UNORDERED CONSTRAINT SATISFACTION GAMES

Lauri Ahlroth and Pekka Orponen

Department of Information and Computer Science and
Helsinki Institute for Information Technology HIIT
Aalto University, Finland
firstname.lastname@aalto.fi

Abstract. We consider two-player constraint satisfaction games on systems of Boolean constraints, in which the players take turns in selecting one of the available variables and setting it to *true* or *false*, with the goal of maximising (for Player I) or minimising (for Player II) the number of satisfied constraints. Unlike in standard QBF-type variable assignment games, we impose *no order* in which the variables are to be played. This makes the game setup more natural, but also more challenging to control. We provide polynomial-time, constant-factor approximation strategies for Player I when the constraints are parity functions or threshold functions with a threshold that is small compared to the arity of the constraints. Also, we prove that the problem of determining if Player I can satisfy all constraints is PSPACE-complete even in this unordered setting, and when the constraints are disjunctions of at most 6 literals (an unordered-game analogue of 6-QBF).

1 Introduction

An instance of a *constraint satisfaction problem* (CSP) comprises a set of m non-constant Boolean functions $C = \{c_1, \ldots, c_m\}$ over a common set of n variables $X = \{x_1, \ldots, x_n\}$. In this work we only consider *Boolean* CSP's, so that each constraint $c \in C$ is a mapping $c : \{0,1\}^n \to \{0,1\}$. (The truth values *false*, *true* are identified with the integers $0, 1$ as usual.) A constraint c is *satisfied* by a truth assignment $x \in \{0,1\}^n$ if $c(x) = 1$. The most commonly studied versions of CSPs ask whether all of the given constraints can be satisfied simultaneously (typically an NP-complete problem), or if not, then what is the fraction of constraints that can be satisfied by a polynomial-time approximation algorithm.

Another perspective is to consider the *combinatorial games* [11, 17] defined by CSP instances. Here two players, I and II, take turns in setting values of the variables in X, with Player I's goal being to eventually satisfy as many of the constraints in C as possible, and Player II's goal being the opposite. In this framework one can again ask if Player I has a *(comprehensive) winning strategy*, i.e. whether she can satisfy *all* the constraints, no matter what Player II does. Or if a winning strategy does not exist or is hard to compute, then what is the fraction of constraints that Player I can satisfy by a polynomial-time computable approximate strategy against Player II.

Such *constraint satisfaction games* have applications in e.g. formal methods [1, 16] and adversarial planning [2, 3], but are of course also intrinsically interesting. A

paradigmatic example is QBF, which can be viewed as a two-player version of Satisfiability, where the players take turns in assigning values to the variables in a prespecified order [17]. As is well-known, QBF is PSPACE-complete [18], similarly to many other terminating two-player games [9]. Recently there have also been major developments in developing a taxonomy of such quantified constraint satisfaction games, relating their complexity to the structure of the constraints involved [5, 6, 15]. This work follows the example of the quite comprehensive taxonomy on the approximability of CSPs according to the structure of their defining constraints, as achieved in [14].

However, to our knowledge, all existing studies of constraint satisfaction games consider the model where players assign values to the variables in a *predefined (quantification) order*. This is a somewhat unnatural restriction if one views CSPs in the general framework of combinatorial games, and also leads to an overly pessimistic view on the approximability of games of this type. For instance, the two published studies [8, 12] addressing the approximability of (ordered) constraint satisfaction games in the sense of maximising the number of constraints won by Player I both present only *negative* results on the existence of approximate strategies. In particular, both of these papers show that for some $\varepsilon > 0$, MAX-QBF is PSPACE-hard to approximate within $1-\varepsilon$ times optimum. Additionally, articles [4, 7] consider the somewhat different task of optimising a supplementary objective function across the comprehensive winning strategies for Player I.

A symptom of the unnaturality of variable-ordered games is that they effectively allow one of the players to make many consecutive moves, by forcing the other player to play dummy variables that do not affect the final outcome. In the extreme case, one can e.g. have *all* the even-indexed variables be dummies, so that the outcome is totally indifferent to the moves of Player II. By allowing a free choice of variables neither player can be discriminated in this way, and for this reason the unordered game has a more interesting structure than standard QBF, both as a game and in the sense of approximability.

In this paper, we wish to initiate the study of such *unordered* constraint satisfaction games and their approximate strategies. We first define the model in Section 2. In the subsequent Sections we establish our *positive* results on the approximability of unordered constraint satisfaction games. In Section 3 we show that in a game where the constraints are parity functions (linear equations mod 2) with bounded variable co-occurrences, Player I can always win close to $\frac{1}{2}$ of the constraints. In Section 4 we show that if the constraints are threshold functions of arity k and the threshold is $\mu = O\left(\frac{k}{\log k}\right)$, then Player I can win a fraction of the clauses which approaches 1 exponentially as k increases. A weaker result holds when μ satisfies merely $\mu \leq \frac{k}{2}$. Note that the strong version of the result applies in particular to games on disjunctive constraints, since disjunctions are threshold functions with $\mu = 0$.

