Constraint Games Revisited

Anthony Palmieri, Arnaud Lallouet
Huawei Technologies Ltd, French Research Center
Greyc, University of Caen - Normandie
{anthony.palmieri,arnaud.lallouet}@huawei.com

Abstract

Constraint Games are a recent framework proposed to model and solve static games where Constraint Programming is used to express players preferences. In this paper, we rethink their solving technique in terms of constraint propagation by considering players preferences as global constraints. It yields not only a more elegant but also a more efficient framework. Our new complete solver is faster than previous state-of-the-art and is able to find all pure Nash equilibria for some problems with 200 players. We also show that performances can greatly be improved for graphical games, allowing some games with 2000 players to be solved.

1 Introduction

Game theory [Fudenberg and Tirole, 1991] is a highly successful paradigm for modeling interactions and decision in presence of multiple agents. In the classical model [Von Neumann and Morgenstern, 1944; Nash, 1951], all agents, called *players*, play an action simultaneously and are given a reward, or *utility*, which depends also on the actions of the other players. The decision problem consists in finding which action each agent should play. Since each agent is able to control his action, he would rather choose the one which gives him a maximal reward knowing the actions taken by the other players. Whenever all players are unable to improve unilaterally their reward by changing their own action, the situation is called a *Nash equilibrium*, which is the best known solution concept for this decision problem.

This model with actions and utility is classically represented by an utility matrix for each player. Unfortunately, the size of these matrices grows exponentially with the number of players, which makes the representation tractable only for small games. This has motivated the introduction of compact representations like Graphical Games [Kearns *et al.*, 2001], Boolean Games [Harrenstein *et al.*, 2001], Action-Graph Games [Jiang *et al.*, 2011] or the more recent *Constraint Games* [Nguyen *et al.*, 2013; Nguyen and Lallouet, 2014] where action space and utilities are described using Constraint Programming. A related scheme is the one of Asynchronous DCOP [Grinshpoun *et al.*, 2013; Wahbi and Brown, 2016] in which all players share the same constraints but with different costs.

In Constraint Games, user preferences are represented by a Constraint Satisfaction Problem (CSP) or a Constraint Optimization Problem (COP) whose respective satisfaction and maximization define preferred situations. Constraint Games are able to model easily a wide variety of games, from classical games of the Gamut suite [Nudelman *et al.*, 2004] to games in more specialized fields like network or scheduling games [Nguyen and Lallouet, 2014]. An algorithm called *Conga* (and hereinafter called *Conga 1.0*) based on tree-search is used to explore the space of joint players actions and proposes two improvements over naive search. The first one is a memoization of previously encountered best responses in order to eliminate duplicate computations. The second one is a lazy pruning of inconsistent solutions obtained by comparing the theoretical and actual counting of deviations in a node's subspace and backtracking when all deviations have been evaluated. It improves over the previous state-of-the-art solver Gambit [McKelvey *et al.*, 2016]. However, although effective and theoretically sound, this pruning only works well in practice at the very last levels of the search tree (level n or sometimes $n-1$ if n is the number of variables), opening the way to further improvements.

In this paper, we propose to consider the player's preference as a global constraint and we provide a complete understanding of its filtering. We prove first that the complete filtering up to arc-consistency is intractable and we further propose for it an approximation scheme. This new vision no longer relies on an ad-hoc algorithm but instead allows to fully reuse the framework of Constraint Programming. We have implemented a new solver called *Conga 2.0* to demonstrate the efficiency of this improved filtering, showing a great improvements over Conga on a selection of games.

In addition, the handling of graphical games in our new framework is straightforward. By giving the dependency graph, we are able to limit the scope of the global preference constraints to the variables it actually depends on. It allows to handle games with a high number of players very efficiently. Since the global constraint is triggered by the player's objective, the solver is also able to take advantage of some hidden dynamic dependencies. It happens in particular in disjunctive cases when we are in a situation where a subset of the dependencies are enough to assign the objective.

2 Background

For a family of sets $E = (E_i)_{i \in I}$ and a subset $J \subseteq I$ of indices, we denote by $E|_J = (E_j)_{j \in J}$ the projection of E on J and by E^J the Cartesian product $\prod_{j \in J} E_j$.

2.1 Game Theory

In this paper, we are only interested in static games of complete and perfect information.

Definition 1 (Game)**.**
A game *is a 3-tuple $G = (\mathcal{P}, A, u)$ where:*

- *$\mathcal{P}$ is a finite set of players.*
- *$A = (A_i)_{i \in \mathcal{P}}$ where $A_i \neq \emptyset$ is the set of actions that can be played by player i. We call* strategy *the choice of an action by Player i and* strategy profile *a tuple $s = (s_i)_{i \in \mathcal{P}}$ where $s_i \in A_i$. The set of strategy profiles is denoted by $A^{\mathcal{P}}$.*
- *$u = (u_i)_{i \in \mathcal{P}}$ where $u_i : A^{\mathcal{P}} \to \mathbb{R}$ is the utility function of player i.*

We denote by s_{-i} a strategy profile for all players but i, and by $(s_i, s_{-i}) = s$ the strategy profile obtained by concatenation of s_i and s_{-i}. It is implicitely assumed that all players want to maximize their own utility. The standard representation of the utility function of a player is a n-dimensional matrix representing the value associated to each strategy profiles of the game.

