Strategic Constraint Satisfaction Problems

Christian Bessiere, Guillaume Verger

Strategic Constraint Satisfaction Problems

Christian Bessiere and Guillaume Verger

LIRMM, CNRS/University of Montpellier, France
{bessiere,verger}@lirmm.fr

Abstract. The Quantified constraint satisfaction problem (QCSP) has been introduced to express situations in which we are not able to decide the value of some of the variables (the universal ones). Despite the expressiveness of QCSP, many problems, such as two-player games or motion planning of robots, remain difficult to express. In this paper, we propose Strategic CSP, an extension of QCSP where universal variables adapt their domain to be compatible with previous choices. This new framework permits an easy encoding of many two-player games. We give examples of such encodings and provide a preliminary experimental evaluation.

1 Introduction

The constraint satisfaction problem (CSP) consists in finding values for variables such that a set of constraints are satisfied. This framework is useful to express and solve many real applications. However, some problems require dealing with features that are hard to express in the classical CSP framework. Hence, several extensions of the basic CSP have been proposed to deal with these various features. For instance, in the quantified CSP framework [1], variables are either existentially or universally quantified (whereas in the basic CSP all variables are existentially quantified). The idea behind universal variables is that they must be able to take any of their values because they represent uncontrollable parameters such as meteorological events. QCSP is also presented as fitting well two-player adversarial games, where we want to find a winning strategy for player 1. In this case, decisions of player 1 are encoded as existential variables whereas decisions of player 2 are universal variables (we do not know what move she will perform). Unfortunately, as soon as there are some rules that constrain future moves depending on past moves, it is difficult for QCSPs to encode such games. The reason is that a QCSP will look for solutions for any possible move of player 2 while some of them have been made impossible by previous moves. This feature of building solutions that are consistent only with values remaining possible at the time we instantiate them would also be useful in some real applications such as motion planning for robots.

In this paper, we propose Strategic CSPs (SCSPs), a new framework that is devoted to problems containing variables that must be able to take any of their values *still possible* at the time we instantiate them. We show how this framework permits to easily encode two-player games by providing SCSP models for a few

well-known simple games. We provide experiments in which we show the cost of finding a winning strategy in a two-player game.

The rest of the paper is organized as follows. In Section 2 we give some background on CSPs and QCSPs. Section 3 shows the limitations of the QCSP framework. In Section 4 we present the strategic CSP framework and we provide a theoretical analysis. Section 5 shows a few examples of game encodings with this framework. Section 6 contains preliminary experimental results. Section 7 discusses related work. In Section 8 we summarize our contribution and give some perspectives.

2 Background

A *constraint network* $N = (X, D, C)$ consists of a finite set of variables $X = \{X_1, \ldots, X_n\}$, a set of domains $D = \{D(X_1), \ldots, D(X_n)\}$, where the domain $D(X_i)$ is the finite set of values that variable X_i can take, and a set of constraints $C = \{c_1, \ldots, c_e\}$. Each constraint c_k is defined by the ordered set $var(c_k)$ of the variables it involves, and by the set $sol(c_k)$ of combinations of values on $var(c_k)$ satisfying it. A *solution* to a constraint network is an assignment of a value from its domain to each variable such that every constraint in the network is satisfied. When only a subset $Y \subseteq X$ of variables are assigned and that all constraints c with $var(c) \subseteq Y$ are satisfied, we say that the assignment is *locally consistent*. A value v_i for a variable X_i is *consistent with* a subset X' of the variables iff the network $(X' \cup \{X_i\}, D', C' \cup \{X_i = v_i\})$ has solutions, where D' is simply the set of domains in D of the variables in X' and C' is the set of the constraints $c_k \in C$ such that $var(c_k) \subseteq X' \cup \{X_i\}$.

Given a constraint network $N = (X, D, C)$, the constraint satisfaction problem (CSP) is the problem of deciding whether there exists an assignment in D for the variables in X such that all constraints in C are satisfied. In a logical formulation, we write, "$\exists X_1 \ldots \exists X_n, C$?"

