Constraint-Based Symmetry Detection in General Game Playing

Paper 1939

Abstract

Symmetry detection is a promising approach for reducing the search tree of games. In General Game Playing (GGP), where any game is compactly represented by a set of rules in the Game Description Language (GDL), the state-of-the-art methods for symmetry detection rely on a *rule graph* associated with the GDL description of the game. Though such rule-based symmetry detection methods can be applied to various tree search algorithms, they cover only a limited number of symmetries which are apparent in the GDL description. In this paper, we develop an alternative approach to symmetry detection in stochastic games that exploits constraint programming techniques. The minimax optimization problem in a GDL game is cast as a stochastic constraint satisfaction problem (SCSP), which can be viewed as a sequence of *one-stage* SCSPs. Minimax symmetries are inferred according to the microstructure complement of these one-stage constraint networks. Based on a theoretical analysis of this approach, we experimentally show on various games that the recent stochastic constraint solver MAC-UCB, coupled with constraint-based symmetry detection, significantly outperforms the standard Monte Carlo Tree Search algorithms, coupled with rule-based symmetry detection. This constraint-driven approach is also validated by the excellent results obtained by our player during the last GGP competition.

1 Introduction

The topic of *General Game Playing* (GGP) is to develop artificial agents capable of playing a wide variety of games, without any human intervention [Genesereth *et al.*, 2005]. In a GGP tournament, the rules of an unknown game are supplied to the agent and, after a short period of deliberation, the agent must be able to play this game effectively against other players. The game rules are represented in the *Game Description Language* (GDL) [Love *et al.*, 2008]. Basically, a GDL program is a set of first-order logical clauses specifying the initial state of the game, the legal moves of each player, the effects of these moves, and the scores obtained at terminal states. Recent versions of GDL [Schiffel and Thielscher, 2014; Thielscher, 2016] are expressive enough to cover various classes of finite-horizon games including, in particular, *stochastic games* in which a distinguished "chance" player is used to simulate the probabilistic effects of joint actions.

Conceptually, any turn-based two-player zero-sum (deterministic or stochastic) game has an optimal, *minimax value function* that specifies the expected outcome of the game, for every possible state, under perfect play by all players. In theory, such games may be solved by recursively computing the value function in a search tree of size b^d, where b is the number of legal moves per state, and d is the number of moves needed to reach a terminal state. Yet, even for games of moderate size, exhaustive search is unfeasible in GGP tournaments, due to the short deliberation time allowed to players.

To this point, *symmetry detection* is a popular inference method for reducing exploration in a search space, by transferring knowledge between equivalent regions of this space. As games typically involve many equivalent states and moves, such similarities can be exploited for transferring the value function between nodes of the search tree. Furthermore, since most symmetry detection methods are implemented using graph automorphism algorithms [McKay and Piperno, 2014], the main component for inferring game symmetries is a *graphical structure* from which a permutation group is induced. Ideally, this permutation group should be as close as possible to the group of *minimax symmetries*, which preserve the optimal strategies of the game.

Most approaches to symmetry detection in GGP rely on some *rule graph* associated with the GDL representation [Kuhlmann and Stone, 2007; Schiffel, 2010; Zhang *et al.*, 2015]. Basically, a rule graph contains nodes for the terms, atoms, and rules of the GDL representation, together with edges connecting these symbols. Any automorphism of this rule graph is a symmetry that preserves the game semantics. However, only a restricted subset of game symmetries can be recognized from such automorphisms, due to the *syntactical bias* imposed by the GDL representation.

In this paper, we propose an alternative approach to symmetry detection in stochastic games, inspired from constraint programming techniques. A *Constraint Satisfaction Problem* (CSP) consists in a set of variables, each associated with a domain of values, and a set of constraints specifying allowed combinations of values for subsets of

variables. A CSP solution is an assignment of variables to values that satisfies all constraints. A common approach for detecting solution symmetries in CSPs is to use the *microstructure complement* of the network [Jégou, 1993; Cohen *et al.*, 2006]. This structure is a hypergraph, whose nodes are associated with variable-value pairs, and whose hyperedges joint sets of nodes which are "disallowed" by some constraint. Any automorphism of this hypergraph, called *constraint symmetry*, is a solution symmetry.

Since stochastic games are *sequential* decision problems, they generally cannot be reduced to CSPs. Instead, our approach is to encode a GDL program and its minimax objective function into a *Stochastic Constraint Satisfaction Problem* (SCSP) [Walsh, 2002; Tarim *et al.*, 2006; Hnich *et al.*, 2012]. This problem involves variables, domains and constraints, but the variables are partitioned into *decision* variables, encoding state descriptions and players' actions, and *stochastic* variables, encoding the actions of the chance player. Any solution of the SCSP is a minimax strategy of the GDL game. By construction, such a network can be viewed as a sequence of *one-stage* SCSPs, each specifying a game turn. Our symmetry detection method uses the microstructure complement of these one-stage SCSPs, from which constraint symmetries form a subgroup of the group of minimax symmetries. Importantly, these microstructure complements can also be exploited for detecting symmetries between *approximate* minimax strategies, which are found by combining constraint propagation and bandit exploration.

We provide an experimental evaluation of this approach using the MAC-UCB algorithm for solving SCSPs [Koriche *et al.*, 2016]. We show on various games that MAC-UCB, coupled with constraint-based symmetry detection, significantly outperforms Monte Carlo Tree Search algorithms, coupled with rule-based symmetry detection. We also emphasize that this approach was implemented in the "Woodstock" player, which has won the 2016 International GGP competition.

2 Games and Symmetries

2.1 Stochastic Games

The problems under consideration in this study are finite-horizon combinatorial games that incorporate probabilistic choices. A *game signature* consists in a tuple (I, F, A), where $I = \{1, \cdots, k\}$ is the set of *players*, F is a finite set of *fluents* used to describe game states, and A is a finite set of *actions* or *moves*. A *stochastic game* over the signature (I, F, A) is a tuple $\mathcal{G} = (s_1, S_\dagger, L, P, u)$, where:

- $s_1 \in 2^F$ is the *initial state*, and $S_\dagger \subseteq 2^F$ is the set of terminal states,
- $L : I \times 2^F \to 2^A$ maps each player i and each state s to the set $L_i(s)$ of *legal moves* of i at s,
- $P : 2^F \times A^k \times 2^F \to [0, 1]$ maps each state s and joint action $\boldsymbol{a} \in \boldsymbol{L}(s)$ to a probability distribution $P(\cdot \mid s, \boldsymbol{a})$ over 2^F, where $\boldsymbol{L}(s) = L_1(s) \times \cdots \times L_k(s)$,
- $u : I \times S_\dagger \to \mathbb{R}$ maps each player i and each terminal state s to the utility $u_i(s)$ of i at s.

A *history* (or *finite play*) of length T in $\mathcal{G}$ is a sequence of the form $(s_1, \boldsymbol{a}_1, s_2, \cdots, s_{T-1}, \boldsymbol{a}_T, s_T)$ where $\boldsymbol{a}_t \in \boldsymbol{L}(s_t)$ and $P(s_{t+1} \mid s_t, \boldsymbol{a}_t) > 0$, for $t \in \{1, \cdots, T-1\}$. $\mathcal{G}$ is called a *T-horizon game* if every history of length T in $\mathcal{G}$ includes a terminal state in $S_\dagger$, and conversely, every terminal state in $S_\dagger$ is included in some history of length T in $\mathcal{G}$.