In Section 5 we establish our fundamental *negative* result showing that also the unordered game analogue of the QBF problem, the Game on Boolean Formulas (GBF), is PSPACE-complete, even when the constraints are disjunctions of at most 6 literals. We conclude in Section 6 with a summary and suggestions for some further research directions.

2 The Game on Boolean Formulas

Our generic example of an unordered constraint satisfaction game is the *Game on Boolean Formulas (GBF)*. An instance of this game is given by a set of m non-constant Boolean formulas $C = \{c_1, \ldots, c_m\}$ over a common set of n variables $X = \{x_1, \ldots, x_n\}$. We refer to the formulas in C as *clauses* even though we do not in general require them to be disjunctions. If all the clauses are disjunctions of *width* $\leq k$, i.e. with at most k literals per clause, then we refer to the game as k-GBF.

A game on C proceeds so that on each turn the player to move selects one of the previously nonselected variables and assigns a truth value to it. Player I starts, and the game ends when all variables have been assigned a value. In the decision version of GBF, the question is whether Player I has a comprehensive winning strategy, by which she can make all clauses satisfied no matter what Player II does. In the positive case we say that the instance is *GBF-satisfiable*. In the maximisation version MAX-GBF, the objective of Player I is to maximise the number of satisfied clauses $m_t := |\{c \in C \mid c(x_1, x_2, \ldots x_n) = 1\}|$, and the objective of Player II is to maximise the number $m_f := m - m_t$ of unsatisfied clauses, making this a zero-sum game.

Recall that a Boolean formula $P(x_1, x_2, \ldots x_n)$ is QBF-satisfiable if Player I can ensure that P becomes satisfied when the players alternate selecting values to $x_1, x_2, \ldots x_n$ *in the given order*. The standard (MAX-)QBF problem can be viewed as a version of (MAX-)GBF, with the additional requirement that the variables must be played in the predetermined order $x_1, x_2, \ldots, x_n$.

3 An approximate strategy for even-odd games

In an *even-odd game* the GBF clauses $c \in C$ are of the form $\sum_{i \in J_c} x_i \equiv d_c \pmod 2$, where for each $c \in C$, $J_c \subseteq \{1, \ldots, n\}$ and $d_c = 0$ or $d_c = 1$. Without loss of generality one can assume that there are no negative literals.

Theorem 1. *In an even-odd game on n variables and m clauses, Player I has a polynomial-time strategy that secures her at least a fraction of* $\frac{1}{2} - \frac{n\delta}{4m}$ *of all clauses, where* $\delta = \max_{i \neq i'} |\{c \mid x_i \in c, x_{i'} \in c\}|$.

Proof. For $1 \leq i \leq \frac{n}{2}$, denote the number of clauses completing as *true* (resp. *false*) due to Player I's ith move as α_i ($\bar{\alpha}_i$) and completing as *true* (*false*) due to Player II's ith move as β_i ($\bar{\beta}_i$). A clause *completes* when its last variable is set to a value. The strategy for Player I is simply to always make a move maximising the difference $\alpha_i - \bar{\alpha}_i$.

When Player I is selecting the variable for her ith move, the variable y selected by Player II on his ith move is also available. Player I's ith move creates at most δ new opportunities for II to complete clauses with y, so by the strategy of I it must be the case that

$$\alpha_i - \bar{\alpha}_i \geq \bar{\beta}_i - \beta_i - \delta,$$

which can be rearranged as

$$\bar{\alpha}_i + \bar{\beta}_i \leq \alpha_i + \beta_i + \delta.$$

Now the total number of clauses has the upper bound

$$m = \sum_i (\alpha_i + \beta_i + \bar{\alpha}_i + \bar{\beta}_i) \leq 2\sum_i (\alpha_i + \beta_i) + \frac{n}{2}\delta = 2m_t + \frac{n}{2}\delta.$$

Hence the fraction of satisfied clauses is at least $\frac{m_t}{m} \geq \frac{1}{2} - \frac{n\delta}{4m}$.

