Optimizing the Core Computation with Social Networks

Luigi Sauro[1] and **Serena Villata**[2]

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 this work we investigate two different types of social 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 that show how they effectively improve the computation of the core.

1 Introduction

Cooperative Boolean Games [5] are a new particular family of boolean games [7, 6]. In such type of games, agent's primary aim is to achieve its 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 its actions. As in typical coalitional games, an agent can have the necessity to cooperate with the others because it does not have sufficient control to ensure its 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: 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 improves 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 social 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 graphs 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 social dependence networks, representing two different levels of abstraction of a cooperative boolean game. Abstract dependence networks have already be used by Bonzon et al. [2] to define stable coalitions which enable to divide the entire problem in subproblems and then combine their solutions. While Bonzon et al. [2] 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. In Bonzon et al. [3], the authors 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.

After Castelfranchi [4], Boella et al. [1] and Sauro [9] show how to use social dependence networks to discriminate among different potential coalitions during the coalition formation process. They develop a criterion of admissibility called 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. This criteria has only a qualitative connotation, it 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.

Refined dependence networks essentially provide a graph representation of a cooperative boolean game where the numerical information about costs is abstracted and actions are simply partitioned in free and costly actions. We present a reduction, called Δ-reduction, to pass from a cooperative boolean game G to a cooperative boolean game (CBG) G', simpler to be solved because less actions can be executed. In Sauro et al. [10], the introduction of abstract and refined dependence networks is provided. We extend the results presented in [10] with the definition of the algorithm allowing the application of the Δ-reduction and the experimental results it allows to achieve.

The reminder of the paper is as follows. In section 2 we formally define cooperative boolean games. Section 3 shows the optimization techniques relative to the core-membership problem. Sections 4 and 5 present the algorithms and the experimental results. Conclusions end the paper.

2 Cooperative Boolean games

A Cooperative Boolean Game (CBG) [5] 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) typi-

[1] Dipartimento di Scienze Fisiche, University of Naples
[2] Dipartimento di Informatica, University of Turin

cally incurs some cost, Dunne et al. [5] define the cost function $cost$ in such a way that $cost(p)$ denotes the cost of performing action p (i.e., setting p to $\top$). Agents' secondary goal is to minimize its costs. Summing up briefly, if the only way an agent can achieve its goal is by making all its variables true, then the agent prefers to to this rather than not achieve its goal but if there are different ways to achieve it, then the agent prefers always those that minimize costs.

Definition 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 $\Phi \rightarrow \Re_+$, $\gamma_1, ..., \gamma_n$ 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. [5], $cost_i(\xi)$ 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. [5], we define an utility function that is always positive if the valuation ξ satisfies the goal of the agent i and otherwise it is always negative. However, this is not an utility function in the classical way as discussed by von Neumann and Morgenstern [8] such as, for example, a function $u : X \rightarrow \mathbf{R}$ from choices to the real numbers assigning a real number to every outcome in a way that captures the agent's preferences over both simple and compound lotteries. In [5], 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: $\xi_1 \succeq_i \xi_2$ iff $u_i(\xi_1) \geq u_i(\xi_2)$. The best utility for an agent is reached if it has its goal satisfied without performing any action (utility of $\mu + 1$) while the worst utility for an agent is reached in the case it does not get its goal satisfied but it performs actions such as to set all its variables true (utility $-cost_i(\Phi_i)$).

Typically, an agent has to cooperate with the other agents because it does not have sufficient control to ensure its goal is satisfied. However, aiming at minimizing its 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 just 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 $\xi_1 = \xi_2 \; mod \; D$.

D blocks ξ_1 through ξ_2 since ξ_2 could do better than ξ_1 only by flipping the value of some of the variables under the control of D. From Dunne et al. [5], the definition of blocked valuation is as follows:

Definition 2 (Blocked Valuation)
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: $\forall i \in D : \xi_2 \succ_i \xi_1$.*

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.

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 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 have a cardinality equal to $2^{|\Phi|}$.

3 Reducing the generation space

In Sauro et al. [10], we introduce two optimization techniques that enable to reduce the generation space. Both of them are based on the notion of social dependence network that can be defined starting from a CBG. These social 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 social dependence network, we named Abstract Dependence Networks, is the same already used in Bonzon et al. [2] as decomposition method for the Nash Equilibrium in Boolean Games without costly actions. In Sauro et al. [10] we essentially extend the results in Bonzon et al. [2] to the framework defined in Dunne et al. [5].

