Dependency in Cooperative Boolean Games

Luigi Sauro[a], Serena Villata[b]

[a]*Dipartimento di Scienze Fisiche, University of Naples*
[b]*Dipartimento di Informatica, University of Turin*

Abstract

Cooperative boolean games are a family of coalitional games where agents may depend on each other for the satisfaction of their personal goals. In Dunne et al. [1] the authors define as solution concept the notion of *core* showing that several decision problems, such as core-emptiness, are Π_2^p-complete. In this work we investigate how to improve the computation of the core. In particular, we introduce two different types of dependence networks, *abstract dependence networks* and *refined dependence networks*, that are used to define the notion of stable coalitions and Δ-reduction, respectively. Stable coalitions enable to focus on a subset of the agents and use results to determinate the core of the whole game. Δ-reduction prunes the search space by returning a set of actions that are not admissible to be executed. We present an algorithm based on stable coalitions and a Δ-reduction implemented in Prolog and experimental results showing how they effectively improve the computation of the core.

Key words: Coalition formation, dependence networks, stable sets, core

1. Introduction

Cooperative Boolean Games [1] (CBG) are a particular family of boolean games [2, 3]. In such type of games, agents' primary aim is to achieve their individual goal, which is represented as a propositional logic formula over some set of boolean variables. Each agent is assumed to exercise a unique control over some subset of the overall set of boolean variables, and the set of truth assignments for these variables corresponds to the set of actions the agent can take. As secondary aim, each agent wants to minimize the cost associated to the execution of her actions. As in typical coalitional games, an agent can have the necessity to cooperate with the others because she

does not have sufficient control to ensure her goals are satisfied. However, the desire to minimize costs leads to preferences over possible coalitions, and hence to strategic behavior. One of the solution concepts proposed for such games is the notion of core that strengthens the Nash equilibrium criterion: a truth assignment of the boolean variables (strategy profile) is in the core in case no subset of agents can unilaterally deviate from it and improve the rewards of each of its members.

In this paper, we propose a new step to make the computation of the core easier by means of the dependence networks associated to the cooperative boolean game. The reasons of our choice are twofold: first, we present a number of abstractions that allow to reduce the search space by means of a set of criteria principally based on graph visit algorithms which are computationally tractable; second, we underline a number of hidden properties in the notion of core showing how, in certain cases, this notion is too much strict and, thus, it can lead to counterintuitive results.

We define two kinds of dependence networks, representing two different levels of abstraction of a cooperative boolean game. Abstract dependence networks have already be used by Bonzon et al. [4] to define stable coalitions which enable to divide the entire problem in subproblems and then combine their solutions. While Bonzon et al. [4] show that the notion of stability is complete with respect to the pure Nash equilibrium with non costly actions, we show in this paper that the notion of stability is complete also with respect to the solution concept of the core in case of cooperative boolean games with costly actions. Refined dependence networks essentially provide a graph representation of a cooperative boolean game where the numerical information about costs is abstracted and actions are partitioned in free and costly actions. We present a reduction, called Δ-reduction, to pass from a cooperative boolean game G to a CBG G', simpler to be solved because less actions can be executed.

Dependence networks enable to reduce the generation space, that is the set of truth assignments on which core-membership has to be checked. We also define optimization techniques that improve the core-membership problem and provide experimental results of a Prolog implementation.

The reminder of the paper is as follows. In section 2 we formally define cooperative boolean games and the core. Section 3 introduces the dependency relations in cooperative boolean games by defining the two types of dependence networks. Section 4 shows the optimization techniques relative to the core-membership problem. Sections 5 and 6 present the algorithms to

simplify the computation of the core and the experimental results. Section 7 gives an informal introduction to the various kinds of boolean games and dependence networks studied in multiagent systems. Conclusions end the paper.

2. Cooperative Boolean games

A Cooperative Boolean Game [1] consists of a set of agents $1, \ldots, n$ which desire to accomplish personal goals. Goals are represented by propositional formulas γ_i over some set of boolean variables Φ. Each agent controls a (possibly empty) subset $\Phi_i \subseteq \Phi$ of the overall set of boolean variables. The notion of control is used to mean that agent i has the unique ability within the game to set the value (either $\top$ or $\bot$) of each variable $p \in \Phi_i$. Variables are assumed to be initially false, this way of setting a variable $p \in \Phi$ to be $\top$ has the meaning of *performing the action* p, whereas setting $p \in \Phi$ to $\bot$ means *doing nothing*. Since action (as opposed to inaction) typically incurs some cost, Dunne et al. [1] define the cost function *cost* in such a way that $cost(p)$ denotes the cost of performing action p (i.e., setting p to $\top$). As in [1], we do not consider the cost about refraining from doing something even if there may be contexts in which it is possible that both doing and not doing an action have costs. Agents' secondary goal is to minimize their costs. Summing up briefly, if the only way an agent can achieve her goal is by making all her variables true, then the agent prefers to do that rather than not achieve her goal but if there are different ways to achieve it, then the agent prefers always those that minimize costs.

Definition 1 (Cooperative Boolean Game [1]).
A cooperative boolean game G is a $(2n+3)$-tuple

$$G = \langle A, \Phi, cost, \gamma_1, ..., \gamma_n, \Phi_1, ..., \Phi_n \rangle$$

where $A = \{1, ..., n\}$ is a set of agents, $\Phi = \{p, q, ...\}$ is a finite set of boolean variables, cost is a cost function defined in $\Phi \to \Re_+$, $\gamma_1, ..., \gamma_n$ are the boolean formulas over Φ which represent the goals of the agents and $\Phi_1, ..., \Phi_n$ is a partition of Φ over n, with the intended interpretation that Φ_i is the set of boolean variables under the control of agent i.

A subset ξ of Φ is a valuation, where the usual meaning is that the value of the variables belonging to ξ is true and the value of the other ones is

false. $\xi \models \phi$ means that ϕ is true under the valuation ξ under the standard propositional semantics.

Valuations intuitively correspond to possible strategies of the agents, in Dunne et al. [1], $cost_i(\xi)$ denotes the cost to agent i of valuation $\xi \subseteq \Phi$, that is,

$$cost_i(\xi) = \sum_{v \in (\xi \cap \Phi_i)} cost(v)$$

As in Dunne et al. [1], we define a utility function that is always positive if the valuation ξ satisfies the goal of the agent i and otherwise it is always negative. If μ represents the total cost of all variables, the utility for agent i of a valuation ξ, $u_i(\xi)$ is defined as:

$$u_i(\xi) = \begin{cases} 1 + \mu - cost_i(\xi) & \text{if } \xi \models \gamma_i \\ -cost_i(\xi) & otherwise \end{cases}$$

This utility function leads to a preference order $\succeq_i$ over valuations: $\xi_1 \succeq_i \xi_2$ if and only if $u_i(\xi_1) \geq u_i(\xi_2)$. The best utility for an agent is reached if she has her goal satisfied without performing any action (utility of $\mu + 1$) while the worst utility for an agent is reached in the case she does not get her goal satisfied but she performs actions such as to set all her variables true (utility $-cost_i(\Phi_i)$).

Typically, an agent has to cooperate with the other agents because she does not have sufficient control to ensure her goal is satisfied. This means that an agent may not have under her control all the variables she needs to be set as $\top$ in order to achieve her goal. However, aiming at minimizing her costs, an agent is preferentially unwilling to execute one of the actions under its control. Therefore, agents cooperate only in the case in which a cooperative solution is preferable to the alternatives, either because the agents cannot achieve their goals independently or their can reduce their respective cost. The utility achieved by agents within a coalition depends not only on their actions but on the actions of the whole set of agents involved in the game.

A coalition D is represented as a (sub)set of agents, $D \subseteq A$ and thus it does not represent any kind of particular relationship among the members composing it. Let $D \subseteq A$ be a coalition, then Φ_D denotes the set of variables under the control of some member of D, $\Phi_D = \bigcup_{i \in D} \Phi_i$. If a valuation ξ_2 is the same as a valuation ξ_1 except at most in the value of the variables controlled by D then $\xi_1 = \xi_2 \; mod \; D$.

The meaning of "D blocks ξ_1 through ξ_2" is that ξ_2 could do better than ξ_1 only flipping the value of some of the variables under the control of D. From Dunne et al. [1], the definition of blocked valuation is as follows:

Definition 2 (Blocked Valuation [1]).
A valuation ξ_1 is blocked by a coalition $D \subseteq A$ through a valuation ξ_2 if and only if:

1. *ξ_2 is a feasible objection by coalition D which means that $\xi_2 = \xi_1$ mod D.*
2. *D strictly prefers ξ_2 over ξ_1: for all $i \in D : \xi_2 \succ_i \xi_1$.*

The blocked valuation allows us to define the core of a cooperative boolean game. The core is a fundamental concept behind coalitional game theory. A valuation is in the core if and only if no coalition has an incentive to defect.