Example 1 (Wolf, Lamb and Cabbage Game (WLC))**.**
Three agents, Wolf (W), Lamb (L) and Cabbage (C) receive an invitation for a party. Each of them has the choice to come or not at this event. Each agent has his own preferences about meeting the others participants. Wolf would be happy to see Lamb but is indifferent about Cabbage's presence. Lamb would like to see Cabbage but only if Wolf is not coming. And Cabbage is a plant and is indifferent to everything. This can be translated in numerical preferences as depicted in Figure 1. The action of P coming to the party corresponds to p and the reverse to $\overline{p}$. For instance w, l and c represent respectively the action of coming for Wolf, Lamb and Cabbage. The complete representation of the game requires $3 \times 2^3 = 24$ integers.

The basic solution concept for a static game is called *Nash equilibrium* and corresponds to a state where each player has no incentive to change his strategy assuming the other players do not change theirs. Note that various other solution concepts have been proposed in the litterature, among them the mixed Nash equilibrium [Nash, 1951] is one of the most popular. A best response for Player i corresponds to the situation where the player play his best option according to the others players (who have already fixed their strategies).

	l, c	$l, \overline{c}$	$\overline{l}, c$	$\overline{l}, \overline{c}$
w	1	1	0	0
$\overline{w}$	0	0	0	0

Wolf's payoff

	w, c	$w, \overline{c}$	$\overline{w}, c$	$\overline{w}, \overline{c}$
l	0	0	1	0
$\overline{l}$	1	1	0	0

Lamb's payoff

	w, l	$w, \overline{l}$	$\overline{w}, l$	$\overline{w}, \overline{l}$
c	0	0	0	0
$\overline{c}$	0	0	0	0

Cabbage's payoff

Figure 1: WLC game in normal form

Definition 2 (Best Response)**.**
A strategy profile s is a best response *for player i if and only if $\forall s'_i \in A_i,\ u_i(s) \geq u_i((s'_i, s_{-i}))$.*

Definition 3 (Pure Nash Equilibrium)**.**
A strategy profile s is a Pure Nash Equilibrium *(or PNE) if and only if s is a best response for all players.*

We call $\mathbb{NE}(G)$ the set of pure Nash equilibria of a game G.

Example 2 (Example 1 continued)**.**
The WLC game has 3 PNE. The first one happens when all players choose not to come to the party ($\overline{wlc}$). The two others happens when Wolf chooses to come and Lamb to skip ($w\overline{lc}$ and $w\bar{l}c$). In this case, Lamb is indifferent to the venue of Cabbage.

2.2 Constraint Games

Constraint Programming is a way to naturally represent and solve combinatorial problems. Let V be a set of variables having $D = (D_x)_{x \in V}$ as domains. A constraint $c = (W, T)$ is defined over a subset $W \subseteq V$ by a relation given by a table $T \subseteq D^W$. A Constraint Satisfaction Problem (CSP) is a triple (V, D, C) where C is a set of constraints. Its solutions $sol(C)$ are the tuples which satisfy all the constraints, i.e. $sol(C) = \{t \in D^V \mid \forall (W, T) \in C, t|_W \in T\}$. Preferred solutions can be chosen by giving an optimization condition $\min(x)$ or $\max(x)$ where $x \in V$.

Constraint Games [Nguyen and Lallouet, 2014] are a way to give games a compact representation by using Constraint Programming to represent utility functions. In a Constraint Game, the strategy of a player is represented by the joint values of the variables he owns, and utility by an optimization condition.

Definition 4 (Constraint Game)**.**
A Constraint Satisfaction Game *(or CSG) is a 4-tuple $(\mathcal{P}, V, D, G)$ where $\mathcal{P}$ is a finite set of players, V is a finite set of variables composed of a family of disjoint sets $(V_i)_{i \in \mathcal{P}}$ for each player and a set V_E of* existential *variables disjoint of all the players variables, D is defined as for CSP, and $G = (G_i)_{i \in \mathcal{P}}$ is a family of CSP on V representing the* goal *of each player.*

Note that [Nguyen and Lallouet, 2014] has introduced satisfaction and optimization variants of Constraint Games. A Constraint Optimization Game (COG) is a variant $(\mathcal{P}, V, D, G, opt)$ where $opt = (opt_i)_{i \in \mathcal{P}}$ and $\forall i \in \mathcal{P}, opt_i \in V$ is the variable whose value defines the utility function u_i of Player i. All players want to maximize their utility. Without further information, we simply call COGs Constraint Games. In addition, Constraint Games are able to represent easily *hard constraints* that define situations which are globally possible or forbidden [Rosen, 1965]. PNE and best responses can only be seeked in the satisfiable part of the hard constraints.

It is easy to prove that they provide a large increase in expressivity since it is impossible to find a matrix representation for a game with hard constraints by giving unsatisfiable profiles any numerical value for utility. However, they also change the set of equilibria: some of them may be forbidden by the hard constraints and new ones may appear if the best response that would have occured without the hard constraints is actually unsatisfiable. In this paper, we will only consider games *without* hard constraints.