The quantified extension of the CSP allows some of the variables to be universally quantified. A *quantified* constraint network consists of variables $X = \{X_1, \ldots, X_n\}$, a set of domains $D = \{D(X_1), \ldots, D(X_n)\}$, a quantifier sequence $\Phi = (\phi_1 X_1, \ldots, \phi_n X_n)$, where $\phi_i \in \{\exists, \forall\}, \forall i \in 1..n$, and a set of constraints C. Given a quantified constraint network, the Quantified CSP (QCSP) is the question "$\phi_1 X_1 \ldots \phi_n X_n, C$?". For instance, if $D(X_1) = D(X_2) = D(X_3) = \{1, 2\}$ and $C = \{X_1 \neq X_2, X_2 \neq X_3\}$, the QCSP $\exists X_1 \forall X_2 \exists X_3 C$ is inconsistent because there does not exist any value for X_1 such that any value of X_2 satisfies $X_1 \neq X_2$. If $D(X_1) = \{1, 2, 3\}$, $\exists X_1 \forall X_2 \exists X_3 C$ is satisfiable because if $X_1 = 3$, for any value of X_2 we can find one for X_3 such that C is satisfied. (If X_2 takes 1, X_3 takes 2 and vice-versa.) The QCSP is PSPACE-complete.

3 Limitations of QCSPs

QCSPs are devoted to problems where there is some uncontrollable/unpredictable event that we cannot decide. Assignments of the 'decision' variables (those with

the existential quantifier) must be consistent with any value of the unpredictable variables. For instance, in [2], we must find quantities of water (or fertilizer) to put on plants such that whatever the amount of rain, the plants finally receive an adequate amount that permits their growth.

QCSPs are also claimed to be well-adapted to express the problem of the existence of a winning strategy in two-player games, where two players perform some actions (or moves) each at his turn, until one of them reaches a "winning state" (see work by Nightingale [3]). Examples of such games include chess, noughts and crosses, Connect4, checkers, etc. Constraints permit to describe the allowed moves a player can perform, existential variables express the decisions taken by player 1, and universal variables those player 2 could take. Unfortunately, the fact that a universal variable X_i must be consistent with $X_1, \ldots, X_{i-1}$ whatever the value it takes in $D(X_i)$ makes very complex the encoding of games where actions at previous stages rule out possibilities for forthcoming moves (which is the case for most games).

Take for instance a simple game where there are k tokens on a table, numbered $1, 2, ..k$. Players alternatively pick a token on the table. The game stops in a winning state for player 2 as soon as the sum of tokens in the hands of player 1 is a multiple of 3. Otherwise, player 1 wins when the table is empty. It could seem natural to encode this game as the following QCSP ($k = 5$): $\exists X_1 \forall X_2 \exists X_3 \forall X_4 \exists X_5, C$, where $D(X_i) = \{1..5\} \forall i$, and $C = \{X_i \neq X_j \mid i \neq j\} \cup \{X_1 mod 3 \neq 0; (X_1 + X_3) mod 3 \neq 0; (X_1 + X_3 + X_5) mod 3 \neq 0\}$. However, this model is trivially inconsistent: given a universal variable X_j, for all $i < j$, there exists a value in $D(X_j)$ which is equal to the value X_i has already been assigned to. We need to express the fact that X_j can take any value/token *remaining* on the table (not any *initially available* token). This problem can be solved by using conditional constraints the left hand sides of which control if we are in an allowed state: $X_1 mod 3 \neq 0$, $X_1 \neq X_2 \wedge X_1 \neq X_3 \wedge X_2 \neq X_3 \rightarrow (X_1 + X_3) mod 3 \neq 0$, and $\bigwedge_{\forall i,j,i \neq j} (X_i \neq X_j) \rightarrow (X_1 + X_3 + X_5) mod 3 \neq 0$, This technique generates large number of conditional constraints with many arguments in the left hand side. It is a burden in the modelling phase and it leads to weak propagation in the solving phase. We can note that Nightingale bypassed this issue by encoding two-player games as a QCSP containing a single constraint involving all variables. In his encoding, a tuple satisfies the constraint iff it is a winning strategy for player 1. This approach can be costly both in time and space.