Definition 3 (Core).
Given a cooperative boolean game G, $\xi \in core(G)$ if and only if it is not blocked by any coalition.

Definition 3 is a straightening of the well-known Nash equilibrium which in non-cooperative game theory [5] is usually called Strong Nash Equilibrium (SNE). This form of solution satisfies at the same time some requirements of both the cooperative and non-cooperative game theory. From the solution criteria developed in cooperative game theory it *borrows* efficiency, a strategy cannot be a solution if the agents or a part of them can obtain better results. From non-cooperative game theory, it assumes that agents are suspicious and that agreements cannot be enforced, that is at any time agents may betray agreements. To better understand the features of SNE, let us consider the following example.

Example 1. *Let G be a CBG involving two agents. $\Phi_1 = \{a\}$ and $\Phi_2 = \{b\}$, $\gamma_1 = \gamma_2 = a \vee b$, $cost(a) = cost(b) = 1$. The CBG can be represented by the following payoff matrix:*

a/b	1	0
1	$(1,1)$	$(2,3)$
0	$(3,2)$	$(0,0)$

On the one hand, $(0,0)$ is the only Nash equilibrium in this game, that is not efficient as the two agents collaborating can obtain better results, $(1,1)$, and

hence it is not a SNE. On the other hand, as $(1,1)$ *is not a Nash equilibrium, it is not a SNE too. Indeed, if* $(1,1)$ *is proposed as an agreement each agent would unilaterally betray it obtaining better results.*

Note that whether Strong Nash Equilibrium corresponds to the notion of core depends on how a strategic game is translated into a cooperative game (see Myerson [6]). For example, if we use for the strategic game in Example 1 the minimax representation of a corresponding cooperative game - as introduced by von Neumann and Morgensten [7] - then the strategy $\{1,1\}$ is in the core. This kind of representation is called *offensive* because it is assumed that for a certain coalition D, the agents in $A \setminus D$ *work* in order to minimize the outcomes of D. On the contrary if we adopt a *defensive* representation where the primary aim of the agents is to maximize their own utility, the SNE represents the core of the corresponding cooperative game. Note that if we change the goal to $\neg a \vee \neg b$, then we get $(3,3)$ for $\neg a, \neg b$, and $(0,0)$ for a, b thus $(3,3)$ is in the core. The question is what about if the meaning of the action a and b is of the kind "to refrain to do ..."? If we consider the positive representation of the actions we get the results shown before while if we take into account the negative one, we get different results. The analysis and discussion of the negative representation of the actions is left for future research.

As notion of stability for cooperative boolean games, the core is appealing but there are cases in which it would be empty. Example 1 is a example of game with an empty core. However, there are a number of cases in which the core is not empty. Let us consider the following example:

Example 2. *Consider a game where we have four agents (*$A = \{1,2,3,4\}$*) who want to go on holidays to the seaside or to the mountains. We represent with the boolean variable a to go to the seaside and with b to go to the mountains. Agent i going to the seaside is represented by setting* a_i *to true whereas the holiday to the mountains is represented by* b_i*. For each agent i,* $cost(a_i) = 2$ *and* $cost(b_i) = 1$*. Agent 1 is in love with agent 2 and he wants to go everywhere with her, thus his goal is represented by* $\gamma_1 = (a_1 \wedge a_2) \vee (b_1 \wedge b_2)$*. Agent 2 is in love with agent 1 but she cannot tolerate the change of temperature of the mountains thus her goal is* $\gamma_2 = a_1 \wedge a_2$*. Agent 3 is in love with 2 and, as he is jealous of agent 1, he would like to stay with 2 without the presence of 1, so his goal is* $\gamma_3 = (a_2 \wedge a_3 \wedge \neg a_1) \vee (b_2 \wedge b_3 \wedge \neg b_1)$*. Agent 4 is in love with agent 3, who does not like her, but she is not able to swim thus*

her goal is $\gamma_4 = b_3 \wedge b_4$. Let us say that agent i is satisfied given a valuation ξ if $\xi \models \gamma_i$, i.e., if agent i's goal is satisfied. It can be verified that $\{a_1, a_2\}$ is in the core.

We focus now on the problem of computing the core of a CBG. In general, this problem can be seen as a generation and test problem: generate a strategy, then check whether it belongs to the core according to Definition 3. Thus, it is useful in the following to consider a *generation space*, that is the set of strategies ξ we need to check for core-membership, and a *test space*, the set of strategies ξ' we search to establish whether ξ belongs to the core.

Clearly, if no optimization is implemented both the generation and the test space consist in the set of all strategies and unfavorable they have a cardinality equal to $2^{|\Phi|}$. However, in the following sections we study several optimization techniques which enable to reduce both the generation and the test space.

3. Reducing the generation space

In this section we present two optimization techniques that enable to reduce the generation space. Both of them are based on the notion of dependence networks that can be defined starting from a CBG. These networks, on the one hand, explicitly represent the inter-dependencies among agents according to their goals, on the other hand they abstract from the quantitative aspects of a game associated to the cost function. In particular, the first type of dependence network, we named Abstract Dependence Networks, is the same already used in Bonzon et al. [4] as decomposition method for the Nash Equilibrium in Boolean Games without costly actions. Here, we essentially extend the results in Bonzon et al. [4] to the framework defined in Dunne et al. [1].

Abstract Dependence Networks describe only which agents can play a role in the satisfaction of an agent's goal, therefore they abstract from how they can contribute and in particular if they have to execute a costly action. As this information may help in the study of the core and reduce the generation space, we define another type of dependence network, Refined Dependence Networks, which, as the name suggests, refines Abstract Dependence Networks by representing how agents contribute to the satisfaction of a goal and whether this involves a positive cost (without quantifying it). Starting from Refined Dependence Networks, we define a method called Δ-reduction

to reduce the admissible strategies and we prove the completeness of the Δ-reduction with respect to the computation of the core.

3.1. Abstract Dependence Networks

As in Bonzon et al. [4], in order to correctly establish the dependencies among agents we need to define which variables are relevant for the satisfaction of the agents' goal. A variable p is said irrelevant for a formula ϕ in case there exists an equivalent formula ϕ' where p does not occur. With $RV_G(i)$ we represent the set of all variables $p \in \Phi$ that are relevant for γ_i, whereas $RA_G(i)$ is the set of agents $j \in A$ such that j controls at least one relevant variable of i. The notions of relevant agent and relevant variable are crucial in what follows since they allow to define dependence networks and the way to prune them. Using the notion of relevant agents, we define a dependence network where nodes represent agents and an edge from i to j represents the dependence of i on j ($j \in RA_G(i)$).

Definition 4 (Abstract Dependence Network).
Given the cooperative boolean game $G = \langle A, \Phi, cost, \gamma_1, ..., \gamma_n, \Phi_1, ..., \Phi_n \rangle$, *the* abstract *dependence network of* G *is the directed graph* $ADN(G) = \langle N, R \rangle$ *such that the set of nodes* N *correspond to the agents in* G, $N = A$, *and* $(i, j) \in R$ *if and only if* $j \in RA_G(i)$.

As in Bonzon et al. [4] we say that a set of agents in $ADN(G)$ is stable in case it is *closed* under the relation R. $R(C)$ is the set of players from which C may need some action in order to be satisfied.

Definition 5 (Stable set).
Given a directed graph $\langle N, R \rangle$, $C \subseteq N$ *is stable if and only if* $R(C) \subseteq C$, *i.e. for all* $i \in C$, *for all* j *such that* $(i, j) \in R$ *then* $j \in C$.

Note that the notion of stable set is not related to the strategic criterion with the same name originally introduced by von Neumann and Morgenstern [7].

Definition 6 (ADN Projection).
Let $G = \langle A, \Phi, cost, \gamma_1, ..., \gamma_n, \Phi_1, ..., \Phi_n \rangle$ *be a cooperative boolean game,* $ADN(G) = \langle N, R \rangle$ *be the corresponding abstract dependence graph and* $C = \{i_1, \ldots, i_m\} \subseteq N$ *be a stable set, the projection of* G *on* C *is defined by* $G_C = \langle C, \Phi_C, cost_C, \gamma_{i_1}, ..., \gamma_{i_m}, \Phi_{i_1}, ..., \Phi_{i_m} \rangle$, *where* $cost_C : \Phi_C \rightarrow \Re_+$ *is the restriction of cost on* Φ_C.

By definition the goals occurring in the projection of a cooperative boolean game on a stable set C do not contain variables controlled by agents outside C, therefore the projection is itself a cooperative boolean game.

In Bonzon et al. [4] the authors claim, by restricting a boolean game G to the projection G_C of a stable set C, that if a strategy profile ξ_C in G_C is not a Nash equilibrium, then all of its extensions in G are not a Nash equilibrium. Here we extend this result to the case of the core in cooperative boolean games with costly actions.