A variable $x \in V_i$ is said to be *controlled* by player i. The meaning of a Constraint Game $(\mathcal{P}, V, D, G, opt)$ is a game $(\mathcal{P}, A, u)$ in which the set of actions of a player i is defined by the different assignments of his controlled variables: $A_i = D^{V_i}$. In full generality, the handling of existential variables is related to a precise definition with the *ceteris paribus* principle [Bienvenu *et al.*, 2010]. For the purpose of this paper, we consider that they are all functionnally determined by the assignment of controlled variables and thus any problem state is fully defined by such an assignment (in particular, the variables opt_i are assigned). A more general treatment of this problem is left for future work. We denote by $V_C = V \backslash V_E$ the set of controlled variables and by $A_{-i} = D^{V_C \backslash V_i}$ the set of states of all players but i. When a profile s satisfy the goal G_i of Player i, the utility u_i of this player is given by the value of the variable to be maximized: $u_i(s) = opt_i$. If a profile s does not satisfy the goal G_i, then this profile is never preferred by Player i. In this case, it is equivalent to consider that its associated utility is assigned to $u_i(s) = -\infty$.

Example 3 (Example 1 continued)**.**
We can express the WLC game by a Constraint Satisfaction Game. First we can remark that each player has only two strategies and dichotomic preferences, thus making this problem expressible with boolean games [Harrenstein et al.*, 2001; Bonzon* et al.*, 2006]. We name x_W, x_L and x_C the respective optimization variables of Wolf, Lamb and Cabbage. Then we can state using boolean algebra notations: $x_W = wl$, $x_L = \overline{w}lc + w\bar{l}$ and $x_C = 0$.*

In a Constraint Game, the notion of best response and PNE are the same as in matrix games. We recall from [Nguyen and Lallouet, 2014] that determining whether a game has a PNE in a Constraint Satisfaction Game is Σ_2^P-complete.

2.3 Game Solving

Not many algorithms have been proposed to find PNE. The Gambit solver [McKelvey *et al.*, 2016] proposes an enumeration procedure for finding PNE called *enumpure*. It simply checks all strategy profiles for being a PNE as shown in procedure *Enum* of Algorithm 2.1. The *Enum* procedure calls for each strategy profile a function *isNash* which checks for each player with the function *Deviation* whether the current strategy is a best response or not for this player. The main interest of this algorithm is that it provides a complete search which outputs all equilibria. This naive algorithm has not been improved until recently by the *Conga 1.0* algorithm [Nguyen and Lallouet, 2014] for Constraint Games. *Conga 1.0* is a tree-search algorithm which memorizes the best responses already found and uses a counter for pruning some actions that are never best responses.

Algorithm 2.1 Enum

```
1:  procedure ENUM(game: G = (P, A, u))
2:      for all s ∈ A^P do
3:          if isNash(s) then
4:              print(s)
5:  function ISNASH(strategy profile: s): boolean
6:      for all i ∈ P do
7:          if Deviation(s, i) then return false
8:      return true
9:  function DEVIATION(strategy profile: s, player: i): boolean
10:     for all s'_i ∈ A_i, s'_i ≠ s_i do
11:         if u_i(s_i, s'_{-i}) > u_i(s) then return true
12:     return false
```

3 Preferences as Global Constraints

Although more efficient than the *Enum* algorithm used in Gambit, the *Conga 1.0* algorithm can be still improved by using constraint propagation instead of an ad-hoc pruning algorithm. We propose a new solver for Constraint Games also based on a tree search where a player can own multiple variables which are instanciated separately by a regular Constraint Programming solver. Note that there is a main search tree which defines the region of the search space where an equilibrium is seeked. Like in Constraint Programming, a search state S is defined by giving each variable a current domain: $S = (S_x)_{x \in V}$ with $S_x \subseteq D_x$. We use the notations $S_i = S|_{V_i}$ and $S_{-i} = S|_{V_C \backslash V_i}$. Search consists in a series of domain reductions with an alternation of deterministic consistency steps and non-deterministic branching steps [Rossi *et al.*, 2006].

We propose to implement as constraints the preferences of the players. For this, it is useful to consider how preferences are defined from the goal of the players. Preferences have been widely studied in the litterature from the point of view of knowledge representation, especially using logic [von Wright, 1963; Bienvenu *et al.*, 2010]. Generally, it is widely accepted that a preference is a preorder $\succcurlyeq$ on a set of outcomes Ω. It does not have to be total. We have $s \succcurlyeq s'$ whenever s is preferred to s'. A utility function $\Omega \to \mathbb{R}$ as defined in games defines a natural preference where $s \succcurlyeq s' \leftrightarrow u(s) \geq u(s')$. But games also add a notion of controllability: an outcome can only be compared to a controllable one, thus making the preorder partial.