As usual, a pure stationary strategy is a map from states to actions. Of particular interest in this study is the *minimax* strategy for which the value function (for player i) is

$$V_i^*(s) = \begin{cases} \max_{a_i} \min_{\boldsymbol{a}_{-i}} Q_i^*(s, \boldsymbol{a}) & \text{if } s \notin S_\dagger \\ u_i(s) & \text{otherwise} \end{cases}$$

$$\text{where } Q_i^*(s, \boldsymbol{a}) = \sum_{s' \in S} P(s' \,|\, s, \boldsymbol{a}) V_i^*(s') \tag{1}$$

and where a_i ranges over $L_i(s)$, and $\boldsymbol{a}_{-i}$ ranges over the projection of $\boldsymbol{L}(s)$ onto $I \setminus \{i\}$. This strategy is optimal for the important class of *turn-based two-player zero-sum* stochastic games [Condon, 1992], given by $k = 2$, $u_1(s) = -u_2(s)$ for $s \in S_\dagger$, and $L_1(s) = \{\top\}$ or $L_2(s) = \{\top\}$ for $s \in S_{\mathcal{G}}$, where $\top$ is a distinguished "noop" action with no effects.

Evaluating the minimax value function in combinatorial games is generally intractable. A natural approach to alleviate this issue is to approximate V_i^* by a "depth-bounded" minimax value that limits the recursion to some fixed depth d, and applies a heuristic evaluation function when d is reached [Lanctot *et al.*, 2013]. Given a map $\hat{V}_i : S \to \mathbb{R}$ such that $\hat{V}_i(s) = u_i(s)$ when $s \in S_\dagger$, the *depth-d minimax* strategy is

$$V_i^d(s) = \begin{cases} \max_{a_i} \min_{\boldsymbol{a}_{-i}} Q_i^d(s, \boldsymbol{a}) & \text{if } d > 0 \text{ and } s \notin S_\dagger \\ \hat{V}_i(s) & \text{otherwise} \end{cases}$$

$$\text{where } Q_i^d(s, \boldsymbol{a}) = \sum_{s' \in S} P(s' \,|\, s, \boldsymbol{a}) V_i^{d-1}(s'). \tag{2}$$

2.2 Game Symmetries

Intuitively, a symmetry in a set O of objects is a permutation over O that leaves some property of those objects unchanged. The particular group of symmetries that we obtain depends on the property we choose to preserve. In this paper, two main types of game symmetries are distinguished: *structural* symmetries that preserve the game structure, and *minimax* symmetries that preserve the minimax strategies of the game. Given a stochastic game $\mathcal{G}$ with signature (I, F, A), a structural symmetry of $\mathcal{G}$ is a bijective function σ on $I \cup F \cup A$ such that for all $i \in I$, $f \in F$, $s, s' \in S_{\mathcal{G}}$, and $\boldsymbol{a} \in A^k$,

$$\sigma(i) = i, \sigma(a) \in A, \sigma(f) \in F \tag{3a}$$
$$\sigma(f) \in s_1 \text{ iff } f \in s_1 \tag{3b}$$
$$\sigma(s) \in S_\dagger \text{ iff } s \in S_\dagger \tag{3c}$$
$$\sigma(a) \in L_i(\sigma(s)) \text{ iff } a \in L_i(s) \tag{3d}$$
$$P(\sigma(s') \mid \sigma(s), \sigma(\boldsymbol{a})) = P(s' \mid s, \boldsymbol{a}) \tag{3e}$$
$$u_i(\sigma(s)) = u_i(s) \tag{3f}$$

where $\sigma(s) = \{\sigma(f) \,|\, f \in s\}$, and $\sigma(\boldsymbol{a}) = (\sigma(a_1), \cdots, \sigma(a_k))$. A minimax symmetry is a bijective function σ over $I \cup F \cup A$ satisfying conditions (3a-3c) together with

$$V_i^*(\sigma(s)) = V_i^*(s) \tag{4}$$

for all $i \in I$ and $s \in S_{\mathcal{G}}$. By extension, *depth-d minimax symmetries* are defined by replacing (4) with

$$V_i^d(\sigma(s)) = V_i^d(s) \tag{5}$$

Any structural symmetry is a minimax symmetry, since the conditions (3d-3f) imply that two states with equivalent subgames must have the same minimax value. Yet, structural symmetries and depth-d minimax symmetries are generally incomparable, excepted for $d = T$ in T-horizon games.

3 Rule-Based Symmetry Detection

3.1 GDL Representations

In the GGP setting, games are represented by first-order stratified logic programs, using the GDL language that includes several keywords specifying the game dynamics. Stochastic games are captured by an extension of GDL that describes the chance player `random` [Schiffel and Thielscher, 2014].

Example 1. In the standard Tic-Tac-Toe game, two players 1 and 2 take turns by marking the cells in a grid until a row, column or diagonal is filled by a player. We focus here on a randomized variant of this game for the 2×2 grid; the players 1 and 2 can only choose a row or a column, and the exact position in the chosen line is drawn at random by the chance player. For instance, the rules

$$\texttt{next}(\texttt{row}(\texttt{X},\texttt{I})) \leftarrow \texttt{does}(\texttt{I},\texttt{chooseRow}(\texttt{X}))$$
$$\texttt{next}(\texttt{col}(\texttt{Y},\texttt{I})) \leftarrow \texttt{does}(\texttt{I},\texttt{chooseCol}(\texttt{Y}))$$
$$\texttt{next}(\texttt{cell}(\texttt{X},\texttt{Y},\texttt{I})) \leftarrow \texttt{does}(\texttt{random},\texttt{mark}(\texttt{X},\texttt{Y},\texttt{I}))$$

specify the effects of players' actions. Namely, if a player `I` selects a row `X` (resp. a column `Y`) in the current state, then the fluent $\texttt{row}(\texttt{X},\texttt{I})$ (resp. $\texttt{col}(\texttt{Y},\texttt{I})$) is true in the next state, and if the chance player `random` marks a position $(\texttt{X},\texttt{Y})$ with player `I` in the current state, then the fluent $\texttt{cell}(\texttt{X},\texttt{Y},\texttt{I})$ is true in the next state. Alternatively, the rules:

$$\texttt{legal}(\texttt{random},\texttt{mark}(\texttt{X},\texttt{Y},\texttt{I})) \leftarrow \texttt{true}(\texttt{row}(\texttt{X},\texttt{I})),\texttt{true}(\texttt{cell}(\texttt{X},\texttt{Y},*)) \quad (6a)$$

$$\texttt{legal}(\texttt{random},\texttt{mark}(\texttt{X},\texttt{Y},\texttt{I})) \leftarrow \texttt{true}(\texttt{col}(\texttt{Y},\texttt{I})),\texttt{true}(\texttt{cell}(\texttt{X},\texttt{Y},*)) \quad (6b)$$

capture the legal moves of the chance player: marking a cell $(\texttt{X},\texttt{Y})$ with `I` is possible if the position $(\texttt{X},\texttt{Y})$ is available $(*)$, and the row `X` or column `Y` is chosen by player `I`.

For a set `R` of GDL rules, the *dependency graph* of `R` associates a node with each literal `L` in `R`, and an arc $(\texttt{L},\texttt{L}')$ if `L` occurs in the body of some rule with head $\texttt{L}'$.

Because GDL is Turing complete [Saffidine, 2014], even for "valid" programs [Love *et al.*, 2008], we focus here on a *propositional* fragment of this language (including `random`), denoted P-GDL, satisfying two properties:

(7a) the number of ground instances of functional terms and the size of clauses are *bounded*, and

(7b) the dependency graph of the program is *acyclic*.