Proposition 1.
Given a stable set C, if ξ is in core(G), then ξ_C is in core(G_C), where ξ_C is the projection of ξ on the variables controlled by C, $\xi_C = \xi \cap \Phi_C$.

Proof 1. *Let $\bar{\xi}$ be a generic valuation in G and $\bar{\xi}_C$ the projection of $\bar{\xi}$ on the variables controlled by C. Clearly, for all $i \in C$, $cost_i(\bar{\xi}) = cost_i(\bar{\xi}_C)$. Furthermore, as C is stable, for all $i \in C$, $RV_G(i) = RV_{G_C}(i)$ and hence $\bar{\xi} \models \gamma_i$ if and only if $\bar{\xi}_C \models \gamma_i$. This entails that for two generic valuations $\bar{\xi}$ and $\hat{\xi}$ and for all $i \in C$, $u_i(\bar{\xi}) \leq u_i(\hat{\xi})$ if and only if $u_i(\bar{\xi}_C) \leq u_i(\hat{\xi}_C)$.*

Assume that there exists a valuation ξ'_C that blocks ξ_C, i.e. there exists a $C' \subseteq C$ such that $\xi'_C = \xi_C$ mod C' and for all $i \in C'$ $\xi'_C \succ_i \xi_C$. Now let $\xi_{-C} = \xi \cap \Phi_{A \setminus C}$ and $\xi' = \xi'_C \cup \xi_{-C}$. Clearly, $\xi' = \xi$ mod C' and, since for all $i \in C$ $u_i(\xi) \leq u_i(\xi')$ if and only if $u_i(\xi_C) \leq u_i(\xi'_C)$, ξ is blocked by C' through ξ'.

Finally, given two coalitions D_1 and D_2, we say that two relative strategies ξ_{D_1} and ξ_{D_2} are consistent if and only if for each agent $i \in D_1 \cap D_2$, $\Phi_i \cap \xi_{D_1} = \Phi_i \cap \xi_{D_2}$. The following two propositions hold - the proof is straightforward and it is left to the reader.

Proposition 2.
Let ξ be a strategy blocked by a coalition D through ξ' and C be a stable set such that $C' = C \cap D \neq \emptyset$, then ξ is blocked by C' through ξ'_C.

Proposition 3.
Given the stable sets $C_1, \ldots, C_n$ and the relative strategies $\xi_{C_1}, \ldots, \xi_{C_n}$ depending on the stable sets, if

1. *for all $1 \leq i \leq n$, $\xi_i \in$ core(G_{C_i});*
2. *$\xi_{C_1}, \ldots, \xi_{C_n}$ are consistent;*

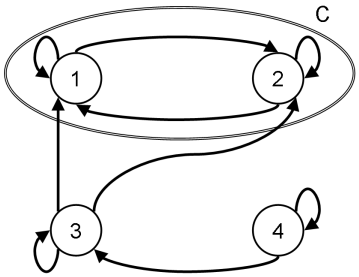

Figure 1: Abstract Dependence Network of Example 3 with the stable set C.

3. $A = \bigcup_{i=1}^{n} C_i$,

then $\bigcup_{i=1}^{n} \xi_i \in core(G)$.

Propositions 1 and 3 provide a way to decompose the problem of determining the core of a CBG G into the subgames G_{C_i}, where $C_1, \ldots, C_n$ are stable sets that involve all the agents in G. Then, once each $core(G_{C_i})$ is determined, it remains to gather any union ξ of consistent strategies $\xi_{C_i} \in core(G_{C_i})$, with $1 \leq i \leq n$. Due to Proposition 3, $\xi \in core(G)$, whereas, Proposition 1 ensures that in this way we find all the elements in $core(G)$, indeed if $\xi' \in core(G)$, then $\xi'_{C_i} \in core(G_{C_i})$, with $1 \leq i \leq n$ and hence ξ' is the union of consistent solutions in each G_{C_i}.

Example 3 (Continued). *Let us consider the abstract dependence network of the game presented in Example 2:* $ADN = \langle N, E \rangle$ *where* $N = \{1, 2, 3, 4\}$ *and* $R = \{(1, 2), (2, 1), (3, 1), (3, 2), (4, 3)\}$ *(see Figure 1). Following Definition 5 the set* C *composed by agents 1 and 2 is a stable set (all the edges that go out from agent 1 enter in agent 2 and converse), so we can consider first the projection of the game on* C *following Definition 6. Valuations are represented as* $uvyz \in \{0, 1\}^4$*, where* u *is the value of* a_1*,* v *that of* a_2*,* y *the value of* b_1 *and* z *that of* b_2*. Following Definition 6, it can be found that* 1100 *is the only strategy in the core. Due to Propositions 1, also in the complete game all the strategies containing either* b_1 *or* b_2 *and not containing one of* a_1 *and* a_2 *are blocked.*

3.2. Refined Dependence Networks

As seen before, Abstract Dependence Networks can be safely used to split the original problem in subproblems without loosing solutions. However, Abstract Dependence Networks may hide some useful information that

can also be used to prune some strategies that cannot belong to the core - and hence to reduce the search space. For this reason we define another notion of dependence network at a lower level of abstraction and we call it Refined Dependence Networks (RDNs). These networks may seem actually equivalent to the boolean game itself, except for the cost of the variables. These costs, however, are not a minor point since two boolean games that result in the same RDN can have different solutions depending in these costs.

A Refined Dependence Network represents how the goals can be satisfied by means of AND-arcs among the agents whose single edges are labeled with literals. Furthermore, costly actions are *marked* in a set Δ.

Definition 7 (Refined Dependence Network).
A Refined Dependence Network is an AND-dependence graph

$$\langle N, \Phi, \Delta, E, \Phi_1, ..., \Phi_n \rangle$$

where N is the set of nodes, Φ is the set of boolean variables, $\Delta \subseteq \Phi$ is the subset of variables p with $cost(p) > 0$, $E \subseteq N \times 2^{(N \times Litt(\Phi))}$ where $Litt(\Phi) = \Phi \cup \{\neg p | p \in \Phi\}$ and $\Phi_i \subseteq \Phi$ where n is the cardinality of N.

In the following, given a literal l, we denote by $|l|$ the corresponding boolean variable, that is if $l \in \Phi$, then $\mid l \mid = l$ whereas if $l = \neg p$, $\mid l \mid = p$. Furthermore, to simplify the formalism, we represent an AND-arc $(i, \{(j_1, l_1), \ldots, (j_m, l_m)\})$ as the set of triples $\{(i, j_1, l_1), \ldots, (i, j_m, l_m)\}$. Of course, a set as $\{(1, 3, p), (3, 4, q)\}$ has no meaning in our context.

As already done for Abstract Dependence Networks, we use Refined Dependence Networks to reveal the structure of interdependencies among agents. First, we assume that the goals of the agents do not contain irrelevant variables and are given in disjunctive normal form, i.e. $\gamma_i = \gamma_{i_1} \vee \cdots \vee \gamma_{i_m}$ where each γ_{i_j} is a conjunction of literals. To simplify again the formalism we describe respectively γ_i as a set of γ_{i_j} and each γ_{i_j} as a set of literals - the empty set has the usual meaning respectively of $\bot$ referred to γ_i and $\top$ referred to the γ_{i_j}. Roughly, each AND-arc e outgoing the agent i corresponds to a $\gamma_{i_j} \in \gamma_i$, where each single edge that composes e is labeled with a literal occurring in γ_{i_j} and reaches the agent that controls the corresponding variable. The set Δ consists of the actions that have a strictly positive cost.

Definition 8 (From CBGs to RDNs).
Given the cooperative boolean game $G = \langle A, \Phi, cost, \gamma_1, ..., \gamma_n, \Phi_1, ..., \Phi_n \rangle$,

RDN$(G) = \langle N, \Phi, \Delta, E, \Phi_1, ..., \Phi_n \rangle$ *is such that* $N = A$, $\Delta = \{p \in \Phi \mid cost(p) > 0\}$ *and* $\{(i, j_1, l_1), \ldots, (i, j_m, l_m)\} \in E$ *if and only if* $\{l_1, \ldots, l_m\} \in \gamma_i$ *and for all* $1 \leq h \leq m$, $\mid l_h \mid \in \Phi_{j_h}$.