Some real applications have characteristics that are difficult to express in the QCSP framework. For instance, in the TWIG robot motion planning project [4], we want to find a sequence of basic moves for a robot such that the final state is some expected position. Each move is activated by one of the robot motors (activating a wheel, a arm, a leg, etc.). The consequence of the move is checked by some sensors that observe the new position of the robot after the move. The next move of the robot must be consistent with the observed position of the robot; that is, for any possible observed position (a value for a universal variable), the robot must be able to find a new movement that is on a path to the goal. What these possible observed positions can be? Among all

the initially possible positions, they are the subset of those that are possible given the position of the robot before the motion action and the type of action executed. This subset is seldom a singleton value because of some unexpected external parameters that affect the robot. For instance, the "jumping 1 meter ahead" action does not have the same result if the robot is on a flat ground or on a 10% slope, or with/without a bag on the shoulders. Hence, as in the case of two-player games, we need an extra feature to express remaining states.

4 The Strategic CSP Framework

To overcome the limitation of QCSPs to express games or any situation where future contingencies are constrained by previous decisions, we introduce a new formalism that we call the Strategic CSP.

To be as general as possible, we chose to separate the variables in two different sets. A first set contains the state variables, that is, those that are used to express the rules of the game (or the constraints of the world). The second set contains the quantified variables, that is, those that are decided by player 1 or player 2 (or those for which we can choose the value and those that must be able to take any of their remaining values).

Definition 1 (Strategic Constraint Network). *A Strategic constraint network is composed of:*

- *a set of state variables* $X = \{X_1, \ldots, X_n\}$,
- *a set of quantified variables* $Y = \{Y_1, \ldots, Y_m\}$,
- *a set of domains* $D = \{D(X_1), \ldots, D(X_n),\ D(Y_1), \ldots, D(Y_m)\}$, *where the domain* $D(X_i)$ *(resp.* $D(Y_i)$*) is the finite set of values that variable* X_i *(resp.* Y_i*) can take,*
- *a quantifier sequence* $\Phi = (\phi_1 Y_1, \ldots, \phi_m Y_m)$, *where* $\phi_i \in \{\exists, \mathring{\forall}\}$ *for all* $i \in 1..m$
- *a set of constraints* $C = \{c_1, \ldots, c_e\}$, *where* $var(c_k) \subseteq X \cup Y$ *for all* $k \in 1..e$

A variable quantified with $\mathring{\forall}$ (called "universal" by language abuse) must be able to take any of its values that are still consistent after the assignment of all the quantified variables preceding it in the sequence. In the following, `head` denotes the first element in a sequence, and `tail` denotes the sequence where the first element has been removed.

Definition 2 (Strategic CSP). *A Strategic CSP (SCSP)* (X, Y, D, Φ, C) *is satisfiable iff:*

- $Y = \emptyset$ *and* (X, D, C) *is a satisfiable CSP, or*
- $\mathtt{head}(\Phi) = \exists Y_i$ *and there exists a value* $v \in D(Y_i)$ *such that* $(X \cup \{Y_i\}, Y \setminus \{Y_i\}, D, \mathtt{tail}(\Phi), C \cup \{Y_i = v\})$ *is satisfiable, or*
- $\mathtt{head}(\Phi) = \mathring{\forall} Y_i$ *and for every value* $v \in D(Y_i)$, *if* (Y_i, v) *is consistent with* X, *then* $(X \cup \{Y_i\}, Y \setminus \{Y_i\}, D, \mathtt{tail}(\Phi), C \cup \{Y_i = v\})$ *is satisfiable.*

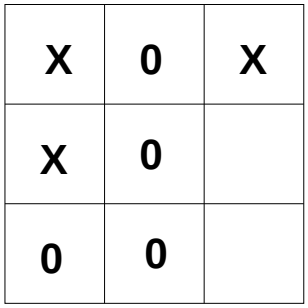

Fig. 1. Tic-tac-toe: The noughts win

Note that deciding which values have to be tried for a universal variable Y_i is NP-hard because it requires testing which values of Y_i are consistent with all state variables and all already instantiated quantified variables. This requires checking the consistency of the values of Y_i in the subnetwork containing all constraints involving state variables and quantified variables $Y_1, \ldots, Y_i$. In fact, we will see later that in all our examples the models are naturally structured in such a way that the test of consistency of a universal variable is linear. We now prove that SCSP has the same complexity as QCSP.

