# Stochastic Constraint Programming for General Game Playing with Imperfect Information

**Frédéric Koriche, Sylvain Lagrue, Éric Piette,** and **Sébastien Tabary**
CRIL Univ. Artois - CNRS UMR 8188, France-62307 Lens
{koriche,lagrue,epiette,tabary}@cril.fr

## Abstract

The game description language with incomplete information (GDL-II) is expressive enough to capture partially observable stochastic multi-agent games. Unfortunately, such expressiveness does not come without a price: the problem of finding a winning strategy is NEXP$^{NP}$-hard, a complexity class which is far beyond the reach of modern constraint solvers. In this paper, we identify a PSPACE-complete fragment of GDL-II, where agents share the same (partial) observations. We show that this fragment can be cast as a decomposable stochastic constraint satisfaction problem (SCSP) which, in turn, can be solved using general-purpose constraint programming techniques. Namely, we develop a constraint-based sequential decision algorithm for GDL-II games which exploits constraint propagation and Monte Carlo sampling based. Our algorithm, validated on a wide variety of games, significantly outperforms the state-of-the-art general game playing algorithms.

## 1 Introduction

Of all human activities, games convey one of the most illustrative examples of intelligent behavior. A player has indeed to face complex tasks, such as understanding abstract rules, evaluating the current situation, choosing the best possible move and, ultimately, devising a winning strategy. In Artificial Intelligence, the General Game Playing (GGP) challenge [Genesereth and Thielscher, 2014; Genesereth *et al.*, 2005] is to develop computer players that understand the rules of previously unknown games, and learn to play these games well without human intervention.

In General Game Playing, the rules of an input game are described using a high-level, declarative representation formalism, called Game Description Language (GDL). The first version of this language (GDL-I) is restricted to deterministic games with complete information [Love *et al.*, 2008]. While a GDL-I game may involve multiple players, each player has complete knowledge about the current game state, the past actions of her opponents, and the deterministic effects of the joint action.

In order to alleviate these restrictions, Schiffel and Thielscher [2011; 2014] recently proposed a new game description language (GDL-II) for representing games with *incomplete information*. In a GDL-II game, players may have restricted access to the current game state, and the effects of their joint actions are uncertain. As such, GDL-II is expressive enough to capture *partially observable stochastic games (POSGs)*, which cover a wide variety of multi-agent sequential decision problems. However, such expressiveness does not come without a price: from a computational viewpoint, the problem of finding a winning strategy in a POSG is NEXP$^{NP}$-complete [Goldsmith and Mundhenk, 2007], a complexity class which is far beyond the standard PSPACE complexity class of games with complete information. Although several algorithms have been devised for tackling specific instances of POSGs, such as Contract Bridge [Ginsberg, 2001] and Poker [Bowling *et al.*, 2015], they are dedicated programs which heavily rely on human knowledge about game rules and evaluation functions. By contrast, the task of developing *general* game players for POSGs appears extremely challenging due to their complexity barrier.

Despite this theoretical issue, several GGP algorithms have been recently developed for solving restricted, yet expressive, fragments of GDL-II. They include, for example, mono-agent deterministic games with incomplete information [Geißer *et al.*, 2014], and multi-agent stochastic games with complete information [Koriche *et al.*, 2016]. In particular, the last approach relies on Constraint Programming techniques which have proved to be successful in practice. Our present study aims at extending the range of GDL-II games which can be expressed in terms of Stochastic Constraint Satisfaction Problems (SCSPs).

The main contribution of this article can be summarized as follows: (i) we present an important PSPACE-complete fragment of GDL-II, where players share the same (partial) observations, and which can be expressed as a SCSP; (ii) we extend MAC-UCB, a sequential decision algorithm that exploits constraint propagation and Monte Carlo sampling, to GDL-II games; (iii) we provide a comparative experimental study on various GDL games, including deterministic games, stochastic games, and partially observable stochastic games (with shared information). Experimental results show that our constraint programming technique outperforms the current general game playing algorithms.

## 2 General Imperfect Information Games

The problems under consideration in GGP are *finite sequential and synchronous* games. Each game involves a finite number of players, and a finite number of states, including one distinguished initial state, and one or several terminal states. On each round of the game, each player has at her disposal a finite number of actions (called "legal moves"); the current state of the game is

updated by the simultaneous application of each player's action (which can be "noop" or do nothing). The game starts at the initial state and, after a finite number of rounds, ends at some terminal state, in which a reward is given to each player. In a *stochastic* game, a distinguished player, often referred to as "chance", can choose its actions at random according to a probability distribution defined over its legal moves. In an *incomplete information* game, some aspects (called "fluents") of the current game state are not fully revealed to the agents. We shall focus on stochastic *shared information* games in which, at each round, all agents have the same (possibly incomplete) information about the game state.

## 2.1 GDL-II Syntax

GDL is a declarative language for representing finite games. Basically, a GDL program is a set of rules described in first-order logic. Players and game objects (coins, dice, locations, pieces, etc.) are described by constants, while fluents and actions are described by first-order terms. The atoms of a GDL program are constructed over a finite set of relation symbols and variable symbols. Some symbols have a specific meaning in the program, and are described in Table 1. For example, in the TicTacToe game, `legal(alice,mark(X,Y))` indicates that player `alice` is allowed to mark the square `(X,Y)` of the board. In GDL-II, the last two keywords of the table are added to represent stochastic games (`random`), and partially observable games (`sees`).

Table 1: GDL-II keywords

| Keywords | Description |
|---|---|
| `role(P)` | `P` is a player |
| `init(F)` | the fluent `F` holds in the initial state |
| `true(F)` | the fluent `F` holds in the current state |
| `legal(P,A)` | the player `P` can take action `A` in the current state |
| `does(P,A)` | the player `P` performs action `A` |
| `next(F)` | the fluent `F` holds in the next state |
| `terminal` | the current state is terminal |
| `goal(P,V)` | the player `P` gets reward `V` in the current state |
| `sees(P,F)` | the player `P` perceives `F` in the current state |
| `random` | the "chance" player |

The rules of a GDL program are first-order Horn clauses. For example, the rule:

$$\texttt{sees(alice,cell(X,Y,o))} \leftarrow \texttt{does(alice,mark(X,Y))}$$

states that alice sees the effect of marking squares of the board. In order to represent a finite sequential game, a GDL program must obey to syntactic conditions, defined over the terms and relations occurring in rules, and the structure of its rule set. We refer the reader to [Love *et al.*, 2008; Thielscher, 2010] for a detailed analysis of these conditions.