Example 4. *Let G be a cooperative boolean game defined by $A = \{1, 2, 3, 4\}$, $\Phi = \{a, b, c, d, e\}$, $cost(a) = cost(b) = cost(c) = cost(d) = cost(e) = 1$, $\gamma_1 = a$, $\gamma_2 = c \wedge e$, $\gamma_3 = b \wedge c$, $\gamma_4 = d$, $\Phi_1 = \{b, e\}$, $\Phi_2 = \{d\}$, $\Phi_3 = \{a\}$, $\Phi_4 = \{c\}$. The associated refined dependence network $RDN_G = \langle A, \Phi, \Delta, E, \Phi_1, ..., \Phi_4 \rangle$, where E is composed by the following dependencies: $\{(1, 3, a)\}$, $\{(3, 1, b), (3, 4, c)\}$, $\{(2, 1, e), (2, 4, c)\}$, $\{(4, 2, d)\}$.*

Given a Refined Dependence Network *RDN*$(G) = \langle N, \Phi, \Delta, E, \Phi_1, ..., \Phi_n \rangle$, we mean with $R_E \subseteq N \times N$ the binary relation such that $(i, j) \in R_E$ just in the case there exists an AND-arc $e \in E$ that starts from i and reaches j, i.e. for some literal l, $(i, j, l) \in e$. It is easy to see that *ADN*$(G) = \langle N, R_E \rangle$ and hence *RDN*(G) describes G at a lower level of abstraction with respect to *ADN*(G).

We want to use Refined Dependence Networks to impose some constraints to the set *core*(G). To this scope some preliminary results are needed. A boolean variable $a \in \Phi_i$ is said to be *unfavorable* if and only if $a \in \Delta$, i.e. $cost(a) > 0$, and for each $\{l_1, \ldots, l_m\} \in \gamma_i$, $a \notin \{l_1, \ldots, l_m\}$. In the following we denote by $[i]^-$ the set of unfavorable variables of the agent i.

Proposition 4.
Given a cooperative boolean game G and an agent $i \in A$, for each $a \in [i]^-$, $\xi \in core(G)$ implies $a \notin \xi$.

Proof 2. *$a \in [i]^-$ means that $cost(a) > 0$ and a does not occur (positive) in γ_i. Assume that $\xi \models \gamma_i$, then, for some $\{l_1, \ldots, l_m\} \in \gamma_i$, $\xi \models \{l_1, \ldots, l_m\}$. Since $a \notin \{l_1, \ldots, l_m\}$, this means that also $\xi \setminus \{a\} \models \{l_1, \ldots, l_m\}$.*

Now it remains to show that if for each ξ, $\xi \models \gamma_i$ implies $\xi \setminus \{a\} \models \gamma_i$, then $\xi \in core(G)$ implies $a \notin \xi$. Assume that $a \in \xi$, clearly $\xi \setminus \{a\} = \xi$ mod $\{i\}$ as $a \in \Phi_i$. Furthermore, as $cost_i(\xi \setminus \{a\}) < cost_i(\xi)$ and by hypothesis $\xi \models \gamma_i$ implies $\xi \setminus \{a\} \models \gamma_i$, then $\xi \prec_i \xi \setminus \{a\}$. But this means that ξ is blocked by i through $\xi \setminus \{a\}$.

Note that, according to Proposition 4, it can be easily seen in Example 4 that, as all the variables are unfavorable, the core can contain only the empty strategy. We also prove that a goal depending on an unfavorable

variable a can be reduced into one that does not depend on a without affecting the possible solutions. More precisely, we define the notion of reduction as follows.

Definition 9 (Δ-reduction).
Given a cooperative boolean game G and an unfavorable variable $a \in [i]^-$, we say that the cooperative boolean game G' is a Δ-reduction of G just in the case it is obtained from G applying the following steps:

1. *remove a from Φ_i;*
2. *remove from each γ_i any conjunction of type $\{l_1, \ldots, a, \ldots, l_n\}$;*
3. *replace in each γ_i any conjunction of type $\{l_1, \ldots, \neg a, \ldots, l_n\}$ with $\{l_1, \ldots, l_n\}$.*

Proposition 5.
Let G' be a Δ-reduction of a cooperative boolean game G, $core(G) \subseteq core(G')$.

Proof 3. *For each valuation ξ that does not contain the unfavorable variable a, it clearly holds for each agent i that $u_i(\xi)$ in G is equal to $u_i(\xi)$ in G'. Therefore, if in G', ξ'_1 is blocked by C through ξ'_2, then the same holds in G and hence $core(G) \subseteq core(G')$.*

Note that the converse does not hold. Consider Example 5:

Example 5. *Let G be a cooperative boolean game composed by 3 agents and such that $\Phi_1 = \{a\}$, $\Phi_2 = \{b, c\}$ and $\Phi_3 = \{d\}$, $\gamma_1 = b$, $\gamma_2 = a \vee (c \wedge d)$ and $\gamma_3 = c \wedge b$, $cost(a) = cost(b) = 1$ and $cost(c) = cost(d) = 2$. The boolean variable a is an unfavorable variable then the Δ-reduced game G' is such that $\Phi_1 = \emptyset$, $\Phi_2 = \{b, c\}$ and $\Phi_3 = \{d\}$, $\gamma_1 = b$, $\gamma_2 = c \wedge d$ and $\gamma_3 = c \wedge b$. The function cost is the same as in G. It is easy to see that $\{c, d\} \in core(G')$ whereas in G it is blocked by $\{1, 2\}$ through $\{a, b\}$.*

Note that the previous results do not use *quantitative* values of the cost function but only the fact that an action has a strictly positive value, therefore they reside in the level of abstraction of RDNs. We can now define a procedure on $RDN(G)$ which uses unfavorable variables and Δ-reductions to reduce the search space in finding the core.

Definition 10 (Reduction rule).
Let $RDN(G)$ be a Refined Dependence Network and let ω denoting a valuation initially set to $\emptyset$, the reduction rule $RDN(G)$ is given by applying exhaustively the following rule:

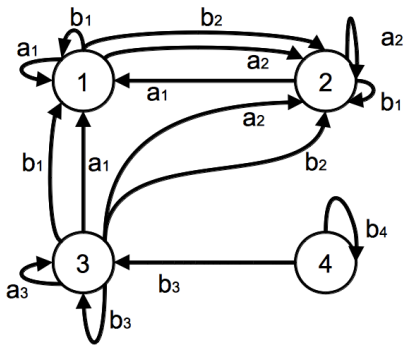

Figure 2: Refined Dependence Network of Example 6.

Condition: *for some $a \in \Phi_i \cap \Delta$, there does not exist an AND-arc e outgoing from i such that $(i, i, a) \in e$ (i.e. a is unfavorable).*

Action: *remove any AND-arc e' such that $(j, i, a) \in e'$, a from Φ_i and add a to ω.*

Due to Propositions 4 and 5, the valuation ω we obtain from Definition 10 constraints the strategies in $core(G)$ to be also in $\Phi \setminus \omega$.

Example 6 (Continue). *Let us consider again the cooperative boolean game of Example 2. In the corresponding RDN all the actions involve some positive cost, thus $\Delta = \Phi$. The AND-arcs are shown in Figure 2 (without connections for simplicity of the figure) and all the edges are labeled with the boolean variable they represent. By exhaustively applying the rule in Definition 10, b_2 satisfies the given condition, therefore we remove the AND-arc $\{(1, 2, b_2), (1, 1, b_1)\}$ and the AND-arc $\{(3, 3, b_3), (3, 2, b_2), (3, 1, \neg b_1)\}$. After these deletions, also the boolean variables b_1 and b_3 satisfy the condition in Definition 10, and hence also the AND-arc $\{(4, 3, b_3), (4, 4, b_4)\}$ has to be removed. Finally, also b_4 satisfies the condition, therefore as output ω is equal to $\neg b_1 \wedge \neg b_2 \wedge \neg b_3 \wedge \neg b_4 \wedge \neg a_4$, therefore the only strategies that can be in the core are the subsets of $\{a_1, a_2\}$. The resulting RDN after the application of the Δ-reduction is depicted in Figure 3.*

3.3. Discussion

In this section we discuss additional insights from our study, which on the one hand may be seen as a kind of criticism to the concept of core, but

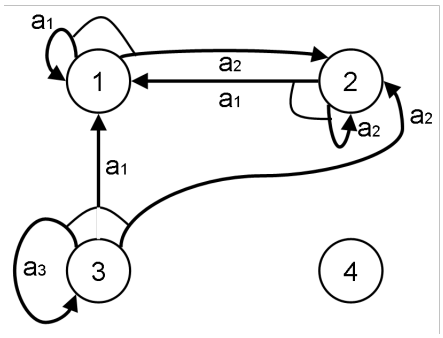

Figure 3: Refined dependence network of Example 6 after the Δ-reduction.

on the other hand also shows how theorems can be used to deal with these problems.

We have shown so far, and other examples can be found in Dunne et al. [1], that in some cooperative boolean games the core may not exist. In such cases it could be useful to identify the reason why the core does not exist, to obtain at least some partial answers. For example, consider the game given by the union of two unrelated games, such as Examples 1 and 2. In economics this may be seen just as a misrepresentation of the scenario, because we are joining two unrelated games but Proposition 1 tells us that the lack of a core in the sub-game of Example 1 leads to the lack of a core in the combined game of Examples 1 and 2, even if Example 2 has a nonempty core.