For a Constraint Game $(\mathcal{P}, V, D, G, opt)$, the set of outcomes is defined by the search space D^V, and the preference associated to a player is induced by the utility given by the value of his optimization variable. Since Player i only controls the variables V_i, we can fully describe his preference relation by giving the preferred outcomes for each uncontrollable situation defined by the other players, i.e. the set of best responses in A_i of Player i associated to each partial state of A_{-i}. This relation has been introduced in [Gottlob *et al.*, 2005] in the context of graphical games under the name of *Nash constraint*. The Nash constraint N_i of Player i is defined by $N_i = (V_C, \{(s_i, s_{-i}) \mid s_{-i} \in A_{-i} \wedge \forall s'_i \in A_i, u((s_i, s_{-i})) \geq u((s'_i, s_{-i}))\})$. Theorem 4.3 of [Gottlob *et al.*, 2005] adapted to Constraint Games states that $\mathbb{NE} = sol(\bigcup\{N_i \mid i \in \mathcal{P}\})$.

w	c	l
0	0	0
0	0	1
0	1	1
1	0	0
1	1	0

(a) Nash Constraint N_{Lamb}

w	c	l	utility
0	0	0	0
0	0	1	0
0	1	1	1
1	0	0	1
1	1	0	1

(b) Ext. Nash constraint eN_{Lamb}

Figure 2: Nash constraints for Lamb in extension

In this paper, we propose to implement players preferences using global constraints and study filtering algorithms for them. The Nash constraint N_i contains at least one entry for each partial profile in A_{-i}. This property makes it unsuitable for filtering values from the other players' variables. Moreover, its representation in extension is exponential.

Example 4 (Example 1 continued)**.**
The Nash constraint for Lamb is defined in Figure 2(a). Note that if Wolf and Cabbage are not coming, Lamb is indifferent and thus has both coming *and* not coming *as best responses.*

Since the Nash constraints N_i contains all best responses of Player i for all the uncontrollable situations defined by the other players joint strategies, any strategy for i that does not belong to the projection $N_i|_{V_i}$ is called a *never best response*. We first state an intractability result:

Proposition 1. *In a Constraint Satisfaction Game, deciding whether an assignment $nbr \in A_i$ for Player i is a never best response is Π_2^P-complete.*

Proof. Membership is immediate.

For hardness, we reduce a QCSP [Bordeaux and Monfroy, 2002] $Q = \forall X \exists Y C$ to a 2-players 0-sum CSG $G = (\{1,2\}, V_1 = X \cup \{x,a\}, V_2 = Y \cup \{y,b\}, G = (a = b) \vee (C \wedge (Y \neq nbr) \wedge (x = y))$ where $Y \neq nbr$ stands for the assignment of Y is different of the tuple nbr and x, y, a, b are new variables whose domain contains at least two elements. Since the game is 0-sum, we only need to specify the goal G for Player 1 and take Player 2's goal as the negation of G.

If Q is valid, then for all $s^X \in D^X$, there is a $s^Y \in D^Y$ such that C is true. Let v_x and v_a be the respective values of x and a set by Player 1. If $s^Y = nbr$, then Player 2 assigns $b \neq v_a$. If $s^Y \neq nbr$, then Player 2 assigns $y = v_x$. Hence nbr is a never best response.

Conversely, if Q is not valid, then it does exist a tuple $s^X \in D^X$ such that forall $s^Y \in D^Y$, C is false. If $s^Y = nbr$, then Player 2 assigns $b = v_a$. If $s^Y \neq nbr$, then Player 2 assigns $b \neq v_a$. Hence nbr is a best response. □

Arc-consistency filtering amounts to removing all values involved only in tuples which are never best responses. From Proposition 1 and the exponential representation argument, we infer that filtering the Nash constraint to arc-consistency is intractable even for dichotomic preferences. Thus we introduce three approximations in order to keep the problem tractable. We recall that a propagator for a constraint is a function that is *i)* correct, *ii)* contracting, *iii)* monotonic, and *iv)* singleton complete [Apt, 1999]. The respect of these properties ensures the correct behavior of the propagator when placed in the solver.

The first approximation consists in using only the objective value. We call *extended Nash Constraint* $eN_i = (V_C \cup \{opt_i\}, \{(s, o_i) \mid s \in N_i\})$ the Nash constraint N_i augmented with the value of the player's objective (see Figure 2(b)). Instead of removing exact inconsistent values, we will remove inconsistent objective values. The actual pruning on decision variables is done by back-propagation of arc-consistency through the constraints defining the objective. It means that we can approximate eN_i by ooN_i (objective optimal Nash constraint) defined by $ooN_i = (V_C \cup \{opt_i\}, \{(s, o_i) \mid \exists (s', o_i) \in eN_i \wedge o_i \in eN_i|_{opt_i}\})$. However, only looking at the objective value is not enough. In different regions of the seach space, two tuples s and s' may have the same objective value o_i for Player i despite s is a not a best response for s_{-i} and s' is one for s'_{-i}, hence the approximation.

Still, computing the exact objective values can be difficult because the subspace generated by the variables V_i of Player i may be huge. In our second approximation, we replace the set of exact objective values by the interval where they lie. This is defined by the constraint $mooN_i = (V_C \cup \{opt_i\}, \{(s, o_i) \mid \exists (s', o'_i) \in eN_i, o_i \geq o'_i\})$. Since $sol(eN_i) \subseteq sol(ooN_i) \subseteq sol(mooN_i)$, any filtering of $mooN_i$ is correct with respect to eN_i. Note that we only need to update the lower bound because we have a maximization problem.