As in Bonzon et al. [2], 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 CBG $G = \langle A, \Phi, cost, \gamma_1, ..., \gamma_n, \Phi_1, ..., \Phi_n \rangle$*, 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* $(i, j) \in R$ *iff* $j \in RA_G(i)$*.*

As in Bonzon et al. [2] 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 iff* $R(C) \subseteq C$*, i.e. for all* $i \in C$*, for all* j *such that* $(i, j) \in R$ $j \in C$*.*

Definition 6 (ADN Projection)
Let $G = \langle A, \Phi, cost, \gamma_1, ..., \gamma_n, \Phi_1, ..., \Phi_n \rangle$ *be a CBG,* $ADN(G) = \langle N, R \rangle$ *the corresponding abstract dependence graph and* $C = \{i_1, \ldots, i_m\} \subseteq N$ *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 the set of boolean variable of C,* Φ_C*.*

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

By using stable sets, 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 called Refined Dependence Networks (RDNs). These social 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.

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 a AND-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 costly variables* [3]*,* $E \subseteq N \times 2^{(N \times Litt(\Phi))}$ *where* $Litt(\Phi) = \Phi \cup \{\neg p | p \in \Phi\}$ *and* $\Phi_i \subseteq \Phi$ *where* n *is the cardinality of* N*.*

Given a literal l, we denote by $|l|$ the corresponding boolean variable. If $l \in \Phi$, then $| l |= l$ whereas if $l = \neg p$ then $| l |= p$. 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.

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 [11]. 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.

[3] The set of variables with an associated cost.

Definition 8 (From CBGs to RDNs)
Given the CBG $G = \langle A, \Phi, c, \gamma_1, ..., \gamma_n, \Phi_1, ..., \Phi_n \rangle$*,* $RDN(G) = \langle N, \Phi, \Delta, E, \Phi_1, ..., \Phi_n \rangle$ *is such that* $N = A$*,* $\Delta = \{p \in \phi \mid c(p) > 0\}$ *and* $\{(i, j_1, l_1), \ldots, (i, j_m, l_m)\} \in E$ *iff* $\{l_1, \ldots, l_m\} \in \gamma_i$ *and for all* $1 \leq h \leq m$*,* $| l_h | \in \Phi_{j_h}$*.*

Example 1 *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* $RDN_G = \langle A, \Phi, E, \Phi_1, ..., \Phi_n \rangle$*, 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 1 *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 1 $a \in [i]^-$ *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 1, if all the variables are unfavorable, the core can contain only the empty strategy. We 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 *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* $\{l_1, \ldots, l_n\}$*.*

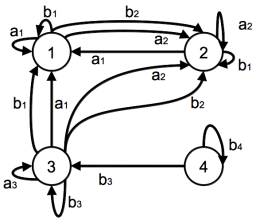

Figure 1. Refined Dependence Network of Example 1.

Proposition 2 *Let G' be a Δ-reduction of a CBG G, core$(G) \subseteq$ core(G').*

Proof 2 *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').*

The converse does not hold. 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:*

Condition: *for some $a \in \Phi_i \cap \Delta$, there does not exist an AND-arc e outgoing from i such that $(i, j, a) \in e$ (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 1 and 2, the valuation ω we obtain from definition 10 constraints the strategies in *core*(G) to be also in $\Phi \setminus \omega$. For more details and graphical examples about ADNs and RDNs, see [10].

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 CBG. 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. [5] can be taken into account. Let $contrib(\xi)$ be the set of agents that incur some positive cost in ξ ($contrib(\xi) = \{i \mid \exists a \in \xi$ s.t. $a \in \Phi_i$ and *cost*$(a) > 0\}$) and $ben(\xi)$ the *beneficiaries* of ξ ($ben(\xi) = \{i \mid \xi \models \gamma_i\}$). 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. $\xi \in$ *core*$(G) \Longrightarrow contrib(\xi) \subseteq ben(\xi)$
2. $\xi \in$ *core*$(G) \Longrightarrow \xi$ is $contrib(\xi)$ minimal for $\gamma_{contrib(\xi)}$

Each ξ in the generation space such that either $contrib(\xi) \not\subseteq ben(\xi)$ or it is not $contrib(\xi)$ minimal for $\gamma_{contrib(\xi)}$ 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 $contrib(\xi) \not\subseteq ben(\xi)$ 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. [5], checking that ξ is not $contrib(\xi)$ minimal for $\gamma_{contrib(\xi)}$ 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 $contrib(\emptyset) = \emptyset$, 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. $\xi = \text{argmax}_{\bar{\xi}}\, u_i(\bar{\xi})$, 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 $a \in \Phi_i$, *cost*$(a) >$ *cost*$_i(\xi)$, then i has no incentive to perform a. The following theorem expresses these two considerations, the proof is straightforward and it is left to the reader.

