A Message-Passing Approach to Computing Stable Coalitions on Graphs

Paper 814

Abstract

Many real-life settings such as communication networks, or the electricity grid require the development of decentralized mechanisms that allow agents to organize into stable coalitions whose potential membership is restricted by a graph. To meet these challenges, in this paper we develop a novel graphical model representation of coalitional games over arbitrary graphs, the delegation-based model, and exploit this representation to devise the first message-passing algorithm for the formation of core-stable coalition structures. Our algorithm allows agents to simultaneously identify an optimal coalition structure along with a payoff allocation such that the resulting pair is in the core of the game or, alternatively, detect the emptiness of the core.

1 Introduction

Many real-life multiagent settings, such as communication or sensor networks, or, more recently, the *smart electricity grid*, can benefit from the development of *decentralised* algorithms that allow agents to organize into stable coalitions. Moreover, it is natural in such settings to assume that coalition membership can be restricted by some kind of graph, reflecting realistic barriers to the formation of certain teams.

As an illustrating example, consider the need to maintain reliability of supply in the modern electricity network while also minimizing carbon emissions and costs. One key requirement in this domain is that demand should follow supply in order to make optimal use of intermittent sources of renewable energy, while still ensuring that electricity generation and electricity consumption are perfectly matched (Gri 2003). Given this, a promising demand-side management strategy is to promote the formation of coalitions among energy consumers with near-complementary consumption restrictions, by promising them rewards for collectively "flattening" their joint energy consumption curve by time-shifting their individual consumption. In this domain, however, geographical barriers and technological limitations, alongside the need to respond to supply and demand imbalances occurring at a local level, imply that consumers cannot just join any coalition, but rather, that coalition membership is restricted by a graph. In more detail, a coalition among agents is feasible if its members form a vertex-connected induced subgraph. Now, from the Grid's perspective, the emerging coalitions should achieve the highest possible savings in terms of energy consumption. At the same time, it is desirable that coalitions should be as stable as possible, since, it is more efficient for the Grid to interact with specific entities, as this leads to operational savings and the ability to strike long-term binding contracts. Furthermore, the balance of supply and demand will, naturally, be affected if consumers keep moving from coalition to coalition, altering their consumption savings plans. However, consumers are rational utility maximizers, and have to be incentivised to stay in a coalition via appropriate monetary payoffs. Thus, it is necessary in domains like this to devise decentralised processes to allow autonomous rational agents to join together into stable coalitions, that are also optimal from the system designer's point of view. Such problems can be modelled naturally as *coalitional games*, where: *(i)* the set of coalitions with maximum collective value, that is, an optimal coalition structure, has to be identified; and *(ii)* each coalition's value has to be distributed among its members in such way that coalition members have no incentive to break away from the identified optimal structure (Osborne and Rubinstein 1994). When this happens, we say that a game outcome is in *the core*.

Against this background, in this paper we develop a novel graphical model representation for games on graphs, the delegation-based model, and exploit this to devise a decentralised algorithm, *SCF-Graphs*, that allow agents to find a core-stable coalition structure or, alternatively, to detect the emptiness of the core. By so doing, we are proposing the first *decentralised* algorithm performing on *arbitrary* graphs that can deal simultaneously with two key activities usually treated separately in the coalition formation literature: identifying an optimal coalition structure *and* an accompanying payoff allocation so that core-stable elements emerge.

This paper is structured as follows. In Section 2, we review the literature and in Section 3, we describe our novel representation. The stable coalition formation algorithm is presented in Section 4. Finally, we conclude in Section 5.

2 Background

A coalitional ("transferable utility", or "characteristic function") game is traditionally defined as follows. Let $A =$

$\{a_1, \ldots, a_n\}$ be a set of agents. A subset $S \subseteq A$ is termed a coalition. Then, a coalitional game CG is completely defined by its *characteristic function* $v : 2^A \rightarrow \Re$ (with $v(\emptyset) = 0$), which assigns a real value representing (transferable) utility to every feasible coalition (Osborne and Rubinstein 1994). Agents in a coalition are then permitted to freely distribute coalitional utility among themselves. Given a game CG, a *coalition structure* $CS = \{S_1, \ldots, S_k\}$ is an exhaustive disjoint partition of the space of agents into coalitions. We overload notation by denoting by $v(CS)$ the (intuitive) worth of a coalition structure: $v(CS) = \sum_{S \in CS} v(S)$. We also denote the set of all coalition structures by $\mathbf{CS}$.

Assume now that feasible coalitions are determined by a *graph* G: *(i)* each node represents an agent; and *(ii)* a coalition S is allowed to form iff every two agents in S are connected by some path in the subgraph induced by S. We denote the set of agent nodes in G by $A(G)$ and, given a set of agents $A \subseteq A(G)$, by G_A the subgraph of G induced by A.

Definition 1. *A coalitional game CG on a graph G is a tuple $\langle A(G), v, F(G) \rangle$ where: (i) $F(G)$ is the set of all* feasible *coalitions—i.e., coalitions permitted to form given G; and (ii) v is the characteristic function defined over $F(G)$.*

An example of a game on a graph with a cycle in which four agents is given in Figure 1(a). Thus, in Figure 1(a) a_3 can form a coalition with agents a_0 and a_1 ($S = \{013\}$) but not a coalition with a_1 without a_0 ($S = \{13\} \notin F(G)$).

