Towards representing Games in Constraint Programming

PhD student : Thi-Van-Anh Nguyen[1], Supervisor : Arnaud Lallouet[1],
Co-Authors : Lucas Bordeaux[2], and Youssef Hamadi[2]

[1] Université de Caen, GREYC, Campus Côte de Nacre,
Boulevard du Maréchal Juin, BP 5186, 14032 Caen Cedex, France
{thi-van-anh.nguyen,arnaud.lallouet}@unicaen.fr
[2] Microsoft Research, 7 J. J. Thomson Avenue, Cambridge CB3 0FB, United Kingdom
{lucasb,youssefh}@microsoft.com

Abstract. [3] Games theory is a highly successful paradigm for strategic decision making between multiple agents having conflicting objectives. Since a few years, games have been studied in a computational perspective, raising new issues like complexity of equilibria or succinctness of representation. Indeed, the main representation for general games is still a n-dimensional matrix of exponential size called *normal form*. In this paper, we introduce the framework of *Constraint Games* to model strategic interaction between players. A Constraint Game is composed of a set of variables shared by all the players. Among these variables, each player owns a set of *decision variables* she can control and a Constraint Optimization Problem defining her preferences. Since the preferences of a player depend on the decisions taken by the other players, each player may try to improve her position by choosing a better assignment. Pure Nash equilibria are situations in which no player may improve her preferences unilaterally. Constraint Games are thus a generic tool to model general games and can be exponentially more succinct than their normal form. We present here the framework and sketch the resolution of such games.

1 Introduction

The mathematical field of *game theory* [15, 12] has been set up to address problems of strategic decision making when modelling interacting agents with conflicting objectives. Game theory has an incredible success in description of economic processes, but is also used in various other domains such as biology, political sciences or philosophy. One of the most fundamental problems in computational game theory is undoubtedly the computation of a Nash equilibrium [14], which models a situation where no agent has an incentive to change her decision unilaterally. Other issues are numerous and include simultaneous or sequential interaction mode, complete or incomplete knowledge, determinism, coalitions, repetition, etc.

[3] This work is submitted to the CP 2013 doctoral program. Another version is submitted to ICTAI 2013 at the same time. This work is supported by Microsoft Research grant MRL-2011-046.

A game is composed of a set of players, each of them having a set of possible actions they can perform. A game in normal form is represented as a payoff matrix stating for each player the reward she will get for any combination of actions of all players. One important problem of game theory is the compactness of payoff representation because the matrix grows exponentially with the number of player. Surprisingly, although compactness can be achieved by switching to a combinatorial setting for payoffs, there are few attempts to define compact yet generic languages for the expression of games.
We propose *Constraint Games* which use Constraint Satisfaction Problems (CSP) as basic tool for expressing players preferences. In a constraint game, each player controls a set of finite domain variables and their Cartesian product defines a possible action space for the player. In addition, each player owns a CSP on all players variables which defines her satisfaction. Given the partial state defined by the other players moves, a player can choose her assignment in order to satisfy her own CSP. A global solution to such a problem is given by the notion of *pure Nash equilibrium*, in which no player can improve unilaterally her own satisfaction.
We propose four natural variants of this concept that differ from whether optimization is allowed or not and from the definition of the search space. In *Constraint Satisfaction Games* (CSG), the payoff of a player is simply defined as the satisfaction of a CSP. In *Constraint Optimization Games* (COG), the objective of a player is to optimize some value according to some constraints. We also introduce two variants called CSG-HC and COG-HC (HC stands for with *hard constraints*) that allow to constrain the search space in a flexible way, and are therefore easier to solve. For space reasons, we only sketch the resolution of Constraint Games in this paper.