Any P-GDL program `R` can be rewritten into an equivalent ground instance $\texttt{R}_g$ of size polynomial in $|\texttt{R}|$. The ground terms in $\texttt{R}_g$ are partitioned into a set $I^+ = \{0, 1, \cdots, k\}$ of players, where 0 denotes the chance player, a set F of fluent terms, a set A of action terms, and a set U of utility constants. Similarly, the rules in $\texttt{R}_g$ can be partitioned into six categories:

$$\texttt{role}(i)$$
$$\texttt{init}(f)$$
$$\texttt{terminal} \leftarrow \texttt{L}_1, \cdots, \texttt{L}_q$$
$$\texttt{legal}(i,a) \leftarrow \texttt{L}_1, \cdots, \texttt{L}_{q'}$$
$$\texttt{next}(f) \leftarrow \texttt{does}(0,a_0), \cdots, \texttt{does}(k,a_k)$$
$$\texttt{goal}(i,\nu) \leftarrow \texttt{L}_1, \cdots, \texttt{L}_{q''}$$

where $i \in I^+$, $\nu \in U$, $f \in F$ and and $a, a_0, \cdots, a_k \in A$. Furthermore, each $\texttt{L}_j$ is an atom $\texttt{A}_j$ or its negation $\texttt{not}\,\texttt{A}_j$, where $\texttt{A}_j$ is an expression of the form $\texttt{true}(f)$ or $f_1 \neq f_2$.

The *Clark's completion* [Clark, 1978] of $\texttt{R}_g$ is the set of propositional formulas $\texttt{R}_c$, where (i) each symbol `not` in $\texttt{R}_g$ is replaced with $\neg$, (ii) each atom `A` occurring as a head of some rule is associated with its set $\texttt{B}_\texttt{A}$ of bodies, i.e.

$$\texttt{B}_\texttt{A} = \{\texttt{L}_1 \wedge \cdots \wedge \texttt{L}_q \mid \texttt{A} \leftarrow \texttt{L}_1, \cdots, \texttt{L}_q \in \texttt{R}_g\}$$

and (iii) for each atom `A`, all rules with head `A` are replaced with a single formula $\texttt{A} \leftrightarrow \bigvee \texttt{B}_\texttt{A}$. For an atom `A`, we use $\texttt{R}_g \vdash \texttt{A}$ to denote that `A` is true in the (unique) stable model of $\texttt{R}_g$, and we use $\texttt{R}_c \models \texttt{A}$ to denote that $\texttt{R}_c \wedge \neg\texttt{A}$ is unsatisfiable. By Corollary 4.21 in [Ben-Eliyahu and Dechter, 1994], if $\texttt{R}_g$ satisfies (7b), we must have $\texttt{R}_g \vdash \texttt{A}$ iff $\texttt{R}_c \models \texttt{A}$, that is, $\texttt{R}_g$ and $\texttt{R}_c$ are *logically equivalent*.

Let $\mathcal{G}(\texttt{R}) = (s_1, S_\dagger, L, P, u)$ be the stochastic game over the signature (I, F, A), where $I = I^+ \setminus \{0\}$, and such that

$$f \in s_1 \text{ iff } \texttt{R}_g \vdash \texttt{init}(f),$$
$$s \in S_\dagger \text{ iff } \texttt{R}_g \cup \texttt{true}(s) \vdash \texttt{terminal},$$
$$a \in L_i(s) \text{ iff } \texttt{R}_g \cup \texttt{true}(s) \vdash \texttt{legal}(i,a),$$
$$u_i(s) = \nu \text{ iff } \texttt{R}_g \cup \texttt{true}(s) \vdash \texttt{goal}(i,\nu), \text{ and}$$
$$P(s' \mid s, \boldsymbol{a}) = \tfrac{1}{|L_0(s)|} \text{ if } s' \in \rho(s,\boldsymbol{a}), \text{ and 0 otherwise.}$$

Here, $\texttt{true}(s) = \{\texttt{true}(f) \mid f \in s\}$, and $\rho(s,\boldsymbol{a})$ is the set of states s' for which there is an action $a_0 \in L_0(s)$, such that for all $f' \in s'$, $\texttt{R}_g \cup \texttt{true}(s) \cup \{\texttt{does}(i,a_i)\}_{i=0}^k \vdash \texttt{next}(f')$. Thus, $P(\cdot \mid s, \boldsymbol{a})$ is a uniform distribution over the states updated from s, using the moves $(a_0, \boldsymbol{a})$ coupling the action profile $\boldsymbol{a}$ of players, and all possible actions $a_0 \in L_0(s)$.

Example 2. In the ground instance $\texttt{R}_g$ of the GDL program `R` describing the randomized Tic-Tac-Toe game for the 2×2 grid, the rules (6a) and (6b) are each replaced with 8 ground formulas, obtained by instantiating the variables $\texttt{X} \in \{\texttt{x}_1, \texttt{x}_2\}$, $\texttt{Y} \in \{\texttt{y}_1, \texttt{y}_2\}$, and $\texttt{I} \in \{1, 2\}$. Notably, in the ground formulas

$$\texttt{legal}(\texttt{random},\texttt{mark}(\texttt{x}_1,\texttt{y}_2,1)) \leftarrow \texttt{true}(\texttt{row}(\texttt{x}_1,1)),\texttt{true}(\texttt{cell}(\texttt{x}_1,\texttt{y}_2,*)) \quad (8a)$$

$$\texttt{legal}(\texttt{random},\texttt{mark}(\texttt{x}_1,\texttt{y}_2,1)) \leftarrow \texttt{true}(\texttt{col}(\texttt{y}_2,1)),\texttt{true}(\texttt{cell}(\texttt{x}_1,\texttt{y}_2,*)) \quad (8b)$$

$\texttt{row}(\texttt{x}_1,1)$, $\texttt{col}(\texttt{y}_2,1)$ and $\texttt{cell}(\texttt{x}_1,\texttt{y}_2,*)$ are fluent terms, and $\texttt{mark}(\texttt{x}_1,\texttt{y}_2,1)$ is an action term. If we assume that in the current state s, the first player has chosen row $\texttt{x}_1$, and the number of free positions in $\texttt{x}_1$ is y, then the probability distribution $P(\cdot \mid s, \boldsymbol{a})$ over the next states using the action profile $\boldsymbol{a} = (\texttt{mark}(\texttt{x}_1,\texttt{y}_2,1), \top, \top)$ is simply $1/y$.

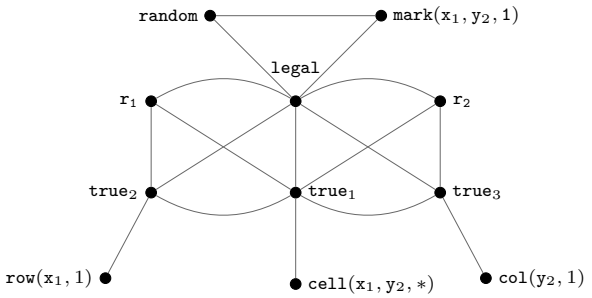

Figure 1: The ground rule graph of formulas (8a) and (8b).

3.2 Rule Graphs

For the propositional fragment of GDL examined in this study, we use a simplified version of the (enhanced) rule graph specified in [Schiffel, 2010]. Formally, the *ground rule graph* of a P-GDL program R is the colored graph $G = (N, E, \chi)$ over $\mathtt{R}_g$, such that

- each ground term $\mathtt{t} \in I^+ \cup F \cup A \cup U$ is mapped to a distinct node $n_\mathtt{t}$; if t is a player $i \in I^+$, then $\chi(n_\mathtt{t}) = i$, if t is a utility value $\nu \in U$, then $\chi(n_\mathtt{t}) = \nu$, if t is a fluent term in F, then $\chi(n_\mathtt{t}) = \mathtt{fluent}$, and if t is an action term in A, then $\chi(n_\mathtt{t}) = \mathtt{action}$;
- each atom A of the form $\mathtt{p}(\mathtt{t}_1, \cdots, \mathtt{t}_q)$ is mapped to a distinct node $n_\mathtt{A}$ with color $\chi(n_\mathtt{A}) = \mathtt{p}$, and a clique over $\{n_\mathtt{A}, n_{\mathtt{t}_1}, \cdots, n_{\mathtt{t}_p}\}$;
- each literal L of the form not A is mapped to a distinct node $n_\mathtt{L}$ with $\chi(n_\mathtt{L}) = \mathtt{not}$, and an edge $(n_\mathtt{L}, n_p)$;
- each rule r of the form $\mathtt{A} \leftarrow \mathtt{L}_1, \cdots, \mathtt{L}_q$ is mapped to a distinct node $n_\mathtt{r}$ with color $\chi(n_\mathtt{r}) = \mathtt{rule}$, and a clique over $\{n_\mathtt{r}, n_\mathtt{A}, n_{\mathtt{L}_1}, \cdots, n_{\mathtt{L}_q}\}$.