Theorem 1 (Complexity). *The Strategic CSP is PSPACE-complete.*

Proof. We reduce QCSP (which is PSPACE-complete) to SCSP. Let (Y, D, Φ, C) be a QCSP. We build a set of variables Y^c containing a copy Y_i^c of Y_i for each $Y_i \in Y$. We define the quantifier sequence $\Phi^c = (\phi_1^c Y_1^c \phi_2^c Y_2^c \ldots \exists Y_1 \exists Y_2 \ldots)$ where $\phi_i^c = \exists$ if $\phi_i = \exists$ and $\phi_i^c = \overset{\circ}{\forall}$ if $\phi_i = \forall$. Note that all variables in Y are existentially quantified in Φ^c. (There are no state variables.) The set C' of constraints contains all constraints of C on Y as in the QCSP and constraints $Y_i^c = Y_i$ for all i. The SCSP $(\emptyset, Y^c \cup Y, D, \Phi^c, C')$ is satisfiable iff the QCSP is satisfiable. Since SCSP is in PSPACE we are done. □

5 Examples

5.1 Tic-tac-toe (noughts and crosses)

Tic-tac-toe, also called noughts and crosses and many other names, is a game between two players, O and X, who alternate in marking the cells in a 3x3 board. A player wins by getting three of her own marks in a horizontal, vertical or diagonal row. The game ends in a draw if the board is filled completely without any player winning.

A natural question that arises in such games is: 'is there a winning strategy for player 1?' (that is, is it possible that player 1 wins if player 2 plays perfectly). We show this question can be encoded in a strategic CSP that is satisfiable iff such a strategy exists.

We build the constraint network on two kinds of variables. The first kind of variables (state variables) represent the board and will support the constraints expressing the rules of the game. The second kind of variables (quantified variables) represent the decisions taken by the players, and is linked to the first kind of variables by channelling constraints that guarantee the coherence of the values taken by all variables.

Model 1 The strategic CSP of Tic-tac-toe contains state variables and quantified variables. The set of state variables contains 9 board-variables and 8 flag-variables that represent the cells on the board and the winning states. They will support constraints expressing the rules of the game (what kind of moves are allowed and what is a winning state). The board-variables are $X_{1.1}, \ldots, X_{3.3}$, where $X_{i.j}$ takes as value the number of the player (1 or 2) who put a mark on the cell of coordinates $i.j$, or 0 if not marked; that is, $D(X_{i.j}) = \{0, 1, 2\}, \forall i, j \in 1..3$. The flag-variables $F_{r.1}, F_{r.2}, F_{r.3}, F_{c.1}, F_{c.2}, F_{c.3}, F_{d.1}, F_{d.2}$ give the status of their associated row ($F_{r,i}$), column ($F_{c,i}$), or diagonal ($F_{d,i}$): Variable $F_{*.i}$ takes 1 if the line is won by player 1, 2 if it is won by player 2, 0 otherwise; $D(F_{*.i}) = \{0, 1, 2\}$. The set of quantified variables contains 9 player-variables $P_1, \ldots, P_9$ that represent the decisions of the players, 4 stopping-variables $Stop_1, Stop_3, Stop_5, Stop_7$ that will permit to stop the game as soon as player 1 has won, and a final-state variable F to decide if player 1 won. Variable P_i takes as value the coordinate of the cell marked at turn i (this is a decision by player 1 if i is odd, by player 2 otherwise). If the game has finished before step i, P_i takes the dummy value 0. That is, $D(P_i) = \{0, 1.1, \ldots, 3.3\}, \forall i \in 1..9$. Boolean variable $Stop_i$ (i is odd) takes value 1 iff player 1 won at move i or earlier; $D(Stop_i) = \{0, 1\}$.

The following constraints ensure that variables take values as described above. We post the channelling constraints that say that X_v takes as value the number of the player who marked it (0 if not marked):

$$P_i = v \rightarrow X_v = ((i-1) mod 2) + 1, \forall i \in 1..9, v \in 1.1..3.3$$

