Upper Confidence Tree-Based Consistent Reactive Planning Application to MineSweeper

Michèle Sebag[1] and Olivier Teytaud[2]

[1] TAO-INRIA, LRI, CNRS UMR 8623,
Université Paris-Sud, Orsay, France
[2] OASE Lab, National University of Tainan, Taiwan

Abstract. Many reactive planning tasks are tackled through myopic optimization-based approaches. Specifically, the problem is simplified by only considering the observations available at the current time step and an estimate of the future system behavior; the optimal decision on the basis of this information is computed and the simplified problem description is updated on the basis of the new observations available in each time step. While this approach does not yield optimal strategies *stricto sensu*, it indeed gives good results at a reasonable computational cost for highly intractable problems, whenever fast off-the-shelf solvers are available for the simplified problem.

The increase of available computational power − even though the search for optimal strategies remains intractable with brute-force approaches − makes it however possible to go beyond the intrinsic limitations of myopic reactive planning approaches.

A consistent reactive planning approach is proposed in this paper, embedding a solver with an Upper Confidence Tree algorithm. While the solver is used to yield a consistent estimate of the belief state, the UCT exploits this estimate (both in the tree nodes and through the Monte-Carlo simulator) to achieve an asymptotically optimal policy. The paper shows the consistency of the proposed *Upper Confidence Tree-based Consistent Reactive Planning* algorithm and presents a proof of principle of its performance on a classical success of the myopic approach, the MineSweeper game.

1 Introduction

This paper focuses on reactive planning, of which power plants maintenance [20] or the MineSweeper game [13,5,18] are typical problem instances. The difficulty of reactive planning is due to the great many uncertainties about the problem environment, hindering the search for optimal strategies. For this reason, most reactive planners are content with selecting the current move based on their only current knowledge.

However, cheap and ever cheaper computational power makes it possible nowadays to aim at the best of both worlds, combining an approximate model of

Y. Hamadi and M. Schoenauer (Eds.): LION 6, LNCS 7219, pp. 220–234, 2012.

the problem under examination with knowledge-based solvers. Taking inspiration of earlier work devoted to the combination of domain knowledge-based heuristics with Monte Carlo Tree Search and Upper Confidence Trees [21,22], this paper shows how fast solvers combined with Upper Confidence Tree approaches can be boosted to optimal planning. A proof of principle of the approach is given on the MineSweeper game, an NP-complete partially observable game. The choice of this game, despite its moderate complexity (indeed there exist many games with EXP or 2EXP computational complexity) is motivated as there exists efficient MineSweeper player algorithms, while many partially observable games still lack efficient algorithms.

The paper is organized as follows. Section 2 introduces the formal background of partially observable Markov decision processes (POMDP). The MineSweeper game is described in section 3. POMDP algorithms are discussed in Section 4 and illustrated on the MineSweeper problem. Section 5 describes our contribution and gives an overview of the proposed UC-CRP (*Upper Confidence Belief Estimation-based Solver*) algorithm, combining the Upper Confidence Tree algorithm with existing MineSweeper algorithms. Section 6 discusses the consistency of existing MineSweeper algorithms and establishes UC-CRP consistency. Section 7 reports on the comparative experimental validation of UC-CRP and the paper concludes with some perspectives for further work.

2 Formal Background

This section introduces the main notations used through the paper. After the standard terminology [24,6], a Markov Decision Process (MDP) is described from its space of states $\mathcal{S}$, its state of actions $\mathcal{A}$, the probabilistic transition model $p(s, a, s')$ describing the probability of arriving in state s' upon triggering action a in state s (with $\sum_{s'} p(s, a, s') = 1$), and a reward function r ($r : \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}$). A policy π maps a (finite sequence of) state(s) onto an action, possibly in a stochastic manner ($\pi : \mathcal{S}^i \times \mathcal{A} \mapsto \mathbb{R}$, with $\sum_a \pi(s, a) = 1$). Given initial state s_0, a probabilistic policy π thus defines a probability distribution on the cumulative rewards.

A partially observable MDP (POMDP) only differs from a MDP as each state s is partially visible through an observation o. Letting $\mathcal{O}$ denote the space of observations, a strategy π maps a (finite sequence of) observation(s) onto an action, possibly in a stochastic manner ($\pi : \mathcal{O}^i \times \mathcal{A} \mapsto \mathbb{R}$, with $\sum_a \pi(o, a) = 1$). In the following, we shall refer to s as *complete state* (including the partially hidden information), whereas observation o is referred to as *state* for the sake of consistency with the terminology used in the MineSweeper literature.