Corollary 1. *In an even-odd game where $\delta\frac{m}{n} \leq \gamma$ for a constant $\gamma < 2$, Player I can secure a constant fraction of all clauses.*

4 Approximate strategies for threshold games

In a *μ-threshold game*, the clauses c are subsets of literals from $\{x_1, \ldots, x_n, \bar{x}_1, \ldots, \bar{x}_n\}$, and a clause $c \in C$ is satisfied if and only if it contains at least $\mu + 1$ true literals $l \in c$. We assume that each clause has at least k literals, that no literal occurs multiple times in the same clause, and that no literal co-occurs with its negation in the same clause. The case $\mu = 0$ corresponds to disjunctive clauses. When all clauses are exactly of length k and $\mu + 1 = k$, the clauses correspond to conjunctions.

In the following, we present an efficient approximate strategy for Player I in the case of small values of μ. The potential function approach used to drive the strategy is inspired by the MAX-SAT approximation algorithm presented in [13], even though the details are very different.

Theorem 2. *Let $\mu \in \mathbf{N}$. In a μ-threshold game on clauses of size at least k, Player I has a polynomial-time strategy that secures her at least a fraction $1 - O(\frac{k^\mu}{2^{k/2}})$ of all clauses.*

To describe the strategy indicated in Theorem 2 we need some more notation. Given a clause c, denote the number of unset literals present in c by $f(c)$, and the number of literals already set to *true* by $t(c)$. Define q as the positive root of $q^2 + q/k - 2 = 0$, which gives

$$\sqrt{2}\frac{1}{1 + 1/(k\sqrt{2})} < q < \sqrt{2}. \tag{1}$$

The *weight* of a clause is defined as

$$u(c) := \begin{cases} q^{-f(c)}k^{-t(c)}, \text{ if } t(c) \leq \mu \\ 0, \text{ else.} \end{cases}$$

Also, the *weight* of a literal l is $u(l) := \sum_{l \in c} u(c)$, and the *potential* of a partially played game is $U := \sum_{c \in C} u(c)$.

Proof (Theorem 2).

Player I's strategy is to always find a literal l of maximum weight $u(l)$ and set l to *true*. We shall prove that this strategy ensures that U is nonincreasing over each pair of moves by the two players.

Note first that if Player I sets literal x to *true* and Player II sets literal y to *true* then $u(x) \geq \max\{u(y), u(\bar{x}), u(\bar{y})\}$.

Over one move the potential changes by $\Delta_x U \leq (q-1)u(\bar{x}) + (q/k-1)u(x)$. After this move the weights change to $u'(c)$, but the new weights still satisfy $u'(c) \leq qu(c)$ clause-wise, implying

$$\Delta_y U \leq (q-1)qu(\bar{y}) + (q/k-1)u'(y) \leq (q^2-q)u(x).$$

Hence

$$\Delta_x + \Delta_y \leq (q-1+q/k-1+q^2-q)u(x) = (q^2+q/k-2)u(x) = 0.$$

This and $\Delta_x \leq 0$ imply that U is nonincreasing over the whole game.

From Eq. (1) one obtains $q^k \geq \sqrt{2}^k e^{-1/\sqrt{2}}$ and further

$$\frac{k^\mu}{q^k} \leq \frac{k^\mu}{2^{k/2}} e^{1/\sqrt{2}}.$$

Now

$$k^\mu m_f \leq U_{end} \leq U_{begin} \leq mq^{-k},$$

implying the final claim since

$$\frac{m_f}{m} \leq \frac{k^\mu}{q^k} \leq \frac{k^\mu}{2^{k/2}} e^{1/\sqrt{2}}.$$

Corollary 2. *In a μ-threshold game where $\frac{k}{\log k}\frac{\log 2}{2} - \frac{1}{\sqrt{2}\log k} - \mu \geq \gamma$ for a constant $\gamma > 0$, Player I can secure a constant fraction of all clauses.*

Theorem 2 gives a positive bound up to $\mu \leq \mu^*$ for some $\mu^* = \theta\left(\frac{k}{\log k}\right)$. For large threshold values $\mu = \Omega\left(\frac{k}{\log k}\right)$ one can apply an elementary algorithm to improve the bound.

Theorem 3. *Let $\mu \in \mathbf{N}$. In a μ-threshold game on clauses of size at least k, Player I has a polynomial-time strategy that secures her at least a fraction of $\frac{k-2\mu}{2k-2\mu}$ of all clauses.*

Proof. Any literal l occurs in $v(l) = |\{c \in C \mid l \in c\}|$ clauses, and we define the weight of a literal as the difference $u(l) := v(l) - v(\bar{l})$. The strategy is to always find a literal l of maximum weight $u(l)$ and set l to *true*.