Constraint $\texttt{Full}(X, Y, Z, G)$ is true iff $(X = Y = Z = k) \leftrightarrow (G = k), k \in 1, 2$. That is, the three board-variables X, Y, Z are all filled by player k iff the flag-variable G is k. We post `Full` on each row, column, and diagonal to guarantee that the flag of a given line correctly indicates if there is a winner on this line:

$$\texttt{Full}(X_{i.1}, X_{i.2}, X_{i.3}, F_{r.i}), \forall i \in 1..3, \text{ /* rows */}$$

$$\texttt{Full}(X_{1.j}, X_{2.j}, X_{3.j}, F_{c.j}), \forall j \in 1..3, \text{ /* columns */}$$

$$\texttt{Full}(X_{1.1}, X_{2.2}, X_{3.3}, F_{d.1}), \text{ and}$$

$$\texttt{Full}(X_{1.3}, X_{2.2}, X_{3.1}, F_{d.2}) \text{ /* diagonals */}$$

We post the constraints that guarantee that player 1 is the winner (at least one line is won by player 1, and no line is won by player 2):

$$max(F_{*.i}) = F \wedge F = 1$$

Note that if we had directly posted the constraint $max(F_{*.i}) = 1$, it would prune value 2 from all $F_{*.i}$ because $F_{*.i}$ are state variables. Hence, constraints $\texttt{Full}(X, Y, Z, F_{*.i})$ would prevent player 2 from filling a cell on a line that would make her the winner, and the SCSP would be satisfiable even if there is no winning strategy for player 1. With F being a quantified variable, constraint $max(F_{*.i}) = F$ is not used to test consistency of universal variables as long as variable F is not instantiated (see Definition 2). Thus, nothing prevents player 2 from winning a line. In this case, the failure comes from the impossibility to instantiate F with value 1. Furthermore, we must avoid player 2 wins a line *after* player 1 already won. For that, we post stopping constraints. They permit player 1 to stop the game as soon as he won. They also prevent player 2 from passing her turn.

$$Stop_i = 1 \leftrightarrow P_{i+1} = 0, \forall i \in \{1, 3, 5, 7\}$$

$$Stop_i = 1 \leftrightarrow P_{i+2} = 0, \forall i \in \{1, 3, 5, 7\}$$

$$Stop_i = 1 \rightarrow Stop_{i+2} = 1, \forall i \in \{1, 3, 5\}$$

Once the network is specified as above, the strategic CSP $(\{X_{1.1}, \ldots, X_{3.3}, F_{*.i}, \ldots\}, \{P_1, Stop_1, P_2, P_3, Stop_3, \ldots, P_9, F\}, (\exists P_1, \exists Stop_1, \mathring{\forall} P_2, \exists P_3, \exists Stop_3, \ldots, \exists P_9, \exists F), \mathcal{C})$ is satisfiable iff there exists a winning strategy for player 1. We see that once all $F_{*.i}$ are instantiated, it imposes a single value for F, and the instantiation is consistent iff that value for F is 1. Variables $Stop_i$ are existential because this is player 1 who decides when to stop.

Model 2 We propose a second model which emphasizes the fact that universal variables do not need to have remaining consistent values once this is their turn to be instantiated. An existential variable without any consistent value means a failure whereas a universal variable without any consistent value means a winning branch.

We transform model 1 the following way: Player-variables P_i do not have the dummy value 0 in their domain: $D(P_i) = \{1.1, \ldots, 3.3\}$. We remove all $Stop_i$ variables and the stopping constraints. We add the constraint

$$\texttt{not-both-1-and-2}(F_{r.1}, F_{r.2}, F_{r.3}, F_{c.1},$$

$$F_{c.2}, F_{c.3}, F_{d.1}, F_{d.2})$$

which holds iff values 1 and 2 do not both appear in the variables $F_{*.i}$.

The strategic CSP $(\{X_{1.1}, \ldots, X_{3.3}, F_{*.i}, \ldots\}, \{P_1, P_2, P_3, \ldots, P_9, F\}, (\exists P_1, \mathring{\forall} P_2, \exists P_3, \ldots, \exists P_9, \exists F), \mathcal{C})$ is satisfiable iff there exists a winning strategy for player 1.

This model can seem a bit surprising. First, it does not contain stopping variables and constraints. These variables and constraints were necessary to ensure that a player does not pass his turn or stops the game before the end. In model 2, the P_i do not contain value 0. So, they are obliged to take a value

which is a regular move. Once one of the players, say k, has won a line, the corresponding $F_{*.i}$ is set to k, and the constraint `not-both-1-and-2`$(F_{*.i})$ guarantees that none of the other moves is winning for the other player. If $k = 2$, this can lead to branches where player 1 cannot play, which is a failure in that branch (and this is what we want since player 2 won). If $k = 1$, this can lead to branches where player 2 cannot play (all its values are inconsistent with constraint `not-both-1-and-2`$(F_{*.i})$), which is a successful branch (and this is what we want since player 1 won).