A well-studied problem, due to its apparent significance, is the *coalition structure generation (CSG) problem*, aiming to identify the coalition structure CS^* with maximal value. The CSG problem is NP-hard (Sandholm *et al.* 1999).

Example 1. *Consider the graph in Figure 1(a) with the following characteristic function $v(A(G)) = 3$, $v(\{i, j, k\}) = 2.5$ for any feasible coalition $\{i, j, k\}$ of size 3, $v(\{i, j\}) = 1$ for any feasible coalition of size 2 and $v(\{i\}) = 0.1$ for any single coalition. Then the optimal coalition structure is composed of the grand coalition that includes all the four agents, $CS^* = \{\{0, 1, 2, 3\}\}$, with a value of 3.*

Now, agents are selfish and thus need to decide how the value of their coalition should be distributed. A vector $\rho = \{\rho_1, \ldots, \rho_n\}$ assigning some payoff to each agent $a_i \in A$ is called an *allocation*. We denote $\sum_{i \in S} \rho_i$ by $\rho(S)$. An allocation ρ is an *imputation* for a given CS, if it is efficient ($\rho(S) = v(S)$ for all $S \in CS$), and individually rational (that is, $\rho_i \geq v(\{i\})$ for all a_i). Note that if ρ is an imputation for CS, then $\rho(A) = v(CS)$.

A game outcome is a (CS, ρ) pair, assigning agents to coalitions and allocating payoffs to agents efficiently. However, this does not mean that a game outcome is necessarily stable; there might be agents that feel there are outcomes that can make them better-off. This raises the question of identifying stable outcomes. The *core* is arguably the main stability solution concept in cooperative games. It is the set of coalition structure-imputation tuples (CS, ρ) such that no feasible coalition has a deviation incentive. Formally:

$$Core(CG) = \{(CS, \rho) : \rho(A(G)) = v(CS) \,\&\, \rho(S) \geq v(S)\ \forall S \in F(G)\}$$

Notice that *only optimal coalition structures might admit an element in the core*. Intuitively, if the current structure is suboptimal then a subset of agents can be made strictly better off by moving to an optimal coalition structure. Note also that the core is a strong solution concept, as it is empty in a plethora of games; therefore, the question of the core non-emptiness is key in many settings.

2.1 Related Work and Discussion

There is an abundance of papers dealing with various aspects of the coalition formation problem—some focusing on CSG, others on the problem of allocating payoff to the agents in some fair or stable manner. For instance, a number of algorithms have been proposed to solve the CSG problem (Michalak *et al.* 2010; Dang and Jennings 2004; Rahwan *et al.* 2009). However, these methods ignore the fact that individual agents—say electricity consumers—have their own preferences, and thus need to be provided with incentives. On the other hand, the provision of such incentives via payoff allocation, has long been studied in cooperative games (Greco *et al.* 2011; Voice *et al.* 2011; Conitzer and Sandholm 2003; Deng and Papadimitriou 1994). However, relevant research to date mostly focuses on characterising outcomes and its complexity, rather than providing algorithms that the agents can use in order to actually form stable coalitions. Moreover, in many occasions the optimality of the grand coalition[1] is assumed. Hence, the payoff allocation problem is kept (artificially) isolated from the CSG one. In our work here, we tackle both problems simultaneously. Furthermore, while most previous work has viewed coalition formation as a centralized problem, not satisfying the decentralisation requirement present in many domains (Dang and Jennings 2004; Rahwan *et al.* 2009), we provide a *decentralized* algorithm to identify core-pairs (coalition structure, payoff allocation).

Regarding graph-restricted games,, there has been some work on *cooperative solution concepts* in graph-restricted games (van den Brink 2006; Meir *et al.* 2011) but none of these works focus on how core-stable structures emerge. One exception is the work of (Demange 2004) that proved that when the graph restricting a game is a *tree* there always exists an element in the core; and, moreover, presented a process that outputs a core element for tree games. Here we extend that work, by providing an algorithm that can compute core-stable coalition structures on *general* graphs.

3 Problem Representation

In this section, we describe our approach to represent coalitional games on graphs. First, we introduce a hierarchical arrangement of the graph, the so-called *delegation-based graph*, that enables coalition formation through delegation. Next, we show how to map any delegation-based graph into a graphical model representation.

3.1 Delegation-based graphs

The basic idea behind our representation is the use of a hierarchical arrangement that enables delegation on the formation of coalitions. In particular, our representation is based on arranging agents in a graph G into a pseudotree structure (PT). Following (Dechter 2003),

[1] The grand coalition is the coalition of all agents.

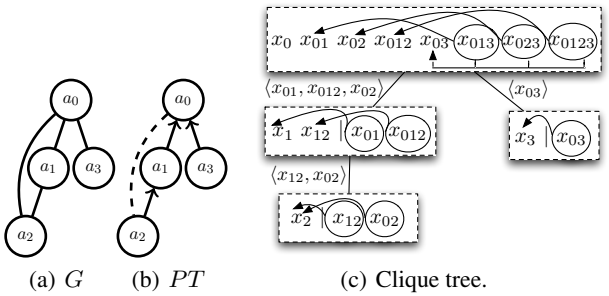
Definition 2. *A pseudotree PT of G is a rooted tree with agents $A(G)$ as nodes and the property that any two agents that share an edge in G are on the same branch in PT.*