The feasible set $\mathcal{F}(o)$ denotes the set of complete states compatible with (a sequence of) observation(s) o. A consistent belief state estimation $p(s|o)$ is an algorithm sampling the uniform probability distribution on the feasible set $\mathcal{F}(o)$, given observation o. An asymptotically consistent belief state estimation is an algorithm yielding a probability distribution on $\mathcal{S}$, and converging toward the uniform probability distribution on $\mathcal{F}(o)$ as the computational effort goes to infinity.

3 The MineSweeper Game

MineSweeper is a widely spread game, which has motivated a number of studies as a model for real problems [12], or a challenge for machine learning [7] or genetic programming [15], or for pedagogical reasons [4] (including a nice version aimed at teaching quantum mechanics [11]).

3.1 The Rule

MineSweeper is played on a $h \times w$ board. The player starts by choosing a location on the board; thereafter, M mines are randomly placed on the board, anywhere except on the chosen location in order to avoid the immediate death out of bad luck. The location of the mines defines the (unknown) complete state of the game. The level of difficulty of the game is defined by (h, w, M). At each turn, the player selects an uncovered location ℓ (Fig. 1). Then,

- Either there is a mine in ℓ (immediate death) and the player has lost the game;
- Or there exists no mine and the player is informed of the number of mines in the 8-neighborhood of ℓ. Usually, when there is no mine in the 8-neighborhood of ℓ the player is assumed to automatically play all neighbors of ℓ, which are risk-less moves, in order to save up time[1].

The player wins iff she plays all non-mine $h \times w - M$ locations and avoids the immediate death termination. In each time step the current observation is made of the locations selected so far and the number of mines in their 8-neighborhood; by construction, the observation is consistent with the complete state s (the actual locations of the M mines).

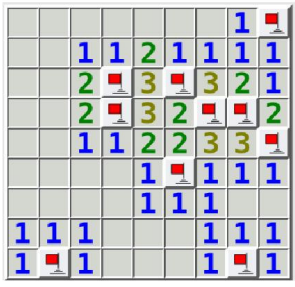

Fig. 1. The MineSweeper game, an observation state

[1] In some variants, e.g. the Linux Gnomine, the initial location is such that there is no mine in its 8-neighborhood. While this rule efficiently reduces the probability of loss out of bad luck, it is not widely used in the literature. For this reason, and for the sake of fair comparisons with the state of the art, this rule will not be considered in the remainder of the paper unless stated otherwise.

MineSweeper algorithms fall in two categories. In the former category are artificial intelligences (AIs) tackling the underlying Markov Decision Process, which is quite hard to analyze and solve [18]. In the latter category are AIs based on heuristics, and relying on Constraint Solving algorithms (CSP) [23]. In particular, a widely used heuristic strategy is based on evaluating the feasible set made of all complete states consistent with the current observation, counting for each location the percentage of states including a mine in this location (referred to as probability of immediate death), and playing the location with lowest probability of immediate death.

Two facts need be emphasized. On the one hand, CSP is a consistent belief state estimator [18]. On the other hand, the strategy of selecting the location with minimal probability of immediate death is suboptimal [9], irrespective of the computational effort involved in computing the feasible set (more in section 6).

3.2 State of the Art

The MineSweeper game is solved until 4x4 board [18]. Although it is only NP-complete in the general case [5] as already mentioned, it is more complex than expected at first sight [13,5,18].

The best CSP-based published methods have a probability of success of 80% at the beginner level (9x9, 10 mines), 45% at the intermediate level (16x16, 40 mines) and 34% in expert mode (16x30, 99 mines). Another, limited search method proposed by Pedersen [19] achieves a breakthrough with 92.5% success (respectively 67.7% success) at the beginner (resp. intermediate) level.

The importance of the initial move must be emphasized, although it is not specifically considered in CSP approaches. Our claim is that the good performances of some MineSweeper players can be attributed to good initialization heuristics or opening books, combined with standard CSP. However, as shown in [9] the initial move is not the only reason for CSP approaches being suboptimal; many small patterns lead to suboptimal moves with non-zero probability.