**Example 1** *"Matching Pennies" is a well-known game involving two players, who place a penny (or coin) on the table, with the payoff depending on whether pennies match. We consider here a variant named "Hidden Matching Pennies" in which alice plays against the chance player (random) playing two pennies, one sees by alice and the other hidden. During the first round, the chance player chooses tails or heads for its two pennies and, during the second round, alice places a coin on the table; alice wins 100 points if all the three sides are heads or tails and 50 points if at least one side of the chance player and its side are similar. The corresponding GDL program is described in Figure 1;*

## 2.2 GDL-II Semantics

For a positive integer $n$, let $[n] = \{1,\cdots,n\}$. For a finite set $S$, let $\Delta_S$ denote the probability simplex over $S$, that is, the space of all probability distributions over $S$. Various descriptions of incomplete information games have been proposed in the literature (see e.g. [Schiffel and Thielscher, 2011]). We focus here on a variant of [Geißer *et al.*, 2014].

Formally, a *partially observable stochastic game with legal actions (POSG)*, is a tuple $G = \langle k,S,s_0,S_g,A,L,P,B,R\rangle$, where:

- $k \in \mathbb{N}$ is the number of players,
- $S$ is a finite set of states, including a distinguished initial state $s_0$, and a subset $S_g$ of goal (or terminal) states.
- $A$ is a finite set of *actions*. As usual an *action profile* is a tuple $\boldsymbol{a} = \langle a_1, \dots, a_k\rangle \in A^k$; by $a_p$, we denote the action of player $p$, and by $\boldsymbol{a}_{-p}$ the action profile $\langle a_1,\dots,a_{p-1},a_{p+1},\dots,a_k\rangle$ of the remaining players.
- $L : [k] \times S \to 2^A$ defines the set of *legal actions* $L_p(s)$ of player $p$ at state $s$; we assume that $L_p(s) = \emptyset$ for all $s \in S_g$. $P : S \times A^k \hookrightarrow \Delta_S$ is the *partial transition probability function*, which maps each state $s \in S$ and each action profile $\langle a_1,\dots,a_k\rangle \in L(s)$ to a probability distribution over $S$.
- $B : [k] \times S \to \Delta_S$ is the *belief function* which maps every player $p \in [k]$ and every state $s \in S$ to a probability distribution $B_p(s)$ over $S$, capturing the belief state of $p$ at $s$.
- $R : [k] \times S_g \to [0,1]$ is the *reward function* which maps every player $p \in [k]$ and every goal state $s \in S_g$ to a value $R_p(s) \in [0,1]$, capturing the reward of $p$ in $s$.

With these notions in hand, the POSG $G$ associated to a GDL-II program `G` is defined as follows. Let `B` denote the Herbrand base (i.e. the set of all ground terms) of `G`; $A$ (resp. $F$) is the set of all ground action (resp. fluent) terms occurring in `B`. We use $\texttt{G} \models \texttt{A}$ to denote that atom `A` is true in the unique answer set of `G`. The number $k$ of ground terms `p` such that $\texttt{role(p)} \in \texttt{G}$, determines the set of players.

Each state $s$ is a subset of $F$. Notably, the initial state $s_0$ is $\{f : \texttt{G} \models \texttt{init(f)}\}$, and any terminal state is a set of fluents $s = \{f_1,\cdots,f_n\}$ such that $\texttt{G} \cup s^{\texttt{true}} \models \texttt{terminal}$, where $s^{\texttt{true}}$ is the set of facts $\{\texttt{true(f}_1\texttt{)},\cdots,\texttt{true(f}_n\texttt{)}\}$. The set $L_p(s)$ of legal actions for player $p$ at state $s$ is given by $\texttt{G} \cup s^{\texttt{true}} \models \texttt{legal(p,a)}$. In particular, $L_0(s)$ denotes the set of legal actions for the chance player (`random`). Any action profile (extended to the chance player) $\boldsymbol{a} = \langle a_0,a_1,\dots,a_k\rangle \in L_0(s) \times L_1(s) \times \cdots \times L_k(s)$ determines a successor $s'$ of $s$ given by $\{f : \texttt{G} \cup s^{\texttt{true}} \cup \boldsymbol{a}^{\texttt{does}} \models \texttt{next(f)}\}$, where $\boldsymbol{a}^{\texttt{does}}$ is the set of facts $\{\texttt{does(a}_0\texttt{)},\texttt{does(a}_1\texttt{)},\dots,\texttt{does(a}_k\texttt{)}\}$. The probability distribution $P(s,\boldsymbol{a}_{-0})$ over all those successors is the uniform distribution, i.e. $P(s,\boldsymbol{a}_{-0})(s') = 1/|L_0(s)|$.

The belief state $B_p(s)$ of player $p$ at any successor $s'$ of $s$ is given by the joint distribution $\prod_{f \in F} P(f)$, where $P(f) = 1$ if $\texttt{G} \cup s^{\texttt{true}} \cup \boldsymbol{a}^{\texttt{does}} \models \texttt{sees(p,f)}$, and $P(f) = 1/2$ otherwise. Finally, the reward $R_p(s)$ of player $p$ at a terminal state $s$ is the value $v$ such that $\texttt{G} \cup s^{\texttt{true}} \models \texttt{goal(p,v)}$.

```
% roles
role(alice).
role(random).
% side of a coin
side(tails).
side(heads).
% initial state
init(coin(unset)).
init(coins(unset,unset)).
init(control(random)).
% legal moves
legal(random,choose(S1,S2)) ← true(control(random)), side(S1),side(S2).
legal(alice,play(S)) ← true(control(alice)), side(S).
legal(P,noop) ← not(true(control(P))).
% game update
next(coins(S1,S2)) ← does(P,choose(S1,S2)).
next(coin(S)) ← does(P,play(S)).
next(coins(S1,S2)) ← not(true(control(random)), true(coins(S1,S2)).
next(coin(S)) ← not(true(control(alice)), true(coin(S)).
next(control(alice)) ← true(control(random)).
% the percepts
sees(alice,coins(S1,_)) ← does(random,choose(S1,S2)).
% terminal states
terminal ← not(true(coin(unset))), not(true(coins(unset,unset))).
goal(alice,100) ← true(coin(S)), true(coins(S,S)).
goal(alice,50) ← or(true(coin(S1)),true(coin(S2))), true(coins(S1,S2)), distinct(S1,S2).
goal(alice,0) ← true(coin(S1)), true(coins(S2,S2)), distinct(S1,S2).
```