Figure 1(b) shows a pseudotree, rooted at agent a_0, of the cyclic graph G in Figure 1(a). A PT has two kinds of edges: tree-edges (bold lines) that link parent with children (e.g. a_2 is child of a_1); and pseudoedges (dashed lines) that link pseudoparents with pseudochildren (e.g. a_2 is pseudochild of a_0). Let's denote $A(PT)$ the set of agent' nodes in PT and PT_i the subtree of PT rooted at a_i. Thus, in Figure 1(b), PT_1 is a tree rooted at a_1 composed of agents a_1, a_2. Finally, given an agent $a_i \in A(PT)$ we will denote as Ch_i its children, An_i its ancestors (the set composed of its parent and its pseudoparents), and D_i its descendants (the set composed of its children and pseudochildren) in PT. Then, in Figure 1(b), $Ch_2 = D_2 = \emptyset$ and $An_2 = \{a_0, a_1\}$. Then, given a game on a graph $CG = \langle A(G), v, F(G)\rangle$ and a pseudotree PT over G, the partial ordering that PT defines among agents allows us to partition the set of feasible coalitions into $|A|$ disjoint sets $\{\mathbf{L}_i | a_i \in A\}$, one per agent. The set of (leading) coalitions $\mathbf{L}_i$ contains all the feasible coalitions in which a_i is the leader in the hierarchy, that is all coalitions that include agent a_i but no agent up a_i in PT, $\mathbf{L}_i = \{S \in F(G_{A(PT_i)}) | \{i\} \in S\})$.

Futhermore, the pseudotree ordering also allows us to formulate a delegation mechanism that guides coalition formation. In more detail, in our representation, an agent a_i can ensure the participation of other agents in one of its leading coalitions $S \in \mathbf{L}_i$ not directly through them but instead delegating to some agent a_j the formation of a subcoalition that includes a_j and some other agents accessible from a_j. Since this subcoalition is required for the formation of coalition S, we call it **required coalition** of S. Thus, in Figure 1(b), agent a_0, for its leading coalition $\{012\}$ will not negotiate the participation of a_2 directly with him but instead a_2 will negotiate with agent a_1 for the required coalition $\{12\}$. Thus, a_0 delegates to a_1 the formation of a coalition that includes agents in PT_1. Next, we describe the main steps of a distributed procedure that allow agents to compute the set of requiring and leading coalitions on a graph. Thus, at the end of this process, each agent will know not only its set of leading coalitions $\mathbf{L}_i$ but also the so-called requiring mapping $\mathbf{Req_i}$, that given a coalition returns its set of required coalitions in $\mathbf{L}_i$. We refer to the set of coalitions that does not form part of $\mathbf{L}_i$ but requires some coalition in $\mathbf{L}_i$ as a_i requiring coalitions ($\mathbf{R}_i$).

Procedure 1. Each agent a_i uses the PT partial ordering to order its descendants $D_i = \{d^1, \ldots, d^m\}$ from higher to lower. For example, in Figure 1(b), a_0 can order its descendants as a_1,a_2,a_3 (as far as a_1 is placed before a_2 the order is valid). Then, a_i will proceed to generate its set of leading coalitions $\mathbf{L}_i$ into two steps: first, generating a set of basic coalitions, and second, generating a set of composed coalitions, that result from combination of basic ones. Finally, each agent computes its set of requiring coalitions $\mathbf{R}_i$.

Step 1. (Basic coalitions). In this step, each agent a_i will generate the set of *basic* coalitions. For each descendant $d^{j=1\ldots m}$, a_i generates all coalitions S such that $\{d^j\} \subseteq S \subseteq A(PT_i) \setminus \{E\}$ where $E = \{d_k | k < j\}$ stands for all descendants placed before d^j in the ordering. For example, in Figure 1(b), agent a_0 will generate the set of coalitions that include: (i) a_1 and other agents reachable from a_1 in PT_1 ($\{1\}$,$\{12\}$); (ii) a_2 and other agents reachable from a_2 in PT_1 excluding a_1 ($\{2\}$); and (iii) a_3 and other agents reachable from a_3 in PT_1 excluding a_1 and a_2 ($\{3\}$). Each agent a_i records for each coalition S a set of frontiers nodes, F, these are nodes that are reachable from S but not included in S. For example, in Figure 1(b) the set of frontiers for $\{1\}$ is $\{2\}$ since a_2 is reachable from a_1 but not included. For each these coalitions S, a_i will send a message $req_{i\to k}(S)$ to a_k, being a_k the agent in S with highest position in PT, requiring coalition S. Thus, in Figure 1(b), a_0 will require coalition $\{12\}$ for $\{012\}$ through a_1. Finally, a_i stores each generated coalition S, adding a_i, ($S \cup \{i\}$), as well as a_i's single coalition ($\{i\}$), as part of its leading coalitions $\mathbf{L}_i$. Since the only coalition that a basic coalition $S \in \mathbf{L}_i$ requires in $\mathbf{L}_i$ is itself, mapping $Req_i(S)$ returns $\{S\}$.



(a) G (b) PT (c) Clique tree.

Figure 1: Example of (a) a graph with a cycle (G); (b) a pseudotree PT of G; and (c) a clique tree of PT.