Proposition 1. *Let* R *be a* P-GDL *program and* G *be its ground rule graph. Then, any automorphism of* G *is a structural symmetry of* $\mathcal{G}(\mathtt{R})$.

Example 3. As depicted in Figure 1, the ground rule graph of the ground formulas (8a) and (8b) includes two similar four-cliques. An automorphism of this graph can be obtained by permuting the nodes $\mathtt{r}_1$, $\mathtt{true}_2$ and $\mathtt{row}(\mathtt{x}_1, 1)$, with $\mathtt{r}_2$, $\mathtt{true}_3$ and $\mathtt{col}(\mathtt{y}_2, 1)$, respectively.

Note that the group of automorphisms of G depends on how the game is described by $\mathtt{R}_g$. For example, suppose that two actions a and a' have the same preconditions and effects, but $\mathtt{R}_g$ uses two rules $\mathtt{legal}(i, a) \leftarrow \mathtt{A}$ and $\mathtt{legal}(i, a) \leftarrow \mathtt{not\ A}$ for action a, and one fact $\mathtt{legal}(i, a')$ for a'. Then, no automorphism in G can permute a and a' because the neighbors of n_a and $n_{a'}$ are not equivalent.

4 Constraint-Based Symmetry Detection

The key idea of this study is to translate GDL programs into stochastic constraint networks, from which the microstructure complement can be used for detecting various types of symmetries. We focus on a slight generalization of the original SCSP model [Walsh, 2002] that incorporates conditional probability distributions over stochastic variables.

Formally, a *Stochastic Constraint Satisfaction Problem* is a tuple $\mathcal{N} = (X, Y, D, C, P, \theta)$, where

- $X = (x_1, \cdots, x_n)$ is an ordered finite set of variables,
- $Y \subseteq X$ is the set of *stochastic variables*,
- $D = \{D(x_i)\}_{i=1}^n$ is a set of *finite domains* each associated with a variable in X,
- C is a set of *constraints*,
- $P = \{P_y\}_{y \in Y}$ is a set of *conditional probability tables* each associated with a variable in Y, and
- $\theta \in [0, 1]$ is a threshold value.

Any variable in $X \setminus Y$ is called a *decision variable*. Given an ordered subset $Z = (z_1, \cdots, z_m)$ of variables in X, we denote by $D(Z)$ the relation $D(z_1) \times \cdots \times D(z_m)$. An *instantiation* on Z is a tuple $\boldsymbol{v} \in D(Z)$, also written as a set of pairs variable-value $\{(z_1, v_1), \cdots, (z_m, v_m)\}$. The instantiation $\boldsymbol{v}$ is *complete* if $Z = X$. For a subset $Z' \subseteq Z$, we use $\boldsymbol{v}_{|Z'}$ to denote the projection $\{(z_i, v_i) : z_i \in Z'\}$ of $\boldsymbol{v}$ onto Z'.

As usual, each constraint $c \in C$ is defined over a set of variables $scp_c \subseteq X$, called the *scope* of c, and consists in a mapping from $D(scp_c)$ to $\{0, 1\}$. Any instantiation $\boldsymbol{v}$ on scp_c for which $c(\boldsymbol{v}) = 1$ (resp. $c(\boldsymbol{v}) = 0$) is called *allowed* (resp. *disallowed*). Each conditional probability table P_y is defined over a set of variables $scp_y \subseteq X$ occurring *before* y in X, and maps every instantiation $\boldsymbol{v}$ on scp_y to a probability distribution $P_y(\cdot \mid \boldsymbol{v})$ over $D(y)$.

The probability $P(\boldsymbol{v})$ and the consistency $C(\boldsymbol{v})$ of a complete instantiation $\boldsymbol{v}$ are respectively given by

$$P(\boldsymbol{v}) = \prod_{y \in Y} P_y(\boldsymbol{v}_{|\{y\}} \mid \boldsymbol{v}_{|scp_y}) \text{ and } C(\boldsymbol{v}) = \prod_{c \in C} c(\boldsymbol{v}_{|scp_c})$$

An instantiation $\boldsymbol{v}$ is *globally consistent* (GC) if it can be extended to a complete instantiation $\boldsymbol{v}'$ such that $C(\boldsymbol{v}') = 1$.

A *policy* π for $\mathcal{N}$ is a tree, in which nodes are labeled according to the ordering X; the root is labeled with x_1, and each child of a node x_i is labeled with x_{i+1}. Each outgoing edge of x_i is labeled with a value in $D(x_i)$; decision nodes have a unique child, and stochastic nodes y_i have $|D(y_i)|$ children. Each leaf is labeled with the value $C(\boldsymbol{v})$ of the complete instantiation $\boldsymbol{v}$ connecting that leaf to the root. The *expected consistency* of a policy π is given by

$$C(\pi) = \sum_{\boldsymbol{v}} P(\boldsymbol{v}) C(\boldsymbol{v})$$

where $\boldsymbol{v}$ ranges over the root to leaf paths of π. A *solution* of $\mathcal{N}$ is a policy π such that $C(\pi) \geq \theta$. Note that in the particular case where $\theta = 1$, π is a solution of $\mathcal{N}$ iff every instantiation in π is GC.

Finally, $\mathcal{N}$ is *T-stage* SCSP if X is partitioned into T stages, i.e. $X = (Z_1, Y_1, \cdots, Z_T, Y_T)$, where $(Z_1, \cdots, Z_T)$ is a partition of $X \setminus Y$, and $(Y_1, \cdots, Y_T)$ is a partition of Y.

4.1 From GDL to SCSP

Based on the framework suggested in [Koriche *et al.*, 2016], any P-GDL program $\mathtt{R}$ can be encoded into a multi-stage SCSP, where each stage simulates a game turn. Formally, let $\mathtt{R}_g$ be the ground instance of $\mathtt{R}$, where $I^+ = \{0, 1, \cdots, k\}$ is the set of players, A is the set of actions, F is the set of fluents, and U is the set of utility values. Let $\mathtt{R}_c$ be the Clark's completion of $\mathtt{R}_g$, and T be a positive integer. Then, the T-stage SCSP of $\mathtt{R}$ is the tuple $\mathcal{N}_{\mathtt{R},T} = (X, Y, D, C, P, \theta)$ such that $\theta = 1$, and

- $X = (Z_1, Y_1, \cdots, Z_T, Y_T)$, where each Z_t includes a Boolean variable ω_t indicating the presence of a terminal state, a set of Boolean variables $\{\mathtt{f}_{j,t}\}_{j=1}^n$, each associated with a distinct fluent in F, a set of variables $\{\mathtt{a}_{i,t}\}_{i=1}^k$ with domain A, each associated with a distinct player i, and a set of variables $\{\mathtt{u}_{i,t}\}_{i=1}^k$ with domain U each associated with a player i. Each Y_t includes a single variable $\mathtt{a}_{0,t}$ with domain A, associated with $\mathtt{random}$;
- $C = C_1 \cup \cdots \cup C_T$, where C_1 includes constraints encoding the unary clauses over $\mathtt{init}$ in $\mathtt{R}_c$, and for every stage t, C_t includes a constraint $c_{\mathtt{A},t}$ encoding the rule $\mathtt{A} \leftrightarrow \bigvee \mathtt{B}_\mathtt{A}$ in $\mathtt{R}_c$. Constraints over $\mathtt{legal}$ for the same player i are merged into a single constraint $\mathtt{legal}_{i,t}$;
- $P = \{P_1, \cdots, P_T\}$, where P_t encodes the conditional probability distribution of $\mathtt{a}_{t,0}$ at stage t. The scope of P_t is given by the scope $\{\mathtt{f}_{j,t}\}_{j=1}^m$ of the constraint $\mathtt{legal}_{0,t}$ (minus $\mathtt{a}_{0,t}$), and for each instantiation $\boldsymbol{v}$ on this scope, $P_t(\cdot \mid \boldsymbol{v})$ is the uniform distribution over the set legal moves $\{a \in A \mid \mathtt{legal}_{0,t}(a, \boldsymbol{v}) = 1\}$.

