Hypervolume indicator and dominance reward based multi-objective Monte-Carlo Tree Search

Weijia Wang, Michèle Sebag

JMLR: Workshop and Conference Proceedings 1–30

Hypervolume indicator and Dominance reward based Multi-objective Monte-Carlo Tree Search

Weijia Wang WEIJIA.WANG@LRI.FR
Michèle Sebag MICHELE.SEBAG@LRI.FR
LRI, CNRS UMR 8623 & INRIA-Saclay, Université Paris-Sud, 91405 Orsay, Cedex FRANCE

Abstract

Concerned with multi-objective reinforcement learning (MORL), this paper presents MOMCTS, an extension of Monte-Carlo Tree Search to multi-objective sequential decision making, embedding two decision rules respectively based on the hypervolume indicator and the Pareto dominance reward. The MOMCTS approaches are firstly compared with the MORL state of the art on two artificial problems, the two-objective Deep Sea Treasure problem and the three-objective Resource Gathering problem. The scalability of MOMCTS is also examined in the context of the NP-hard grid scheduling problem, showing that the MOMCTS performance matches the (non-RL based) state of the art albeit with a higher computational cost.

Keywords: reinforcement learning, Monte-Carlo tree search, multi-objective optimization, sequential decision making

1. Introduction

Reinforcement learning (RL) (Sutton and Barto, 1998; Szepesvári, 2010) addresses sequential decision making in the Markov decision process framework. RL algorithms provide guarantees of finding the optimal policies in the sense of the expected cumulative reward, relying on the thorough exploration of the state and action spaces. The price to pay for these optimality guarantees is the limited scalability of mainstream RL algorithms w.r.t. the size of the state and action spaces.

Recently, Monte-Carlo Tree Search (MCTS), including the famed Upper Confidence Tree algorithm (Kocsis and Szepesvári, 2006) and its variants, has been intensively investigated to handle sequential decision problems. MCTS, notably illustrated in the domain of Computer-Go (Gelly and Silver, 2007), has been shown to efficiently handle medium-size state and action search spaces through a careful balance between the exploration of the search space, and the exploitation of the best results found so far. While providing some consistency guarantees (Berthier et al., 2010), MCTS has demonstrated its merits and wide applicability in the domain of games (Ciancarini and Favini, 2009) or planning (Nakhost and Müller, 2009) among many others.

This paper is motivated by the fact that many real-world applications, including reinforcement learning problems, are most naturally formulated in terms of multi-objective optimization (MOO). In multi-objective reinforcement learning (MORL), the reward associated to a given state is d-dimensional (e.g. cost, risk, robustness) instead of a single scalar value (e.g. quality). To our knowledge, MORL was first tackled by Gábor et al. (1998); introducing a lexicographic (hence total) order on the policy space, the authors show the convergence of standard RL algorithms under the total order assumption. In

practice, multi-objective reinforcement learning is often tackled by applying standard RL algorithms on a scalar aggregation of the objective values (e.g. optimizing their weighted sum; see also Mannor and Shimkin (2004); Tesauro et al. (2007)).

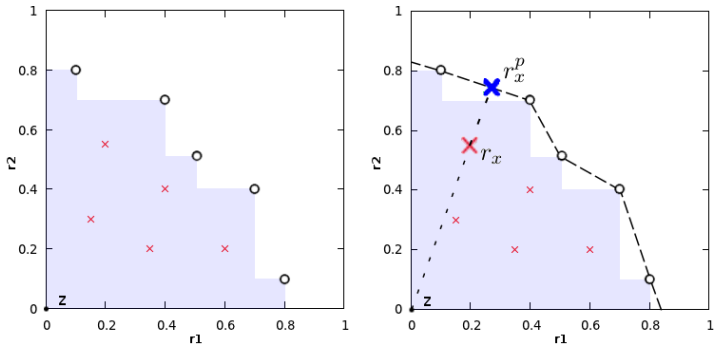
In the general case of antagonistic objectives however (e.g. simultaneously minimize the cost and the risk of a manufacturing process), two policies might be incomparable (e.g. the cheapest process for a fixed robustness; the most robust process for a fixed cost): solutions are partially ordered, and the set of optimal solutions according to this partial order is referred to as Pareto front (more in section 2). The goal of the so-called multiple-policy MORL algorithms (Vamplew et al., 2010) is to find several policies on the Pareto front (Natarajan and Tadepalli, 2005; Chatterjee, 2007; Barrett and Narayanan, 2008; Lizotte et al., 2012).