2 Constraint Games

Constraints and CSP. Let V be a set of variables and $D = (D(X))_{X \in V}$ be the family of their (finite) domains. For $W \subseteq V$, we denote by D^W the set of tuples on W, namely $\Pi_{X \in W} D(X)$. Projection of a tuple (or a set of tuples) on a variable (or a set of variables) is denoted by $|$. For example, for $t \in D^V$, $t|_W = (t_X)_{X \in W}$ and for $E \subseteq D^V$, $E|_W = \{t|_W \mid t \in E\}$. For $W, U \subseteq V$, the join of $A \subseteq D^W$ and $B \subseteq D^U$ is $A \bowtie B = \{t \in D^{A \cup B} \mid t|_W \in A \ \wedge \ t|_U \in B\}$. When $W \cap U = \emptyset$, we denote the join of tuples $t \in D^W$ and $u \in D^U$ by (t, u). A *constraint* $c = (W, T)$ is a couple composed of a subset $W = \mathit{var}(c) \subseteq V$ of variables and a relation $T = \mathit{sol}(c) \subseteq D^W$ (called *solutions*). A *Constraint Satisfaction Problem* (or CSP) is a set of constraints. We denote by $\mathit{var}(C) = \bigcup_{c \in C} \mathit{var}(c)$ its set of variables and by $\mathit{sol}(C) = \bowtie_{c \in C} \mathit{sol}(c)$ its set of solutions.
Constraint Satisfaction Games. Let $\mathcal{P}$ be a set of players and V a set of variables. Each player i is given a set of *controlled* variables $V_i \subseteq V$. The sets $(V_i)_{i \in \mathcal{P}}$ are disjoint. Thus each variable is controlled by at most one player. A variable which is not controlled by any player is called an *existential* variable and belongs to V_E.

Definition 1 (Constraint Satisfaction Game). *A* Constraint Satisfaction Game *(or CSG) is a 4-uplet $(\mathcal{P}, V, D, G)$ where $\mathcal{P}$ is a finite set of players, V is a set of variables composed of a family of disjoint sets (V_i) for each player $i \in \mathcal{P}$ and a set V_E of* existential *variables disjoint of all the players variables, $D = (D(X))_{X \in V}$ is the family of their domains and $G = (G_i)_{i \in \mathcal{P}}$ is a family of satisfiable CSP on V.*

The CSP G_i is called the *goal* of the player i. The intuition behind CSG is that, while a player i can only control her own subset of variables V_i, her satisfaction will depend also on variables controlled by all the other players. A controlled variable is called a *decision variable*. The intuition behind existential variables is that they are existentially quantified (but most of the time they will be functionally defined from decision variables).

Example 1. We consider the following CSG: the set of players is $\mathcal{P} = \{X, Y, Z\}$. Each player owns one variable: $V_X = \{x\}, V_Y = \{y\}$ and $V_Z = \{z\}$ with $D(x) = D(y) = D(z) = \{0, 1, 2\}$. The goals are $G_X = \{x \neq y, x > z\}$, $G_Y = \{x \leq y, y > z\}$ and $G_Z = \{x+y = z\}$.

A *strategy* for player i is an assignment of the variables V_i controlled by player i. A *strategy profile* $s = (s_i)_{i \in \mathcal{P}}$ is the given of a strategy for each player. A strategy profile s is *winning* for i if it satisfies the goal of i: $s \in sol(G_i)$. A CSG can be interpreted as a classical game with a boolean payoff function which takes value 1 when the player's CSP is satisfied and 0 when not. The boolean payoff 3-dimensional matrix of Example 1 in normal form is depicted in Figure 1.

$z=0$		y		
		0	1	2
x	0	(0,0,1)	(0,1,0)	(0,1,0)
	1	(1,0,0)	(0,1,0)	**(1,1,0)**
	2	(1,0,0)	(1,0,0)	(0,1,0)

$z=1$		y		
		0	1	2
x	0	(0,0,0)	(0,0,1)	(0,1,0)
	1	(0,0,1)	(0,0,0)	**(0,1,0)**
	2	(1,0,0)	(1,0,0)	**(0,1,0)**

$z=2$		y		
		0	1	2
x	0	(0,0,0)	(0,0,0)	**(0,0,1)**
	1	(0,0,0)	**(0,0,1)**	*(0,0,0)*
	2	**(0,0,1)**	*(0,0,0)*	*(0,0,0)*

Fig. 1. Boolean payoff matrix of Example 1. Nash equilibria are depicted in bold and italics (italics stands for equilibria in which no player is satisfied).