On the positive side, an analysis based on dependence networks also indicates a way to deal with the lack of a core. The ADN of two groups, which do not interact with each other, is composed of two disconnected sub-graphs. Therefore, it makes sense to calculate the sub-games corresponding to each disconnected component, and compute the core for these sub-games. The lack of a core in a sub-game could lead to a warning. If there is a strategy in one of the sub-games that is in the core of the sub-game, then this could be a partial solution to the game.

The notion of core used in coalitional games models solutions which involve some kind of reciprocity. However, there are also many examples where reciprocity is apparent, but the core does not identify it as a solution. For example, if you buy a book on eBay, then the transaction of sending money and receiving the book is a basic kind of reciprocity. It can be modeled by a goal of the buyer to receive the book, and the goal of the receiver to receive money, both goals make the agent dependent on the other agent, and

the cycle represents that there is a possibility to cooperate. However, due to Proposition 5 if a costly action does not play a role in the satisfaction of the owner agent that it will never do, so even simple exchanges are not admissible. In other words, according to the core, such a solution will never be done, because an agent would be better off by not sending the book (or not sending the money). Again, it may be argued that this example should not be modeled in this way. But it seems to be a natural way to model it, and it is the way it is typically modeled in dependence networks.

The solution would be to rewrite such games into new games such that the core and thus CBGs can be used in a wider set of situations. Typically, if two agents want to do a transaction, then there is a way to enforce this transaction. The game can be rewritten in a way such that both agents desire both the satisfaction of their own goal and the goal of the other agent. If the buyer and the seller both want the exchange, i.e. not only to receive the book but to receive the book together with paying for it, then the exchange will be incorporated in the core. Alternatively, it also suggests a revised notion of core where such reciprocity cycles are taken into account.

4. Reducing the test space

In the previous section, we have seen how to determinate the set of unfeasible actions ω such that the generation space becomes the powerset of $\Phi \setminus \omega$.

However, even if core-membership has to be checked just for one strategy (for example the *empty* strategy $\emptyset \subseteq \Phi \setminus \omega$), the test space still remains the set of all strategies $\xi' \subseteq \Phi$, which is exponentially large in the size of a cooperative boolean game. For this reason, in this section we study optimization techniques that reduce the test space.

Before studying new optimization techniques, it is reasonable to look for previous results that can be reused for our end. In particular, two properties shown in Dunne et al. [1] can be taken into account. Let $\mathit{contrib}(\xi)$ be the set of agents that incur some positive cost in ξ ($\mathit{contrib}(\xi) = \{i \mid \exists a \in \xi \text{ s.t. } a \in \Phi_i \text{ and } \mathit{cost}(a) > 0\}$) and $\mathit{ben}(\xi)$ the *beneficiaries* of ξ ($\mathit{ben}(\xi) = \{i \mid \xi \models \gamma_i\}$). Furthermore, we write $\xi \subset_C \xi'$ if $\xi \cap \Phi_C \subset \xi' \cap \Phi_C$ and say that ξ is C minimal for ϕ in case $\xi \models \phi$ and no $\xi' \subset_C \xi$, $\xi' \models \phi$. Then, the following properties hold:

1. $\xi \in \mathit{core}(G) \Longrightarrow \mathit{contrib}(\xi) \subseteq \mathit{ben}(\xi)$
2. $\xi \in \mathit{core}(G) \Longrightarrow \xi$ is $\mathit{contrib}(\xi)$ minimal for $\gamma_{\mathit{contrib}(\xi)}$

Each ξ in the generation space such that either $\mathit{contrib}(\xi) \not\subseteq \mathit{ben}(\xi)$ or it is not $\mathit{contrib}(\xi)$ minimal for $\gamma_{\mathit{contrib}(\xi)}$ can be discarded without checking whether it is blocked, thus, both the properties attempt to reduce the test space to the empty set. Whether they can be profitably used as optimization technique clearly depends on the computational complexity.

Since the condition $\mathit{contrib}(\xi) \not\subseteq \mathit{ben}(\xi)$ can be easily computed, the first property is a possible optimization technique (actually it has been implemented in our framework). On the contrary, with a proof that is essentially the same as Theorem 1 in Dunne et al. [1], checking that ξ is not $\mathit{contrib}(\xi)$ minimal for $\gamma_{\mathit{contrib}(\xi)}$ has the same computational complexity as checking that ξ belongs to the core (coNp-complete). Therefore, it has not been considered as a possible candidate.

Note that, since $\mathit{contrib}(\emptyset) = \emptyset$, the empty strategy cannot be discarded according to the first property for at least one strategy we still have a test space of exponential size. For this reason, we consider some other criterion to reduce the test space. Clearly, as in the case of unfeasible actions, we need a computationally easy way to select a subset of 2^{Φ} that is complete with respect to the core-membership problem.

On the one hand, when a strategy ξ is optimal for an agent i, i.e. $\xi = \mathrm{argmax}_{\bar{\xi}}\, u_i(\bar{\xi})$, no other strategy can be strictly more profitable for i, and hence i does not have any incentive to participate to a coalition D in order to block ξ. On the other hand, assume that ξ satisfies the goal γ_i, and for a given $a \in \Phi_i$, $\mathit{cost}(a) > \mathit{cost}_i(\xi)$, then i has no incentive to perform a. The following proposition expresses these two considerations, the proof is straightforward and it is left to the reader.

Proposition 6.
Given a cooperative boolean game G, a strategy ξ and an agent i. Assume that D blocks ξ through ξ'. We have that:

1. $\xi = \mathit{argmax}_{\bar{\xi}}\, u_i(\bar{\xi}) \Longrightarrow i \notin D$;
2. $\xi \models \gamma_i$, $a \in \Phi_i$ *and* $\mathit{cost}(a) > \mathit{cost}_i(\xi) \Longrightarrow a \notin \xi'$.

Proposition 6 can be used to reduce the test space as follows. If ξ is optimal for the agents $O = \{i_1, \ldots, i_m\}$, then a strategy ξ' that blocks ξ is such that $\xi \cap \Phi_j = \xi' \cap \Phi_j$, for all $j \in O$. Thus, the test space is reduced to $\bigcup_{h \notin O} \Phi_h$. Furthermore, if $\mathit{cost}(a) > \mathit{cost}_i(\xi)$, the test space can be further reduced to those strategies ξ' such that $a \notin \xi'$.

Finally, note that to decide whether a strategy ξ is optimal for an agent i does not require to check for all the other strategies $\bar{\xi}$ that $u_i(\xi) \geq u_i(\bar{\xi})$. As we are considering goals in disjunctive normal form, i.e. $\gamma_i = \{\gamma_{i_1}, \ldots, \gamma_{i_m}\}$, we just have to consider the strategies $\xi_{i_j} = \{a \in \Phi \mid a \in \gamma_{i_j}\}$. Then, the maximal value u_i^m of the $u_i(\xi_{i_j})$, with $1 \leq j \leq m$, corresponds to the maximal utility the agent i can obtain. Therefore, we simply have to check that $u_i(\xi) = u_i^m$.

5. Algorithms

We have used the results of Sections 3 and 4 to implement in Prolog a procedure find_core that, given as input a cooperative boolean game G, returns *core*(G). A procedural description of find_core is given in Figure 4. Essentially, find_core consists of two procedures, FIND_CORE and CORE_MEM.

In FIND_CORE, the ADN of G is instantiated and it is partitioned in its smallest subgraphs $\mathsf{A} = \{A_1, \ldots, A_n\}$ that are pairwise disconnected. Clearly, each A_i represents a stable coalition and no agent occurs in two distinct A_i and A_j (lines 2-3).

For each $A \in \mathsf{A}$, the projection G' of A is computed (line 6) and the core of G' is determined as follows. First, the RDN of G' is instantiated and, according to Definition 10, DELTA_RED applies the Δ-reduction by calculating the set ω of unfeasible actions. Then, for each ξ not containing actions in ω, the core-membership of ξ is decided by CORE_MEM. If $\xi \in$ *core*(G'), then it is added to CORE$'$.

According with Propositions 1 and 3 the core of G is computed by gathering any union of strategies in each *core*(G'). This is done incrementally in line 14.

In CORE_MEM, first it is checked whether *contrib*$(\xi) \notin$ *ben*(ξ), in affirmative case ξ cannot belong to the core and hence false is returned. Line 4 computes the set of agents O such that ξ is optimal. According to Proposition 6, such agents do not have any incentive in modifying their strategies. Thus, their strategies are *freezed* in ξ_O. Then, given the actions TA of the remaining agents, actions a of the beneficiary agents with *cost*$(a) >$ *cost*(ξ) are discarded – again according to Proposition 6.

After this last optimization, the core membership is *brutally* computed and a strategy blocking ξ is searched in the set of strategies $\xi_O \cup \xi'$, where ξ' is a subset of the resulting TA.