Figure 1: GDL program of "Hidden Matching Pennies"

### 2.3 A PSPACE Fragment of GDL-II

A game with *shared information* is any partially observable stochastic game $G$ in which, at each round, all players share the same belief state, i.e. $B_1(s)=\cdots=B_k(s)$ for all states $s\in S$. We use here $B(s)$ to denote the common belief state in $s$. Remark that any game with shared information can be converted into a fully observable stochastic game, by replacing the transition function $P$ and the belief function $B$ with a new transition function $Q:S\times A^k\hookrightarrow\Delta_S$ defined by:

$$Q(s,\boldsymbol{a})(s')=\prod_{t\in S}P(s,\boldsymbol{a})(t)\cdot B(t)$$

for all $\boldsymbol{a}\in L(s)$. In other words, $Q(s,\boldsymbol{a})(s')$ is the probability of observing $s'$ after performing the action profile $\boldsymbol{a}$ in state $s$. Since any game $G$ with shared information is a stochastic game, a joint policy for $G$ is a map $\boldsymbol{\pi}:S\to A^k$, where $\pi_p(s)$ is the policy of player $p$, and $\boldsymbol{\pi}_{-p}(s)$ is the joint policy of other players. Given a threshold vector $\boldsymbol{\theta}\in[0,1]^k$, we say that $\boldsymbol{\pi}$ is a *winning policy* for player $p$ if the expected reward of $p$ w. r. t. $\boldsymbol{\pi}$ is greater than $\theta_p$.

Based on the notion of shared information, we now examine several restrictions of GDL-II programs which together guarantee that the problem of finding a winning policy is in PSPACE. Namely, a GDL-II program G is *depth-bounded* if the number of ground terms in the Herbrand universe of G is polynomial in $|\mathtt{G}|$. If G is bounded and each rule of G has a constant number of variables, then G is *propositional*. For an integer $T$, G is of *horizon* $T$ if any terminal state is reachable after at most $T$ rounds. Finally, G is *information sharing* if for every player $p$, every fluent $f$, every state $s$, and every action profile $\boldsymbol{a}$, if $\mathtt{G}\cup s^{\mathtt{true}}\cup\boldsymbol{a}^{\mathtt{does}}\models\mathtt{sees(p,f)}$, then $\mathtt{G}\cup s^{\mathtt{true}}\cup\boldsymbol{a}^{\mathtt{does}}\models\mathtt{sees(q,f)}$, for all players $q\in[k]$.

**Theorem 1** *Let $\mathcal{G}_T\subseteq$ GDL-II be the fragment propositional, information sharing programs of horizon $T$. Then, the problem of finding a winning policy in $\mathcal{G}_T$ is* PSPACE*-complete.*

**Proof 1 (Sketch)** *Since $\mathcal{G}_T$ includes full-information stochastic games as a special case, the problem is* PSPACE*-hard. For any finite and depth-bounded game $\mathtt{G}\in\mathcal{G}_T$, a winning policy can be found in $f(|\mathtt{G}|)$ time and $g(|\mathtt{G}|)$ space using a stochastic-alternating Turing Machine (TM), i.e. a TM which includes stochastic states (for* random*), existential states (for player $p$), and universal states (for all other players). Since* G *is propositional, the number of fluents and the number of actions are polynomial in $|G|$, which together imply that $g(|\mathtt{G}|)$ is polynomial. At each game state, the stochastic-alternating TM can guess a game action profile using its existential states. Since $k\leq|\mathtt{G}|$, the next game state can be found using a polynomial number of universal states and stochastic states. This, together with fact that the TM will find a terminal game state in at most $T$ rounds, implies that $f(|\mathtt{G}|)$ is also polynomial. Finally, since any stochastic-alternating TM using polynomial time and space can be simulated by* NPSPACE *(see e.g. [Bonnet and Saffidine, 2014]) then, using Savitch's theorem* NPSPACE = PSPACE*, it follows that $\mathcal{G}_T$ is in* PSPACE*, which yields the result.*

## 3 The SCSP Framework

Borrowing the terminology of [Walsh, 2002], stochastic constraint networks extend the standard CSP framework by introducing

stochastic variables in addition to the usual decision variables. We focus here on a slight generalization of the original SCSP model that captures conditional probability distributions over stochastic variables.

**Definition 1** *A* Stochastic Constraint Satisfaction Problem (SCSP) *is a 6-tuple* $N = \langle \mathcal{V}, \mathcal{Y}, \mathcal{D}, \mathcal{C}, \mathcal{P}, \theta \rangle$*, such that* $\mathcal{V} = (V_1, \cdots, V_n)$ *is a finite tuple of variables,* $\mathcal{Y} \subseteq \mathcal{V}$ *is the set of* stochastic variables*,* $\mathcal{D}$ *is a mapping from* $\mathcal{V}$ *to* domains *of values,* $\mathcal{C}$ *is a set of* constraints*,* $\mathcal{P}$ *is a set of* conditional probability tables*, and* $\theta \in [0,1]$ *is a* threshold*.*

- *Each constraint in* $\mathcal{C}$ *is a pair* $C = (scp_C, val_C)$*, such that* $scp_C$ *is a subset of* $\mathcal{V}$*, called the* scope *of* $C$*, and* $val_C$ *is a map from* $\mathcal{D}(scp_C)$ *to* $\{0,1\}$*.*
- *Each conditional probability table in* $\mathcal{P}$ *is a triplet* $(Y, scp_Y, prob_Y)$*, where* $Y \in \mathcal{Y}$ *is a stochastic variable,* $scp_Y$ *is a subset of variables occurring before* $Y$ *in* $\mathcal{V}$*, and* $prob_Y$ *is map from* $\mathcal{D}(scp_Y)$ *to a probability distribution over the domain* $\mathcal{D}(Y)$*.*

By $\mathcal{X}$, we denote the set $\mathcal{V} \setminus \mathcal{Y}$ of *decision variables*. If $Y \in \mathcal{Y}$ is a stochastic variable and $\tau \in \mathcal{D}(scp_Y)$ is a tuple of values in the conditional probability table of $Y$, then we use $P(Y \mid \tau)$ to denote the distribution $prob_Y(\tau)$. In particular, if $y \in \mathcal{D}(Y)$, then $P(Y = y \mid \tau)$ is the probability that $Y$ takes value $y$ given $\tau$.