Step 2. (Composed coalitions). In this step, each agent a_i will generate the set of *composed* coalitions. For each descendant $d^{j=1\ldots m}$ a_i will combine all coalitions reachable from d^j with all compatible coalitions reachable from d^{j+1}, from d^{j+2}, ..., and until d^m, storing at each step the new coalitions as reachable from d^j. Thus, in Figure 1(b), a_0 will combine coalitions reached from a_1 with coalitions from a_2 (storing any new coalition as reachable from a_1) and coalitions reachable from a_1 with coalitions from a_3. Two coalitions, S reachable from d^j, and S' reachable from $d^{k>j}$, are compatible if S' does not contain any agent in S or in its frontiers ($S' \cap S \cap F_S \neq \emptyset$). Thus, in Figure 1(b), a_1 will not combine $\{1\}$ from a_1 and $\{2\}$ from a_2 because a_2 is a frontier for $\{1\}$. A composed coalition is generated as $\{S \cup S'\}$ with frontier agents $\{F_S \cup F_{S'}\}$. Thus, in Figure 1(b), a_0 the result of combining $\{2\}$ from a_2 with $\{3\}$ from a_3 is a composed coalition $\{2, 3\}$ reachable from a_2 with $F = \{2\} \cup \emptyset$. For each composed coalition resulting from the combination of S with S', a_i computes its set of requiring coalitions $Req_i(S \cup S' \cup \{i\})$ as $\{S, S'\}$ if S is a basic coalition or, alternatively, as $\{Req_i(S), S'\}$ if S is a composed coalition. Thus, a_0 computes the required coalitions for $\{2, 3\}$ as $Req_1(\{0, 2, 3\}) = \{\{0, 2\} \cup \{1, 3\}\}$. Finally, a_i stores each composed coalition S including $\{a_i\}$ ($S \cup \{i\}$) as part of its leading coalitions $\mathbf{L}_i$.

Step 3. (Requiring coalitions). In this step, each agent a_i will process the incoming requiring messages. For each requiring message $req_{j\to i}(S)$ received, agent a_i adds the re-

quirement to its requiring mapping $Req_i(S \cup \{j\}) = \{S\}$ and coalition $S \cup \{j\}$ to its set of requiring coalitions $\mathbf{R}_i$.

3.2 Delegation-based model

In this section we provide a new mapping between any graph-restricted game and a graphical model, and more concretely, a junction tree (Aji and McEliece 2000; Dechter 2003). Our representation of the problem builds on the delegation-based graph explained in section above. In our encoding each agent is responsible for a clique composed of several decision variables and a value function. Thus, in this new mapping, each agent $a_i \in A(G)$ creates the following two sets of binary decision variables:

- X_i^L A set of option (leading) variables that includes one variable x_S for each coalition $S \in \mathbf{L}_i$.
- X_i^R A set of option (requiring) variables that include one variable x_S for each coalition $S \in \mathbf{R}_i$.

Since these two sets of variables stand for all local options of agent a_i when forming a coalition, we refer to them as a_i's local variables $X_i = X_i^R \cup X_i^L$. Figure 1(c) shows the clique tree for the PT graph arrangement in Figure 1(b) where each dashed box surrounds the set of option variables that an agent is responsible for (that form part of its clique) and arrows join a requirement variable to its required variable in X_i [2]. To illustrate how the representation works, we analyze the option coalition variables that a_1 is responsible for. a_1 has four variables encoding each of its four local options to form a coalition. One one hand, a_1 can form two coalitions as a leader: its single coalition ($\{1\}$) and a coalition with agent a_2 ($\{1,2\}$). Thus, $X_1^L = \{x_1, x_{12}\}$. On other hand, a_1 or a_1 together with a_2 can join a coalition through a_0. Thus, $X_1^R = \{x_{01}, x_{012}\}$. Moreover, notice that from all configurations of a_i's variables only a subset will be feasible - i.e. guarantee a_i's internal coherence as decision maker. One the one hand, each agent a_i should consider to join one and only one coalition. On the other hand, a coalition can not be formed if any of its required coalitions is not composed. As a result, the set of X_i's feasible configurations, that we refer to $\mathbf{X_i}$, contains a configuration per variable $x_S \in X_i$, $\mathbf{x}_S$, in which: (i) x_S and all coalition variables required by S ($\{S' | S' \in Req_i(S)\}$) are set to 1; and (ii) the rest of variables are set to 0. Thus, in Figure 1(c), the set of feasible configurations of a_1, $\mathbf{X_1}$, contains four configurations, one per variable in X_1 – i.e., configuration $\mathbf{x_{012}}$ sets x_{012} and its required variable x_{12} to 1 and the rest of variables to 0.

Moreover, each agent a_i has a value function $V_i : \mathbf{X}_i \rightarrow \Re$ that for each $x_S \in X_i$: if S is a leading coalition returns $v(S)$, otherwise is 0. For example, in Figure 1(c), the value function of agent a_1 returns the value of coalitions $\{1\}$ and $\{12\}$ for its respective configurations $\mathbf{x_1}$ and $\mathbf{x_{12}}$.

Then, given this representation, the CSG problem can be cast as the problem of finding the feasible configuration for all agents variables $X = X_1, \ldots, X_{|A(G)|}$ that maximises the following value function: $F(\mathbf{X}) = \bigotimes_{a_i \in A(G)} V_i$ where $\bigotimes$ stands for the combination (also called join) operator (Dechter 2003). Recall that the combination of two functions $F_i(\mathbf{X_i}) \otimes F_j(\mathbf{X_j})$ is a function defined over the joint set of feasible configurations of X_i and X_j [3].