Input: A cooperative boolean game $G = \langle A, \Phi, cost, \gamma_1, ..., \gamma_n, \Phi_1, ..., \Phi_n \rangle$
Output: $core(G)$

$\mathsf{CORE} \leftarrow \{\emptyset\}$;
$ADN \leftarrow \mathsf{CONVERT\_ADN}(G)$;
$\mathsf{A} \leftarrow \mathsf{PAIRWISE\_DIS}(ADN)$;
forall $A \in \mathsf{A}$ **do**
  $\mathsf{CORE}' \leftarrow \emptyset$;
  $G' \leftarrow \mathsf{PROJECT\_CBG}(G, A)$;
  $RDN \leftarrow \mathsf{CONVERT\_RDN}(G')$;
  $\omega \leftarrow \mathsf{DELTA\_RED}(RDN)$;
  forall $\xi \subseteq \Phi[G'] \setminus \omega$ **do**
    if $\mathsf{CORE\_MEM}(\xi, G')$ **then**
      $\mathsf{CORE}' \leftarrow \mathsf{CORE}' \cup \{\xi\}$
    end
  end
  $\mathsf{CORE} \leftarrow \{\xi \mid \xi = \xi_1 \cup \xi_2$ where $\xi_1 \in \mathsf{CORE}$ and $\xi_2 \in \mathsf{CORE}'\}$
end
return CORE;

Algorithm 1: FIND_CORE

Input: A cooperative boolean game G and a strategy ξ
Output: True if $\xi \in core(G)$, false otherwise.

if $contrib(\xi) \not\subseteq ben(\xi)$ **then**
  return false;
end
$O \leftarrow \{i \mid \xi = \text{argmax}_{\bar{\xi}}\, u_i(\bar{\xi})\}$;
$\xi_O \leftarrow \xi \cap \bigcup_{i \in O} \Phi_i$;
$TA \leftarrow \bigcup_{j \notin O} \Phi_j$;
forall $j \in ben(\xi) \setminus O$ **do**
  $TA \leftarrow TA \setminus \{a \mid a \in \Phi_j$ and $cost(a) > cost_j(\xi)\}$;
end
forall $\xi' \subseteq TA$ **do**
  if $\mathsf{BLOCKED}(\xi, \xi_O \cup \xi')$ **then**
    return false;
  end
end
return true;

Algorithm 2: CORE_MEM

Figure 4: The find_core algorithm

As noted in Bonzon et al. [4], a further optimization could be in principle possible by removing some arc (i, j) in $ADN(G)$ in case all the actions of j that occur in γ_i are irrelevant. However, selecting irrelevant variables a in a formula ϕ means to check the validity of $\phi_\top \leftrightarrow \phi_\bot$, where $\phi_\top$ and $\phi_\bot$ are obtained by substituting each occurrence of a in ϕ with $\top$ and $\bot$ respectively. Thus, this would add a coNP-complete step just to deal with a few cases of *pathological* goals and without being sure that, by removing an edge, two subgraphs result disconnected. For this reason, $ADN(G)$ is instantiated from the initial goals, without determining irrelevant variables.

6. Experimental results

We have implemented a generator that enables to instantiate CBGs of different *shapes*. The generator takes as input 6 parameters: the number of agents N_A, the number of actions per agent N_{Ac}, the number of disjuncts N_D, the number of conjuncts N_C, the minimal and maximal cost values C_{min} and C_{max}. As output we obtain a CBG with N_A agents that control N_{Ac} variables each. An agent's goal takes the form $\gamma_1 \vee \cdots \vee \gamma_{N_D}$ where each γ_i consists in a conjunction $l_1 \wedge \cdots \wedge l_{N_C}$ of randomly generated literals. Finally, an integer cost value from C_{min} to C_{max} is randomly assigned to each propositional variable. In the figures below, the ordinate axis represents the run-time expressed in seconds where, for each input setting, the reported results correspond to the mean over 20 runs of find_core.

In Figure 5 (top), the number of actions per agent as well as the structure of goals remain fixed ($N_{Ac} = N_D = N_C = 2$). Also, C_{min} is fixed to 0 whereas the X axis corresponds to the number of agents N_A. Different lines correspond to different values of the maximal cost C_{max}. As can be seen, performances strongly depend on C_{max} and the framework behaves better by increasing it. This is only apparently surprising since costless actions cannot be discarded by applying Δ-reduction or in lines 7-9 of CORE_MEM. Now, as the cost of each action a is randomly chosen in the range $0, \ldots, C_{max}$, the smaller C_{max} is the higher is the probability that $cost(a) = 0$. In particular, for $C_{max} = 2$ approximatively 33% of the actions cannot be unfeasible, this rate decreases to 25% and 10% for $C_{max} = 3$ and $C_{max} = 9$, respectively. Furthermore, dealing with randomly generated CBGs, by setting $N_D = N_C = 2$ the resulting ADNs are very likely to be strongly connected and hence stable coalitions practically do not affect performances.