Proposition 3 *Given a CBG G, a strategy ξ and an agent i. Assume that D blocks ξ through ξ'. We have that:*

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

Proposition 3 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 *cost*$(a) >$ *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 it is not required to check for all the other strategies $\bar{\xi}$ with $u_i(\xi) \geq u_i(\bar{\xi})$. 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 utility $u_i(\xi_{i_j})$, with $1 \leq j \leq m$, corresponds to the maximal utility agent i can obtain. Therefore, we simply have to check that $u_i(\xi) = u_i^m$.

5 Algorithms

We use the results in Sections 3 and 4 to implement in Prolog a procedure find_core that, given as input a CBG G, returns *core*(G). A procedural description of find_core is given in Algorithm 1. 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 calculates 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 $\xi \in \mathit{core}(G')$, then it is added to CORE′.

The core of G is computed by gathering any union of strategies in each $\mathit{core}(G')$. This is done incrementally in line 14. In CORE_MEM, first it is checked whether $\mathit{contrib}(\xi) \subseteq \mathit{ben}(\xi)$, in affirmative case ξ cannot belong to the core and hence false is returned. CORE_MEM computes in line 4 the set O of agents such that ξ is optimal. According to Proposition 3, 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 $\mathit{cost}(a) > \mathit{cost}(\xi)$ are discarded – again according to Proposition 3.

Input: A cooperative boolean game
$G = \langle A, \Phi, \mathit{cost}, \gamma_1, ..., \gamma_n, \Phi_1, ..., \Phi_n \rangle$
Output: $\mathit{core}(G)$
1 CORE ← {∅};
2 *ADN* ← CONVERT_ADN(G);
3 A ← PAIRWISE_DIS(*ADN*);
4 **forall** $A \in$ A **do**
5 CORE′ ← ∅;
6 G' ← PROJECT_CBG(G, A);
7 *RDN* ← CONVERT_RDN(G');
8 ω ← DELTA_RED(*RDN*);
9 **forall** $\xi \subseteq \Phi[G'] \setminus \omega$ **do**
10 **if** CORE_MEM(ξ, G') **then**
11 CORE′ ← CORE′ ∪ {ξ}
12 **end**
13 **end**
14 CORE ← $\{\xi \mid \xi = \xi_1 \cup \xi_2$ where $\xi_1 \in$ CORE and $\xi_2 \in$ CORE′$\}$
15 **end**
16 **return** CORE;

Algorithm 1: FIND_CORE

Input: A cooperative boolean game G and a strategy ξ
Output: True if $\xi \in \mathit{core}(G)$, false otherwise.
1 **if** $\mathit{contrib}(\xi) \not\subseteq \mathit{ben}(\xi)$ **then**
2 **return** false;
3 **end**
4 $O \leftarrow \{i \mid \xi = \text{argmax}_{\bar{\xi}}\, u_i(\bar{\xi})\}$;
5 $\xi_O \leftarrow \xi \cap \bigcup_{i \in O} \Phi_i$;
6 $TA \leftarrow \bigcup_{j \notin O} \Phi_j$;
7 **forall** $j \in \mathit{ben}(\xi) \setminus O$ **do**
8 $TA \leftarrow TA \setminus \{a \mid a \in \Phi_j$ and $\mathit{cost}(a) > \mathit{cost}_j(\xi)\}$;
9 **end**
10 **forall** $\xi' \subseteq TA$ **do**
11 **if** BLOCKED($\xi, \xi_O \cup \xi'$) **then**
12 **return** false;
13 **end**
14 **end**
15 **return** true;

Algorithm 2: CORE_MEM

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.

As noted in Bonzon et al. [2], a further optimization could be in principle possible by removing some arc (i, j) in $\mathit{ADN}(G)$ 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 $\mathit{ADN}(G)$ is instantiated from the initial goals, without determining irrelevant variables.