4 POMDP Algorithms

Many POMDP algorithms feature two components. The first one is in charge of estimating the belief state, i.e. the probability distribution on the complete state space conditioned by the current observation. The second component is in charge of move selection, based on the estimated belief state. Both components are tightly or loosely coupled depending on the problem and the approach.

CSP-based MineSweeper approaches feature a consistent belief state estimation and a trivial move selection algorithm (select the move with lowest immediate death probability). Let us briefly illustrate the main approaches proposed for belief state estimation in the MineSweeper context.

4.1 Rejection Method

The simplest approach is the rejection method; its pseudo-code is given in Alg. 1. It proceeds by uniformly drawing a complete state and rejecting it if not consistent with the observation. By construction, the rejection method thus is consistent (i.e. it proposes an exact belief state estimation) and slow (moderate to large computational efforts are required to ensure a correct estimate).

Algorithm 1. The rejection method for consistent belief state estimation.

Input: observations.
Output: a (uniformly sampled among consistent states) complete state.
$s \leftarrow$random complete state
//uniform selection in [0,1]
while $U[0,1] \geq C \times Likelihood(s|o)$ **do**
  $s \leftarrow$random complete state
end while
Return s

Parameter C is such that $C \times Likelihood(s|o) \leq 1$. In the case where the likelihood of an observation o given complete state s, is binary (0 or c), then $C = 1/c$ is the optimal choice, where state s is accepted iff it is consistent with observation o. In the MineSweeper case, $c = 1$: testing whether a complete state is consistent with the current observation is trivial.

Despite its simplicity, the rejection method is reported to outperform all other methods in [9], which is attributed to the fact that it is the only consistent method. In particular, we shall see that the Markov-Chain Monte-Carlo method (below) is only *asymptotically* consistent.

4.2 Constraint Solving (CSP)

Constraint Solving approaches usually rely on enumerating or estimating the whole feasible set, including all complete states which are consistent with the observations; their pseudo-code is given in Alg. 2. Indeed many implementations thereof have been proposed. Note that as far as the feasible set is exhaustively determined, CSPs are optimal belief state estimation algorithms in the MineSweeper context. The feasible set enables to compute for each location its probability of being a mine. However CSP-based approaches involve an intrinsically myopic move selection, optimizing some 1-step ahead reward (the probability of immediate death). As already mentioned, this greedy selection policy is not optimal.

Note that CSP-based approaches also enable to compute the exact probability of transition to a given state (which we shall use in Section 5), which has never been exploited within a tree search to our best knowledge.

Algorithm 2. The CSP algorithm for playing MineSweeper.

Input: observation o; total number M of mines
for each location ℓ on the $h \times w$ board **do**
  $Nb(\ell) \leftarrow 0$
end for
for each feasible state s (positioning of the M mines consistent with o) **do**
  for each location ℓ **do**
    If s includes a mine in ℓ, $Nb(\ell) \leftarrow Nb(\ell) + 1$
  end for
end for
Select move ℓ uniformly chosen among the set of (uncovered) locations with minimum $Nb(\ell)$.

4.3 Markov-Chain Monte-Carlo Approaches

Markov-Chain Monte-Carlo (MCMC) algorithms, of Metropolis-Hastings (MH) is a widely studied example, achieve the asymptotically consistent estimation of the belief state. In the MineSweeper context, and more generally when observations are deterministic with binary likelihood depending on the belief state, Metropolis-Hastings boils down to Alg. 3, under the assumption of a symmetric transition kernel (that is, the probability of mutating from state s to state s' is equal to the probability of mutating from state s' to state s).

Note that the asymptotic properties of MH are not impacted by the initialization to a consistent complete state (line 3 of Alg. 3), which only aims at saving up time. According to [9] and for the mutation used in the MineSweeper context however, the MH distribution is significantly biased by the choice of the initial state, making the overall MH algorithm weaker than the simple rejection algorithm (Alg. 1).

Algorithm 3. The Metropolis-Hastings algorithm with symmetric transition kernel in the case of deterministic binary observations.