Since consistent objective value correspond to one best response, the minimum bound of the objective should be set to the maximum value of the minimum bound of the best responses on the current search state S:

$$Mm(S) = \max_{s_i \in A_i} \min_{s_{-i} \in S_{-i}} (s_i, s_{-i})|_{opt_i} \qquad (1)$$

Thus we need to find (or approximate) the *maximin* of the objective on the current subspace. The exact computation of this maximin on a search state S requires a traversal of the subspace S_{-i}, and for each tuple, a test for deviation in A_i. Unfortunately, a traversal of S_{-i} is too costly. The third approximation consists in using arc-consistency to reject impossible objective values.

To implement this filtering, we use an auxiliary solver with Branch & Bound on a tree-search limited to Player i's variables to maximize the minimum value of the objective on the subspace at a node n of the main search tree. In order to compute deviations for Player i, all variables of V_i are reset to their original domain A_i, *including* those which were assigned before n. The domain of the variables of the other players are given by S_{-i} (some are already assigned at node n and some are not). Whenever the minimum bound of the objective is pruned to a value b, a new constraint $opt_i > b$ is posted for the rest of the search. For a search state S, we replace in formula (1) the traversal of the search space S_{-i} by an arc-consistency check. This amounts to compute $Mm_{AC}(S)$, the best estimation arc-consistency can provide for the maximum value of the minimum bound of the objective: $Mm_{AC}(S) = \max_{s_i \in A_i} lb(AC((s_i, S_{-i})))$ where AC denotes arc-consistency applied to a search state. Formally, the operator $BB_i : D^V \rightarrow D^V$ implemented by the global

constraint is defined by:

$$BB_i(S) = \begin{cases} S_{opt_i} & = & S_{opt_i} \cap [Mm_{AC}(S)..ub(S_{opt_i})] \\ S_x & = & S_x, \forall x \neq opt_i \end{cases}$$

When all variables are assigned, computing deviations amounts to checking whether a complete assignment is a best response for Player i.

Proposition 2. *BB is a propagator for* eN_i.

Proof. Let S and S' be two search states. First, because AC is correct, we have $Mm_{AC}(S) \leq Mm(S)$. Then, if some value is pruned by Mm_{AC}, then it has to be pruned also by Mm.

- correctness: $S \cap \mathbb{NE} \subseteq BB(S)$. Take $s \in S \cap \mathbb{NE}$ and suppose $s \notin BB(S)$. Let $s|_{opt_i} = o_i$ and let $\alpha_i = Mm_{AC}(S)$. Since $s \notin BB(S)$, we have $o_i < \alpha_i$. Then there exists $s'_i \in A_i$ such that $(s'_i, s_{-i})|_{opt_i} = o'_i > o_i$. Hence we have $o_i < \alpha_i \leq o'i$ and (s'_i, s_{-i}) is a deviation for s and $s \notin \mathbb{NE}$, contradiction.
- contractness: $BB(S) \subseteq S$. Since the propagator only remove values, contractness is always true.
- monotony: $S \subseteq S' \rightarrow BB(S) \subseteq BB(S')$. Take $s \in BB(S)$, we have $s \in S$ since BB is contracting and $s \in S'$ by definition. Let $s|_{opt_i} = o_i$, $\alpha_i = Mm_{AC}(S)$ and $\beta_i = Mm_{AC}(S')$. Since $s \in BB(S)$, we have $\alpha_i \leq o_i$. Since AC is monotonic, we have $\beta_i \leq \alpha_i$. Thus $\beta_i \leq o_i$ and $s \in BB(S')$.
- singleton completeness: ensured by construction.

□

4 Experiments

We have implemented a new solver for Constraint Games on top of the constraint solver Choco v4 [Prud'homme *et al.*, 2017]. This new solver is composed of an interface to post goals for the different players and the implicit post of the new global constraints as defined in Section 3. An important aspect is that it does not require a modification of the classical Constraint Programming framework (search and consistency).

The scope of the preference constraint for Player i is the set of all controlled variables V_C plus Player i's optimization variable opt_i. However, the propagator is called whenever the objective of the player has been updated. Upon the call, it revises opt_i by launching a new search tree in a distinct solver only on the variables of V_i, having copied the current state S_{-i} of all the other players' variables. During this search, Branch & Bound is applied to the lower bound of the objective and the approximation of its maximin value on the current subspace S_{-i} is returned. For all experiments, we have applied a lexicographic heuristics on the choice of variables and a min value heuristics on the choice of the values.

We have performed experiments on classical games of the Gamut suite [Nudelman *et al.*, 2004]: Minimum Effort Game (MEG), Travelers Dilemma (TD), Dispersion Game (DG), Collaboration Game (CG), ArmRaces (AR), ElFarol Bar Game (EFBG) [Arthur, 1994] and Colonel Blotto (CB) [Roberson, 2006]. For space reasons, we invite the reader to refer to the original papers to get a full description of the games. They represent a wide range of games, with many, few or no equilibria. In all games, the parameters are given by two numbers: the number of players and the number of actions. For example, "MEG 5.5" denotes the Minimum Effort Game with 5 players, each one having 5 actions. One exception is the Colonel Blotto game for which the second number is the number of battlefields and the last one the number of troops each player can deploy.