Discussion We have observed in Section 4 that testing which values of the universal variables remain consistent is NP-hard. However, in some cases, the subnetwork involving state variables and already instantiated quantified variables can fit a polynomial class (tree-structured networks for instance).

How is it in model 1 of Tic-tac-toe? The constraints involving state variables $X_{i.j}$ are the channelling constraints, the `Full` constraints, and the $max(F_{*.i}) = F$ constraint. If P_i is the universal variable to instantiate, all channelling constraints involving $P_j, j < i$, are unary constraints because P_j has already been instantiated. So, intractability cannot come from them. `Full` constraints are functional from the $X_{i.j}$ to $F_{*.i}$, which means that any combination of values for the $X_{i.j}$ belongs to a satisfying tuple. The only other constraint involving $F_{*.i}$ is $max(F_{*.i}) = F$, which will be considered only when instantiating the quantified variable F, which is after all universal variables P_i. Thus, arc consistency is enough to guarantee that a value for a universal variable is consistent with state variables and previous instantiations of quantified variables.

How is it in model 2? Here, in addition to the constraints in model 1, there is the `not-both-1-and-2` constraint on the $F_{*.i}$. This last constraint cannot force different $F_{*.i}$ to take different non null values. Hence, two `Full` constraints overlapping on some $X_{i.j}$ do not constrain it more than the most restrictive `Full` constraint alone. Thus, as in model 1, arc consistency is sufficient.

Interestingly, in our two models, consistency for values of universal variables is easy to check. This is not surprising because constraints involving state variables are either channelling constraints or constraints expressing the rules of the game. The former are functional and the latter cannot be complex to verify because the rules of a game must be easy to follow by human players.

5.2 Connect 4

We now show that a more complex game like Connect4 can be encoded with the same principle.

Connect4 (see Fig 2) is a two-player game that is played on a vertical board with 6 rows and 7 columns. The players have 21 discs each, distinguished by color. The players take turns in dropping discs in one of the non-full columns. The disc then occupies the lowest unoccupied cell on that column. A player wins by placing four of their own discs consecutively in a line (row, column or diagonal), which ends the game. The game ends in a draw if the board is filled completely without any player winning.

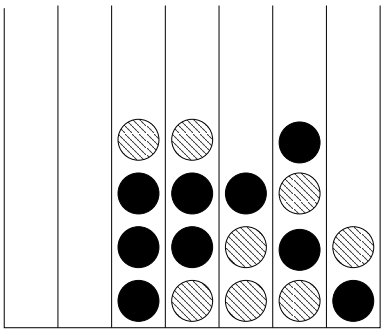

Fig. 2. Connect4: The blacks win

A Model We use the same technique as in the second model of Tic-tac-toe, except that we need to keep the ordering of the moves of each player because a disc placed recently cannot be placed below an older one in the same column. The strategic CSP of Connect4 contains state variables and quantified variables. The set of state variables contains 42 board-variables $X_{1.1}, \ldots, X_{6.7}$, where $X_{i.j}$ represents the cell on the ith row, jth column and takes as value the rank at which a disc was put in that cell. That is, $D(X_{i.j}) = \{1..42\}, \forall i, j$. There are also flag-variables $F_{k_1}, F_{k_2} \ldots$, one for each sequence of four cells on the same line. F_{k_i} will give the status of their associated line; that is, 1 if the line is won by player 1, 2 if it is won by player 2, 0 otherwise; $D(F_{k_i}) = \{0, 1, 2\}$. The set of quantified variables contains 42 player-variables $P_1, \ldots, P_{42}$ that represent the decisions of the players. Variable P_i takes as value the coordinate of the cell filled at turn i (this is a decision by player 1 if i is odd, by player 2 otherwise). That is, $D(P_i) = \{1.1, \ldots, 6.7\}, \forall i \in 1..42$.

The following constraints ensure that variables take values as described above.