Input: observation o, number T of iterations.
Output: a (nearly uniformly sampled) complete state consistent with o.
Init: Find state s consistent with o by constraint solving
Select a number T of iterations by heuristic methods
  // in [9], T depends on the number of UCT-simulations.
for $t = 1 \dots T$ **do**
  Let s' be a mutation of s
  if s' is consistent with o **then**
    $s \leftarrow s'$
  end if
end for

Several MH variants have been investigated by [9], with disappointing results compared to the rejection method. This partial failure is blamed by the authors

on the asymptotic MH consistency (as opposed to the "real" consistency of the rejection method). Indeed one cannot exclude that some MH variant, e.g. with some optimal tuning of its many parameters, may yield much better results. Still, it is suggested that a robust problem-independent MH approach is yet to be found.

In summary, our goal is to propose a fully consistent belief state estimation algorithm, such that it is faster than the rejection method and easier to adjust than MCMC.

5 Upper Confidence Tree-Based Consistent Reactive Planning

The contribution of the present paper is the UC-CRP algorithm, combining the consistent CSP-based belief state estimation and the consistent MDP solver UCT to achieve a non-trivial move selection. The Upper Confidence Tree algorithm is first briefly reminded for the sake of self-containedness, before detailing UC-CRP.

5.1 Upper Confidence Tree

Referring the reader to [10,14,17] for a comprehensive introduction, let us summarize the main steps of the Upper Confidence Tree algorithm (Fig. 2). UCT gradually constructs a search tree initialized to a single root node, by iterating tree-walks also referred to as *episodes*. Each episode involves two phases. In the so-called bandit part, the current action is selected after the Multi-Armed Bandit setting, using the Upper Confidence Bound formula [3] on the basis of the average reward and number of trials associated to each action, and the current state is updated accordingly until i) reaching a new state s (not already stored in the tree); or ii) reaching a final state. In the first case, UCT switches to the so-called Monte-Carlo or random phase, where an action is randomly selected until reaching a final state. Upon reaching a final state, the reward of the episode is computed, and the counter and average reward indicators attached to every stored node of the episode are updated.

UCT is well known for its ability to work on problems with little or no expert information. In particular, it does not require any heuristic evaluation function, although it enables to use prior knowledge encoded in heuristic strategies. These heuristic strategies can be used in the bandit or random parts of the algorithm, or in the episode evaluation. In particular, long-term effects of the decisions taken in an episode can be accounted for through simulating complete runs (as in $TD(1)$ methods, see e.g. [6]).

Indeed, a number of UCT variants have been considered in the literature, involving how to choose the actually selected action (last line in Fig. 2), or controlling the number of actions considered in each step of the bandit part, e.g. using progressive widening.

- Let the tree root be initialized to the current state s_0
- Repeat *// simulate an episode*
  - Bandit part *// bandit part of episode*
    - $s = s_0$ *// start at the root*
    - Iterate :
      Select $a(s)$ using the UCB formula and draw $s' \sim p(s, a, s')$.
      If s' is not yet stored in the tree, goto Monte-Carlo part.
      If s' is a final state, goto Scoring part $s \leftarrow s'$
  - Monte-Carlo part *// Monte-Carlo part of episode*
    - Add s' as son node of s, a. $s \leftarrow s'$
    - Repeat until s is a final state
      Select action a randomly; $s' \sim p(s, a, s'); s \leftarrow s'$
  - Scoring part *// scoring part of episode.*
    Let r be the overall reward associated to the episode. For each node (s, a) of the episode which is stored in the tree,
    - Update average reward $r(s, a)$ by r ($r(s, a) \leftarrow \frac{n(s,a)*r(s,a)+r}{n(s,a)+1}$) and likewise update the reward variance $v(s, a)$.
    - Increment counter $n(s, a)$ by 1;

  until reaching the allowed number of episodes/the computational budget.

- Play the action played most often from the root node s_0.

Fig. 2. Sketch of the UCT algorithm

5.2 UC-CRP = Belief State Estimation + Upper Confidence Trees

UC-CRP is an UCT variant embedding several CSP-based MineSweeper modules.

The first module is the single point strategy (SPS) found in PGMS[2]. SPS achieves a limited constrained propagation. Assuming that k out of the n neighbors of some location ℓ are mines, with m non-mine neighbor locations and $n - m - p$ yet uncovered locations, then two particular cases are considered. In the first case, $k = p$: it is easily seen that none of the uncovered neighbors is a mine, therefore all these neighbors can be played automatically to save up time (as already mentioned, most standard MineSweeper AIs include this constraint propagation heuristics). In the second case, $k = n - m - p$ and one likewise sees that all uncovered neighbors are mines. This case can also be automatically handled through the constraint propagation (which only involves risk-less moves).