The goal of this paper is to extend MCTS to multi-objective sequential decision making. The proposed scheme called MOMCTS basically aims at discovering several Pareto-optimal policies (decision sequences, or *solutions*) within a single tree. MOMCTS requires one to modify the exploration of the tree to account for the lack of total order among the nodes, and the fact that the desired result is a set of Pareto-optimal solutions (as opposed to, a single optimal one). A first possibility considers the use of the hypervolume indicator (Zitzler and Thiele, 1998), which measures the the MOO quality of a solution w.r.t. the current Pareto front. Specifically, taking inspiration from (Auger et al., 2009), this indicator is used to define a single optimization objective for the current path being visited in each MCTS tree-walk, *conditioned on the other solutions* previously discovered. MOMCTS thus handles a single-objective optimization problem in each tree-walk, while eventually discovering several decision sequences pertaining to the Pareto-front. This approach, first proposed by Wang and Sebag (2012), suffers from two limitations. On the one hand, the hypervolume indicator computation cost increases exponentially with the number of objectives. Secondly, the hypervolume indicator is not invariant under the monotonous transformation of the objectives. The invariance property (satisfied for instance by comparison-based optimization algorithms) gives robustness guarantees which are most important w.r.t. ill-conditioned optimization problems (Hansen, 2006).

Addressing these limitations, a new MOMCTS approach is proposed in this paper, using Pareto dominance to compute the instant reward of the current path visited by MCTS. Compared to the first approach − referred to as MOMCTS-hv in the remainder of this paper, the latter approach − referred to as MOMCTS-dom− has linear computational complexity w.r.t. the number of objectives, and is invariant w.r.t. the monotonous transformation of the objectives.

Both MOMCTS approaches are empirically assessed and compared to the state of the art on three benchmark problems. Firstly, both MOMCTS variants are applied on two artificial benchmark problems, using MOQL (Vamplew et al., 2010) as baseline: the two-objective Deep Sea Treasure (DST) problem (Vamplew et al., 2010) and the three-objective Resource Gathering (RG) problem (Barrett and Narayanan, 2008). A stochastic transition model is considered for both DST (originally deterministic) and RG, to assess the robustness of both MOMCTS approaches. Secondly, the real world NP-hard problem of grid scheduling Yu et al. (2008) is considered to assess the performance and scalability of MOMCTS methods comparatively to the (non-RL-based) state of the art.

The paper is organized as follows. Section 2 briefly introduces the formal background. Section 3 describes the MOMCTS-hv and the MOMCTS-dom algorithm. Section 4 presents the experimental validation of MOMCTS approaches. Section 5 discusses the strengths and limitations of MOMCTS approaches w.r.t. the state of the art and the paper concludes with some research perspectives.



(a) (b)

Figure 1: The left figure shows vectorial rewards in the two-dimensional objective plane. The non-dominated vectorial rewards are depicted as black circles. The hypervolume indicator of these solutions w.r.t reference point z in the lower-left corner is the surface of the shaded region. The right figure shows the perspective projection r_x^p of r_x on the piecewise linear envelope of the Pareto front (section 3.1.1).

2. Formal background

Assuming the reader's familiarity with the reinforcement learning setting (Sutton and Barto, 1998), this section briefly introduces the main notations and definitions used in the rest of the paper.

A Markov decision process (MDP) is described by its state and action space respectively denoted $\mathcal{S}$ and $\mathcal{A}$. The transition function ($p : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \mapsto [0, 1]$) gives the probability $p(s, a, s')$ of reaching state s' by executing action a in state s. The (scalar) reward function is defined from the state $\times$ action space onto $\mathbb{R}$ ($r : \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}$).