Consider the numbers of *true* and *false* literal occurrences

$$T = |\{(c,l) \mid c \in C,\ l \in c,\ l \textit{ true}\}|,$$
$$F = |\{(c,l) \mid c \in C,\ l \in c,\ l \textit{ false}\}|.$$

The strategy used by Player I always targets a literal with maximum weight. As the moves never alter the weights of the other literals, the sum of weights of the literals set to *true* is always nonnegative. Especially this holds at end of the game. Hence $T \geq F$, and $T \geq \frac{m}{2}k$. As each unsatisfied clause contains at most μ *true* literals, T can be bounded as

$$\frac{m_t + m_f}{2} k \leq T \leq \mu m_f + k m_t.$$

Rearranging yields $\frac{m_t}{m_f} \geq \frac{k-2\mu}{k}$ and finally

$$\frac{m_t}{m} \geq \frac{k-2\mu}{2k-2\mu} = \frac{1}{2} - \frac{\mu}{2(k-\mu)}.$$

The result of Theorem 3 yields a positive lower bound on the number of clauses won by Player I up to a threshold of $\mu < \frac{k}{2}$. Actually no strategy can guarantee a positive fraction of the clauses to her if $\mu \geq \frac{k}{2}$. To verify this, consider an instance that has different variables in each clause, and Player II always responds by playing a literal *false* in the same clause where Player I just played. Independent of Player I's strategy, all clauses end up *false*. Naturally, this observation does not preclude approximation bounds with respect to the value of an optimal solution, or positive results under more structural assumptions on the instance.

5 PSPACE-completeness of the GBF problem

We now prove that the decision problem version of GBF is PSPACE-complete. We present two versions of this result. Theorem 4 is proved by a direct but nontrivial reduction from QBF. However since this reduction yields a GBF instance with only a *single* constraint formula, we present also a second version, Theorem 5, which has a more complex proof but yields a constraint system consisting of disjunctions of width ≤ 6.

Theorem 4. *The problem of deciding GBF-satisfiability of a Boolean formula is PSPACE-complete.*

Proof. The existence of a winning strategy for Player I can clearly be determined in polynomial space by a systematic depth-first min-max search of all the play sequences.

To show PSPACE-hardness we design a reduction from QBF satisfiability. Fix a QBF instance, which consists of a Boolean formula $P(x_1, x_2 \ldots x_n)$. We construct a Boolean formula $\tilde{P}$ where Player I has a GBF winning strategy if and only if P is QBF-satisfiable. The instance $\tilde{P}$ is going to be essentially P embedded with gadgets that force the players to respect the predefined order of variables.

We present explicitly a gadget that requires Player I to start from playing x_i. Given a Boolean formula $R(x_1, \ldots x_n)$, consider a GBF with the formula

$$\begin{aligned} &R^i(a_i, b_i, c_i, d_i, e_i, x_1, \ldots x_{i-1}, x_{i+1}, \ldots x_n) := \\ &[R(x_1, \ldots x_{i-1}, a_i, x_{i+1}, \ldots x_n) \wedge e_i \wedge (a_i \vee c_i) \wedge (b_i \vee d_i)] \vee (a_i \wedge d_i) \vee (c_i \wedge b_i). \end{aligned} \tag{2}$$

For sake of presentation we omit the subscript i from the variables $S_i = \{a_i, b_i, c_i, d_i, e_i\}$.

The idea behind the construction is to have two copies of the variable x_i that end up with equal values, $a = b$. Also the negation $\bar{x}_i$ is represented by two variables, $c = d \neq b$. e is a dummy variable that switches turn. The structure of the gadget guarantees that after Player I fixes a value to x_i, the players continue filling the remaining gadget variables $S_i = \{a, b, c, d, e\}$ accordingly. A deviation from this guideline leads to a quick loss for the deviating player. Also, Player I cannot postpone the decision on the value of variable x_i to a later point without losing the game.

Let us define a generalised version of GBF imposing some restrictions on the order of variables. In a *GBF under queue* $q = (x_{i_1}, x_{i_2}, \ldots x_{i_r})$, $0 \leq r \leq n$, the jth move must be done on the queue variable x_{i_j} as long as $j \leq r$. For $j > r$ the variables can be chosen freely among the unset variables as in standard GBF. Also, define a concatenation of queues as $q \circ q' = (x_{i_1}, \ldots x_{i_r}, x'_{i_1}, \ldots x'_{i'_r})$.

Lemma 1. *Player I has a GBF winning strategy on R^i if and only if Player I has a GBF winning strategy on R under queue (x_i).*

Proof. The proof needs some tedious case-by-case analysis. To prune out obvious inferior moves on R^i, note that the gadget has no negated variables. We observe that Player I is always better off by playing a value *true* than a value *false* in the gadget variables S_i. The opposite holds for Player II. Further, let us encode game positions by concatenating the variables selected by the players, with # denoting any variable outside S_i. For example, the game that started with moves $a = 1;\ x_2 = 0;\ e = 1;\ d = 0$ in this order would be compressed as $a\bar{\#}e\bar{d}$. We also use $*$ as a wild character with no set restrictions.