The second module is a constraint solver, determining the feasible set of all complete states consistent with the current observation. Note that least constrained variables are used first; whereas most constrained variables should be used first when looking for one single solution, it is better to use least constrained variables first when looking for all solutions. CSP provides useful information at a moderate computational cost, including: i) the probability for each location to cover a mine, and thus the set of risk-less moves if any; ii) the exact probability distribution of the next state (i.e. how will be the board after the next

[2] `http://www.ccs.neu.edu/home/ramsdell/pgms/`

Table 1. The *Upper Confidence Tree-based Consistent Reactive Planning* combines UCT with problem domain-based solvers CSP and SPS (section 5.2). The CSP components are used to both select risk-less or optimal moves, and provide a belief state estimate (*). The notion of risk-less moves and moves with minimal risk depend on the considered problem. In the MineSweeper context, a risk-less move is one with 0 probability of immediate death.

Feature	In UC-CRP-Mine Sweeper
Node creation	One per simulation
Progressive widening	Double
Leaf evaluation	Monte-Carlo simulation
Bandit phase	Upper Confidence Bound formula with variance estimates [1]
Random phase Single point strategy (SPS)	if possible, returns a risk-less move/action
Constraint Satisfaction Solver (CSP)	If possible select a risk-less move (CSP can find all such moves if any)
Otherwise	with probability 0.7 Select move with minimum risk
Otherwise (*)	Randomly draw a complete state s from observation o and select a risk-less move after s
Forced moves [25]	Select a risk-less move (estimated by CSP)

move), i.e. the probabilistic transition model $p(o, a, o')$. Notably, with $p(o, a, o')$ the POMDP setting boils down to an MDP.

Finally, UC-CRP involves several UCT variants together with the above modules, depicted in Table 1. The variance-based UCB formula [1] is given as, where $n(s)$ is the number of visits to state s, $r(s, a)$ and $v(s, a)$ respectively are the empirical mean and variance of the reward gathered over episodes visiting (s, a), and $\alpha \geq 2$ is an integer value:

$$\text{Select argmax} \left\{ r(s,a) + \sqrt{\frac{2v(s,a)\log\left(4n(s)^{\alpha}\right)}{n(s)}} + \frac{16\log\left(4n(s)^{\alpha}\right)}{3n(s)} \right\}$$

The double progressive widening detailed in [8] is used to control the branching factor of the tree, i.e. the number of considered moves in each node.

Note that UC-CRP requires more than assessing the probability of mine in every location conditioned by the current observation; it requires the full probability distribution over the complete states conditioned by the current observation.

6 Consistency Analysis

This section, devoted to the analytical study of the considered algorithms, establishes the consistency of UC-CRP in the general case and the inconsistency of CSP-based MineSweeper approaches.

6.1 Consistency of UC-CRP

Let us first examine the consistency of UCT approaches in the MDP setting.

UCT (Upper Confidence Trees) is known to be a consistent MDP solver [14] in the finite case. While the presented approach uses heuristics in the evaluation of leaves by Monte-Carlo simulations, this does not affect the consistency proof in [14]. While heuristics are also used in the tree part of UCT, the upper-confidence-bound proof from [16,2], used in [14], is independent of heuristics too. A last additional component of UC-CRP compared to UCT [14] is progressive widening [8]. Progressive widening however has no asymptotic impact since the number of actions and the number of random outcomes is finite. The UCT-variant involved in UC-CRP thus is a consistent solver in the MDP case.

Indeed MineSweeper is not an MDP: as far as the complete state is unknown it is a POMDP. However, as shown by [18], the hidden information can be exactly estimated by CSP. Indeed the CSP-based move selection (select the move with lowest probability of immediate death) is not optimal as shown by [9]; nevertheless the estimation of the hidden information soundly enables to cast the POMDP MineSweeper problem into an MDP one.

As a consequence, UC-CRP is a consistent MDP solver, and it inherits from its CSP component the property of being an asymptotically optimal reactive planner.

6.2 Inconsistency of CSP-Based Approaches

Let us consider the 3×3 MineSweeper problem with 7 mines. As stated above, UC-CRP is asymptotically consistent (since UCT is an asymptotically consistent MDP solver and the CSP or rejection method provides an exact transition model). Therefore, UC-CRP finds the optimal strategy in the $(3, 3, 7)$ MineSweeper setting. Quite the contrary, the CSP cannot be optimal due to the uniform selection of the initial move.