6 Experimental results

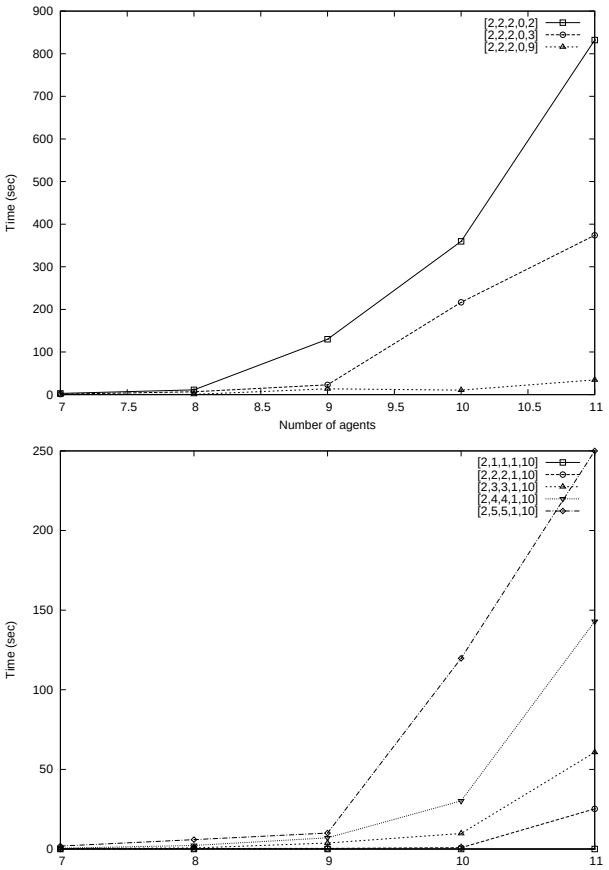

Figure 2. Experiments varying the costs of variables and the size of goals

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, 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 the figures the following tuple is used for the experiments $[N_{Ac}, N_D, N_C, C_{min}, C_{max}]$.

In Figure 2.a, 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 2.b 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[4], 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.

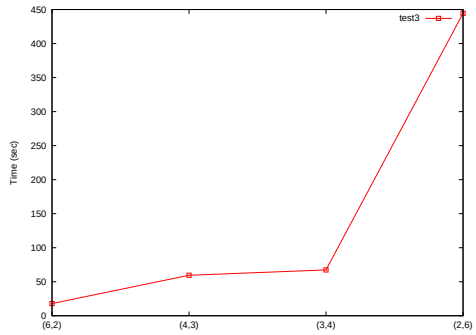

Figure 3. Varying the number of disjuncts and conjuncts in a goal

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 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 2, bigger 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 3 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.

[4] 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.

7 Conclusion

In this paper we present an approach to optimize the computation of the core in cooperative boolean games [5] which is essentially based on dependence networks [9, 11]. The problem of computing the core of a cooperative boolean game G is a typical generation and test problem. We provide optimization techniques using two kinds of social dependence networks for the generation phase and extending some the results provided by Dunne et al. [5] for the test phase.

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.

REFERENCES

[1] G. Boella, L. Sauro, and L. van der Torre, 'Strengthening admissible coalitions', in *17th European Conference on Artificial Intelligence, ECAI*, pp. 195–199, (2006).

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

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

[4] C. Castelfranchi, 'Social power. a point missed in multi-agent, dai and hci.', in *Decentralized AI - Proceedings of MAAMAW'90*. Elsevier Science Publishers, (1990).

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

[6] P. Harrenstein, *Logic in Conflict*, Ph.D. dissertation, Utrecht University, 2004.

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

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

[9] L. Sauro, *Formalizing admissibility criteria in coalition formation among goal directed agents*, Ph.D. dissertation, University of Turin, 2005.

[10] L. Sauro, L. van der Torre, and S. Villata, 'Dependency in cooperative boolean games', in *Agent and Multi-Agent Systems: Technologies and Applications, 3rd KES International Symposium, KES-AMSTA*, volume 5559 of *Lecture Notes in Computer Science*, pp. 1–10, (2009).

[11] J. Sichman and R. Conte, 'Multi-agent dependence by dependence graphs', in *1st International Joint Conference on Autonomous Agents & Multiagent Systems, AAMAS*, pp. 483–490, (2002).