Based on this construction, we can observe that for each stage $X_t = (Z_t, Y_t)$, the scopes of the constraints $\mathtt{terminal}_t$, $\mathtt{legal}_{i,t}$ and $\mathtt{goal}_{i,t}$, together with the scope of the table P_t, are all covered by the variables in X_t. Thus, the only component that connects a stage and its successor is the constraint $\mathtt{next}_t$, which determines the values of fluents at turn $t+1$, given the actions simultaneously played in turn t.

Example 4. Consider again the randomized Tic-Tac-Toe game for the 2×2 grid. By Clark's completion, the ground formulas (8a) and (8b) are merged into a single formula:

$$\begin{aligned}&\mathtt{legal}(\mathtt{random}, \mathtt{mark}(\mathtt{x}_1, \mathtt{y}_2, 1)) \leftrightarrow \\ &\mathtt{true}(\mathtt{row}(\mathtt{x}_1, 1)), \mathtt{true}(\mathtt{cell}(\mathtt{x}_1, \mathtt{y}_2, *)) \vee \\ &\mathtt{true}(\mathtt{col}(\mathtt{y}_2, 1)), \mathtt{true}(\mathtt{cell}(\mathtt{x}_1, \mathtt{y}_2, *))\end{aligned}$$

At each stage t, this formula is encoded into a $\mathtt{legal}_{0,t}$ constraint with scope $\mathtt{mark}_{0,t}$, $\mathtt{row}(\mathtt{x}_1, 1)_t$, $\mathtt{col}(\mathtt{y}_2, 1)_t$ and $\mathtt{cell}(\mathtt{x}_1, \mathtt{y}_2, *)_t$. The relation of this constraint states that the value $(\mathtt{x}_1, \mathtt{y}_2, 1)$ is allowed for the action variable $\mathtt{mark}_{0,t}$ iff the fluent variable $\mathtt{cell}(\mathtt{x}_1, \mathtt{y}_2, *)_t$ is true, and at least one of the fluent variables $\mathtt{row}(\mathtt{x}_1, 1)_t$ and $\mathtt{col}(\mathtt{y}_2, 1)_t$ is true. Similarly, by Clark's completion, the ground formula

$$\mathtt{next}(\mathtt{cell}(\mathtt{x}_1, \mathtt{y}_2, 1)) \leftarrow \mathtt{does}(\mathtt{random}, \mathtt{mark}(\mathtt{x}_1, \mathtt{y}_2, 1))$$

is replaced with

$$\mathtt{next}(\mathtt{cell}(\mathtt{x}_1, \mathtt{y}_2, 1)) \leftrightarrow \mathtt{does}(\mathtt{random}, \mathtt{mark}(\mathtt{x}_1, \mathtt{y}_2, 1))$$

and then encoded at each stage t into a $\mathtt{next}_{0,t}$ constraint which states the fluent variable $\mathtt{cell}(\mathtt{x}_1, \mathtt{y}_2, 1)_{t+1}$ is true at turn $t+1$ iff the value of $\mathtt{mark}_{0,t}$ at turn t is $(\mathtt{x}_1, \mathtt{y}_2, 1)$.

4.2 Microstructure complements

We have now all ingredients in hand to extend the notion of *microstructure complement* [Cohen *et al.*, 2006] to stochastic constraint networks. Given an SCSP $\mathcal{N}$, the microstructure complement of $\mathcal{N}$ is the hypergraph $\mathcal{H}$, where the set of nodes is the collection of all variable-value pairs $\{(x, v) \mid x \in X, v \in D(x)\}$, and the set of hyperedges is the union over all constraints $c \in C$ of the instantiations on scp_c which are disallowed by c. Thus, $\boldsymbol{v} = \{(z_1, v_1), \cdots, (z_k, v_k)\}$ is a hyperedge of $\mathcal{H}$ iff $\{z_1, \cdots, z_k\}$ is the scope of some constraint $c \in C$ for which $C(\boldsymbol{v}) = 0$.

By extension, the microstructure complement of $\mathcal{N}_{\mathtt{R},T}$ is the colored hypergraph $\mathcal{H}_{\mathtt{R},T}$, wherein nodes and hyperedges are defined as above, and the coloring function χ of $\mathcal{H}_{\mathtt{R},T}$ maps each node (x, v) to a triplet, given as follows:

- if x is a fluent variable $\mathtt{f}_{j,t}$, then $\chi(x, v) = (t, \star, \star)$,
- if x is a terminal variable ω_t, then $\chi(x, v) = (t, \star, v)$,
- if x is an action variable $\mathtt{a}_{i,t}$ then $\chi(x, v) = (t, i, \star)$, and
- if x is a utility variable $\mathtt{u}_{i,t}$ then $\chi(x, v) = (t, i, v)$,

where $\star$ is an arbitrary symbol disjoint from $I^+ \cup U$. By construction, different colors are assigned to variables of different type, and to variables occurring at different stages. Any color-preserving automorphism of $\mathcal{H}_{\mathtt{R},T}$ is called a *constraint symmetry* of $\mathcal{N}_{\mathtt{R},T}$.

Proposition 2. *Let* $\mathtt{R}$ *be the* P-GDL *program of a* T*-horizon game. Then, any constraint symmetry of* $\mathcal{N}_{\mathtt{R},T}$ *is a structural symmetry of* $\mathcal{G}(\mathtt{R})$.

It is important to keep in mind that any solution policy of $\mathcal{N}_{\mathtt{R},T}$ is just a "legal strategy" that satisfies all game rules. In order to encode minimax policies, let $\mathcal{N}^*_{\mathtt{R},T}$ be the extension of $\mathcal{N}_{\mathtt{R},T}$ to the constraints $C^* = (C_1^*, \cdots, C_T^*)$, where $C_t^* = C_t \cup \{V_{i,t}^*, Q_{i,t}^*\}_{i=1}^k$. Here, $V_{i,t}^*$ and $Q_{i,t}^*$ are used to encode the minimax strategy (1). Namely, the scopes of $V_{i,t}^*$ and $Q_{i,t}^*$ are respectively given by $\{\mathtt{f}_{j,t}\}_{j=1}^n \cup \{\mathtt{u}_{i,t}\}$ and $\{\mathtt{f}_{j,t}\}_{j=1}^n \cup \{\mathtt{a}_{i,t}\}_{i=1}^k \cup \{\mathtt{u}_{i,t}\}$, where $\mathtt{u}_{i,t}$ is the utility variable of player i. The state-value constraint $V_{i,t}^*$ maps any state s to

$$\mathtt{u}_{i,t} = \begin{cases} \max_{a_i} \min_{\boldsymbol{a}_{-i}} Q_{i,t}^*(s, \boldsymbol{a}) & \text{if } \omega_t = 0, \\ \mathtt{goal}_{i,t}(s) & \text{otherwise.} \end{cases}$$

Correspondingly, the action-value constraint $Q_{i,t}^*$ maps any state s and action profile $\boldsymbol{a}$ to

$$\mathtt{u}_{i,t} = \sum_{a_0} P_t(a_0 \mid s, \boldsymbol{a}) V_{i,t+1}^*(s)$$

where a_0 ranges over all values in $D(\mathtt{a}_{0,t})$ such that $\mathtt{legal}_{0,t}(a_0, s) = 1$. By enforcing global consistency over $\mathcal{N}^*_{\mathtt{R},T}$, we must have $Q_{i,t}^*(s, \boldsymbol{a}) = V_{i,t}^*(s)$, which in turn implies that only optimal actions for player i are kept in the instantiations $(s, \boldsymbol{a})$ which are GC. The network $\mathcal{N}^d_{\mathtt{R},T}$ is defined analogously using the d-depth minimax strategy (2).