2.1. Multi-objective optimization

In multi-objective optimization (MOO), each point x in the search space $\mathcal{X}$ is associated with a d-dimensional reward vector r_x in $\mathbb{R}^d$, referred to as vectorial reward in the following. With no loss of generality, it is assumed that each objective is to be maximized.
Given two points $x, x' \in \mathcal{X}$ with $r_x = (r_1, \ldots, r_d)$ and $r_{x'} = (r'_1, \ldots, r'_d)$ their associated vectorial rewards, r_x is said to dominate, or Pareto-dominate, $r_{x'}$ (noted $r_x \succeq r_{x'}$) iff r_i is greater than or equal to r'_i for $i = 1 \ldots d$. The dominance is strict (noted $r_x \succ r_{x'}$) if $r_x \succeq r_{x'}$ and $r_i > r'_i$ for some i (Fig. 1(a)). As mentioned, Pareto-dominance defines a partial order relation on $\mathbb{R}^d$ and thus on $\mathcal{X}$. The Pareto front is defined as follows:

Definition 1 *Given $A \subset \mathbb{R}^d$ a set of vectorial rewards, the set P_A of non-dominated points in A is defined as:*

$$P_A = \{r \in A : \nexists r' \in A \ s.t. \ r' \succ r\}$$

The Pareto front is made of all non-dominated vectorial rewards. By abuse of language, P_A is referred to as the set of Pareto-optima in A.

Two different categories of MOO problems are distinguished depending on whether they correspond to a convex or non-convex Pareto front. The convex Pareto front can be identified by solving a set of single objective optimization problems defined on the weighted

sum of the objectives, referred to as linear scalarization of the MOO problem (as done in MOQL, section 4.2.1). When dealing with non-convex Pareto fronts (for instance, the DST problem (Vamplew et al., 2010), and the ZDT2 and DTLZ2 test benchmarks (Deb et al., 2002)) however, the linear scalarization approach fails to discover the non-convex parts of the Pareto front (Deb, 2001). Although many MOO problems have a convex Pareto front, especially in the two-objective case, the discovery of the non-convex Pareto front remains the main challenge for MOO approaches (Deb et al., 2000; Beume et al., 2007)[1].

2.2. Monte-Carlo Tree Search

Let us describe the best known MCTS algorithm, referred to as Upper Confidence Tree (UCT) (Kocsis and Szepesvári, 2006) and extending the Upper Confidence Bound algorithm (Auer et al., 2002) to tree-structured spaces. UCT simultaneously explores and builds a search tree, initially restricted to its root node, along N tree-walks a.k.a. simulations. Each tree-walk involves three phases:

The **bandit phase** starts from the root node and iteratively selects an action/a child node until arriving in a leaf node. Action selection is handled as a multi-armed bandit problem. The set $\mathcal{A}_s$ of admissible actions a defines the possible child nodes (s,a) of node s; the selected action a^* maximizes the Upper Confidence Bound:

$$\hat{r}_{s,a} + \sqrt{c_e \ln(n_s)/n_{s,a}} \tag{1}$$

over a ranging in $\mathcal{A}_s$, where n_s stands for the number of times node s has been visited, $n_{s,a}$ denotes the number of times a has been selected in node s, and $\hat{r}_{s,a}$ is the average reward collected when selecting action a from node s. The first (respectively the second) term in Eq. (1) corresponds to the exploitation (resp. exploration) term, and the exploration vs exploitation trade-off is controlled by parameter c_e. Upon the selection of a^*, the next state is drawn from the transition model depending on the current state and a^*. In the remainder of the paper, a tree node is labeled with the sequence of actions followed from the root; the associated reward is the average reward collected over all tree-walks involving this node.

The **tree building phase** takes place upon arriving in a leaf node s; some action a is (uniformly or heuristically) selected and (s,a) is added as child node of s. Accordingly, the number of nodes in the tree is the number of tree-walks.