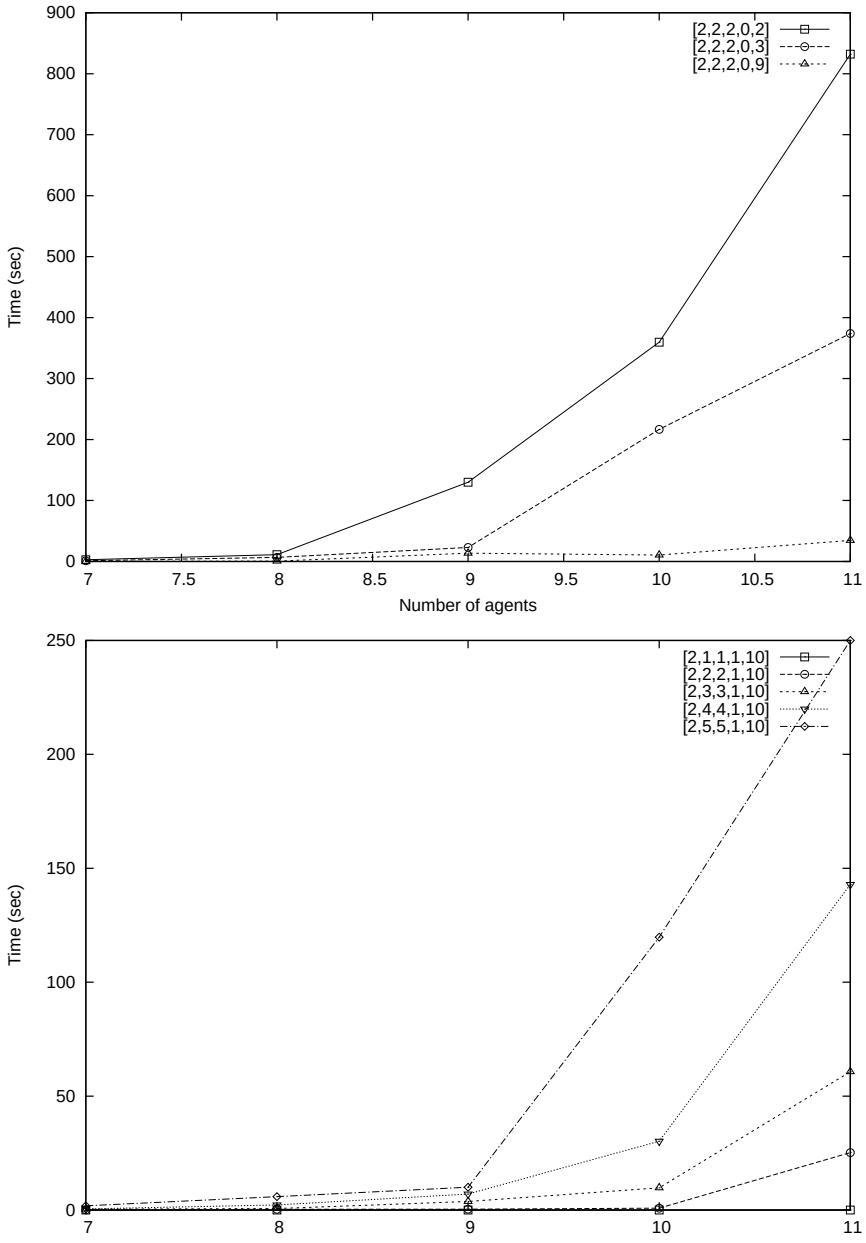

Figure 5: Experiments varying the costs of variables and the size of goals

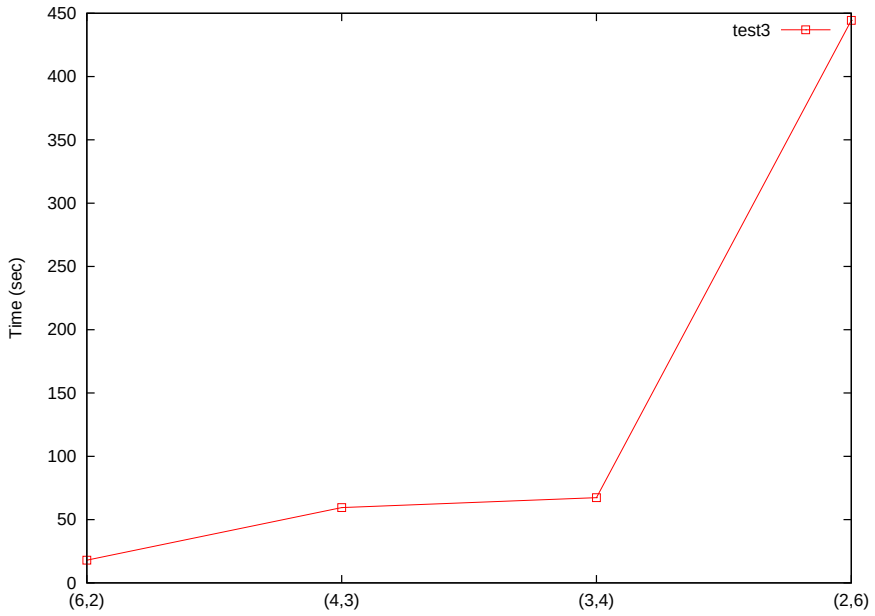

Figure 6: Varying the number of disjuncts and conjuncts in a goal

Figure 5 (bottom) shows how the framework behaves by increasing the *size* of goals. As before, $N_{Ac} = 2$ and the X axis corresponds the number of agents N_A, on the contrary the cost range is fixed to $1 - 10$ – note that all variables are costly. Each line corresponds to different goal sizes, N_D is set equal to N_C and their value varies from 1 to 5.

With goals composed by a single literal $N_D = N_C = 1$, even with 11 agents[1], find_core returns in 0.2 seconds. By comparing with the previous figure, one of the factors that improves so much performances is that the resulting ADNs present quite often two or more disconnected components and hence stable coalitions actually decompose the original game.

As said before, with $N_A = N_C = 2$ stable coalitions practically do not affect performances anymore. However, by generating random CBGs, goals are still enough *small* with respect to the number of possible literals that it is very likely that all – or all except – actions result to be unfeasible. So, in most

[1]Note that $N_A = 11$ and $N_{Ac} = 2$ mean $2^{22} \simeq 4 \cdot 10^6$ possible strategies composing initially both the generation and the test spaces.

of the cases, the find_core reduces the search space to the only empty strategy and hence basically this setting measures performances of CORE_MEM. For larger values of N_D and N_C, performances get worse. One reason is that the probability that a costly action is unfeasible decreases with the number of literals occurring in a goal. Thus, the Δ-reduction becomes less and less effective.

In Figure 5, larger goals are obtained by increasing both N_D and N_C at the same time, but which of these two values has a higher impact on performances? Figure 6 shows a set of experiments where the number of agents, actions per agents and cost range is fixed (respectively $N_A = 7$, $N_{Ac} = 3$, $C_{min} = 1$ and $C_{max} = 10$). The X axis represents different pairs of values for (N_D, N_C) such that the total number of literals composing a goal is constant, namely $(6, 2), (4, 3), (3, 4), (2, 6)$. As before, we can see a valuable variability in performances and, to better understand why large N_C jeopardizes more than large N_D, we run a version of our framework with only Δ-reduction on values $(6, 2)$ and $(2, 6)$. The run-times obtained –respectively 457 and 449 seconds– are very close to the value obtained in the original framework for $(2, 6)$. This means that the optimizations designed to decide the core membership perform better with goals composed by several disjuncts of small size that those with few disjuncts of big size.

Experimental results show somewhat clearly the applicability of the presented optimizations. On the one hand, large games with hundreds or thousands of agents, such as social networks, cannot be tackled by our approach. However, considered the Π_2^p-completeness of the problem, such games were not supposed to be affordable. Furthermore, at the best of our knowledge no other approach has already made any kind of game-theoretical reasoning scalable. On the other hand, for medium-sized systems, such as the dependency graphs defined in TROPOS, the computation of core can be computed in reasonable time. Evenmore, performances strongly improve when some simple properties are satisfied: the actions have positive and spreaded costs and the goals consist in disjuncts of small size. This means that the input games can be easily inspected to foresee the effectiveness of our technics.

7. Related Work

Boolean games are a particular kind of games for expressing compactly two-players zero-sum static games where players' utility functions are binary and described by a single boolean formula, and the strategies available to

a player consist of truth assignments to each of a given set of propositional variables controlled by the player. The classical solution concept of these games consists in the Nash equilibrium. This kind of games has been presented the first time and, then, deeply analyzed by Harrenstein et al. [2] and Harrenstein [3].

Another class of games is represented by cooperative games [8]. A cooperative game is a game where groups of players called coalitions may enforce cooperative behavior. Thus, the game can be viewed as a competition between coalitions of players, rather than between individual players. Classical solution concepts of this kind of games are the stable sets and the core. These two class of games are based on similar models: in both of them there are decision variables and they are composed by single agents' goal represented by logical formula.

Cooperative Boolean Games derive in part from non-cooperative boolean games as proposed by Harrenstein et al. [2, 3] and further developed by Bonzon et al. [4]. In non-cooperative boolean games, as in CBGs, agents have goals represented by propositional formulae, and control some set of variables, but there is no cost element, and strategic concerns arise largely from considerations about how other agents will try to satisfy their goals. CBGs are also descended from Qualitative Coalitional Games (QCGs) [9] and Coalitional Resource Games (CRGs) [10]. In a QCG, an agent's desires are represented as goals that are either satisfied or unsatisfied and coalitions have different choices available to them, where each choice allows to achieve some subset of the overall set of possible goals, and there is no cost element for achieving goals while CRGs are a generalization of QCGs in which the accomplishment of goals is assumed to require the consumption of resources of various kinds. The difference between CBGs and CRGs is that in CRGs the key issue consists in finding an efficient resource usage, and in highlighting potential conflicts between coalitions with respect to resource bounds while in CBGs these issues are not involved since no resource bound exists.

In Dunne et al. [1], they concentrate their attention on the solution concept of the core and the stable sets. On the one hand, complexity issues are investigated and, in particular, several decision problems, such as core-emptiness, are shown to be Π_2^p-complete. On the other hand, Dunne et al. [1] prove that the core of a game satisfies some general properties, but a discussion whether these properties can be used to optimize the computation of the core is missed.

Dependence networks, firstly defined by Emerson [11], have been devel-

oped in the context of multiagent systems by Conte and Sichman [12] as a kind of social network representing how each agent depends on other agents to achieve the goals she cannot achieve alone. The notion of agent dependence used to define dependence networks is related to the concept of social power, introduced in the field of Artificial Intelligence by Castelfranchi [13]. As previously said, it may be possible that two (or more) agents cannot achieve their goals independently or they can achieve their goals independently but collaborating they obtain a cost reduction. In these cases, we can represent this collaboration by means of dependencies, depicted using the dependence networks of Conte and Sichman [12]. Informally, an agent a depends on an agent b if there is at least a boolean variable under the control of b which is included in the propositional formula composing the goal of agent a. Agent b has the power to see to the goal of agent a since the one (or more) of the boolean variables composing the goal of a is under her control. Sichman [14] presents coalition formation using a dependence-based approach. This model introduces the notion of dependence situation, which allows an agent to evaluate the susceptibility of other agents to adopt her goals, since agents are not necessarily supposed to be benevolent and therefore automatically adopt the goals of each other. In this dependence-based model, coalitions can be modeled using dependence networks. Another approach using dependence networks, with the support of argumentation theory, for coalition formations has been presented by Boella et al. [15, 16].

Boella et al. [17] and Sauro [18] show how to use dependence networks to discriminate among different potential coalitions during the coalition formation process. In these works, they assume that a coalition is effectively formed only when all its members agree on it and they cannot deviate from what was established in the agreement, once they decide to enter it. They develop a criterion of admissibility called, from latin, *do-ut-des* property describing a condition of reciprocity: an agent gives a goal only if this fact enables it to obtain, directly or indirectly, the satisfaction of one of its own goals. Moreover, they define another criterion, called the indecomposable *do-ut-des* property, which strengthens the previous one. In the indecomposable do-ut-des property, differently from the *do-ut-des* property, the decomposability of a coalition in independent sub-coalitions is considered as a discriminant for the admissibility of the coalition itself. These two criteria have only a qualitative connotation and thus, they [17, 18] cannot be directly applied to the solutions developed in game theory. In this approach goals are not structured and they do not represent explicitly the costs of the actions.