We post the channelling constraints that say that X_v takes as value the rank at which it is marked (0 if not marked):

$$P_i = v \leftrightarrow X_v = i, \forall i \in 1..42, v \in 1.1..6.7$$

We post constraints that guarantee that a recent disc is not put below an older one:

$$X_{i+1.j} > X_{i.j}, \forall i, j$$

We define a constraint that detects a winner on a line of four cells: $\texttt{Line}(W, X, Y, Z, G)$ holds iff:

- W, X, Y, Z are all even and $G = 2$, or
- W, X, Y, Z are all odd and $G = 1$, or
- W, X, Y, Z do not all have the same parity and $G = 0$.

We post this constraint on all lines of four consecutive cells:

$$\texttt{Line}(X_{i.j}, \ldots, X_{i.j+3}, F_{k_i}) \wedge \ldots$$

$$\wedge \texttt{Line}(X_{i.j}, \ldots, X_{i+3.j}, F_{k_l}) \wedge \ldots$$

We add the constraint

$$\texttt{not-both-1-and-2}(F_{k_1}, F_{k_2}, \ldots)$$

We post the constraints that guarantee that player 1 is the winner (at least one line is won by player 1, and no line is won by player 2):

$$max(F_{*.i}) = F \wedge F = 1$$

Once the network is specified as above, the strategic CSP $(\{X_{1.1}, \ldots, X_{6.7}, F_{k_1}, \ldots\}, \{P_1, P_2, P_3, \ldots, P_{42}, F\}, (\exists P_1, \mathring{\forall} P_2, \exists P_3, \ldots, \mathring{\forall} P_{42}, \exists F), \mathcal{C})$ is satisfiable iff there exists a winning strategy for player 1.

In addition to the constraints similar to those in model 2 of Tic-tac-toe, this model contains constraints '>' on state variables. These constraints being linear, arc consistency is sufficient to check consistency of values of universal variables.

6 Experimental Evaluation

We provide very preliminary experiments that show that the SCSP framework is a promising approach for solving two-player games. We used a very brute-force algorithm, which is merely a classical CP toolkit (Choco [5]) in which we postpone propagation of constraints that contain quantified variables until all their quantified variables are instantiated, and we force extending universal variables to solutions for all their remaining values. All our experiments were run on a 1.6GHz PC with 128Mb of memory.

We tested the SCSP framework on the two games presented in Section 5. For Tic-tac-toe, we experimented on the two models proposed in Section 5.1 to see what type of model (with or without stopping variables) is more efficient and to see if our approach is practicable. We obtained 5.74 sec. for model 1 and 2.05 sec. for model 2. This shows that the simpler model (without stopping variables) is the faster. These results can be compared to those obtained in [3], where Nightingale also tested on two different models. One of them was solved in 26.20 sec. and the other in 13.78 sec. on a PC twice as fast as ours (3GHz with 1Gb of memory).

Concerning Connect4, the classical instance presented in Section 5.2 (6×7 board with lines of length 4) is far too difficult. We tested various sizes of the board as presented in [6] for their experimentation of QBF encodings of Connect4: square boards of sizes $k \times k, k \in 2..9$ where players try to put 2, 3 or 4 discs on the same line. Table 1 reports results obtained by Gent and Rowley on their QBF encoding and results obtained with our SCSP model of Section 5.2. Their computer was slightly faster than ours (1.7GHz) with the same memory. We can see in Table 1 that SCSP is almost always faster with the notable exception of the $(5, 5, 4)$ instance. It would be worth seeing if it is confirmed on other instances, like $(5, 5, 5)$, but Gent and Rowley's paper contains

Table 1. Results of Gent's QBF encoding (left) and our SCSP encoding (right). Square boards of width and height k, requiring c discs in a line to win. Cutoff at 3600 sec.

QBF			Instance			SCSP		
Time(s)	#vars	#clauses	k	c	Solution	Time(s)	#vars	#constraints
0.00	238	631	2	2	T	0.00	15	27
0.03	1,117	3,330	3	2	T	0.06	39	110
4.23	8,374	27,384	5	2	T	0.23	123	720
22.80	17,314	57,891	6	2	T	0.45	183	1,439
137.59	32,043	108,838	7	2	T	1.72	255	2,602
0.07	901	2,610	3	3	F	0.20	27	98
8.10	2,880	9,381	4	3	T	1.83	57	295
451.41	7,174	24,984	5	3	T	98.04	99	696
-	15,154	55,011	6	3	-	-	153	1,409
3.08	2,432	7,461	4	4	F	3.89	43	281
703.92	6,174	21,384	5	4	F	-	79	676
-	13,282	49,539	6	4	-	-	127	1,383

only those reported here. Finally, we can note that the encoding is extremely simpler in SCSP than in QBF: In [6], the description of the Connect4 model takes 11 pages whereas we needed less than a column.