Given a subset $\mathcal{U} = (V_1, \cdots, V_m) \subseteq \mathcal{V}$, an *instantiation* of $\mathcal{U}$ is an assignment $I$ of values $v_1 \in \mathcal{D}(V_1), \cdots, v_m \in \mathcal{D}(V_m)$ to the variables $V_1, \cdots, V_m$, also written $I = \{(V_1, v_1), \ldots, (V_m, v_m)\}$. An instantiation $I$ on $\mathcal{U}$ is *complete* if $\mathcal{U} = \mathcal{V}$. Given a subset $\mathcal{U}' \subseteq \mathcal{U}$, we use $I_{|\mathcal{U}'}$ to denote the restriction of $I$ to $\mathcal{U}'$, that is, $I_{|\mathcal{U}'} = \{(V_i, v_i) \in I : V_i \in \mathcal{U}'\}$. The *probability* of $I$ is given by:

$$P(I) = \prod_{Y \in \mathcal{Y} : scp_Y \subseteq \mathcal{U}} P(Y = I_{|Y} \mid I_{|scp_Y})$$

Correspondingly, the *utility* of an instantiation $I$ on $\mathcal{U}$ is given by

$$val(I) = \sum_{C \in \mathcal{C} : scp_C \subseteq \mathcal{U}} val(I_{|scp_C})$$

An instantiation $I$ is called *consistent* with a constraint $C$ if $val(I_{|scp_C}) = 1$, that is, $I$ can be projected to a tuple satisfying $C$. By extension, $I$ is *locally consistent* if $val(I) = 1$, that is, $I$ satisfies every constraint in $C$. Finally, $I$ is *globally consistent* (or *consistent*, for short) if it can be extended to a complete instantiation $I'$ which is locally consistent.

A *policy* $\pi$ for the network $N$ is a rooted tree where each internal node is labeled by a variable $V$ and each edge is labeled by a value in $\mathcal{D}(V)$. Specifically, nodes are labeled according to the ordering $\mathcal{V}$: the root node is labeled by $V_1$, and each child of a node $V_i$ is labeled by $V_{i+1}$. Decision nodes $X_i$ have a unique child, and stochastic nodes $Y_i$ have $|\mathcal{D}(Y_i)|$ children. Finally, each leaf in $\pi$ is labeled by the utility $val(I)$, where $I$ is the complete instantiation specified by the path from the root of $\pi$ to that leaf. Let $\mathcal{L}(\pi)$ be the set of all complete instantiations induced by $\pi$. Then, the *expected utility* of $\pi$ is the sum of its leaf utilities weighted by their probabilities. Formally,

$$val(\pi) = \sum_{I \in \mathcal{L}(\pi)} P(I) val(I)$$

A policy $\pi$ is a *solution* of $N$ if its expected utility is greater than or equal to the threshold $\theta$, that is $val(\pi) \geq \theta$. We mention in passing that if $\theta = 1$, then $\pi$ is a solution of $N$ if and only if $val(I) = 1$ for each path $I$ in $\mathcal{L}(\pi)$ such that $P(I) \neq 0$.

A *(decision) stage* in a SCSP is a tuple of variables $\langle X_i, Y_i \rangle$, where $X_i$ is a subset of decision variables, $Y_i$ is a subset of stochastic variables, and decision variables occurs before any stochastic variable [Hnich *et al.*, 2012]. By extension:

**Definition 2** *A* $T$-stage stochastic constraint satisfaction problem *is an SCSP* $N = \langle \mathcal{V}, \mathcal{Y}, \mathcal{D}, \mathcal{C}, \mathcal{P}, \theta \rangle$*, in which* $\mathcal{V}$ *can be partitioned into* $T$ stages*, i.e.* $\mathcal{V} = (\langle \mathcal{X}_1, \mathcal{Y}_1 \rangle, \ldots, \langle \mathcal{X}_T, \mathcal{Y}_T \rangle)$*, where* $\{\mathcal{X}_i\}_{i=1}^T$ *is a partition of* $\mathcal{X}$*,* $\{\mathcal{Y}_i\}_{i=1}^T$ *is a partition of* $\mathcal{Y}$*, and* $scp_{Y_i} \subseteq \mathcal{X}_i$ *for each* $i \in \{1, \ldots, T\}$ *and each* $Y_i \in \mathcal{Y}_i$*. If* $T = 1$*,* $N$ *is called a* one-stage SCSP*, and denoted* $\mu$SCSP*.*

Note that the problem of finding a winning policy in a SCSP is PSPACE-complete. The problem remains in PSPACE for $T$-stage $k$-player SCSPs, as each stage of the problem is in $\text{NP}^{\text{PP}}$.

## 4 An SCSP Representation of Games

In this section, we present a constraint-based representation of games. Namely, a GDL-II game G is first cast as a stochastic constraint network $N$, which encodes the rule of the game as a set of constraints. Then, $N$ is enriched by a new set of constraints describing the players' solution concepts. The final stochastic constraint network, capturing both the game rules and the players' strategies, can be solved using standard SCSP algorithms.