The **random phase** starts from the new leaf node (s,a) and iteratively (uniformly or heuristically) selects an action until arriving in a terminal state u; at this point the reward r_u of the whole tree-walk is computed and used to update the reward estimates in all nodes (s,a) visited during the tree-walk:

$$\begin{aligned} \hat{r}_{s,a} &\leftarrow \frac{1}{n_{s,a}+1}\left(n_{s,a} \times \hat{r}_{s,a} + r_u\right) \\ n_{s,a} &\leftarrow n_{s,a}+1; \qquad n_s \leftarrow n_s + 1 \end{aligned} \tag{2}$$

Additional heuristics have been considered, chiefly to prevent over-exploration when the number of admissible arms is large w.r.t the number of simulations (the so-called many-armed bandit issue (Wang et al., 2008)). The Progressive Widening (PW) heuristics (Coulom, 2006) will be used in the following, where the allowed number of child nodes of s is initialized to 1 and increases with its number of visits n_s like $\lfloor {n_s}^{1/b} \rfloor$ (with b usually set to 2 or 4). The Rapid Action Value Estimation (RAVE) heuristic is meant to guide the exploration of the search space (Gelly and Silver, 2007). In its simplest version, $RAVE(a)$

1. Notably, the chances for a Pareto front to be convex decreases with the number of objectives.

is set to the average reward taken over all tree-walks involving action a. The RAVE vector can be used to guide the tree-building phase[2], that is, when selecting a first child node upon arriving in a leaf node s, or when the Progressive Widening heuristics is triggered and a new child node is added to the current node s. In both cases, the selected action is the one maximizing $RAVE(a)$. The RAVE heuristic aims at exploring earlier the most promising regions of the search space; for the sake of convergence speed, it is clearly desirable to consider the best options as early as possible.

3. Overview of MOMCTS

The main difference between MCTS and MOMCTS regards the node selection step. The challenge is to extend the single-objective node selection criterion (Eq. (1)) to the multi-objective setting. Since there is no total order between points in the multi-dimensional space, as mentioned, the most straightforward way of dealing with multi-objective optimization is to get back to single-objective optimization, through aggregating the objectives into a single one; the price to pay is that this approach yields a single solution on the Pareto front. Two aggregating functions (the hypervolume indicator and the cumulative discounted dominance reward) aimed at recovering a total order among points in the multi-dimensional reward space conditionally to the search archive, will be integrated within the MCTS framework.

The MOMCTS-hv algorithm is presented in section 3.1 and its limitations are discussed in section 3.2. The MOMCTS-dom algorithm aimed at overcoming these limitations is introduced in section 3.3.

3.1. MOMCTS-hv

3.1.1. Node selection based on hypervolume indicator

The hypervolume indicator (Zitzler and Thiele, 1998) provides a scalar measure of solution sets in the multi-objective space as follows.

Definition 2 *Given $A \subset \mathbb{R}^d$ a set of vectorial rewards, given reference point $z \in \mathbb{R}^d$ such that it is dominated by every $r \in A$, then the hypervolume indicator (HV) of A is the measure of the set of points dominated by some point in A and dominating z:*

$$HV(A; z) = \mu(\{x \in \mathbb{R}^d : \exists r \in A \ s.t. \ r \succeq x \succeq z\})$$

where μ is the Lebesgue measure on $\mathbb{R}^d$ (Fig.1(a)).

It is clear that all dominated points in A can be removed without modifying the hypervolume indicator ($HV(A; z) = HV(P_A; z)$). As shown by Fleischer (2003), the hypervolume indicator is maximized iff points in P_A belong to the Pareto front of the MOO problem. Auger et al. (2009) show that, for $d = 2$, for a number K of points, the hypervolume indicator maps a multi-objective optimization problem defined on $\mathbb{R}^d$, onto a single-objective optimization problem on $\mathbb{R}^{d \times K}$, in the sense that there exists at least one set of K points in $\mathbb{R}^d$ that maximizes the hypervolume indicator w.r.t. z.