The success rate can be analytically computed on this toy MineSweeper problem. If the initial move is the center location, this location necessarily has 7 neighbor mines. The probability of winning thus is $\frac{1}{8}$ as there are 7 mines in the 8 neighbors (Fig. 3).

If the initial move is located on a side of the board, then the number of mines in its neighborhood is: 5 with probability $\frac{3}{8}$ (with probability of winning $\frac{1}{3}$) and 4 with probability $\frac{5}{8}$ (with probability of winning $\frac{1}{5}$). Overall, the probability of winning when playing a side location as initial move thus is $\frac{1}{4} = \frac{3}{8} \times \frac{1}{3} + \frac{5}{8} \times \frac{1}{5}$.

Finally, if the initial move is located in a corner of the board, then the number of mines in its neighborhood is 2 with probability $\frac{3}{8}$ (with probability of winning $\frac{1}{3}$) and 3 with probability $\frac{5}{8}$ (with probability of winning $\frac{1}{5}$). The overall

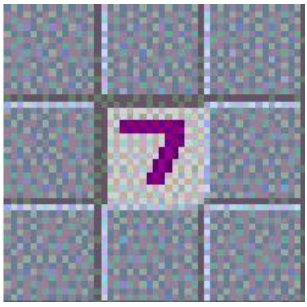

Fig. 3. A bad initial move in 3x3 with 7 mines

probability of winning when playing a corner location as initial move thus is $\frac{1}{4} = \frac{3}{8} \times \frac{1}{3} + \frac{5}{8} \times \frac{1}{5}$.

An optimal strategy thus selects a side or corner location as first move, yielding a probability $\frac{1}{4}$ of winning. Quite the contrary, CSP-based MineSweeper algorithms (rightly) considers that all moves have the same probability of immediate death and thus the first move is uniformly selected. The probability of winning thus is $\frac{1}{9} \times \frac{1}{8} + \frac{8}{9} \times \frac{1}{4} = \frac{17}{72}$, which is less than the probability $\frac{1}{4}$ of winning of an optimal strategy (and asymptotically reached by UC-CRP).

7 Experimental Validation

The goal of the experiments is to study both the asymptotic optimality and the comparative performances of UC-CRP. The optimality of UC-CRP is studied using the GnoMine Custom mode (the first move has no mine in its 8-neighborhood), as this setting facilitates the analytical study. The standard MineSweeper setting (where the first move is only assumed to be not a mine) is also considered for the fair comparison of UC-CRP with the state of the art CSP-PGMS algorithm.

All experiments are done on a 16-cores Xeon 2.93GHz Ubuntu 2.6-32-34. The computational effort is measured using a single core per experiment.

7.1 A Gnomine Custom Mode: 15 Mines on a 5x5 Board

Let us consider the 5x5 board with 15 mines (Fig. 4). Under the Gnomine custom mode, playing the center location implies that all mines are located on the sides, thus yielding a sure win. Interestingly, UC-CRP finds this optimal strategy (that humans easily find as well), while CSP-based approaches still uniformly select their first move. In such case, the probability of playing location $(2, 2)$ (up to rotational symmetry) is $\frac{4}{25}$, yielding a loss probability of $\frac{1}{2}$. Likewise, the probability of playing (2,3) (up to rotational symmetry) is $\frac{4}{25}$, yielding a loss probability of $\frac{1}{4}$. The overall probability of loosing the game is at least $\frac{1}{2} \times \frac{4}{25} + \frac{1}{4} \times \frac{4}{25} = \frac{3}{25}$ (indeed, the actual loss probability is bigger since the side

Table 2. Comparative performances of UC-CRP and CSP-PGMS

Format	CSP-PGMS	UC-CRP
4 mines on 4x4	64.7 %	**70.0% ± 0.6%**
1 mine on 1x3	100 %	100% (2000 games)
3 mines on 2x5	22.6%	**25.4 % ± 1.0%**
10 mines on 5x5	8.20%	9% (p-value: 0.14)
5 mines on 1x10	12.93%	**18.9% ± 0.2%**
10 mines on 3x7	4.50%	**5.96% ± 0.16%**
15 mines on 5x5	0.63%	**0.9% ± 0.1%**

locations are even worse initial moves), showing that CSP approaches do not find the optimal strategy as opposed to UC-CRP.

