space is at least one order of magnitude smaller than the scale-free network counterpart, justifying the above mentioned improved performance. This experimental result can be further motivated by comparing the number of edges removed by an edge contraction in the two considered synergy graph topologies. Figures 7 and 8 respectively show a *scale-free network* and *community network*, both with 13 nodes and 19 edges: in the first case, the contraction of the edge $(\{A\}, \{C\})$ produces only one couple of parallel edges, which will be collapsed in the resulting graph, whereas, in the second example, contracting an edge in the highlighted *community* generates 5 couple of redundant edges, then removed in the final graph. Therefore, the presence of these highly connected subgraphs featured by *community networks* results in a greater number of deleted edges at each algorithmic step, hence producing a smaller search space.

As a final remark, it is important to note that the scalability of CFSS indirectly benefits from this performance improvement: in fact, with the same time limit for a single experiment (10^5 seconds), our approach is able to solve bigger instances w.r.t. scale-free networks, whereas DyCE is still limited by its exponential memory requirements.

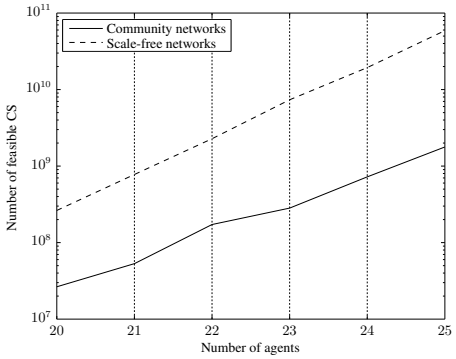


Fig. 5: Number of feasible coalition structures

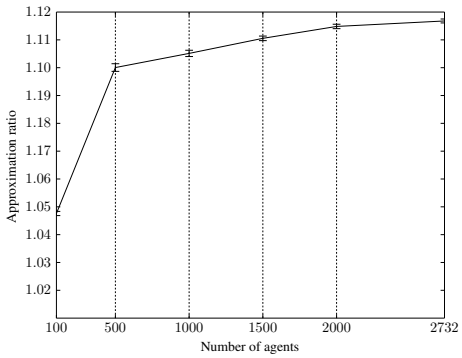
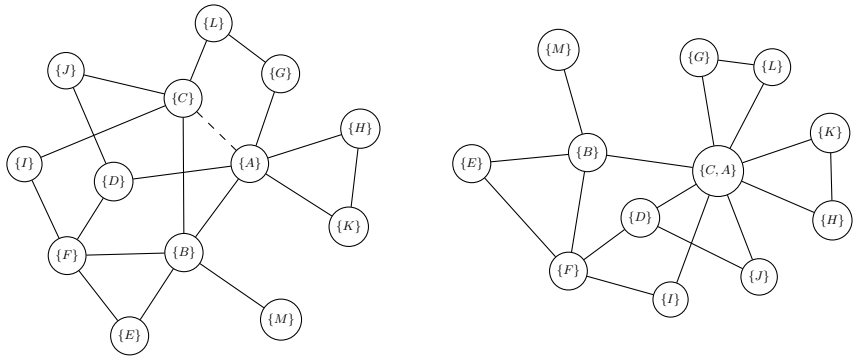


Fig. 6: Approximation ratio



(a) Before contraction

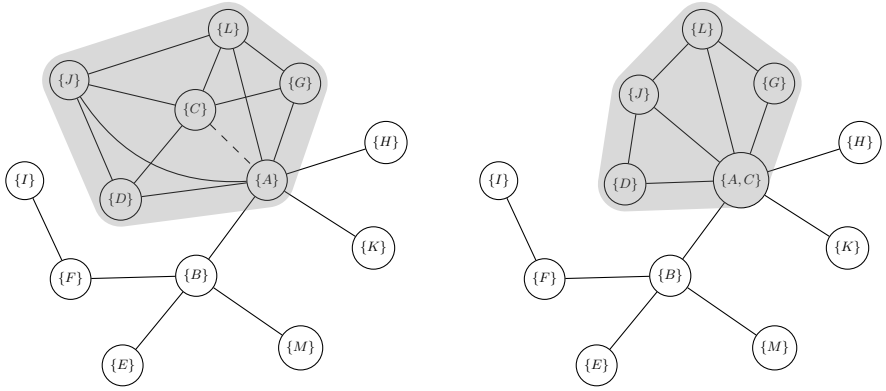
(b) After contraction

Fig. 7: Example of an edge contraction in a *scale-free network*

5.4 Anytime performance

Figure 6 shows the value of the approximation ratio (i.e., the ratio between the provided bound and the solution returned by CFSS) obtained in 6 particular configurations for the *collective energy purchasing* scenario (using *scale-free networks* with $n \in \{100, 500, 1000, 1500, 2000, 2732\}$ and $m = 4$) and considering a time limit of 100 seconds.[8] For each configuration, we plot the average and the standard error of the mean over 20 repetitions. 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. In the worst case, CFSS provides an approximation ratio of 1.12 and thus solutions that are at least 88% of the optimal. This confirms the effectiveness of this bounding technique applied to the energy domain, which allows us to provide solutions and quality guarantees for problems involving a very high number of agents.