Now, assume we are given a configuration $\mathbf{x}^*$. The following mapping allows us to recover the optimal coalition structure CS^* in CG.

Definition 3 (Ω)**.** *Ω maps any configuration $\mathbf{x}$ into a coalition structure CS composed of all coalitions S activated in $\mathbf{x}$ ($x_S = 1 \in \mathbf{x}$) for which it does not exist any other coalition S' such that its coalition $x_{S'}$ is activated in $\mathbf{x}$ and that contains all agents in S ($\nexists x_{S'} = 1 \in \mathbf{x}$ such that $S \subset S'$).*

In a distributed setting, agents can use mapping Ω over its optimal configuration $\mathbf{x_i^*}$ restricted to variables X_i. In this case, observe that mapping Ω will return either a leading coalition $S \in \mathbf{L}_i$ that stands for the coalition that the agent will form or a requiring coalition in $S \in \mathbf{R}_i$, that stands for the ancestor through which a_i and other agents down a_i in S are joining a coalition through. For example, in the game of Figure 1(a) using the characteristic function of Example 1, the mapping Ω over a_1's optimal configuration $\mathbf{x}_1 = \{x_1{=}0, x_{12}{=}1, x_{01}{=}0, x_{012} = 1\}$ recovers coalition $\{012\}$ (a_1 together with a_2 join a coalition through a_0).

4 The *SCF-Graphs* algorithm

In this section we introduce the *SCF-Graphs* algorithm (whose pseudocode is outlined in Algorithm 1) that allow agents to self-organise into stable coalition structures or detect the core emptiness on any graph-restricted coalitional game. *SCF-Graphs* requires a preprocessing phase, in which agents compile the game $CG = \langle A(G), v, F(G) \rangle$ into a pseudotree PT and generate leading coalitions and requiring mappings as described in Section 3.1. As a result of this preprocessing each agent a_i in SCF-Graphs starts knowing its parent (a_p) and children (Ch_i) in the PT as well as its leading coalitions $\mathbf{L}_i$, the requiring mapping Req_i and the characteristic function v. Building on this, each agent a_i initializes its local decision problem (line 1) by creating its set of local decision variables X_i, including leading and requiring sets, and its value function V_i as explained in Section 3.2. Then, *SCF-Graphs* runs into three phases (**demand propagation**, **main**, **offer propagation**) during which agents exchange two kinds of messages: (i) **demand messages**, to exchange demands through tree-edges for requiring coalitions whose activation is required by some agent up to PT; and (ii) **offer messages**, to propagate offers through tree- and pseudo- edges for required coalitions to agents down PT.

In what follows we describe the main phases of the *SCF-Graphs* algorithm using the trace in Figure 2 of an execution of *SCF-Graphs* over the the pseudotree arrangement of the CG cyclic graph game in Figure 1(c).

Each agent a_i starts the **demand propagation phase** waiting until receiving a *demand* message from each of its children $a_j \in Ch_i$ (line 3). Figure 2(a) depicts the demand messages exchanged during this phase. Thus, agent a_1 waits until it receives the *demand* message from a_2 that contains a function over feasible configurations of variables $X_{21} =$

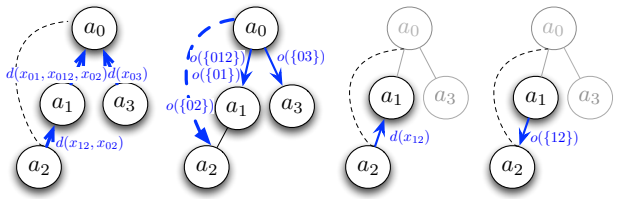
[2] Whenever a variable lies between two dashed boxes, it means that both agents need to keep a copy of it

[3] $\bigotimes$ stands for the generalisation of the join operator over a set of functions: $\bigotimes_{f_1,\ldots,f_{|M|}} = f_1 \otimes (f_2 \otimes \ldots (f_{|M|-1}) \otimes f_{|M|})$

Algorithm 1 SCF-Graphs

a_i knows $\langle a_p, Ch_i, Req_i(\mathbf{L_i} \cup \mathbf{R_i}), v\rangle$ and runs:

1: $\langle X_i = X_i^L \cup X_i^R, V_i(\mathbf{X_i})\rangle \leftarrow$ ***initialize***$(Req_i(\mathbf{L_i} \cup \mathbf{R_i}), v)$;
2: **/*PHASE I: DEMAND PROPAGATION*/**
3: Wait for the demands $d_{j\to i}(\mathbf{X_{ji}})$ from each child $a_j \in Ch_i$;
4: $\langle P_i(\mathbf{X_i^P}), \rho_i\rangle \leftarrow$ ***demand_propagation()***;
5: **/*PHASE II: MAIN PHASE*/**
6: $CORE \leftarrow true$;
7: **while** $X_i^P \neq X_i^L \neq X_i$ (payment depends on some alien coalition or an offer is missing) **do**
8: ***wait_for_messages();***
9: **for all** offers $o_{j\to i}(S)$ received **do**
10: $X_i^L = X_i^L \cup \{x_S\}$; /*Add x_S as leading variable*/
11: $\mathbf{x_i^*} = \mathbf{x_i^*} \cup (x_S = \mathbf{x_S^*})$; /*Add decision*/
12: $V_i(\mathbf{x}_S) = o(S)$; /*Add the offer */
13: **if** $a_j == a_p$/*Offer received from the parent*/ **then**
14: Mark a_i as the new root;
15: **end if**
16: **end for**
17: $\langle P_i(\mathbf{X_i^P}), \rho_i\rangle \leftarrow$ ***demand_propagation()***;
18: **end while**
19: **/*PHASE III: OFFER PROPAGATION*/**
20: $\mathbf{x_i^*}$=$arg\max_{\mathbf{x_i}\in\mathbf{X_i^P}\setminus X_i^R} P_i(\mathbf{x_i^*};\mathbf{x})$/*Compute optimal configuration*/
21: $S_i^* \leftarrow \Omega(\mathbf{x_i^*})$; /*Retrieve the optimal coalition*/
22: **if** $P_i(\mathbf{x_i^*}) \neq \rho_i$ **then**
23: $CORE = false$;
24: **end if**
25: **for all** $S \in \mathbf{L_i}$ /*For each local coalition*/ **do**
26: $X_o = \emptyset$; /*Coalitions already offered*/
27: $\mathbf{x_o} = \bigcup_{x_S\in X_i}(x_S = 0)$; /*Demand of variables already offered*/
28: **while** $X_o \neq Req_i(S)$ /*Offer through required coalitions*/ **do**
29: Let $x_{S'}$ be the next variable in $Req_i(S) \setminus X_o$;
30: $X_o = X_o \cup \{x_{S'}\}$;/*Mark $x_{S'}$ as offered*/
31: $\mathbf{x_{s'\cap o}} = (X_o = 1; X_i \setminus X_o = 0)$; /*The joint demand of $x_{S'}$ and other variables already offered*/
32: $o_S = P_i(\mathbf{x_S}) - \rho_i - \sum_{j\in Ch_i}(d_{j\to i}(\mathbf{x_{s'\cap o}}) - d_{j\to i}(\mathbf{x_o}))$;
33: $\mathbf{x_o} = \mathbf{x_{s'\cap o}}$;
34: $o(S') = max(o(S'), o_S)$;
35: **end while**
36: **for all** $S \in \mathbf{L_i} | Req_i(S) = \{S\}$ /*If basic coalition*/ **do**
37: Let a_k be the agent in $S\setminus\{i\}$with highest position in PT;
38: $o_{i\to k}(S) \leftarrow \langle o(S), (x_S = \mathbf{x_S^*}) \in \mathbf{x_i^*}\rangle$;
39: **end for**
40: **end for**
41: Send offer messages $\{o_{i\to k}(\mathbf{S})\}$ out to respective agents;
42: $CORE \leftarrow$ **propagate_core_status**$(CORE)$;
43: **return** $\langle \rho_i, S_i^*, CORE\rangle$;
44: **EndAlgorithm**
45: **Procedure demand_propagation()**
46: $P_i(\mathbf{X_i^P}) = V_i \otimes \bigotimes_{j\in Ch_i} d_{j\to i}$; /*Compute payment function*/
47: $\rho_i = \max_{x_S\in X_i^L} P_i(\mathbf{x_S})$; /*Compute agent's payment*/
48: **if** a_i is not the root /*Compute demands*/ **then**
49: $X_{ip} = X_i^P \setminus X_i^L$; /*Set of variables on which depends a_i's payment excluding $a_i's$ leading variables*/
50: $d_{i\to p}(\mathbf{X_{ip}}) = \max_{\mathbf{x}\in\mathbf{X_i^P}\setminus X_{ip}} P_i(\mathbf{X_{ip}};\mathbf{x}) - \rho_i$;
51: Send $d_{i\to p}(\mathbf{X_{ip}})$ to a_p;
52: **end if**
53: **return** $\langle P_i(\mathbf{X_i^P}), \rho_i\rangle$
54: **EndProcedure**



(a) (b) (c) (d)

Figure 2: Messages exchanged at different steps of the execution of *SCF-Graphs* over the clique tree in Figure 1.

$\{x_{12}, x_{02}\}$ before computing the demand message for its parent a_0. To compute its demand for its parent, each agent needs first to compute its **payment function** and its **expected payment** (Procedure **demand_propagation**, lines 44-53). Each agent a_i computes its payment function, $P_i(\mathbf{X_i^P})$, as the combination of its value function, $V_i(\mathbf{X}_i)$, and the (last) *demand* messages from its children that substract the amount required for agents down a_i (line 45). Thus, in Figure 2(a), agent a_1 will combine its value function $V_1(\mathbf{X_1})$ with the demand received from a_2. Notice that the payment function depends on a set of variables X_i^P, that is the set of variables of a_i (X_i) and the set of variables in the scope of any demand received from a child $a_j \in Ch_i$ (X_{ji}). Thus, in Figure 2(a), the payment function of a_1 will depend not only on its variables X_1 but also on some alien coalition variable, namely x_{02}, that is propagated from a_2. Then, a_i computes its expected payment ρ_i as the highest payment he can get on any of its leading coalitions, that is for any configuration of a variable in X_i^L (line 46). Thus, in Figure 2(a), a_1 computes the highest payment he gets among variables x_1,x_{12}. After that, if agent a_i is not the root, a_i sends a message to its parent a_p that summarizes its payment function P_i over all possible configurations of variables in X_{ji} (filtering out its leading variables X_i^L) and substracts its expected payment ρ_i (lines 47-51). Thus, in Figure 2(a), a_1 summarises its payment function P_1 over variables $\{x_{01}, x_{012}, x_{02}\}$ (filtering out its leading variables $X_1^L = \{x_1, x_{12}\}$). The result of this summarization is, for each coalition variable x_S in X_{ip}, the demand that agent a_j will need to satisfy to agents in $A(PT_i)$ to ensure their participation in S. In case of an alien variable $x_S \notin X_i$, this payment is not simply the amount demand by agents down PT_i but considers the potential loss of a_i when restricting to coalitions that does not contain any agent in S (the maximum offer a_i is going to offer to agents in S to join). Thus, in Figure 2(a), the demand of agent a_1 to its parent a_0 for x_{02} is the maximum between the amount demanded from a_2 to join this coalition and the potential loss of a_1 from moving to a coalition ($\{1\}$) that does not include a_2.