Let P denote the archive of non-dominated vectorial rewards measured for every terminal state u (section 2.2). It then comes naturally to define the value of any MCTS tree node as follows.

2. Another option is to use a dynamically weighted combination of the reward $\hat{r}_{s,a}$ and $RAVE(a)$ in Eq. (1).

Let us associate to each node (s,a) in the tree the vector $\overline{r}_{s,a}$ of the upper confidence bounds on its rewards:

$$\overline{r}_{s,a} = \left(\hat{r}_{s,a;\ i} + \sqrt{c_i \ln(n_s)/n_{s,a}} \right)_{i=1}^{d} \tag{3}$$

with c_i the exploration vs exploitation parameter for the i-th objective (Eq. (1)).

An upper-bound $V(s,a)$ on the value of (s,a) is given by considering the hypervolume indicator of $\overline{r}_{s,a}$ w.r.t. archive P.

$$V(s,a) = HV(P \cup \{\overline{r}_{s,a}\}; z)$$

While $V(s,a)$ does provide a scalar value of a node (s,a) conditioned on the solutions previously evaluated, it takes on a constant value if $\overline{r}_{s,a}$ is dominated by some vectorial reward in P. In order to differentiate these dominated points, we consider the perspective projection $\overline{r}^p_{s,a}$ of $\overline{r}_{s,a}$ onto $\mathcal{P}$, the piecewise linear surface in $\mathbb{R}^d$ including all $r_u \in P$ (Fig. 1(*b*)). Let $\overline{r}^p_{s,a}$ denote the (unique) intersection of line $(\overline{r}_{s,a}, z)$ with $\mathcal{P}$ (being reminded that z is dominated by all points in P and by $\overline{r}_{s,a}$). The value function associated to (s,a) is then defined as the value of $\overline{r}_{s,a}$, minus the Euclidean distance between $\overline{r}_{s,a}$ and $\overline{r}^p_{s,a}$. Finally, the value of (s,a) is defined as:

$$W(s,a) = \begin{cases} V(s,a) & \text{if } \overline{r}_{s,a} \text{ is non-dominated in } P \\ V(s,a) - \| \overline{r}^p_{s,a} - \overline{r}_{s,a} \|_2 & \text{otherwise} \end{cases} \tag{4}$$

The Euclidean distance term here sets a penalty for dominated points, increasing with their distance to the linear envelope $\mathcal{P}$ of P. Note that Eq. (4) sets a total order on all vectorial rewards in $\mathbb{R}^d$, where non-dominated points are ranked higher than dominated ones.

3.1.2. MOMCTS-hv algorithm

MOMCTS-hv differs from MCTS in only three respects (Algorithm 1). Firstly, the selected action a^* is the one maximizing value function $W(s,a)$ instead of the UCB criterion (Eq. (1)). Secondly, MOMCTS-hv maintains the archive P of all non-dominated vectorial rewards evaluated in previous tree-walks. Upon arriving in a terminal state u, MOMCTS-hv evaluates the vectorial reward r_u of the tree-walk. It then updates $\hat{r}_{s,a}$ for all nodes (s,a) visited during the tree-walk, and it updates archive P if r_u is non-dominated. Thirdly, the RAVE vector (section 2.2) is used to select new nodes in the tree-building phase. Letting $RAVE(a)$ denote the average vectorial reward associated to a, letting $RAVE^p(a)$ denote the perspective projection of $RAVE(a)$ on the approximated Pareto front $\mathcal{P}$, then the action selected is the one minimizing

$$\| RAVE^p(a) - RAVE(a) \|_2 \tag{5}$$

MOMCTS-hv parameters include i) the total number of tree-walks N, ii) the b parameter used in the progressive widening heuristic (section 2.2); iii) the exploration vs exploitation trade-off parameter c_i for every i-th objective; and iv) the reference point z.