[8] Other values for m show a similar behaviour (not reported here).



(a) Before contraction (b) After contraction

Fig. 8: Example of an edge contraction in a *community network*

The above mentioned experiments were also repeated considering similar *community network* topologies, producing equivalent results and demonstrating the insignificant impact of the synergy graph topology on the performance of this bounding technique. In our experiments, the bound is assessed at the root, without any frontier expansion. Thus, the bound can be computed almost instantly, 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, as proven in [3].

6 Conclusions

In this paper we considered the coalition structure generation problem on two particular synergy graph topologies, *scale-free networks* and *community networks*. Specifically, we extend the analysis of the CFSS algorithm that we recently proposed in [3], evaluating its performances on community networks. Our empirical evaluation shows that CFSS outperforms DyCE, the state-of-the-art algorithm, solving bigger instances with better or comparable runtimes. Moreover, the scalability of our approach improves considering community networks (while the counterpart is still limited by its exponential memory requirements), as a result of the reduced search space produced by this particular topology. Finally, our algorithm provides approximate solutions with good quality guarantees (i.e., with an approximation ratio of 1.12 in the worst case) for systems of unprecedented scale (i.e., more than 2700 agents).

Future work will look at improving the study on community networks, investigating how different network features (e.g., clustering coefficient) impact the performance of our approach. Moreover we aim at extending the algorithm to work on other hard optimisation problems, such as the winner determination problem for combinatorial exchanges, where $m + a$ functions also apply.

References

1. Aiello, W., Chung, F., Lu, L.: A random graph model for massive graphs. In: STOC. pp. 171–180 (2000)
2. Albert, R., Barabási, A.L.: Statistical mechanics of complex networks. Rev. Mod. Phys. 74(1), 47–97 (2002)
3. Bistaffa, F., Farinelli, A., Cerquides, J., Rodríguez-Aguilar, J.A., Ramchurn, S.D.: Anytime coalition structure generation on synergy graphs. Accepted at: AAMAS (2014), http://db.tt/Z9MUgzic
4. Demange, G.: On group stability in hierarchies and networks. Journal of Political Economy 112(4), pp. 754–778 (2004), http://www.jstor.org/stable/10.1086/421171
5. Farinelli, A., Bicego, M., Ramchurn, S., Zucchelli, M.: C-link: A hierarchical clustering approach to large-scale near-optimal coalition formation. In: IJCAI. pp. 106–112 (2013)
6. Grätzer, G.: Lattice Theory: Foundation. Springer (2011)
7. Kolda, T.G., Pinar, A., Plantenga, T., Seshadhri, C.: A scalable generative graph model with community structure. CoRR abs/1302.6636 (2013)
8. Kolda, T.G., Pinar, A., et al.: Feastpack (2014), http://www.sandia.gov/ tgkolda/feastpack/
9. Lancichinetti, A., Kivelä, M., Saramäki, J., Fortunato, S.: Characterizing the community structure of complex networks. CoRR abs/1005.4376 (2010)
10. Myerson, R.B.: Graphs and cooperation in games. Mathematics of Operations Research 2(3), pp. 225–229 (1977), http://www.jstor.org/stable/3689511
11. Rahwan, T., Jennings, N.R.: An improved dynamic programming algorithm for coalition structure generation. In: AAMAS. pp. 1417–1420 (2008)
12. Sandholm, T., Larson, K., Andersson, M., Shehory, O., Tohmé, F.: Coalition structure generation with worst case guarantees. AIJ 111(1), 209–238 (1999)
13. Shehory, O., Kraus, S.: Methods for task allocation via agent coalition formation. AIJ 101(1-2), 165–200 (1998)
14. Voice, T., Polukarov, M., Jennings, N.: Coalition structure generation over graphs. JAIR 45, 165–196 (2012)
15. Voice, T., Ramchurn, S., Jennings, N.: On coalition formation with sparse synergies. In: AAMAS. pp. 223–230 (2012)