Consider all possible first player moves.

Case a**:** If Player I starts with a, the threat of $a * d$ forces Player II to continue as $a\bar{d}$. Now, the threat of $a\bar{d} * \bar{b}$ forces Player I to continue as $a\bar{d}b$. The threat $a\bar{d}b *$ c enforces $a\bar{d}b\bar{c}$ and Player I is forced to continue to $a\bar{d}b\bar{c}e$. Variables in S get fixed and the game continues with Player II's turn on the truncated game $R_{i,1} = R(x_1, \ldots x_{i-1}, 1, x_{i+1}, \ldots x_n)$.

Case b**:** As Case a with swapping variables as $a \leftrightarrow b,\ c \leftrightarrow d$. Variables in S_i get fixed and the game continues with Player II's turn on $R_{i,1}$.

Case c**:** As Case a with swapping variables as $a \leftrightarrow c,\ b \leftrightarrow d$. Variables in S_i get fixed and the game continues with Player II's turn on $R_{i,0} = R(x_1, \ldots x_{i-1}, 0, x_{i+1}, \ldots x_n)$.

Case d**:** As Case a with swapping variables as $a \leftrightarrow d,\ b \leftrightarrow c$. Variables in S_i get fixed and the game continues with Player II's turn on $R_{i,0}$.

Case e**:** Player II *can* continue to $e\bar{a}$. $e\bar{a} * \bar{c}$ would be a loss for Player I, hence the third move must lead to $e\bar{a}c$. Player II can reply by $e\bar{a}c\bar{b}$, which enforces $e\bar{a}c\bar{b}d$. Variables in S_i are fixed and the game continues with Player II's turn on $R_{i,0}$. For $e\bar{c}, e\bar{b}$ and ed the analysis is similar with variable swaps, leading to fixed S_i and Player II's turn on $R_{i,0}$. Effectively, Player II decides value for x_i on game R and keeps turn - this is no better for Player I than Case a.

Case #: As Case e, with the exception of sixth move being $\bar{e}$, yielding a loss for Player I.

Collecting the case-by-case results, we conclude that Player I has a winning strategy on R' if and only if Player I wins either of the subgames $R_{i,0}$, $R_{i,1}$ with Player II to move first. This condition is equivalent to Player I having a winning strategy on R that first assigns a value to x_i, and the proof is finished.

The lemma easily yields the following corollaries.

Corollary 3. *Let $i \neq j$. Given a Boolean formula $R(x_1, \ldots x_n)$, Player I has a winning strategy on $(\neg(\neg R)^j)^i$ if and only if Player I has a GBF winning strategy on R under queue (x_i, x_j).*

Proof. By a simple role reversal observation Player I has a winning strategy in a game Q if and only if I has no winning strategy in the game $\neg Q$ that is started by Player II. By Lemma 1 and role reversal I has a winning strategy on $(\neg(\neg R)^j)^i$ if and only if I has no winning strategy on at least one of the subgames $(\neg R_{i,0})^j$, $(\neg R_{i,1})^j$. Applying the lemma and role reversal again, the previous is equivalent to Player I winning either both $R_{i,0;j,0}$ and $R_{i,0;j,1}$, or both $R_{i,1;j,0}$ and $R_{i,1;j,1}$. But this is just the condition that Player I has a GBF winning strategy on R with requirements of Player I playing first x_i and Player II playing x_j immediately thereafter.

Corollary 4. *Let $i \neq j$, $x_i, x_j \notin q$, $|q|$ even. Given a Boolean formula $R(x_1, \ldots x_n)$, Player I has a winning strategy on $(\neg(\neg R)^j)^i$ under q if and only if Player I has a GBF winning strategy on R under queue $q \circ (x_i, x_j)$.*

Proof. Assume there is a winning strategy for either of the cases. This is easily converted to a winning strategy in the other: simply play q as the winning strategy suggests. Consider each of the subgames after q has been played. One of them is known to have a winning strategy, whence by Corollary 3 there must be a winning strategy for the other as well.

Corollary 5. *Given a Boolean formula $P(x_1 \ldots x_n)$, there exists a polynomial-time computable Boolean formula $\tilde{P}(y_1 \ldots y_{5n})$ such that P is QBF-satisfiable if and only if $\tilde{P}$ is GBF-satisfiable.*

Proof. Consider

$$\tilde{P} = (\neg(\neg \ldots (\neg(\neg P)^{n'})^{n'-1} \ldots)^2)^1,$$

where $n' = 2\lfloor \frac{n}{2} \rfloor$. Using Corollary 4 one can transform two outermost gadgets into a queue with winning strategies staying equivalent. Repeated use $\frac{n'}{2}$ times implies that existence of a winning strategy in $\tilde{P}$ is equivalent to existence of a winning strategy in P under queue $q = (x_1, x_2, \ldots, x_{n'})$. But this is the same game as QBF play on P with order $x_1, x_2, \ldots x_n$.