We denote by s_{-i} the projection of s on $V - V_i$. Given a strategy profile s, a player i has a *beneficial deviation* if $s_i \notin sol(G_i)$ and $\exists s'_i \in D^{V_i}$ such that $(s'_i, s_{-i}) \in sol(G_i)$. Beneficial deviation represents the fact that a player will try to maximize her satisfaction by changing the assignment of the variables she can control if she is unsatisfied by the current assignment. A strategy profile s is a *Pure Nash equilibrium* (PNE) of the CSG $\mathcal{C}$ if no player has a beneficial deviation.

Example 2 (Example 1 continued). Each solution of G_X is a winning strategy for player X. For example, 100 (which stands for $x = 1, y = 0, z = 0$) is a winning strategy for player X. However, 100 is not a PNE of the CSG because Player Y may deviate from 0 to 1 to get the winning strategy 110 solution of G_Y. Player Z is able to do the same with 101. The strategy profile 120 is a PNE because it is solution for X and Y, and Player Z is unable to deviate because neither 120, 121 or 122 are solution of G_Z. Pure Nash Equilibria are depicted in Figure 1.

Proposition 1 (derived of [3]). *CSG are Σ_2^p-complete.*

Hard Constraints. In some case it is useful to model that the strategy a player is able to choose depends on the strategy also chosen by other players. In other words, some combinations of individual strategies are not allowed to form a strategy profile. This can be easily expressed in the framework of Constraint Games by adding an additional CSP on the whole set of variables in order to constrain the set of possible strategy profiles. A *Constraint Satisfaction Game with hard constraints* (or CSG-HC) is a 5-uplet $(\mathcal{P}, V, D, C, G)$ where $(\mathcal{P}, V, D, G)$ is a CSG and C is a CSP on V. The intended meaning of the

hard constraints is that beneficial deviation is only allowed in the satisfiable subspace defined by the additional CSP. It is useful to distinguish a strategy profile which does not satisfy any player's goal from a strategy profile which does not satisfy the hard constraints. The former can be a PNE if no player has a beneficial deviation while the latter cannot. Therefore hard constraints provide an increase of modelling expressibility without changing the general complexity of CSG.

Constraint Optimization Games. By adding an optimization condition it is possible to represent classical games. A *Constraint Optimization Game* (or COG) is an extension of CSG in which each player tries to optimize his goal. This is achieved by adding for each player i an optimization condition $\min(x)$ or $\max(x)$ where $x \in V_i$ is a variable controlled by i.

Definition 2 (Constraint Optimization Game). *A* Constraint Optimization Game *(or COG) is a 5-uplet* $(\mathcal{P}, V, D, G, Opt)$ *where* $(\mathcal{P}, V, D, G)$ *is a CSG and* $Opt = (Opt_i)_{i \in \mathcal{P}}$ *is a family of optimization conditions for each player of the form* $\min(x)$ *or* $\max(x)$ *where* $x \in V_i$.

A winning strategy for player i is still a strategy profile which satisfies G_i. However, the notion of beneficial deviation needs to be slightly adapted. Given a strategy profile s, a player i having as optimization condition $\min(x)$ (resp. $\max(x)$) has a *beneficial deviation* if $\exists s'_i \in D^{V_i}$ such that $s' = (s'_i, s_{-i}) \in sol(G_i)$ and $s'|_x < s|_x$ (resp $s'|_x > s|_x$). Given this, the notion of solution is the same as for CSG. In addition, COG can be extended with hard constraints the same way CSG are, yielding COG-HC.