### 4.1 Modelling Game Rules

The translation of a $k$-player game $\mathtt{G} \in \mathcal{G}_T$ into a $T$-stage SCSP is specified as follows. We first convert G into a set ground rules $\mathtt{G}'$, whose size is polynomial in $|\mathtt{G}|$ because G is propositional. To $\mathtt{G}'$ we associate a one-stage SCSP $N = \langle [k], G_t, R_t, \{F_t\}, \{A_t\}, A_{t,0}, \{F_{t+1}\} \rangle$, where $[k] = \{1, \cdots, k\}$ is the set of players, $G_t$ is a Boolean variable indicating whether the game has reached a terminal (goal) state, and $R_t$ is the set of reward values of the game. $\{F_t\}$ and $\{F_{t+1}\}$ are the sets of fluents describing the game state at round $t$ and $t+1$, respectively; in order to respect the one-stage property, $\{F_t\}$ is a set of *decision* variables, and $\{F_{t+1}\}$ is a set of *stochastic* variables. $\{A_t\} = \{A_{t,1}, \ldots, A_{t,k}\}$ is a set of decision variables, each $A_{t,p}$ describing the set of possible moves of player $p$. Finally, $A_{t,0}$ is a stochastic variable describing the set of possible moves of the chance player.

The Horn clauses of a GDL program G can naturally be partitioned into `init` rules describing the initial state, `legal` rules specifying the legal moves at the current state, `next` rules capturing the effects of actions, `sees` rules describing the observations of each player on the game, and `goal` rules defining the players' rewards at a terminal state. `init`, `legal` and `next` rules are encoded into hard constraints in the network $N$. The `sees` rules are used to express the conditional probability table $P(f_{t+1} \mid \boldsymbol{f}, \boldsymbol{a})$, of each stochastic variable $F_{t+1}$. The last stochastic variable $A_{t,0}$ is associated to a uniform probability distribution $P(a_0 \mid \boldsymbol{f})$ over the *legal* moves of the chance player. The constraint relation is extracted in the same way as the domains of variables, by identifying all allowed combinations of constants. Similarly, `goal` rules are encoded by a constraint encoding player's rewards at a terminal state.

By repeating $T$ times this conversion process, we therefore obtain a $T$-stage SCSP encoding the $T$-horizon game G. The threshold $\theta$ is set to 1, indicating that all constraints must be satisfied by a solution policy.

### 4.2 Modelling Strategies

Based on the above translation process, the resulting SCSP $N$ encodes only the game rules of the input GDL program. In order to "solve" a game, $N$ must also incorporate, in a declarative way, the solution concepts of players. In essence, these solutions concepts or strategies are captured by two operators $\oplus$ and $\otimes$, together with a set of constraints joining them. In what follows, we write $\boldsymbol{a}_{-0} \in \{A_t\}$ as an abbreviation of $(a_1,\cdots,a_t) \in A_{t,1} \times \cdots \times A_{t,k}$, for specifying an action profile of the $k$ players (excluding the chance player 0). By extension, $\boldsymbol{a}_{-\{0,p\}} \in \{A_t\}_{-p}$ denotes an action profile $(a_1,\cdots,a_{p-1},a_{p+1},\cdots,a_k)$ of $k-1$ players exluding both $p$ and the chance player. The shortcut $\boldsymbol{f} \in \{F_t\}$ is defined similarly.

To incorporate players' strategies, each one-stage SCSP in $N$ is enriched with several constraints. The first constraint $u_{t,0}$, defined over the scope $([k],\{F_t\},\{A_t\},U_0)$, associates to each player $p \in [k]$, each state description $\boldsymbol{f} \in \{F_t\}$, and each action profile $\boldsymbol{a}_{-0} \in \{A_t\}$, the value $u_{t,0}(p,\boldsymbol{f},\boldsymbol{a}_{-0})$ in $U_0$ given by:

$$u_{t,0}(p,\boldsymbol{f},\boldsymbol{a}_{-0}) = \sum_{a_0 \in A_{t,0}} \sum_{\boldsymbol{f}' \in \{F_{t+1}\}} P(a_0 \mid \boldsymbol{f}) P(\boldsymbol{f}' \mid \boldsymbol{f},\boldsymbol{a})\, u^*_{t+1,p}(\boldsymbol{f}')$$

where $\boldsymbol{a} = (a_0,\boldsymbol{a}_{-0})$ is the final action profile of all $k+1$ players (including the chance player), and $u^*_{t+1,p}(\boldsymbol{f}')$ is the utility of player $p$ at state $\boldsymbol{f}'$. Intuitively, $u_{t,0}(p,\boldsymbol{f},\boldsymbol{a}_{-0})$ is the expected utility of player $p$ when the joint action $\boldsymbol{a}_{-0}$ of the $k$ players is applied on the game position $\boldsymbol{f}$. Based on this value, we can define the utility of each player's move. To this end, we associate to each player $p$ a constraint $u_{t,p}$ defined over the scope $(\{F_t\},A_p,U_p)$. This constraint maps each state description $\boldsymbol{f} \in \{F_t\}$ and each action $a_p \in \{A_p\}$ to the value $u_{t,p}(\boldsymbol{f},a_p)$ in $U_p$ given by

$$u_{t,p}(\boldsymbol{f},a_p) = \bigoplus_{\boldsymbol{a}_{-\{0,p\}} \in \{A_t\}_{-p}} u_{t,0}(p,\boldsymbol{f},\boldsymbol{a}_{-0})$$

In a symmetrical way, we can also capture the player's utility of each state. To this end, we associate to each player $p$ a constraint $u^*_{t,p}$ defined over the scope $(\{F_t\},U^*_p)$. This constraint maps each state description $\boldsymbol{f} \in \{F_t\}$ to a value $u^*_{t,p}(\boldsymbol{f})$ in $U^*_p$, defined by the following condition: if $\boldsymbol{f}$ is a terminal state, then $u^*_{t,p}(\boldsymbol{f})$ is the reward of $p$ at $\boldsymbol{f}$, as specified by the goal rules. Otherwise,

$$u^*_{t,p}(\boldsymbol{f}) = \bigotimes_{a_p \in A_{t,p}} u_{t,p}(\boldsymbol{f},a_p)$$

Finally, the optimal play is captured by simply adding the equality constraints $u^*_{t,p}(\boldsymbol{f}) = u_{t,p}(\boldsymbol{f},a_p)$ defined over the scope $(U^*_p,U_p)$. These equalities filter out any player's move that is not optimal. Various solution concepts can be defined according the operators $\oplus$ and $\otimes$. In our experiments, we use the standard *maximin* strategy (for all players) given by $\otimes = \max$ and $\oplus = \min$.