PSPACE-hardness of the GBF-satisfiability for Boolean formulas follows now directly from Corollary 5.

Let us then consider the more restricted GBF instances where the constraint system is in k-conjunctive normal form.

Theorem 5. *Deciding GBF-satisfiability of a system of disjunctions of width ≤ 6 is PSPACE-complete.*

Proof. As in Theorem 4, it is straightforward to see that the problem is in PSPACE.

To establish the PSPACE-hardness of the problem we outline a reduction from the PSPACE-complete *Shannon switching game on vertices (SSG)* [10]. As we shall see, in fact any PSPACE-complete game with an unordered set of binary game variables and a polynomial-time winning predicate would do as a basis for the reduction, but let us for concreteness focus here on the SSG.

An SSG instance is given by an undirected graph $G = (V, E)$ with two distinguished vertices $s, t \in V$. The players take turns in selecting vertices in the graph, excluding s

and t, and Player I wins if she manages to establish an $s-t$ path in G through vertices owned by her. Player II wins if he can keep this from happening until all the vertices have been played. As proved in [10], it is PSPACE-complete to determine if in a given graph (G,s,t) Player I has a winning strategy.

We now show how to effectively construct from an SSG instance (G,s,t) a GBF instance (X,C), where all the clauses in C are disjunctions with at most 6 literals per clause, and Player I has a winning strategy on (G,s,t) if and only if she has a winning strategy on (X,C).

The construction is completely general, and only depends on the high-level characteristics of the SSG game, as indicated earlier. Let the graph G have n vertices, excluding s and t, and let $x_1,\ldots,x_n$ be binary variables indicating whether each vertex i is selected by Player I ($x_i = 1$) or by Player II ($x_i = 0$). Let $W(x_1,\ldots,x_n)$ be a polynomial-time computable predicate indicating whether the resulting configuration is a win for Player I ($W(x) = 1$) or for Player II ($W(x) = 0$).

For a given SSG instance (G,s,t), the predicate $W(x)$ can be effectively expanded into a polynomial-size circuit with, say, p gates. Since $\mathrm{NAND}(x,y) = \overline{x \wedge y} = \bar{x} \vee \bar{y}$ is a universal basis operation for Boolean circuits, we for simplicity and w.l.o.g. assume that all the gates are NAND gates. Let the gates of the W circuit be topologically sorted into a sequence $(g_1,g_2,\ldots,g_p)$, so that each gate g_k computes an intermediate result $z_k \leftarrow \bar{y}_i \vee \bar{y}_j$, where y_i (resp. y_j) is either z_i for $i < k$ (resp. z_j for $j < k$) or one of the input variables $x_1,\ldots,x_n$, and $z_p = 1$ iff $W(x) = 1$.

Let now the variables of the corresponding GBF instance (X,C) be the original input variables $x_1,\ldots,x_n$, together with $2p$ auxiliary variables $z_1,\ldots,z_p$, $a_1,\ldots,a_p$. For each gate g_k of the form $z_k \leftarrow \bar{z}_i \vee \bar{z}_j$ in the W circuit, we introduce a clause

$$c_k \quad = \quad ((z_k \equiv \bar{z}_i \vee \bar{z}_j) \vee (z_k \equiv a_k) \vee (z_i \equiv a_i) \vee (z_j \equiv a_j)).$$

For gates where one or both of the argument literals are input variables, the clauses are similar except that the disjuncts of the form "$x \equiv a$" are omitted for an input variable x.

We observe first that because each clause c_k refers to at most 6 variables, when it is expanded into conjunctive normal form, it expands into some bounded number $d \leq 3^6$ of disjunctions ("miniclauses") $c_{k1},\ldots,c_{kd}$ with at most 6 literals per miniclause, satisfying $c_k \equiv \bigwedge_{l=1}^{d} c_{kl}$.[1] Altogether these miniclauses thus satisfy

$$\bigwedge_{k=1}^{p} c_k \quad \equiv \quad \bigwedge_{k=1}^{p} \bigwedge_{l=1}^{d} c_{kl},$$

and we take as the set of clauses C in our (6-)GBF instance $C := \{c_{kl} \mid k = 1,\ldots,p, l = 1,\ldots,d\}$.