3 Results

Solving games is a remarkably difficult task. We are not aware of any efficient algorithm to solve pure equilibria in normal form games [20]. We first propose a complete algorithm called *CG-enum-1* based on support enumeration. We use a complete static ordering on the variables and enumerate all solutions of all players in lexicographic order. Then each candidate is tested against equilibrium condition and output if successful. Although naive, this algorithm benefits from the power of constraint propagation on hard constraints and compactness of encoding. The second algorithm we propose is a local search algorithm based on tabu search called *CG-tabu-1*. The notion of move is given by the deviation a player may perform and the tabu list is used to forbid a player to be chosen too early after she has moved. Non-tabu players are checked against deviation. If none of them can deviate, then tabu players are also checked to ensure correction. If only a tabu player can deviate, a restart is performed, otherwise the Nash equilibrium is reported. One important point is that modelling games using the CG framework is very easy. We have examples of various scheduling games, location and assignment games. We have implemented a solver called *CG-Solver* on top of the constraint library Choco [18]. This solver allows to express Constraint Games and solves them using two algorithms, *CG-enum-1* and *CG-tabu-1*. An important point is that our solver accepts all constraints provided by Choco, and reuses the constraint propagators already present in the library. We have compared the efficiency with Gambit [19, 13], a game solving suite which uses normal form representation as input. Although it mainly focuses on mixed

equilibria, Gambit includes a solver for pure equilibria called `gambit-enumpure`. Since we do not use the same representation, the comparison is difficult to establish. In summary, since Gambit represents a game in normal form, the size of the input grows exponentially and memory limits are quickly reached. On the other hand, since all data can be quickly accessed, the discovery of equilibria is usually very fast. CG-enum-1 is comparable to Gambit and reports all equilibria for reasonably small games and CG-tabu-1 shows remarkable performances on large satisfiable games (up to 200 players). For space reasons, we cannot report here the examples, algorithms and experiments.

4 Related works and Conclusion

Related works. Works on game theory are too numerous to be mentioned. But Constraint Games inherit from different lines of work that we try to survey. First they provide a compact encoding in the line of boolean games [10, 3]. One of the difference (besides variable domain) is that we provide optimization to model preferences instead of CP-nets [4].*Graphical games* [11] are games in which a player's utility only depends on a subset of the other players. However, like sparse, symmetric, anonymous, local-effect or multimatrix games [16], they are not general games.
From a modelling perspective, solution concepts are heavily discussed in the game theory community, as pure Nash equilibria do not provide a satisfactory notion of solution all the time. The main directions are mixed equilibria [14] and taking a subset of equilibria with additional properties like Pareto-optimality or subgame equilibria [16].
There are few attempts to use Constraint Programming in Game Theory. In [9] is presented a CSP encoding of the reaction operator in graphical games. In [6], it has been proposed to compute a mixed equilibrium using continuous constraints. Games with Hard Constraints (originally called Shared Constraints) [17] are not related to constraint programming but to general constrained optimization. Some other formalism solve one combinatorial problem by multiple agents, either with a predefined assignment of variables to agents like in DCOP [8] or by letting the agents select dynamically their variable like in SAT-Games [21] and Adversarial CSP [7]. Other types of equilibria such as Stackelberg equilibria have been investigated within the QCSP framework [5, 2].
Elimination of dominated strategies can be seen as a form of propagation for games [1]. Several types of domination have been devised, among them the best known are *strong domination, weak domination* and *never best response*. However, this detection is very costly (actually Σ_2^p-complete for boolean games, see [3]).
Conclusion. In this paper, we propose *Constraint Games*, the first framework that allows to model and solve in a natural way strategic games by using Constraint Programming. Constraint Games come in two flavours: Constraint Satisfaction Games (CSG) and Constraint Optimization Games (COG), with or without hard constraints.