## 5 MAC-UCB-II

Based on a fragment of SCSP for GDL games, we now present our resolution technique called MAC-UCB-II, an extension of the MAC-UCB algorithm [Koriche *et al.*, 2016] for GDL-II. As indicated above, the stochastic constraint network of a GDL program is a sequence of $\mu$SCSPs, each associated with a game round $t$ in $\{1,...,T\}$. For each $\mu\text{SCSP}_t$ in $\{1,...,T\}$, MAC-UCB-II searches the set of feasible policies by splitting the problem into two parts: a CSP and a $\mu$SCSP (smaller than the original one). The first part is solved using the MAC algorithm [Sabin and Freuder., 1994; 1997] and the second part with the FC algorithm dedicated to SCSP [Walsh, 2002]. Then, a sampling with confidence bound (UCB) is performed to estimate the expected utility of each feasible solution of $\mu\text{SCSP}_t$.

Recall that the task of sequential decision associated to a strategic game is an optimization problem. Classically, this problem is addressed by solving a sequence of stochastic satisfaction problems. In practice, each time our player has to play, a slot of time is dedicated to choose the next action in which we solve as much $\mu$SCSPs as possible.

MAC-UCB-II adds conditional probabilities in each $\mu\text{SCSP}_t$ based on the sees rules for each fluent.

Figure 2 represents the constraint network of the $t$th $\mu$SCSP returned by our encoding procedure on the GDL program of Example 1. For the sake of clarity, the identifiers representing variables and domains were renamed: H (heads),T (tails),U (unset) denote the different values of a side of a coin. First, the two variables $\texttt{terminal}_t$ and $\texttt{score}_t$ stand for the end of the game and the remaining reward. These variables are extracted from terminal and goal(P,V) keywords. Their associated domain is boolean for the first one and the set of possible rewards for the second one. From the role keyword, two variables $\texttt{control}_t$ and $\texttt{control}_{t+1}$ determine the player for the round $t$ and $t+1$. The domain of these variables corresponds to the set of players defined with the role statement. The states of the game for the round $t$ and $t+1$ are represented by the set of variables derived from the next keyword. The domains of these variables corresponds to the different combinaison of values of the fluents $S$, and $S1$ $S2$. One can extract from the legal keywords the variables $\texttt{choose}_t$ and $\texttt{play}_t$. Note that $\texttt{choose}_t$ is a stochastic variable since it corresponds to legal move of the chance player. The second stochastic variable is $coins_t$. The conditional probability of the fluent $coins_t$ for Alice is directly associated to the corresponding sees rules ( sees(alice,coins(S1,_)) $\leftarrow$ does(random,choose(S1,S2)). ) indicating that Alice percepts only the first coin when Random chooses its two coins. A terminal constraint links the $\texttt{terminal}_t$ variable with the two variable coins. The game is ended when no unset value is set to a coin. In the same manner the goal constraint links the side of the different coins with the associated reward. The next constraints allow the game to change from one state ($t$) to another ($t+1$) depending on the chosen action for the player (variable $\texttt{play}_t$) and the chosen action for random (variable $\texttt{choose}_t$).

Figure 3 illustrates the tree search obtained directly by the SCSP model built by MAC-UCB-II. To simplify, we have renamed the first coin of the random player $R_1$, its second $R_2$ and the coin of Alice $A$. MAC-UCB-II computes an average reward of 75 for the red sub-trees and 25 for the blue sub-trees. Consequently MAC-UCB-II plays always on the red sub-tree depending on the first coin revealed by the random player. We can easily remark that the strategy followed by MAC-UCB-II (to guarantee the best score) is to play the same side of the coin than the one revealed by the random player. Note that, for game

| Variable | Domain |
|---|---|
| $\texttt{terminal}_t$ | {true,false} |
| $\texttt{score}_t$ | {0,50,100} |
| $\texttt{percept}_t$ | {coins(H),coins(T)} |
| $\texttt{coin}_t$ | {H,T,U} |
| $\texttt{coins}_t$ | {HH,TT,HT,TH,UU} |
| $\texttt{control}_t$ | {alice,random} |
| $\texttt{choose}_t$ | {HH,TT,HT,TH,noop} |
| $\texttt{play}_t$ | {H,T,noop} |
| $\texttt{coin}_{t+1}$ | {H,T,U} |
| $\texttt{coins}_{t+1}$ | {HH,TT,HT,TH,UU} |
| $\texttt{control}_{t+1}$ | {alice,random} |

Variables and domains

| $\texttt{coin}_t$ | $\texttt{coins}_t$ | $\texttt{terminal}_t$ |
|---|---|---|
| H | HH | true |
| H | HT | true |
| ⋮ | ⋮ | ⋮ |
| T | TT | true |
| H | HU | false |
| H | UH | false |
| ⋮ | ⋮ | ⋮ |
| U | UU | false |

terminal constraint

| $\texttt{coin}_t$ | $\texttt{coins}_t$ | $\texttt{score}_t$ |
|---|---|---|
| H | HH | 100 |
| T | TT | 100 |
| H | HT | 50 |
| H | TH | 50 |
| T | HT | 50 |
| T | TH | 50 |
| H | TT | 0 |
| T | HH | 0 |

goal constraint

| $\texttt{control}_t$ | $\texttt{play}_t$ |
|---|---|
| alice | H |
| alice | T |
| random | noop |

legal constraints

| $\texttt{control}_t$ | $\texttt{control}_{t+1}$ |
|---|---|
| random | alice |

| $\texttt{coins}_t$ | $\texttt{choose}_t$ | $\texttt{coins}_{t+1}$ |
|---|---|---|
| HH | noop | HH |
| TT | noop | TT |
| HT | noop | HT |
| TH | noop | TH |
| UU | HH | HH |
| UU | TT | TT |
| UU | HT | HT |
| UU | TH | TH |

| $\texttt{coin}_t$ | $\texttt{play}_t$ | $\texttt{coin}_{t+1}$ |
|---|---|---|
| U | noop | U |
| U | H | H |
| U | T | T |

next constraints

| $\texttt{control}_t$ | HH | HT | TH | TT | noop |
|---|---|---|---|---|---|
| random | 1/4 | 1/4 | 1/4 | 1/4 | 0 |
| alice | 0 | 0 | 0 | 0 | 1 |