Proposition 3. *Let* $\mathtt{R}$ *be the* P-GDL *program of a* T*-horizon game. Then, any constraint symmetry of* $\mathcal{N}^*_{\mathtt{R},T}$ *(resp.* $\mathcal{N}^d_{\mathtt{R},T}$*) is a minimax (resp.* d*-depth minimax) symmetry of* $\mathcal{G}(\mathtt{R})$.

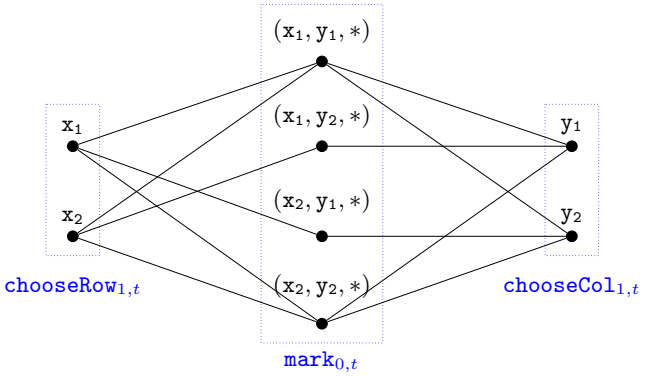

Figure 2: A sub-hypergraph of the micro-structure complement associated with the one-stage SCSP of Example 5.

Though theoretically interesting, the above result can be difficult to implement due to the large size of the microstructure complement of a T-stage SCSP. Fortunately, $\mathcal{N}^*_{\mathtt{R},T}$ can be viewed as a sequence of *one-stage* SCSPs, each associated with a restricted microstructure complement. Concretely, the tth one-stage SCSP of $\mathcal{N}^*_{\mathtt{R},T}$ is the restriction $N^*_{\mathtt{R},t}$ of $\mathcal{N}^*_{\mathtt{R},T}$ to the stage X_t, that is, $N^*_{\mathtt{R},t} = (X_t, Y_t, D, C^*_t, P_t, \theta)$. The corresponding microstructure complement is the projection $H^*_{\mathtt{R},t}$ of $\mathcal{H}_{\mathtt{R},T}$ onto the variables X_t, satisfying the following *backward consistency* condition: if $\boldsymbol{v}_t$ is a tuple on X_t which does not include any hyperedge in $H^*_{\mathtt{R},t}$, then there is an extension $\boldsymbol{v}$ of $\boldsymbol{v}_t$ on $(X_1, \cdots, X_t)$ which does not include any hyperedge in $H^*_{\mathtt{R},1} \cup \cdots \cup H^*_{\mathtt{R},t}$. For the stochastic constraint network $\mathcal{N}^d_{\mathtt{R},T}$, the tth one-stage SCSP $N^d_{\mathtt{R},t}$ and its microstructure complement $H^d_{\mathtt{R},t}$ are defined in a similar way.

Proposition 4. *Let* R *be the* P-GDL *program of a T-horizon game. Then, for every $t \in \{1, \cdots, T\}$, any constraint symmetry of $N^*_{\mathtt{R},t}$ (resp. $N^d_{\mathtt{R},t}$) preserves the minimax (resp. d-depth minimax) strategies of $\mathcal{G}(\mathtt{R})$.*

From a practical viewpoint, constraint symmetries of one-stage SCSPs can be exploited for reducing the search of minimax strategies: once an instantiation $\boldsymbol{v}_t$ on the stage X_t has been proved GC, automorphisms of $H^d_{\mathtt{R},t}$ can be applied to derive symmetric GC instantiations at this stage. Equivalently, if any extension of $\boldsymbol{v}_t$ has reached a dead end, then all symmetric instantiations of $\boldsymbol{v}_t$ at stage X_t can be ignored.

Example 5. Consider the randomized 2×2 Tic-Tac-Toe game after a couple of turns. During the first stage, player 1 selected $\mathtt{chooseRow}(\mathtt{x}_1)$, and the chance player marked $(\mathtt{x}_1, \mathtt{y}_1)$ with 1. Next, player 2 selected $\mathtt{chooseCol}(\mathtt{y}_2)$, and the chance player marked $(\mathtt{x}_2, \mathtt{y}_2)$ with 2. Figure 2 depicts the microstructure complement of the third one-stage SCSP. For the sake of clarity, only action variables of players 0 (random) and 1 are represented. We can here observe that all legal moves for player 1 are symmetrical. Indeed, the automorphism obtained by permuting the symbols "1" and "2" in the values of the action variables, yields a symmetry between rows $\mathtt{x}_1$ and $\mathtt{x}_2$, and a symmetry between columns $\mathtt{y}_1$ and $\mathtt{y}_2$. Analogously, the automorphism given by permuting the symbols "x" and "y" in those values, gives rise to a symmetry between row $\mathtt{x}_1$ (resp. $\mathtt{x}_2$) and column $\mathtt{y}_1$ (resp. $\mathtt{y}_2$).

5 Experiments

Based on our framework, we now present a series of experiments conducted on a cluster of Intel Xeon E5-2643 CPU 3.3 GHz with 64 GB of RAM and four threads under Linux. All problems used in experiments are turn-based two-player zero-sum games. Specifically, we focus on 25 representative games: 20 deterministic GDL games, and 5 stochastic GDL games (with random). For all games, we used the 2015 Tiltyard Open (International GGP Competition) setup: 180s for the *start clock* (deliberation time before the first turn) and 15s for the *play clock* (deliberation time per turn).

MAC-UCB-SYM. In order to implement our constraint-based symmetry detection method, we used (a variant of) the MAC-UCB algorithm developed in [Koriche *et al.*, 2016]. Any GDL description R is first translated into the Clark's completion of its ground instance $\mathtt{R}_c$, which is then encoded into a T-stage SCSP $\mathcal{N}_{\mathtt{R},T}$. The horizon T was fixed to 200. The goal of MAC-UCB is to find d-minimax strategies, that is, solution policies for the SCSP $\mathcal{N}^d_{\mathtt{R},T}$. The MAC (Maintaining Arc Consistency) technique searches solution policies up to depth d, and the UCB (Upper Confidence Bounds) stochastic bandit method estimates the value $\hat{V}(s)$ of states s at depth d. In what follows, MAC-UCB-SYM is the upgrading of MAC-UCB that incorporates symmetry detection using the microstructure complement of one-stage SCSPs. The choice of the depth d was derived by partitioning the deliberation time into three ratios $(r_{\text{MAC}}, r_{\text{UCB}}, r_{\text{SYM}})$, which correspond to the proportions of runtime allocated for MAC, FLAT-UCB, and symmetry detection, respectively. Based on a sensitivity analysis of MAC-UCB-SYM, we used the ratios $(45\%, 30\%, 25\%)$.

MAC-UCB-SYM satisfies the condition of backward consistency by iteratively expanding the partial instantiations of a policy tree rooted at the initial state s_1. For a partial instantiation $\boldsymbol{v}$ on $(X_1, \cdots, X_{t-1})$ which is consistent with all constraints in $(C^d_1, \cdots, C^d_{t-1})$, the algorithm stores in $H^d_{\mathtt{R},t}$ all hyperedges $\boldsymbol{e}_t$ such that $\boldsymbol{v} \cup \boldsymbol{e}_t$ is inconsistent with $N^d_{\mathtt{R},t}$. So, for any independent set $\boldsymbol{v}_t$ of $H^d_{\mathtt{R},t}$, the instantiation $\boldsymbol{v} \cup \boldsymbol{v}_t$ is guaranteed to be consistent with $(C^d_1, \cdots, C^d_t)$.