Game	Param.	#PNE	*enum*	*Conga 1*	*Conga 2*
MEG	7.7	6	30.60	10.63	**0.10**
	8.8	7	–	169.92	**0.13**
	70.70	70	–	–	**480.08**
TD	6.6	1	5.95	2.80	**0.06**
	7.7	1	105.13	39.80	**0.07**
	300.300	1	–	–	**514.73**
DG	6.6	720	2.46	**1.53**	2.02
	7.7	5040	43.36	**21.86**	35.42
	8.8	40320	–	**443.11**	–
CG	7.7	7	32.78	11.92	**2.08**
	8.8	8	–	225.31	**27.85**
	9.9	9	–	–	**469.15**
AR	2.250	1	173.68	**0.64**	1.51
	2.2000	1	–	**457.42**	567.15
	2.2500	1	–	**596.30**	–
EFBG	16.2	12870	116.88	115.44	**87.87**
	17.2	24310	256.77	249.04	**188.80**
	18.2	48620	549.69	539.73	**414.38**
CB	2.3.32	0	474.86	450.24	**54.21**
	2.3.55	0	–	–	**564.96**
	2.4.13	0	388.37	329.25	**19.79**
	2.4.20	0	–	–	**244.75**
	2.6.6	0	278.57	240.43	**16.89**
	2.6.10	0	–	–	**599.01**

Table 1: Runtime of the different methods on Gamut games

We have compared our new approach (called *Conga 2*) to the complete enumeration of the search space (hereafter called *enum*) as implemented in the Gambit solver [McKelvey *et al.*, 2016] and to a custom reimplementation of the Conga algorithm (hereafter called *Conga 1*) as described in [Nguyen and Lallouet, 2014]. The results are presented in Table 1. All times are given in seconds and we applied a maximum runtime of 10 minutes to get the complete set of PNE for each problem. All instances for which the given solver has reached a timeout are indicated by "–". Experiments have been run on a Intel Xeon E5-1660 with 32 GB of RAM, Java 8 and Windows 7.

We can see that Conga 2.0 is most of the time better than Conga 1.0, reaching up to three orders of magnitude on MEG. Exceptions are the Dispersion Game and Arm Race for which there is little propagation. We have presented the results in order to show the limits of each technique when staying under the time limit of 10 minutes.

5 Graphical Aspects

When the utility of a player only depends on a subset of the other players, the game is called *graphical* [Kearns *et al.*, 2001]. More formally, for a profile $s \in A^{\mathcal{P}}$ and two players i and j, we say that $u_i(s)$ is independent of j if $\forall s'_j \in A_j, u_i(s) = u_i(s_{-j}, s'_j)$. If this is not true, then i depends on j for s, denoted by $i \lhd_s j$. We denote by $dep_i(s) = \{j \mid i \lhd_s j\}$ the set of dependencies of Player i for the state s. For $E \subseteq A^{\mathcal{P}}$, we note $dep_i(E) = \bigcup_{s \in E} dep_i(s)$. Finally, Player i *statically depends* on Player j, denoted by $i \lhd j$ if $j \in dep_i(A^{\mathcal{P}})$. A minimal dependency which occurs only in a strict subspace of $A^{\mathcal{P}}$ is called dynamic.

Example 5 (Example 1 continued)**.**
Since Wolf only depends on Lamb and Lamb on Wolf and Cabbage, the WLC game is actually a graphical game, as depicted in Figure 3. By using this dependency scheme, we are able to reduce the size of the matrices to one 1×2*, one* 2×2 *and one* $2 \times 2 \times 2$*, for a total amount of 14 integers. Interestingly, the WLC game owns a dynamic dependency: when Wolf is coming, Lamb does not depend on Cabbage anymore. It is reflected in the Lamb formula* $x_L = \overline{w}lc + w\bar{l}$ *for which* $\bar{l}$ *is a best response to* w *whatever the value of* c*.*

The main purpose of graphical games is to save space in the matrix representation, and also to speed-up the computation of equilibria [Vickrey and Koller, 2002; Ortiz and Kearns, 2002]. We can build the oriented graph of dependencies between player $\mathcal{D} = (\mathcal{P}, \{(i, j) \mid i \lhd j\})$. If this graph is not a clique, then the game is graphical. In this case, it is possible to limit the scope of each global constraint to its associated player and his set of dependencies. It ensures an increased performance in the checking of deviations without relying on a specialized algorithm. When checking for deviations, only the state of the dependent variables is injected in the second solver. In addition, because our global constraint is triggered on updates of the objective, it is able to detect on-the-fly dynamic dependencies. Note that this kind of dependencies may also occur in non-graphical games.