Conditional probability table of $\texttt{choose}_t$

| $\texttt{percept}_t$ | HH | HT | TH | TT |
|---|---|---|---|---|
| $\texttt{coins}_t$(H,_) | 1/2 | 1/2 | 0 | 0 |
| $\texttt{coins}_t$(T,_) | 0 | 0 | 1/2 | 1/2 |

Conditional probability of $\texttt{coins}_t$ associated with the percept of alice

Figure 2: A $\mu$SCSP encoding the GDL program of Hidden Matching Pennies. ($H = heads, T = tails, U = unset$)

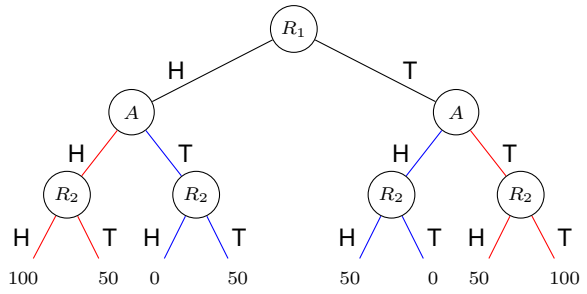


Figure 3: The tree search of MAC-UCB-II for the hidden matching pennies.

with a large tree search, MAC-UCB-II can not completely explore it during the alloted deliberation time. In this case, we use UCB to simulate an expected utility for each non terminal node.

## 6 Experimental Results

Based on our framework and algorithm, we now present a series of experimental results conducted on a cluster of Intel Xeon E5-2643 CPU 3.3 GHz with 64 GB of RAM and four threads under Linux. Our framework was implemented in C++.

We have selected 10 deterministic games described in GDL-I from the Tiltyard server[1], and 15 stochastic games in GDL-II including 5 with no sees rules. A majority of GDL-II games

[1] http://tiltyard.ggp.org/

have been selected from the Dresden server[2]. Experiments have been performed on a large variety of games for a total amount of 9,500 matches. More detailed information about specific game can be found on boardgamegeek.com.

**Setup.** Game competitions have been organized between different general game players. The first player is the multi-player version of the UCT algorithm [Sturtevant, 2008], which is the state-of-the-art algorithm for the deterministic games in GGP. The second player is Sancho version 1.61c [3], a Monte Carlo Tree Search player elaborated by S. Draper and A. Rose, which has won the 2014 International General Game Playing Competition. The third player is [Cazenave, 2015]'s GRAVE algorithm, which implements the Generalized Rapid Action Value Estimation technique, a generalization of the RAVE method [Gelly and Silver, 2007] adapted for GGP. Finally, we also compare our player to CFR [Shafiei *et al.*, 2009], a GGP implementation of the well-known CounterFactual Regret technique used in partially observable games. We did not use HyperPlayer-II algorithm [Schofield and Thielscher, 2015], because it was not available online during experiments and we can not adapt it to our model.

For all matches, we used the 2015 Tiltyard Open (last international GGP competition) setup: 180 seconds for the start clock and 15 seconds for the play clock.

Our algorithm MAC-UCB-II needs to split the play clock time into two parts: exploitation (the MAC part) and exploration (the UCB part). The same parameters as MAC-UCB in [Koriche *et al.*, 2016] was used (90 % of the time dedicated to exploitation and 10 % to exploration).

For all the GDL games, we realized 100 matches between MAC-UCB-II and each other player. For the sake of fairness, the role of players were exchanged during each match.

Our results are summarized in Table 2. The rows are grouped into 4 parts, respectively capturing GDL-I games, GDL-II games with a random player (but no sees rules), GDL-II games with information sharing (all players have the same partial observation of the game state) and one-player GDL-II games with sees rules. Each column reports the average percent of wins for MAC-UCB-II against the selected adversary. For example, the entry of the UCT column for the Chess game (3rd row) indicates that, on average, MAC-UCB-II wins 58% of contests against UCT. Since the fourth group only involves one-player games, the player columns (including MAC-UCB-II) report the average number of times the one-player game is solved, divided by the total number of matches.

**Results on GDL-I.** For deterministic games, UCT and CFR were defeated by MAC-UCB-II, with sometimes a score greater than 80 % (e.g. Reversi Suicide). Sancho, the IGGPC'14 leader, and GRAVE are on average less competitive than MAC-UCB-II. Notable examples are Hex, Connect Four 20x20 against Sancho and Chess against GRAVE. The only exceptions are the Amazons torus 10x10 against GRAVE with a score on average of 46 % and for the Breakthrough suicide against Sancho with 49 %.

**Results on GDL-II.** For the GDL-II games with no sees rules but a random player, the four opponents are beaten with a score higher than 60% for MAC-UCB-II against Sancho, UCT, CFR, and higher than 50% against GRAVE.

For games with partial observations, Sancho does not participate because it is dedicated to GDL-I (modulo a possible simulation of the chance player). For the puzzle games with imperfect information, MAC-UCB-II is the best player for those games except for the Wumpus or GuessDice. However, the GuessDice is not significant because there is no strategy to win, but just chance to guess the Dice. We can note that on the MasterMind, MAC-UCB-II and CFR obtain an equivalent score.