7 Related Work

In [3], Nightingale used QCSPs to encode two-player games. He avoided the limitation of QCSPs by encoding the game in a single constraint. The complexity is thus completely embedded in the constraint, which makes its propagation expensive. Walsh proposed the Stochastic CSP framework to express problems where there is some uncertainty on the value that a given variable can take [7]. Each uncontrollable variable is associated with a probability distribution on its values. Though being more expressive than QCSP, Stochastic CSP still cannot conveniently express the notion of *remaining* values for an uncontrollable variable. Outside the constraint reasoning area, minimax techniques have been used for a long time to solve two-player games. They can be efficient when the space of search is too large to be explored exhaustively and when a good evaluation function is available to compare different situations. However, such techniques do not use propagation of inconsistencies as constraint reasoning permits to do. Finally, Quantified Boolean Formulae (QBF) have been used to encode two-player games. The issue of remaining values is solved by complex encodings involving additional 'indicator' variables and clauses detecting 'illegal' moves. For instance, in [6], an encoding for the Connect4 is proposed. As seen in Section 6, this encoding is extremely complex and often not as efficient as a SCSP formulation solved with a brute-force algorithm. In [8], the weakness of QBF for solving two-player games is pointed out. New encodings are proposed that permit to detect illegal moves as early as possible and a new QBF solver is proposed, that efficiently handles the indicator variables. We can emphasize

here the fact that SCSPs do not need to take care of such optimized encodings since illegal moves are not considered at all.

8 Conclusion and Perspectives

We have shown that there exist problems, such as two-player games, for which we need to express that some variables must be able to take any of their values still available at the time they are instantiated. None of the existing CSP frameworks permit to express this easily. We have proposed the Strategic CSP to tackle this issue. We have shown that Strategic CSPs allow neat encodings of classical two-player games. We provided an initial experimental evaluation that confirms that SCSP is not only a simple framework for expressing problems, but also a way to solve them efficiently compared to other approaches. As for efficiency, we need to carefully study the characteristics of SCSPs to propose efficient algorithms. Our experiments used a standard CSP solver in which constraints involving quantified variables are simply delayed until their last quantified variable is to be instantiated. This could easily be improved by writing propagators that propagate constraints from state variables to quantified ones but not the opposite. Finally, we will use SCSPs in the TWIG robot motion planning project.

Acknowledgements

We want to thank Frederic Koriche, Bruno Zanuttini and all the members of the Coconut group for useful comments and discussions.

References

1. Bordeaux, L., Montfroy, E.: Beyond NP: Arc-consistency for quantified constraints. In: Proceedings CP'02, Ithaca NY (2002) 371–386
2. Fargier, H., Martin-Clouiare, J.L.R., Rellier, J.: Uncertainty and flexibility in constraint satisfaction: a case study and an application to agricultural planning. In: Proceedings of the workshop on Constraint Satisfaction Issues Raised by Practical Applications, Amsterdam, The Netherlands (1994) 21–29
3. Nightingale, P.: Consistency for quantified constraint satisfaction problems. Technical Report CPPod-11-2005, CPPod (2005) short version in Proceedings CP'05, pages 792-796.
4. Paulin, M., Bourreau, E., Dartnell, C., Krut, S.: Modélisation et planification d'actions élémentaires robotiques par apprentissage de réseaux de contraintes. In: Proceedings JFPC'06, Nimes, France (2006) 405–414
5. Choco: A Java library for constraint satisfaction problems, constraint programming and explanation-based constraint solving. URL: http://choco-solver.net (2005)
6. Gent, I., Rowley, A.: Encoding Connect-4 using quantified boolean formulae. In: Proc. 2nd International Workshop on Modelling and Reformulating Constraint Satisfaction Problems, Kinsale, Ireland (2003) 78–93
7. Walsh, T.: Stochastic constraint programming. In: Proceedings ECAI'02, Lyons, France (2002) 111–115

8. Ansotegui, C., Gomes, C., Selman, B.: The Achilles' heel of QBF, Pittsburgh PA (2005) 275–281