In our implementation, we provide the dependency graph as a part of the model. We have performed experiments on classical graphical games: Public Good Game (PGG), Threshold Game of Complement (TGC) [Jackson, 2008] and Road Game (RG) [Vickrey and Koller, 2002]. All games have a fixed number of strategies, which is 2 for PGG and TGC, and 4 for RG. Each model has been run on different topologies with different node degrees. In the circle topology (C), each node is of degree 2 and the only parameter is the number of players. For the tree topology (T), T 21.4.3 means that the game has 21 players connected by a tree of maximal degree 3 on 4 levels. The last topology is the complete bipartite graph (B) whose parameter is the number of players. The results are shown in Table 2, also with a timeout of 10 minutes. In all cases but one, adding the graphical information is highly beneficial. It demonstrate the efficiency of the approach even for graphs of relative high degrees.

Figure 3: Dependency graph of the WLC game.

Game	Topology	#PNE	*enum*	*Conga 1*	*Conga 2*
PGG	C.30	4610	347.60	332.02	**15.21**
	C.40	76725	–	–	**330.50**
	T.21.4.3	0	522.65	339.56	**2.51**
	T.91.9.3	512	–	–	**319.20**
	B.22	2	73.08	9.17	**0.25**
	B.27	2	–	54.16	**0.33**
	B.250	2	–	–	**485.85**
TGC	C.150	2	460.45	311.54	**1.69**
	C.170	2	–	470.89	**2.21**
	C.2000	2	–	–	**479.91**
	T.57.7.3	2	538.98	80.82	**1.04**
	T.1555.7.5	2	–	–	**182.89**
	B.22	2	131.68	15.48	**2.31**
	B.27	2	–	91.50	**10.23**
	B.33	2	–	–	**96.97**
RG	C.10	1119	16.20	10.92	**3.45**
	C.13	9230	167.16	142.43	**97.95**
	C.16	76004	–	–	**519.14**
	T.31.5.3	244	162.42	50.18	**9.49**
	T.57.7.3	2174	–	–	**216.60**
	B.11	96	16.19	**8.90**	10.11
	B.12	3728	140.54	105.18	**63.74**
	B.15	384	–	–	**266.48**

Table 2: Runtime of graphical games on different topologies

6 Conclusion

In this paper, we have proposed a more elegant and efficient vision of Constraint Games. We have fully modeled the potential constraint pruning available in Game Theory, and proved its intractability. We have proposed an efficient filtering algorithm in the general and graphical cases and demonstrated experimentally its efficiency over the state-of-the-art game solver Conga 1.0 [Nguyen and Lallouet, 2014]. However, the main interest of this new approach is to bring game theory closer to the elegant framework of Constraint Programming by viewing agent preferences as constraints.

So far, we limited ourselves to the design of a complete solver because only this type of solver is able to be used as a basis for computing more specialized Nash equilibria, like those optimizing a Social Welfare function or Pareto efficient. Also completeness is required to compute Price of Anarchy and Price of Stability. We did not include theses aspects for lack of space. Another open question is the extension of our filtering to games with hard constraints. As is, our filtering is incorrect in presence of hard constraints because the expected deviation may not be satisfiable. As a consequence, the solver becomes incomplete. In future work, we will also be interested in the computation of one equilibrium, emphasizing the role of a good heuristics and opening comparisons with best response dynamics [Ceppi *et al.*, 2010].

References

[Apt, 1999] Krzysztof R. Apt. The Essence of Constraint Propagation. *Theor. Comput. Sci.*, 221(1-2):179–210, 1999.

[Arthur, 1994] Brian W. Arthur. Complexity in Economic Theory. Inductive Reasoning and Bounded Rationality. *American Economic Review*, 82(2):406–411, 1994.

[Bienvenu *et al.*, 2010] Meghyn Bienvenu, Jérôme Lang, and Nic Wilson. From Preference Logics to Preference Languages, and Back. In F. Lin, U. Sattler, and M. Truszczynski, editors, *KR 2010, Toronto, Ontario, Canada, May 9-13, 2010*. AAAI Press, 2010.

[Bonzon *et al.*, 2006] Elise Bonzon, Marie-Christine Lagasquie-Schiex, Jérôme Lang, and Bruno Zanuttini. Boolean Games Revisited. In G. Brewka, S. Coradeschi, A. Perini, and P. Traverso, editors, *ECAI*, volume 141 of *Frontiers in Artificial Intelligence and Applications*, pages 265–269. IOS Press, 2006.

[Bordeaux and Monfroy, 2002] Lucas Bordeaux and Eric Monfroy. Beyond NP: Arc-Consistency for Quantified Constraints. In P. Van Hentenryck, editor, *CP 2002, Ithaca, NY, USA, September 9-13, 2002*, volume 2470 of *LNCS*, pages 371–386. Springer, 2002.

[Ceppi *et al.*, 2010] Sofia Ceppi, Nicola Gatti, Giorgio Patrini, and Marco Rocco. Local Search Methods for Finding a Nash Equilibrium in Two-Player Games. In J. Xiangji Huang, A. A. Ghorbani, M.-S. Hacid, and T. Yamaguchi, editors, *IAT*, pages 335–342. IEEE Computer Society Press, 2010.

[Fudenberg and Tirole, 1991] D. Fudenberg and J. Tirole. *Game Theory*. The MIT Press, 1991.