3.2. Discussion

Let B denote the average branching factor in the MOMCTS-hv tree, and let N denote the number of tree-walks. As each tree-walk adds a new node, the number of nodes in the tree is $N+1$ by construction. The average length of a tree-path thus is in $\mathcal{O}(\log N)$. Depending on the number d of objectives, the hypervolume indicator is computed with complexity

Algorithm 1 MOMCTS-hv

MOMCTS-hv
Input: number N of tree-walks
Output: search tree $\mathcal{T}$
Initialize $\mathcal{T} \leftarrow$ root node (initial state), $P \leftarrow \{\}$
for $t = 1$ **to** N **do**
  TreeWalk($\mathcal{T}, P,$ root node)
end for
return $\mathcal{T}$

TreeWalk
Input: search tree $\mathcal{T}$, archive P, node s
Output: vectorial reward r_u
if s is not a leaf node, and $\neg(\lfloor (n_s+1)^{1/b} \rfloor > \lfloor (n_s)^{1/b} \rfloor)$ // (PW test is not triggered)
then
  Select $a^* = \text{argmax}\ \{W(s,a), (s,a) \in \mathcal{T}\}$ //Eq. (4)
  $r_u \leftarrow \text{TreeWalk}(\mathcal{T}, P, (s, a^*))$
else
  $\mathcal{A}_s = \{$ admissible actions not yet visited in $s\}$
  Select $a^* = \arg\min\{\parallel RAVE^p(a) - RAVE(a) \parallel_2, a \in \mathcal{A}_s\}$
  Add (s, a^*) as child node of s
  $r_u \leftarrow \text{RandomWalk}(P, (s, a^*))$
end if
Update n_s, n_{s,a^*}, $RAVE(a^*)$ and $\hat{r}_{s,a}$
return r_u

RandomWalk
Input: archive P, state u
Output: vectorial reward r_u
$\mathcal{A}_{rnd} \leftarrow \{\}$ // store the set of actions visited in the random phase
while u is not final state **do**
  Uniformly select an admissible action a for u
  $\mathcal{A}_{rnd} \leftarrow \mathcal{A}_{rnd} \cup \{a\}$
  $u \leftarrow (u, a)$
end while
$r_u = evaluate(u)$ //obtain the vectorial reward of the tree-walk
if r_u is not dominated by any point in P **then**
  Prune all points dominated by r_u in P
  $P \leftarrow P \cup \{r_u\}$
end if
Update $RAVE(a)$ for $a \in \mathcal{A}_{rnd}$
return r_u

$\mathcal{O}(|P|^{d/2})$ for $d > 3$ (respectively $\mathcal{O}(|P|)$ for $d = 2$ and $\mathcal{O}(|P|\log|P|)$ for $d = 3$) (Beume et al., 2009). The complexity of each tree-walk thus is $\mathcal{O}(B|P|^{d/2}logN)$, where $|P|$ is at most the number N of tree-walks.

By construction, the hypervolume indicator based selection criterion (Eq. (4)) drives MOMCTS-hv towards the Pareto front and favours the diversity of the Pareto archive. On the negative side however, the computational cost of $W(s,a)$ is exponential with the number d of objectives. Besides, the hypervolume indicator is not invariant under monotonous transformation of objective functions, which prevents the approach from enjoying the same robustness as comparison-based optimization approaches (Hansen, 2006). Lastly, the MOMCTS-hv critically depends on its hyper-parameters. The exploration vs exploitation (EvE) trade-off parameters $c_i, i = 1, 2, \ldots, d$ (Eq. (1)) of each objective have a significant impact on the performance of MOMCTS-hv (likewise, the MCTS applicative results depend on the tuning of the EvE trade-off parameters (Chaslot et al., 2008)). Additionally, the choice of the reference point z also influences the hypervolume indicator values (Auger et al., 2009)).

3.3. MOMCTS-dom

This section presents a new MOMCTS approach aimed at overcoming the above limitations, which is based on the Pareto dominance test. Notably, this test has linear complexity w.r.t. the number of objectives, and is invariant under monotonous transformation of objectives. As this reward depends on the Pareto archive which evolves along the search, the cumulative discounted dominance(CDD) reward mechanism is proposed to handle the search dynamics.