The first attempts to use dependence networks to represent and simplify the computation of the solution concepts for boolean games are given by Bonzon [19] and Bonzon et al. [4]. Representing these dependencies on a graph, they show how to compute pure-strategy Nash equilibria in a simpler way, without enumerating all combinations of strategies. The notion of dependency between players and variables is used to split up a game into a set of interacting smaller games, which can be solved more or less independently. These properties do not only hold for the standard version of boolean games (with propositional goals and dichotomous preferences) but also for generalized boolean games, where players' preferences are expressed in a compact representation language. This work does not consider costly actions and dependence networks are graphs without labels on the edges representing the variables on which the agents are dependent on. While in Bonzon et al. [4] there is not a definition of coalition, in Bonzon [19] the author defines a coalition as a group of players. A coalition is called efficient if the set of players of the coalition have a common strategy allowing them to achieve their goals. The author [19] studies the properties of efficient coalitions and address computational complexity issues. In Bonzon et al. [20], they extend their previous results with a generalization to non-dichotomous preferences, where agents can not only express plain satisfaction or plain dissatisfaction, but also intermediate levels. This generalization suffices to replace the preference component of a boolean game by an input expressed in a propositional language for compact preference representation. They [20] focus on compact representation languages for ordinal preferences defining a propositional language L for compact representation for ordinal preferences, equipped with a function that maps any input of L to a preference relation. They generalize also the dependency graph between players from boolean games to L-boolean games, for an arbitrary language L.

Koller and Milch [21] introduce a new representation language for multiplayer games called multi-agent influence diagrams (MAIDs). This representation language extends the graphical models developed for probability distributions to a multiagent decision-making context. Like in dependence networks, these diagrams explicitly encode a structure involving the dependence relationships among variables. The authors define a notion of strategic relevance of one decision variable to another: a decision variable x is strategically relevant to a decision variable y when, to optimize the decision rule at y, the agent has to take into account the decision rule at x. They provide a graph-based criterion, called s-reachability, for determining strategic rele-

vance based purely on the graph structure. The paper shows how strategic relevance can be used to detect structures in the games, allowing a subdivision of a large game into a set of interacting smaller games, which can be solved in sequence. The decomposition can lead to savings in the computational cost of computing the Nash equilibria in these games. In multi-agent influence diagrams the player's utility is actually expressed in a compact way as the sum of local utilities, each corresponding to a smaller set of variables. Dependencies between players and variables in such games naturally induce a dependency relation between players, in the same way as we do such that agent i depends on agent j if agent i's utility table refers to a variable that is controlled by j. A first difference with our approach is that Koller and Milch [21] do not consider the structure of cycles if the group of nodes is not a fully connected graph while we, based on Sauro [18], show that this kind of structure provides information about the interaction between the agents. A second difference concerns the fact that [21] refer to the Nash equilibrium solution concept while we refer to the core solution concept. Finally, our approach is not based on *divide et impera* but it provides a way to prune the search space simplifying the core computation. In Vickrey and Koller [22], the authors consider structured game representations, where the interaction between the agents is sparse as in many real-world situations and they consider the relaxed task of finding an approximate equilibrium.

In Sauro et al. [23], abstract and refined dependence networks for cooperative boolean games are introduced and a first version of the Δ-reduction is presented. In this paper, we extend the results of [23] with the definition of the algorithm for the application of the Δ-reduction for computing the core and the experimental results the algorithm allows to achieve.

8. Conclusion

In this paper we present an approach to optimize the computation of the core in Cooperative Boolean Games [1] which is essentially based on dependence networks [18, 12]. The problem of computing the core of a Cooperative Boolean Game is a typical generation and test problem. In this paper, we provide optimization techniques for both the generation phase and the test one. We use two different kinds of dependence networks, abstract and refined dependence networks, for the generation phase and we extend some of the results provided by Dunne et al. [1] for the test phase.

Concerning experimental results, pros and cons of our approach can be highlighted. On the one hand, the advantage, shown by the results of Section 6, is that, also in the case of games with a number of strategies sized in an order of some millions, results are obtained in few minutes. On the other hand, the optimization techniques we propose in this paper are referred only to actions with a positive cost. Actions with no cost have not been analyzed in order to propose new optimization techniques and this is left for future research. Finally, the proposed optimization techniques, concerning the core-membership problem, return satisfactory results if they are applied to particular kinds of goals in which the disjuncts are small.

Dependency between players and variables in such games naturally introduces a dependency relation between players, in the same way as we and Bonzon et al. [4] do. For these reasons, it is interesting to perform a comparison of the computational and performance issues of our approach and these approaches. Our analysis on dependency graphs can remind the experiments on graphs which are addressed in the area of distributed constraint solving. Our dependency graphs have a shape similar to the critical phase in SAT solving. A future research line will investigate the connections between our approach to Cooperative Boolean Games using dependency graphs and the approaches developed in the area of distributed constraint solving. Finally, we can address our methodology and results to the other solution concepts, for example, instead of Strong Nash equilibrium, we could represent the core with a less restrictive notion of stability such as the Coalitional-proof Nash equilibrium. In particular, it has been shown that if a game has only one Nash Equilibrium, then it is also a Coalitional-proof Nash equilibrium.

References

[1] P. E. Dunne, W. van der Hoek, S. Kraus, M. Wooldridge, Cooperative boolean games, in: Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2008), 2008, pp. 1015–1022.

[2] P. Harrenstein, W. van der Hoek, J.-J. Meyer, C. Witteveen, Boolean games, in: Proceedings of the 8th Conference on Theoretical Aspects of Rationality and Knowledge (TARK-2001), 2001, pp. 287–298.

[3] P. Harrenstein, Logic in conflict, Ph.D. thesis, Utrecht University (2004).

[4] E. Bonzon, M.-C. Lagasquie-Schiex, J. Lang, Dependencies between players in boolean games, in: Symbolic and Quantitative Approaches to Reasoning with Uncertainty, 9th European Conference (ECSQARU 2007), 2007, pp. 743–754.

[5] H. Keiding, B. Peleg, Representation of effectivity functions in coalition proof nash equilibrium: A complete characterization, Discussion Papers 99-21, University of Copenhagen. Department of Economics (Mar. 1999).

[6] R. B. Myerson, Game Theory, Harvard University Press, 1997.

[7] J. Neumann, O. Morgenstern, Theory of Games and Economic Behaviour, Princeton Univeristy Press, 1944.

[8] T. Agotnes, W. van der Hoek, M. Wooldridge, On the logic of coalitional games, in: Proceedings of the 5th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2006), 2006, pp. 153–160.

[9] M. Wooldridge, P. E. Dunne, On the computational complexity of qualitative coalitional games, Artif. Intell. 158 (1) (2004) 27–73.

[10] M. Wooldridge, P. E. Dunne, On the computational complexity of coalitional resource games, Artif. Intell. 170 (10) (2006) 835–871.

[11] R. Emerson, Power-dependence relations, American Sociological Review 27 (1962) 31–41.

[12] J. S. Sichman, R. Conte, Multi-agent dependence by dependence graphs, in: Proceedings of the 1st International Joint Conference on Autonomous Agents & Multiagent Systems (AAMAS 2002), 2002, pp. 483–490.

[13] C. Castelfranchi, The micro-macro constitution of power, Protosociology 18 (2003) 208–269.

[14] J. S. Sichman, Depint: Dependence-based coalition formation in an open multi-agent scenario, J. Artificial Societies and Social Simulation (1998) 1 (2).

[15] G. Boella, L. van der Torre, S. Villata, Social viewpoints for arguing about coalitions, in: T. D. Bui, T. V. Ho, Q.-T. Ha (Eds.), PRIMA, Vol. 5357 of Lecture Notes in Computer Science, Springer, 2008, pp. 66–77.

[16] G. Boella, L. van der Torre, S. Villata, Analyzing cooperation in iterative social network design, J. UCS 15 (13) (2009) 2676–2700.

[17] G. Boella, L. Sauro, L. van der Torre, Strengthening admissible coalitions, in: Proceedings of the 17th European Conference on Artificial Intelligence (ECAI 2006), 2006, pp. 195–199.

[18] L. Sauro, Formalizing admissibility criteria in coalition formation among goal directed agents, Ph.D. thesis, University of Turin (2005).

[19] E. Bonzon, Modelisation des interactions entre agents retionnels: les jeux booleens, Ph.D. thesis, Universite Paul Sabatier, Toulouse (2007).

[20] E. Bonzon, M.-C. Lagasquie-Schiex, J. Lang, Dependencies between players in boolean games, Int. J. Approx. Reasoning 50 (6) (2009) 899–914.

[21] D. Koller, B. Milch, Multi-agent influence diagrams for representing and solving games, Games and Economic Behavior 45 (1) (2003) 181–221.

[22] D. Vickrey, D. Koller, Multi-agent algorithms for solving graphical games, in: Proceedings of the Eighteenth National Conference on Artificial Intelligence (AAAI 2002), 2002, pp. 345–351.

[23] L. Sauro, L. van der Torre, S. Villata, Dependency in cooperative boolean games, in: A. Håkansson, N. T. Nguyen, R. L. Hartung, R. J. Howlett, L. C. Jain (Eds.), KES-AMSTA, Vol. 5559 of Lecture Notes in Computer Science, Springer, 2009, pp. 1–10.