Experimentally, UC-CRP (with no expert knowledge besides the gnomine rule) finds the best move in all out of 500 independent runs. The computational cost is 5 seconds on a 16-cores Xeon 2.93GHz Ubuntu 2.6-32-34 (using 1 core only).

Fig. 4. The Gnomine version of the 5x5 board with 15 mines is a sure win: in each case the position of all mines can be deduced as there exists only one non-mine location. The three reported cases cover all possible cases by rotational symmetry.

7.2 Standard MineSweeper Setting

UC-CRP is compared to a CSP-based MineSweeper player[3], selecting corner locations as initial moves.

UC-CRP is allotted a computational budget of 10s per move, except for 10 mines in 5x5 (300 seconds per move) and 10 mines in 3x7 (30s per move). The average winning rate is reported together with the standard deviation on Table 2 for several board sizes. The winning rate of CSP-PGMS is estimated on 100000 games. Note that, while CSP-PGMCS is significantly faster than UC-CRP, its performances do not increase with additional computational time. Overall, UC-CRP outperforms CSP-PGMCS in all cases with a p-value .05, except for the 5×5 with 10 mines MineSweeper, where the p-value is .14.

[3] The CSP-PGMS implementation from `http://www.ccs.neu.edu/home/ramsdell/pgms/` is used.

8 Conclusion and Perspectives

This paper has investigated the tight coupling of Upper Confidence Trees and Constraint Solving to tackle partially observable Markov decision making problems, defining the UC-CRP algorithm. The role of the CSP component in UC-CRP is twofold. On the one hand, it consistently estimates the belief state, turning a POMDP setting into an MDP one; on the other hand, the belief state estimate is exploited to sort out risk-less or heuristically good moves.

The first contribution of the paper is a generic methodology for improving a myopic solver, through its combination with UCT. Our claim is that such approaches, combining consistent asymptotic behaviors and myopic efficient heuristics, will provide artificial intelligences with the best of both worlds: relevant heuristics are used to enforce computational efficiency and yield tractability guarantees; UCT provides asymptotic optimality guarantees, enabling to boost the available heuristics to optimality through computational efforts. Improving a decent heuristic policy to reach optimality with UCT seems to be a simple though general and potentially very efficient strategy.

The second contribution of the paper is to formally establish the consistency of the proposed UC-CRP approach. The third contribution is the empirical validation of the approach, using the difficult MineSweeper problem as proof of concept. It has been emphasized that MineSweeper is a particularly challenging problem. Indeed the CSP approach, which ignores long term effects, is very effective as uncertainties are so big that long-term effects are very unreliable. On this difficult problem, significant improvements on small boards have been obtained compared to [9]. Further, UC-CRP results improve as the computational budget increases while CSP does not benefit from additional computational resources due to its intrinsic myopic limitations.

A key and very promising feature of UC-CRP is that no (manually acquired or programmed) opening book needed be considered; quite the contrary, the good performances of advanced CSP approaches requires specific heuristics to handle the first move selection. Along the same line, UC-CRP flexibly accommodates MineSweeper variants through modifying the only rule module (transition model).

A research perspective aimed at the game community is concerned with a faster implementation of UC-CRP; at the moment UC-CRP relies on a much slower CSP module than e.g. CSP-PGMS. Further assessment of UC-CRP, e.g. comparatively to [19] will be facilitated by using a CSP implementation as fast as CSP-PGMS, enabling to consider expert MineSweeper modes.

Acknowledgements. The authors acknowledge the support of NSC (funding NSC100-2811-E-024-001) and ANR (project EXPLO-RA ANR-08-COSI-004). We thank Laurent Simon, LRI Université Paris-Sud, for fruitful discussions, and we thank the authors of PGMS and PGMS-CSP for making their implementations freely available.