References

1. Apt, K.R.: Uniform proofs of order independence for various strategy elimination procedures. CoRR cs.GT/0403024 (2004)

2. Benedetti, M., Lallouet, A., Vautard, J.: Quantified constraint optimization. In: Stuckey, P.J. (ed.) CP. Lecture Notes in Computer Science, vol. 5202, pp. 463–477. Springer (2008)
3. Bonzon, E., Lagasquie-Schiex, M.C., Lang, J., Zanuttini, B.: Boolean games revisited. In: Brewka, G., Coradeschi, S., Perini, A., Traverso, P. (eds.) ECAI. Frontiers in Artificial Intelligence and Applications, vol. 141, pp. 265–269. IOS Press (2006)
4. Bonzon, E., Lagasquie-Schiex, M.C., Lang, J., Zanuttini, B.: Compact preference representation and boolean games. Autonomous Agents and Multi-Agent Systems 18(1), 1–35 (2009)
5. Bordeaux, L., Monfroy, E.: Beyond np: Arc-consistency for quantified constraints. In: Hentenryck, P.V. (ed.) CP. Lecture Notes in Computer Science, vol. 2470, pp. 371–386. Springer (2002)
6. Bordeaux, L., Pajot, B.: Computing equilibria using interval constraints. In: Faltings, B., Petcu, A., Fages, F., Rossi, F. (eds.) CSCLP. Lecture Notes in Computer Science, vol. 3419, pp. 157–171. Springer (2004)
7. Brown, K.N., Little, J., Creed, P.J., Freuder, E.C.: Adversarial constraint satisfaction by game-tree search. In: de Mántaras, R.L., Saitta, L. (eds.) ECAI. pp. 151–155. IOS Press (2004)
8. Faltings, B.: Distributed Constraint Programming, chap. 20, pp. 699–729. Handbook of Constraint Programming, Elsevier (2006)
9. Gottlob, G., Greco, G., Scarcello, F.: Pure nash equilibria: Hard and easy games. J. Artif. Intell. Res. (JAIR) 24, 357–406 (2005)
10. Harrenstein, P., van der Hoek, W., Meyer, J.J.C., Witteveen, C.: Boolean Games. In: van Benthem, J. (ed.) TARK. Morgan Kaufmann (2001)
11. Kearns, M.J., Littman, M.L., Singh, S.P.: Graphical models for game theory. In: Breese, J.S., Koller, D. (eds.) UAI. pp. 253–260. Morgan Kaufmann (2001)
12. Leyton-Brown, K., Shoham, Y.: Essentials of Game Theory: A Concise Multidisciplinary Introduction. Synthesis Lectures on Artificial Intelligence and Machine Learning, Morgan & Claypool Publishers (2008)
13. McKelvey, R.D., McLennan, A.M., Turocy, T.L.: Gambit: Software tools for game theory (2010), http://www.gambit-project.org
14. Nash, J.: Non-cooperative games. Annals of Mathematics 54(2), 286–295 (1951)
15. Neumann, J.V., Morgenstern, O.: Theory of Games and Economic Behavior. Princeton University Press (1944), http://jmvidal.cse.sc.edu/library/neumann44a.pdf
16. Papadimitriou, C.H.: The complexity of Finding Nash Equilibria, chap. 2, pp. 29–51. Algorithmic game theory, Cambridge University Press (2007)
17. Rosen, J.B.: Existence and uniqueness of equilibrium points for concave n-person games. Econometrica 33(3), 520–534 (July 1965)
18. The Choco Team: Choco : An Open Source Java Constraint Programming Library. Ecole des Mines de Nantes (2008-2013), http://www.emn.fr/z-info/choco-solver/
19. Turocy, T.L.: Towards a black-box solver for finite games: Finding all Nash equilibria with Gambit and PHCpack. In: Stillman, M., Takayama, N., Verschelde, J. (eds.) Software for Algebraic Geometry. IMA volumes in Mathematics and its Applications, vol. 148, pp. 133–148. Springer (2008)
20. Turocy, T.L.: Personal communication (2013)
21. Zhao, L., Müller, M.: Game-sat: A preliminary report. In: SAT (2004)