[Gottlob *et al.*, 2005] Georg Gottlob, Gianluigi Greco, and Francesco Scarcello. Pure Nash Equilibria: Hard and Easy Games. *J. Artif. Intell. Res. (JAIR)*, 24:357–406, 2005.

[Grinshpoun *et al.*, 2013] Tal Grinshpoun, Alon Grubshtein, Roie Zivan, Arnon Netzer, and Amnon Meisels. Asymmetric Distributed Constraint Optimization Problems. *J. Artif. Intell. Res. (JAIR)*, 47:613–647, 2013.

[Harrenstein *et al.*, 2001] Paul Harrenstein, Wiebe van der Hoek, John-Jules Ch. Meyer, and Cees Witteveen. Boolean Games. In Johan van Benthem, editor, *TARK*. Morgan Kaufmann, 2001.

[Jackson, 2008] Matthew O Jackson. *Social and Economic Networks*, volume 3. Princeton university press Princeton, 2008.

[Jiang *et al.*, 2011] Albert Xin Jiang, Kevin Leyton-Brown, and Navin A. R. Bhat. Action-Graph Games. *Games and Economic Behavior*, 71(1):141–173, 2011.

[Kearns *et al.*, 2001] Michael J. Kearns, Michael L. Littman, and Satinder P. Singh. Graphical Models for Game Theory. In Jack S. Breese and Daphne Koller, editors, *UAI*, pages 253–260. Morgan Kaufmann, 2001.

[McKelvey *et al.*, 2016] Richard D McKelvey, Andrew M McLennan, and Theodore L Turocy. *Gambit: Software Tools for Game Theory*, version 16.0.0 edition, 2016.

[Nash, 1951] J.F. Nash. Non-cooperative Games. *Annals of Mathematics*, 54(2):286–295, 1951.

[Nguyen and Lallouet, 2014] Thi-Van-Anh Nguyen and Arnaud Lallouet. A Complete Solver for Constraint Games. In Barry O'Sullivan, editor, *CP 2014, Lyon, France, September 8-12, 2014.*, volume 8656 of *LNCS*, pages 58–74. Springer, 2014.

[Nguyen *et al.*, 2013] Thi-Van-Anh Nguyen, Arnaud Lallouet, and Lucas Bordeaux. Constraint Games: Framework and Local Search Solver. In É. Grégoire and B. Mazure, editors, *ICTAI*, Special Track on SAT and CSP Technologies, pages 963–970. Springer, 2013.

[Nudelman *et al.*, 2004] Eugene Nudelman, Jennifer Wortman, Yoav Shoham, and Kevin Leyton-Brown. Run the GAMUT: A Comprehensive Approach to Evaluating Game-Theoretic Algorithms. In *AAMAS*, pages 880–887. IEEE Computer Society, 2004. `http://gamut.stanford.edu/`.

[Ortiz and Kearns, 2002] Luis E. Ortiz and Michael J. Kearns. Nash Propagation for Loopy Graphical Games. In S. Becker, S. Thrun, and K. Obermayer, editors, *NIPS 2002, December 9-14, 2002, Vancouver, British Columbia, Canada]*, pages 793–800. MIT Press, 2002.

[Prud'homme *et al.*, 2017] Charles Prud'homme, Jean-Guillaume Fages, and Xavier Lorca. *Choco Documentation*. TASC, INRIA Rennes, LINA CNRS UMR 6241, COSLING S.A.S., 2017.

[Roberson, 2006] Brian Roberson. The Colonel Blotto Game. *Economic Theory*, 29(1):1–24, 2006.

[Rosen, 1965] J. B. Rosen. Existence and Uniqueness of Equilibrium Points for Concave n-Person Games. *Econometrica*, 33(3):520–534, July 1965.

[Rossi *et al.*, 2006] Francesca Rossi, Peter van Beek, and Toby Walsh. *Handbook of Constraint Programming (Foundations of Artificial Intelligence)*. Elsevier Science Inc., New York, NY, USA, 2006.

[Vickrey and Koller, 2002] David Vickrey and Daphne Koller. Multi-Agent Algorithms for Solving Graphical Games. In Rina Dechter and Richard S. Sutton, editors, *AAAI 2002, July 28 - August 1, 2002, Edmonton, Alberta, Canada.*, pages 345–351. AAAI Press / The MIT Press, 2002.

[Von Neumann and Morgenstern, 1944] John Von Neumann and Oskar Morgenstern. *Theory of Games and Economic Behavior*. Princeton University Press, 1944.

[von Wright, 1963] Georg H. von Wright. *The Logic of Preference*. Edinburgh University Press, 1963.

[Wahbi and Brown, 2016] Mohamed Wahbi and Kenneth N. Brown. A Distributed Asynchronous Solver for Nash Equilibria in Hypergraphical Games. In G. A. Kaminka, M. Fox, P. Bouquet, E. Hüllermeier, V. Dignum, F. Dignum, and F. van Harmelen, editors, *ECAI 2016, 29 August-2 September 2016, The Hague, The Netherlands*, volume 285 of *Frontiers in Artificial Intelligence and Applications*, pages 1291–1299. IOS Press, 2016.