We used the NAUTY algorithm [McKay and Piperno, 2014] for detecting automorphisms of microstructure complements. Concretely, MAC-UCB-SYM uses a hash table of 32Gb with a hash function specified in [Zobrist, 1990]. For each one-stage SCSP $N^d_{\mathtt{R},t}$, we store in the hash table the tuples $\boldsymbol{v}_t$ on X_t for which the value $Q^d(\boldsymbol{v}_t)$ has been estimated by MAC-UCB.[1] When the algorithm expands a partial instantiation $\boldsymbol{v}$ on $(X_1, \cdots, X_{t-1})$ with an independent set $\boldsymbol{v}_t$ of $H^d_{\mathtt{R},t}$, we detect whether $\boldsymbol{v}_t$ is symmetric to some tuple $\boldsymbol{v}'_t$ in the hash-table. If so, the value $Q^d(\boldsymbol{v}'_t)$ is directly mapped to $Q^d(\boldsymbol{v}_t)$ without exploring the whole subtree up to depth d.

GGP Competitors. MAC-UCB-SYM was compared to several General Game Players: the first player is the (multiagent version) of the UCT algorithm [Sturtevant, 2008], which is the state-of-the-art method for deterministic games. The

[1] Here, $Q^d(\boldsymbol{v}_t) = Q^d_{i,t}(s, \boldsymbol{a})$ where $(s, \boldsymbol{a})$ is the projection of $\boldsymbol{v}_t$ onto the fluent and action variables $\{\mathtt{f}_{j,t}\}_{j=1}^n \cup \{\mathtt{a}_{i,t}\}_{i=1}^k$ of X_t.

Deterministic GDL games						
Game	**MAC-UCB**	**UCT**	**UCT-SYM**	**GRAVE**	**GRAVE-SYM**	**SANCHO**
Amazons torus 10×10	84.2 ($\pm1.2\%$)	96.0 ($\pm2.2\%$)	98.1 ($\pm1.7\%$)	78.4 ($\pm3.4\%$)	86.7 ($\pm2.7\%$)	86.2 ($\pm3.1\%$)
Breakthrough suicide	93.0 ($\pm2.3\%$)	93.1 ($\pm1.6\%$)	81.9 ($\pm3.7\%$)	59.4 ($\pm0.9\%$)	73.2 ($\pm2.9\%$)	77.8 ($\pm4.0\%$)
Chess	76.4 ($\pm2.5\%$)	95.8 ($\pm1.8\%$)	95.3 ($\pm2.1\%$)	88.1 ($\pm4.2\%$)	95.4 ($\pm2.5\%$)	87.9 ($\pm2.1\%$)
Connect Four 20×20	87.5 ($\pm3.5\%$)	97.0 ($\pm1.2\%$)	100.0 ($\pm0.0\%$)	65.1 ($\pm3.1\%$)	88.5 ($\pm2.2\%$)	96.0 ($\pm0.9\%$)
Connect Four Simultaneous	73.7 ($\pm2.8\%$)	87.8 ($\pm1.7\%$)	96.1 ($\pm0.9\%$)	77.2 ($\pm3.4\%$)	93.2 ($\pm3.6\%$)	82.0 ($\pm2.6\%$)
Copolymer with pie	73.9 ($\pm1.5\%$)	87.2 ($\pm2.3\%$)	93.3 ($\pm0.5\%$)	80.2 ($\pm1.6\%$)	91.6 ($\pm1.8\%$)	77.9 ($\pm3.6\%$)
Dots and boxes suicide	65.4 ($\pm1.7\%$)	95.2 ($\pm0.9\%$)	83.7 ($\pm2.4\%$)	80.5 ($\pm1.6\%$)	70.3 ($\pm2.7\%$)	88.0 ($\pm1.5\%$)
English Draughts	85.1 ($\pm2.8\%$)	97.9 ($\pm1.3\%$)	97.4 ($\pm1.3\%$)	70.1 ($\pm4.1\%$)	71.2 ($\pm3.1\%$)	59.3 ($\pm1.5\%$)
Free For All 2P	53.4 ($\pm0.7\%$)	81.5 ($\pm2.4\%$)	84.8 ($\pm1.9\%$)	61.2 ($\pm0.7\%$)	72.3 ($\pm1.6\%$)	71.2 ($\pm2.3\%$)
Hex	84.0 ($\pm1.4\%$)	100.0 ($\pm0.0\%$)	100.0 ($\pm0.0\%$)	74.2 ($\pm3.2\%$)	89.8 ($\pm2.9\%$)	78.1 ($\pm1.5\%$)
Knight Through	81.2 ($\pm2.4\%$)	96.0 ($\pm1.3\%$)	90.6 ($\pm2.6\%$)	82.1 ($\pm3.5\%$)	81.2 ($\pm2.3\%$)	88.1 ($\pm2.6\%$)
Majorities	84.4 ($\pm2.6\%$)	95.7 ($\pm1.6\%$)	96.8 ($\pm1.4\%$)	81.2 ($\pm2.6\%$)	92.3 ($\pm2.3\%$)	87.2 ($\pm3.2\%$)
Pentago	53.1 ($\pm1.5\%$)	84.8 ($\pm3.1\%$)	66.2 ($\pm2.8\%$)	72.1 ($\pm2.3\%$)	58.4 ($\pm2.8\%$)	54.3 ($\pm0.9\%$)
Quarto	54.9 ($\pm1.6\%$)	87.1 ($\pm3.4\%$)	65.6 ($\pm2.6\%$)	65.2 ($\pm3.2\%$)	63.8 ($\pm1.6\%$)	57.9 ($\pm2.3\%$)
Sheep and Wolf	74.8 ($\pm3.2\%$)	92.5 ($\pm2.4\%$)	94.6 ($\pm0.9\%$)	64.4 ($\pm3.7\%$)	63.2 ($\pm3.6\%$)	62.1 ($\pm1.5\%$)
Shmup	58.0 ($\pm1.7\%$)	88.2 ($\pm2.6\%$)	63.7 ($\pm2.2\%$)	55.3 ($\pm1.5\%$)	52.1 ($\pm0.2\%$)	53.0 ($\pm0.6\%$)
Skirmish zero-sum	85.4 ($\pm2.5\%$)	98.7 ($\pm0.4\%$)	100.0 ($\pm0.0\%$)	77.7 ($\pm1.6\%$)	95.8 ($\pm2.3\%$)	67.4 ($\pm4.2\%$)
TicTac Chess 2P	94.9 ($\pm3.4\%$)	97.6 ($\pm0.7\%$)	96.5 ($\pm0.4\%$)	76.3 ($\pm1.3\%$)	93.2 ($\pm2.3\%$)	86.1 ($\pm3.3\%$)
TTCC4 2P	84.4 ($\pm2.3\%$)	97.3 ($\pm1.2\%$)	97.2 ($\pm2.1\%$)	62.4 ($\pm2.3\%$)	85.7 ($\pm3.1\%$)	65.8 ($\pm4.1\%$)
Reversi Suicide	72.2 ($\pm3.2\%$)	100.0 ($\pm0.0\%$)	100.0 ($\pm0.0\%$)	66.1 ($\pm2.7\%$)	78.7 ($\pm2.2\%$)	58.2 ($\pm2.2\%$)
Stochastic GDL games						
Backgammon	92.1 ($\pm2.7\%$)	98.0 ($\pm0.6\%$)	96.1 ($\pm1.4\%$)	70.2 ($\pm4.5\%$)	86.8 ($\pm3.9\%$)	100.0 ($\pm0.0\%$)
Can't Stop	88.2 ($\pm1.7\%$)	94.1 ($\pm2.3\%$)	96.8 ($\pm1.7\%$)	93.4 ($\pm1.3\%$)	93.7 ($\pm3.2\%$)	100.0 ($\pm0.0\%$)
Kaseklau	73.5 ($\pm3.6\%$)	73.4 ($\pm1.3\%$)	72.1 ($\pm0.9\%$)	67.4 ($\pm2.8\%$)	60.2 ($\pm3.2\%$)	88.1 ($\pm2.6\%$)
Pickomino	75.4 ($\pm1.8\%$)	74.3 ($\pm2.4\%$)	82.4 ($\pm2.8\%$)	87.1 ($\pm3.0\%$)	95.6 ($\pm1.0\%$)	92.1 ($\pm2.9\%$)
Yahtzee	87.4 ($\pm1.6\%$)	88.2 ($\pm2.3\%$)	83.1 ($\pm3.3\%$)	64.1 ($\pm3.2\%$)	60.9 ($\pm2.5\%$)	91.8 ($\pm3.3\%$)