Once the *demand propagation* phase is over, agents start the *main phase* (lines 5-22) in which each agent a_i iteratively checks (line 7): (C1) if he has received an offer for all requiring coalitions ($X_i^L \neq X_i$); and (C2) if its payment depends on some alien coalition, ($X_i^P \neq X_i$). If any of these two conditions is not satisfied, it means that agents needs to either receive new demands from children (that get rid of alien variables) or offers from their ancestors (that make an offer for some requiring coalition), so agent a_i waits for messages (line 8). Thus, in Figure 2(a)-(d), after the initial de-

mand propagation phase, agent a_3 waits until receiving an offer message from its parent a_0, whereas agent a_1 needs to wait for a message from its parent a_0 and a new demand from its child a_2 that gets rid of the alien coalition x_{02}. When agent a_i receives an offer message over a coalition S, he adds requiring variable x_S to its set of leading variables X_i^L (given the offer, now a_i is the leader of coalition S) and updates its value function and optimal decision over S (lines 10-12). Additionally, a_i checks if the offer messages comes from its parent in the pseudotree, in which case the agent marks itself as the new root (lines 13-15). Finally, independently if a new offer or demand is received the agent re-runs the demand_propagation procedure recomputing its payment function, its payment and the demand for its parent (line 17). When conditions (C1)(C2) are satisfied (notice that for the root they are always satisfied), each agent a_i starts the **offer propagation phase** (lines 19-40). For example, in Figure 2, a_0, as root, starts the offer propagation phase just after the initial demand propagation phase. First, each agent a_i computes its optimal local configuration $\mathbf{x_i^*}$ given the already inferred decision of its ancestors An_i over requiring coalitions (line 20) and retrieves the optimal coalition S_i^* corresponding to this optimal configuration (line 21). Next, each agent locally checks for the emptiness of the core (lines 22-24). The core is detected as empty if a_i detects that its payment conditioned to the decision of ancestors ($P_i(\mathbf{x_i^*})$) is less than the best payment he can get considering all its local configurations (ρ_i). Finally, each agent a_i can start computing the offers of coalitions requiring some agent down the tree. For each of its leading coalitions $S \in \mathbf{L}_i$, the offer of agent a_i for S is distributed among independent offers for S's required coalitions ($Req_i(S)$). Each agent a_i computes the offer of a coalition S for a required coalition S' as the value of its payment function for S minus the $a_i's$ payment (notice that this includes demands from all agents) canceling the difference between the joint demand for $x_{S'}$ and coalitions still not offered ($\mathbf{x_{S' \cap o}}$) and the joint demand of coalitions still not offered ($\mathbf{x_o}$) (line 32). Thus, in Figure 2(b), a_0 computes the offer of $\{01\}$ for $\{01\}$ (the only requiring coalition of $\{01\}$ in X_1 is itself) as the value of its payment function for $\{01\}$ ($P_0(\mathbf{x_{01}})$) minus its payment (ρ_0) whereas substracting the demand from coalition $\{01\}$. Notice that whenever S is a basic coalition the amount offered to S is just the value of agent a_i for S minus a_i's payment. In contrast, when the coalition is composite the offer is distributed among more than one required coalitions. For example, in Figure 2(b), the offer of a_0 for its leading coalition $\{013\}$ is distributed among its requiring coalitions $\{01\}$ and $\{03\}$. After computing offers, each agent a_i sends for each of its basic coalitions $\{S \in \mathbf{L}_i | Req_i(S) = \{S\}\}$, an offer message to the agent in $S \backslash \{i\}$ with higher position in PT (lines 35-40). At the end of the algorithm (line 41), agents run a distributed procedure[4] to propagate the core status across the graph (emptiness of the core is propagated through the whole graph).

Given the description of *SCF-Graphs* the following Theorem states its correctness:

[4]Any distributed algorithm used for convergence detection can be used for that purpose.

Theorem 1. *Given a game on a graph* $CG = \langle A(G), v, F(G) \rangle$*, then, if the core of* CG *is not empty, the outcome produced by* SCF-Graphs(CG) *belongs to the core of* CG*; otherwise* SCF-Graphs(CG) *outcomes the optimal coalition structure of CG detecting the emptiness of the core.*

Because of lack of space, next we just expose the intuition and sketch for the proof of Theorem 1. The interested reader can find more detailed proofs in (Technical).