Finally, the last five games are partially observable games with information sharing. For instance, the TicTacToe Latent Random 10x10 involves 3 players, including the chance player. The chance player randomly places a cross or a round in the free cases

Table 2: Results of MAC-UCB-II on GDL games.

| Multi-players GDL-I games | | | | |
|---|---|---|---|---|
| **Game** | **UCT** | **CFR** | **GRAVE** | **Sancho** |
| Amazons torus 10x10 | 62 | 73 | 46 | 52 |
| Breakthrough suicide | 71 | 80 | 54 | 49 |
| Chess | 58 | 82 | 71 | 53 |
| Connect Four 20x20 | 76 | 83 | 51 | 72 |
| English Draughts | 69 | 78 | 55 | 51 |
| Hex | 81 | 76 | 64 | 64 |
| Shmup | 75 | 66 | 51 | 51 |
| Skirmish zero-sum | 63 | 77 | 63 | 59 |
| TTCC4 2P | 79 | 83 | 50 | 54 |
| Reversi Suicide | 86 | 86 | 57 | 53 |
| Multi-players GDL-II games with a random player | | | | |
| **Game** | **UCT** | **CFR** | **GRAVE** | **Sancho** |
| Backgammon | 70.0 | 66.7 | 54.6 | 97.5 |
| Can't Stop | 73.1 | 67.5 | 62.1 | 94.3 |
| Kaseklau | 73.6 | 71.5 | 56.2 | 80.3 |
| Pickomino | 65.2 | 60.6 | 60.2 | 74.2 |
| Yahtzee | 72.1 | 72.3 | 53.9 | 72.0 |
| Multi-players GDL-II games with information sharing | | | | |
| **Game** | **UCT** | **CFR** | **GRAVE** | **Sancho** |
| Pacman | 57.2 | 58.9 | 55.1 | × |
| Schnappt Hubi | 71.3 | 57.5 | 56.2 | × |
| Sheep & Wolf | 68.2 | 56.2 | 55.0 | × |
| TicTacToe Latent Random 10x10 | 82.6 | 78.7 | 69.1 | × |
| War (card game) | 72.0 | 69.1 | 66.0 | × |

| One-player GDL-II games with sees rules | | | | |
|---|---|---|---|---|
| **Game** | **MAC-UCB-II** | **UCT** | **CFR** | **GRAVE** |
| GuessDice | 15 | 15.7 | 16 | 16.5 |
| MasterMind | 67.8 | 53.8 | 68.1 | 60.1 |
| Monty Hall | 65.2 | 62.2 | 63.1 | 64.3 |
| Vaccum Cleaner Random | 61.5 | 34 | 46 | 58.8 |
| Wumpus | 32.1 | 40.1 | 44.1 | 51.2 |

[2] http://ggpserver.general-game-playing.de/

[3] http://sanchoggp.github.io/sancho-ggp/

and the other two players observe the squares marked by chance, only when they try to mark it. Schnappt Hubi and Sheep & Wolf are cooperative games, where all agent share their observation in order to beat the chance player. MAC-UCB-II wins with an average score higher than 55% against each opponent, with an average number of wins of about 69% for the TicTacToe Latent Random 10x10 game.

## 7 Conclusion

This paper has presented a consequent forward step on using constraint-based formalisms for GGP by considering a large subclass of GDL-II games. We have identified an important fragment of imperfect information games that can be cast as SCSPs, and which can be solved using general-purpose Constraint Programming techniques. Our sequential decision algorithm for GDL-II games exploits constraint propagation and Monte Carlo sampling. Based on extensive experiments involving various types of games and computer opponents, we showed that general-purpose CP techniques are paying off.

A work in progress is to focus on symmetries in order to strongly decrease the search space. The idea is to exploit symmetries to avoid the useless resolution of symmetrical $\mu$SCSPs. For that, we need to unify and extend the constraint approaches [Cohen *et al.*, 2006] and the ones in GGP [Schiffel, 2010].

## References


[Bonnet and Saffidine, 2014] Edouard Bonnet and Abdallah Saffidine. On the complexity of general game playing. In *Proceedings of CGW*, pages 90–104, 2014.

[Bowling *et al.*, 2015] Michael Bowling, Neil Burch, Michael Johanson, and Oskari Tammelin. Heads-up limit hold'em poker is solved. *Science*, 347(6218):145–149, 2015.

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

[Cohen *et al.*, 2006] Davis 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.

[Geißer *et al.*, 2014] Florian Geißer, Thomas Keller, and Robert Mattmüller. Past, present, and future: An optimal online algorithm for single-player GDL-II games. In *Proceedings of ECAI*, pages 357–362, 2014.

[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 and Thielscher, 2014] Michael R. Genesereth and Michael Thielscher. *General Game Playing*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers, 2014.

[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.

[Ginsberg, 2001] Matthew L. Ginsberg. GIB: imperfect information in a computationally challenging game. *J. Artif. Intell. Res. (JAIR)*, 14:303–358, 2001.

[Goldsmith and Mundhenk, 2007] Judy Goldsmith and Martin Mundhenk. Competition adds complexity. In *Proceedings of NIPS*, pages 561–568, 2007.

[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.

[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.

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

[Sabin and Freuder., 1994] D. Sabin and E.C. Freuder. Contradicting conventional wisdom in constraint satisfaction. In *Proceedings of CP'94*, pages 10–20. Springer, 1994.

[Sabin and Freuder., 1997] D. Sabin and E.C. Freuder. Understanding and improving the mac algorithm. In *Proceedings of CP'97*, pages 167–181. Springer, 1997.

[Schiffel and Thielscher, 2011] Stephan Schiffel and Michael Thielscher. Reasoning about general games described in GDL-II. In *Proceedings of AAAI 2011*, pages 846–851. AAAI Press, 2011.

[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*. AAAI Press, 2010.

[Schofield and Thielscher, 2015] Michael John Schofield and Michael Thielscher. Lifting model sampling for general game playing to incomplete-information models. In *Proceedings of AAAI 2015*, pages 3585–3591, 2015.

[Shafiei *et al.*, 2009] Mohammad Shafiei, Nathan Sturtevant, and Jonathan Schaeffer. Comparing uct versus cfr in simultaneous games. In *IJCAI Workshop on General Game Playing*, 2009.

[Sturtevant, 2008] Nathan R. Sturtevant. An analysis of uct in multi-player games. *ICGA Journal*, 31(4):195–208, 2008.

[Thielscher, 2010] Michael Thielscher. A general game description language for incomplete information games. In *Proceedings of AAAI'10*, pages 994–999, 2010.

[Walsh, 2002] Toby Walsh. Stochastic constraint programming. In *Proceedings of ECAI'02*, pages 111–115, 2002.