Table 1: Results of MAC-UCB-SYM against each GGP player.

second player is Cazenave's GRAVE algorithm (2015), that implements the Generalized Rapid Action Value Estimation technique, a generalization of RAVE [Gelly and Silver, 2007]. The last player is SANCHO (version 1.61c), a Monte Carlo Tree Search method elaborated by S. Draper and A. Rose, which has won the 2014 GGP Competition. Rule-based symmetry detection methods were implemented for UCT and GRAVE, yielding the UCT-SYM player and GRAVE-SYM player, respectively. By analogy with MAC-UCB-SYM, UCT-SYM and GRAVE-SYM used NAUTY for detecting automorphisms in the ground rule graph, together with a hash table for exploiting symmetries. Based on a sensitivity analysis of these players, UCT-SYM (resp. GRAVE-SYM) used 20% (resp. 25%) of its deliberation time for symmetry detection.

Experiments were conducted on $66,000$ game contests, namely, 300 matches for each deterministic game, and 1000 matches for each stochastic game, in order to consider the probabilistic effect of outcomes. For the sake of fairness, the role of players was exchanged during each match.

Results. In Table 1, each column reports the proportion of wins of MAC-UCB-SYM against the selected adversary. For example, the third row of the second column indicates that, for the Chess game, MAC-UCB-SYM wins 95.8% of contests against UCT with a standard deviation of 1.8%.

In light of these results, the following observations can be made: (i) MAC-UCB-SYM outperforms MAC-UCB especially for large games (ex: Amazons torus 10×10), which indicates that constraint-based symmetry detection is an effective method for reducing exploration in large search trees; (ii) MAC-UCB-SYM also dominates UCT, GRAVE, and SANCHO, even for deterministic games which is the class targeted by these algorithms; and (iii) MAC-UCB-SYM is even better against UCT-SYM and GRAVE-SYM for most games. The last point clearly indicates that constraint-based symmetry methods, used to recognize (approximate) minimax symmetries, are more effective than rule-based symmetry detection methods, used to recognize structural symmetries.

6 Conclusion

In this paper, we have shown that constraint-based symmetry detection can prove effective for solving GDL games. Beyond the conclusive results obtained by coupling the stochastic constraint solver MAC-UCB with constraint-based symmetry detection, our framework emerges as a flexible approach for recognizing various types of game symmetries. Notably, a key perspective of further research is to detect *equilibrium symmetries* that preserve (mixed) Nash equilibria, or correlated equilibria, using a constraint-based approach for modeling these solution concepts.

References

[Ben-Eliyahu and Dechter, 1994] Rachel Ben-Eliyahu and Rina Dechter. Propositional semantics for disjunctive logic programs. *Annals of Mathematics and Artificial Intelligence*, 12(1-2):53–87, 1994.

[Cazenave, 2015] Tristan Cazenave. Generalized rapid action value estimation. In *Proceedings of IJCAI*, pages 754–760, 2015.

[Clark, 1978] Keith L. Clark. Negation as failure. In *Logic and Databases*, pages 293–322. Plenum Press, 1978.

[Cohen *et al.*, 2006] David Cohen, Peter Jeavons, Christopher Jefferson, Karen E. Petrie, and Barbara M. Smith. Symmetry definitions for constraint satisfaction problems. *Constraints*, 11(2-3):115–137, 2006.

[Condon, 1992] Anne Condon. The complexity of stochastic games. *Information and Computation*, 96:203–224, 1992.

[Gelly and Silver, 2007] Sylvain Gelly and David Silver. Combining online and offline knowledge in UCT. In *Proceedings of ICML*, pages 273–280, 2007.

[Genesereth *et al.*, 2005] Michael Genesereth, Nathaniel Love, and Barney Pell. General game playing: Overview of the AAAI competition. *AAAI Magazine*, 26(2):62–72, 2005.

[Hnich *et al.*, 2012] Brahim Hnich, Roberto Rossi, S. Armagan Tarim, and Steven Prestwich. Filtering algorithms for global chance constraints. *Artificial Intelligence*, 189:69–94, 2012.

[Jégou, 1993] Philippe Jégou. Decomposition of domains based on the micro-structure of finite constraint-satisfaction problems. In *Proceedings of AAAI*, pages 731–736, 1993.

[Koriche *et al.*, 2016] Frédéric Koriche, Sylvain Lagrue, Éric Piette, and Sébastien Tabary. General game playing with stochastic CSP. *Constraints*, 21(1):95–114, 2016.

[Kuhlmann and Stone, 2007] Gregory Kuhlmann and Peter Stone. Graph-based domain mapping for transfer learning in general games. In *Proceedings of ECML*, pages 188–200, 2007.

[Lanctot *et al.*, 2013] Marc Lanctot, Abdallah Saffidine, Joel Veness, Christopher Archibald, and Mark H.M. Winands. Monte Carlo *-minimax search. In *Proceedings of IJCAI*, pages 580–586, 2013.

[Love *et al.*, 2008] Nathaniel Love, Timothy Hinrichs, David Haley, Eric Schkufza, and Michael Genesereth. General game playing: Game description language specification. Technical Report LG-2006-01, Stanford University, 2008.

[McKay and Piperno, 2014] Brendan D. McKay and Adolfo Piperno. Practical graph isomorphism, II. *Journal of Symbolic Computation*, 60(0):94 – 112, 2014.

[Saffidine, 2014] Abdallah Saffidine. The game description language is Turing complete. *IEEE Transactions on Computational Intelligence and AI in Games*, 6(4):320–324, 2014.

[Schiffel and Thielscher, 2014] Stephan Schiffel and Michael Thielscher. Representing and reasoning about the rules of general games with imperfect information. *J. Artif. Intell. Res. (JAIR)*, 49:171–206, 2014.

[Schiffel, 2010] Stephan Schiffel. Symmetry detection in general game playing. In *Proceedings of AAAI*, 2010.

[Sturtevant, 2008] Nathan R. Sturtevant. An analysis of UCT in multi-player games. *International Computer Games Association Journal*, 31(4):195–208, 2008.

[Tarim *et al.*, 2006] S. Armagan Tarim, Suresh Manandhar, and Toby Walsh. Stochastic constraint programming: A scenario-based approach. *Constraints*, 11(1):53–80, 2006.

[Thielscher, 2016] Michael Thielscher. GDL-III: A proposal to extend the game description language to general epistemic games. In *Proceedings of ECAI*, pages 1630–1631, 2016.

[Walsh, 2002] Toby Walsh. Stochastic Constraint Programming. In *Proceedings of ECAI*, pages 111–115, 2002.

[Zhang *et al.*, 2015] Haifeng Zhang, Dangyi Liu, and Wenxin Li. Space-consistent game equivalence detection in General Game Playing. In *Proceedings of CGW*, pages 165–177, 2015.

[Zobrist, 1990] Albert L. Zobrist. A new hashing method with application for game playing. *International Computer Games Association Journal*, 13(2):69–73, 1990.