Sketch of proof. To prove Th. 1 we first prove that the algorithm always outputs the optimal coalition structure CS^* (Lem. 1). Then, we prove that if the algorithm outputs a payoff allocation, ρ is stable, that is: (i) ρ is an imputation of CS^* (Lem. 2); and (ii) ρ is group rational (Lem. 3). Finally we prove that the algorithm detects rightly the emptiness of the core (Lem. 4). Let's assume a synchronous execution and denote the payment of a_i at iteration t as $\rho_i^t$and $\sum_{a_i \in A} \rho_i^t$ as $\rho^t(A)$. Then, since agents can not compute its final payment until receiving an offer message from its parent, $\rho = \{\rho_1^{l(1)}, \ldots, \rho_{|A|}^{l(|A|)}\}$where $l(i)$ stands for the level of a_i in PT (root has level 0). Let's denote by $A_{<l}$ the set of agents with lower level than l (analogously for $>$,$\geq$,$\leq$). We prove Lem. 1 by showing that the demand propagation phase and the posterior propagation and conditioning of inferred values through offer messages are equivalent[5] to the execution of a dynamic programming procedure (Vinyals *et al.* 2011) that is guarantee to compute the optimal configuration in a junction tree. Moreover, from Lem. 1, we observe that agent's payments that result from running demand propagation are an imputation for $CS^*(v(CS^*) = \rho^0(A))$(Obs. 1). We also observe that an offer from a_i to a_j generates at iteration $l(i) + 1$a new demand propagation from a_j to a_k, being a_k the child of a_i in the path from a_j to a_i. Then, by Lem. 1, at each iteration t agents are solving a CSG problem for a subgame $CG_t = \langle A_{\geq t}, v_t, F(G) \rangle$in which: v_i encodes $A_{<t}$'s payments ($v_i(S) = v(S) - \rho(S \cap A_{<t})$); and coalitions can overlap on $A_{<t}$. Moreover, since *SCF-Graphs* detects core emptiness whenever the value of the optimal coalition structure in any subgame, CS_t^*, differs from CS^*, for Lem. 2-3 we can assume that $\forall t : CS_t^* = CS^*$. Then by Obs.1, Lem. 2 follows from that $\forall t\ v_t(CS^*) = \rho^t(A_{\geq t})$and $v_t(CS^*) = v(CS^*) - \rho(A_{<t})$. Lem. 3 builds on induction showing that at any iteration t $A_{\leq t}$ satisfy all the group rationality inequalities for coalitions composed among themselves ($\forall S \in F(G_{A_{\leq t}}) \rho(S) \geq v(S)$). Finally, Lem. 4 is proven by contradiction (the final payment of agents are computed in a way for which they can not block the existence of an allocation in the core).

5 Conclusions

In this paper we presented a novel graphical model representation scheme to represent graph-restricted games that uses a delegation-based model for coalition formation. Then, building on that representation, we formulate a message-passing algorithm that allow agents not only to form the optimal coalition structure but also an allocation of payments in the core of the game or to detect core emptiness.

[5]upon a particular normalisation of messages and functions

References

Srinivas M. Aji and Robert J. McEliece. The generalized distributive law. *IEEE Transactions on Information Theory*, 46(2):325–343, 2000.

Vincent Conitzer and Tuomas Sandholm. Complexity of determining nonemptiness of the core. In *ACM Conference on Electronic Commerce*, pages 230–231, 2003.

V. D. Dang and N. R. Jennings. Generating coalition structures with finite bound from the optimal guarantees. In *AAMAS*, pages 564–571, 2004.

Rina Dechter. *Constraint processing*. Elsevier Morgan Kaufmann, 2003.

G. Demange. On Group Stability in Hierarchies and Networks. *Journal of Political Economy*, 112(4):754–778, August 2004.

X. Deng and C. Papadimitriou. On the complexity of cooperative solution concepts. *Mathem. of Operation Research*, 19:257–266, 1994.

Gianluigi Greco, Enrico Malizia, Luigi Palopoli, and Francesco Scarcello. On the complexity of compact coalitional games. In *IJCAI*, pages 147–152, 2011.

Grid 2030. a national vision for electricity's second 100 years. Washington DOE, 2003. http://www.oe.energy.gov/DocumentsandMedia/Electric_Vision_Document.pdf.

Reshef Meir, Jeffrey S. Rosenschein, and Enrico Malizia. Subsidies, stability, and restricted cooperation in coalitional games. In *IJCAI*, pages 301–306, July 2011.

Tomasz Michalak, Jacek Sroka, Talal Rahwan, Michael Wooldridge, Peter Mcburney, and Nicholas Jennings. A distributed algorithm for anytime coalition structure generation. In *AAMAS*, pages 1007–1014, May 2010.

M. J. Osborne and A. Rubinstein. *A Course in Game Theory*. MIT Press, Cambridge MA, USA, 1994.

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

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

Technical. Proving correctness for SCF-Graphs. Technical report, . Available at `https://docs.google.com/open?id=0BwSadTGylubwNTBmNmY0OTctNTBkYy00NDNiLWJiM2MtYjlmNTY5NGE3ZDc5`.

R. van den Brink. On Hierarchies and Communication. Tinbergen Discussion Paper 06/056-1, Tinbergen Institute and Free University, 2006.

Meritxell Vinyals, Juan Antonio Rodriguez-Aguilar, and Jesus Cerquides. Constructing a unifying theory of dynamic programming dcop algorithms via the generalized distributive law. *In JAAMAS*, 22(3):439–464, 2011.

Thomas Voice, Maria Polukarov, and Nicholas R. Jennings. Graph coalition structure generation. *CoRR*, abs/1102.1747, 2011.