We now claim that Player I has a winning strategy in the SSG instance (G,s,t) if and only if she has a winning strategy in the corresponding GBF instance (X,C). Note that because Player I wins all the clauses c_k if and only if she wins all the miniclauses c_{kl}, we can argue at the level of the clauses c_k. Note also that Player I wins a clause c_k if and only if she can satisfy any one of the disjuncts $(z_k \equiv \bar{z}_i \vee \bar{z}_j)$, $(z_k \equiv a_k)$, $(z_i \equiv a_i)$,

[1] A more careful analysis shows that in fact $c_k \equiv (z_i \vee z_j \vee z_k \vee \bar{a}_i \vee \bar{a}_j \vee \bar{a}_k) \wedge (\bar{z}_i \vee z_j \vee z_k \vee a_i \vee \bar{a}_j \vee \bar{a}_k) \wedge (z_i \vee \bar{z}_j \vee z_k \vee \bar{a}_i \vee a_j \vee \bar{a}_k) \wedge (\bar{z}_i \vee \bar{z}_j \vee \bar{z}_k \vee a_i \vee a_j \vee a_k)$, i.e. $d = 4$.

$(z_j \equiv a_j)$. Let us call the first one the "gate term" and the others the "escape terms", with correspondingly the z_k the "gate variables" and the a_k the "escape variables".

Suppose first that Player I has a winning strategy on the SSG instance (G,s,t). She will follow this strategy first on the input variables $x_1,\ldots,x_n$, and will then assign to the gate variables z_k the values that they would have in the correct evaluation of the W circuit on the given input vector x. It is not necessary to assign these values in the topological order of the gates $(g_1,\ldots,g_p)$, but it is of course natural to do so unless the actions of Player II require otherwise.

Since Player I has a winning strategy on the SSG instance, she will also win on the GBF instance unless Player II can make some of the gate variables have different values than what they would have in a correct evaluation of $W(x)$. Let us consider the first time in the play where Player II does something else than sets one of the input variables $x_1,\ldots,x_n$ or assigns a gate variable z_k to its correct value. There are two similar cases to consider:

Case *a*: Player II "cheats" by assigning a gate variable z_k to a value which is either wrong or not yet determined (because the input sequence x has not yet been completely played out). Then Player I sets the corresponding escape variable a_k to the same value, and hence wins all the clauses c where variable z_k appears by default. Player I repeats this response as many times as Player II cheats. Eventually Player II must return to "fair" play (or all the remaining clauses become satisfied by Player I's escape responses), and then also Player I can return to her basic strategy.

Case *b*: Player II assigns one of the escape variables a_k to some value. Player I sets the corresponding gate variable z_k to the same value, and wins all the clauses c where variable z_k appears by default. Play continues as in Case *a*.

Suppose then that Player I does *not* have a winning strategy in the SSG instance (G,s,t). How could she nevertheless try to win all the clauses c? Let us again consider the first time in the play where Player I does something else than sets one of the input variables $x_1,\ldots,x_n$ or assigns a gate variable z_k to its correct value on a completed input sequence x. There are again two cases to consider:

Case *a*: Player I cheats by assigning a gate variable z_k to a value which is either wrong or not yet determined (because the input sequence x has not yet been completely played out). Then Player II sets the corresponding escape variable a_k to the *opposite* value. This eliminates the escape terms $z_k \equiv a_k$ from all clauses where variable z_k appears, without changing the satisfiability of those clauses otherwise. Hence eventually any inconsistency in the gate terms introduced by Player I will be discovered.

Case *b*: Player I assigns one of the escape variables a_k to some value. If the corresponding gate variable z_k is already assigned, then Player II does nothing, i.e. continues assigning values to the other gate variables in a consistent way. If z_k is still unassigned, then Player II assigns it to the opposite of a_k. Now if $\bar{a}_k$ is the correct value for z_k, then Player I has gained no advantage in the gate variables. If not, then the present Case *b* reduces to Case *a*: again the escape terms $z_k \equiv a_k$ have been eliminated from all clauses where variable z_k appears, without changing their satisfiability otherwise, and the newly introduced inconsistency will eventually lead to a gate term which Player I cannot satisfy.

Since the reduction from the given SSG instance to the corresponding GBF instance can be computed in polynomial time, and winning strategies for Player I are preserved, we thus conclude that the problem of deciding GBF-satisfiability is PSPACE-hard.