References

1. Audibert, J.-Y., Munos, R., Szepesvari, C.: Use of variance estimation in the multi-armed bandit problem. In: NIPS 2006 Workshop on On-line Trading of Exploration and Exploitation (2006)
2. Auer, P.: Using confidence bounds for exploitation-exploration trade-offs. The Journal of Machine Learning Research 3, 397–422 (2003)
3. Auer, P., Cesa-Bianchi, N., Fischer, P.: Finite-time analysis of the multiarmed bandit problem. Machine Learning 47(2/3), 235–256 (2002)
4. Becker, K.: Teaching with games: the minesweeper and asteroids experience. J. Comput. Small Coll. 17, 23–33 (2001)
5. Ben-Ari, M.M.: Minesweeper as an NP-complete problem. SIGCSE Bull. 37, 39–40 (2005)
6. Bertsekas, D., Tsitsiklis, J.: Neuro-dynamic Programming. Athena Scientific (1996)
7. Castillo, L.P.: Learning minesweeper with multirelational learning. In: Proc. of the 18th Int. Joint Conf. on Artificial Intelligence, pp. 533–538 (2003)
8. Couëtoux, A., Hoock, J.-B., Sokolovska, N., Teytaud, O., Bonnard, N.: Continuous Upper Confidence Trees. In: Coello, C.A.C. (ed.) LION 5. LNCS, vol. 6683, pp. 433–445. Springer, Heidelberg (2011)
9. Couetoux, A., Milone, M., Teytaud, O.: Consistent belief state estimation, with application to mines. In: Proc. of the TAAI 2011 Conference (2011)
10. Coulom, R.: Efficient Selectivity and Backup Operators in Monte-Carlo Tree Search. In: Ciancarini, P., van den Herik, H.J. (eds.) Proc. of the 5th Int. Conf. on Computers and Games, pp. 72–83 (2006)
11. Gordon, M., Gordon, G.: Quantum computer games: quantum Minesweeper. Physics Education 45(4), 372 (2010)
12. Hein, K.B., Weiss, R.: Minesweeper for sensor networks–making event detection in sensor networks dependable. In: Proc. of the 2009 Int. Conf. on Computational Science and Engineering, CSE 2009, vol. 01, pp. 388–393. IEEE Computer Society (2009)
13. Kaye, R.: Minesweeper is NP-complete. Mathematical Intelligencer 22, 9–15 (2000)
14. Kocsis, L., Szepesvári, C.: Bandit Based Monte-Carlo Planning. In: Fürnkranz, J., Scheffer, T., Spiliopoulou, M. (eds.) ECML 2006. LNCS (LNAI), vol. 4212, pp. 282–293. Springer, Heidelberg (2006)
15. Koza, J.R.: Genetic Programming II: Automatic Discovery of Reusable Programs. MIT Press (1994)
16. Lai, T., Robbins, H.: Asymptotically efficient adaptive allocation rules. Advances in Applied Mathematics 6, 4–22 (1985)
17. Lee, C.-S., Wang, M.-H., Chaslot, G., Hoock, J.-B., Rimmel, A., Teytaud, O., Tsai, S.-R., Hsu, S.-C., Hong, T.-P.: The Computational Intelligence of MoGo Revealed in Taiwan's Computer Go Tournaments. IEEE Transactions on Computational Intelligence and AI in Games (2009)
18. Nakov, P., Wei, Z.: Minesweeper, #minesweeper (2003)
19. Pedersen, K.: The complexity of Minesweeper and strategies for game playing. Project report, univ. Warwick (2004)
20. ROADEF-Challenge. A large-scale energy management problem with varied constraints (2010), `http://challenge.roadef.org/2010/`

21. Rolet, P., Sebag, M., Teytaud, O.: Boosting Active Learning to Optimality: A Tractable Monte-Carlo, Billiard-Based Algorithm. In: Buntine, W., Grobelnik, M., Mladenić, D., Shawe-Taylor, J. (eds.) ECML PKDD 2009, Part II. LNCS, vol. 5782, pp. 302–317. Springer, Heidelberg (2009)
22. Rolet, P., Sebag, M., Teytaud, O.: Optimal robust expensive optimization is tractable. In: GECCO 2009, Montréal Canada, 8 p. ACM Press (2009)
23. Studholme, C.: Minesweeper as a constraint satisfaction problem. Unpublished project report (2000)
24. Sutton, R., Barto, A.G.: Reinforcement learning. MIT Press (1998)
25. Teytaud, F., Teytaud, O.: On the Huge Benefit of Decisive Moves in Monte-Carlo Tree Search Algorithms. In: IEEE Conf. on Computational Intelligence and Games, Copenhagen, Denmark (2010)