6 Conclusion and further work

We have presented what is to our knowledge the first study of constraint satisfaction games where the order of playing the variables is not restricted. We have established a number of positive results concerning the existence of polynomial-time approximate strategies for unordered constraint satisfaction games for specific constraint types. Also we have proved that GBF – the unordered analogue of QBF – is PSPACE-complete. Some of the pertinent open questions include:

1. Can one improve the given performance bounds on the strategies for even-odd games (Theorem 1) or threshold games (Theorems 2 and 3)? Or can the approximate strategies for threshold games be extended to e.g. bigger monotone constraint families?
2. Can one prove *inapproximability* results for unordered games, similar to those achieved in [8, 12] for quantifier-ordered games?
3. What is the smallest value of k such that k-GBF satisfiability is PSPACE-complete? In particular, is 3-GBF satisfiability PSPACE-complete?
4. Can one achieve complexity or approximability taxonomies for unordered constraint satisfaction games based on the characteristics of the constraints involved, along the lines of the taxonomies presented in [14] for CSPs or those in [5, 6, 15] for quantifier-ordered games?

Acknowledgments

This research was supported by the Academy of Finland under grants 128823 (L.A.) and 13136660 (P.O.) Part of the work was done during our visit to the Humboldt-Universität zu Berlin in Autumn 2011, and we thank Sebastian Kuhnert from the Institut für Informatik of HU Berlin for helpful discussions on the topic.

References

1. Alur, R., Henzinger, T.A., Kupferman, O.: Alternating-time temporal logic. J. ACM 49(5), 672–713 (2002)
2. Ansótegui, C., Gomes, C.P., Selman, B.: The Achilles' heel of QBF. In: Proc. 20th Natl. Conf. on Artificial Intelligence and 17th Conf. on Innovative Applications of Artificial Intelligence (AAAI/IAAI 2005). pp. 275–281 (2005)
3. Benedetti, M., Lallouet, A., Vautard, J.: QCSP made practical by virtue of restricted quantification. In: Proc. 20th Intl. Joint Conf. on Artificial Intelligence (IJCAI 2007). pp. 38–43 (2007)
4. Benedetti, M., Lallouet, A., Vautard, J.: Quantified constraint optimization. In: Proc. 14th International Conference on Principles and Practice of Constraint Programming (CP 2008). Lecture Notes in Computer Science, vol. 5202, pp. 463–477. Springer (2008)

5. Börner, F., Bulatov, A.A., Chen, H., Jeavons, P., Krokhin, A.A.: The complexity of constraint satisfaction games and QCSP. Inf. Comput. 207(9), 923–944 (2009)
6. Chen, H.: The complexity of quantified constraint satisfaction: Collapsibility, sink algebras, and the three-element case. SIAM J. Comput. 37(5), 1674–1701 (2008)
7. Chen, H., Pál, M.: Optimization, games, and quantified constraint satisfaction. In: Proc. 29th Intl. Symp. on Mathematical Foundations of Computer Science (MFCS 2004). Lecture Notes in Computer Science, vol. 3153, pp. 239–250. Springer (2004)
8. Condon, A., Feigenbaum, J., Lund, C., Shor, P.W.: Probabilistically checkable debate systems and nonapproximability of PSPACE-hard functions. Chicago J. Theor. Comput. Sci. 1995, 4 (1995)
9. Demaine, E.D.: Playing games with algorithms: Algorithmic combinatorial game theory. In: Proc. 26th Intl. Conf. on Mathematical Foundations of Computer Science (MFCS 2001). pp. 18–32 (2001)
10. Even, S., Tarjan, R.E.: A combinatorial problem which is complete in polynomial space. J. ACM 23(4), 710–719 (1976)
11. Fraenkel, A.S.: Complexity, appeal and challenges of combinatorial games. Theor. Comput. Sci. 303(3), 393–415 (2004)
12. Hunt III, H.B., Marathe, M.V., Stearns, R.E.: Complexity and approximability of quantified and stochastic constraint satisfaction problems. Electronic Notes in Discrete Mathematics 9, 217–230 (2001)
13. Johnson, D.S.: Approximation algorithms for combinatorial problems. J. Comput. Syst. Sci. 9(3), 256 – 278 (1974)
14. Khanna, S., Sudan, M., Trevisan, L., Williamson, D.P.: The approximability of constraint satisfaction problems. SIAM J. Comput. 30(6), 1863–1920 (2000)
15. Madelaine, F.R., Martin, B.: A tetrachotomy for positive first-order logic without equality. In: Proc. 26th Ann. IEEE Symp. on Logic in Computer Science (LICS 2011). pp. 311–320 (2011)
16. Ramadge, P., Wonham, W.: The control of discrete event systems. Proc. of the IEEE 77(1), 81–98 (1989)
17. Schaefer, T.J.: On the complexity of some two-person perfect-information games. J. Comput. Syst. Sci. 16(2), 185–225 (1978)
18. Stockmeyer, L.J., Meyer, A.R.: Word problems requiring exponential time: Preliminary report. In: Proc. 5th Ann. ACM Symp. on Theory of Computing (STOC 1973). pp. 1–9 (1973)