3.3.1. Node selection based on cumulative discounted dominance reward

Let P denote the archive of all non-dominated vectorial rewards previously gathered during the search process. A straightforward option would be to associate to each tree-walk reward 1 if the tree-walk gets a vectorial reward r_u which is not strictly dominated by any point in the archive P, and reward 0 otherwise. Formally this boolean dominance reward, called $r_{u;dom}$, is defined as:

$$r_{u;dom} = \begin{cases} 1 & \text{if } \nexists r \in P, r \succ r_u \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

The optimization problem defined by dominance rewards is non-stationary as it depends on the archive P, which evolves along time. To cope with non-stationarity, the reward update proceeds along a cumulative discounted (CD) process as follows. Let $t_{s,a}$ denote the index of the last tree-walk which visited node (s,a), let $\Delta t = t - t_{s,a}$ where t is the index of the current tree-walk, let $\delta \in [0,1]$ be a discount factor, the CD update is defined as:

$$\begin{array}{c} \hat{r}_{s,a;dom} \leftarrow \hat{r}_{s,a;dom} \cdot \delta^{\Delta t} + r_{u;dom}, \ \delta \in [0,1] \\ t_{s,a} \leftarrow t; \quad n_{s,a} \leftarrow n_{s,a} + 1; \quad n_s \leftarrow n_s + 1 \end{array} \tag{7}$$

The reward update in MOMCTS-dom differs from the standard scheme (Eq. (2)) in two respects. Firstly, cumulative instead of average rewards are considered. The rationale for this modification is that a tiny percentage of the tree-walks finds a non-dominated vectorial reward if ever. In such cases, average rewards come to be negligible in front of the exploration term, making the MCTS degenerate to pure random search. The use of cumulative rewards instead tends to prevent this degradation.
Secondly, a discount mechanism is used to moderate the cumulative effects using the discount factor δ ($0 \leq \delta \leq 1$) and taking into account the number Δt of tree-walks since this node was last visited. This discount mechanism is meant to cope with the dynamics of

multi-objective search through forgetting old rewards, thus enabling the decision rule to reflect up-to-date information.
Indeed, the CD process is reminiscent of the discounted cumulative reward defining the value function in Reinforcement Learning (Sutton and Barto, 1998), with the difference that the time-step t here corresponds to the tree-walk index, and that the discount mechanism is meant to limit the impact of past (as opposed to, future) information.

In a stationary context, $\hat{r}_{s,a;dom}$ would converge towards $\frac{1}{1-\delta^{\Delta t}}\bar{r}$, with Δt the average interval of time between two visits to the node. If the node gets exponentially rarely visited, $\hat{r}_{s,a;dom}$ goes to $\bar{r}$. Quite the contrary, if the node happens to be frequently visited, $\bar{r}$ is multiplied by a large factor ($\frac{1}{1-\delta}$), entailing the over-exploitation of the node. However, the over-exploitation is bound to decrease as soon as the Pareto archive moves towards the true Pareto front. While this CDD reward was found to be empirically suited to the MOO setting (see also Maes et al. (2011)), further work is required to analyze its properties.

3.3.2. MOMCTS-dom algorithm

MOMCTS-dom proceeds as standard MCTS except for the update procedure, where Eq. (2) is replaced by Eq. (7). Keeping the same notations B, N and $|P|$ as above, as the dominance test in the end of each tree-walk is linear ($\mathcal{O}(d|P|)$), the complexity of each tree-walk in MOMCTS-dom is $\mathcal{O}(B \log N + d|P|)$, linear w.r.t. the number d of objectives.

Besides the MCTS parameters N and b, MOMCTS-dom involves two additional hyper-parameters: i) the exploration vs exploitation trade-off parameter c_e; and ii) the discount factor δ.

4. Experimental validation

This section presents the experimental validation of the MOMCTS-hv and MOMCTS-dom algorithms.

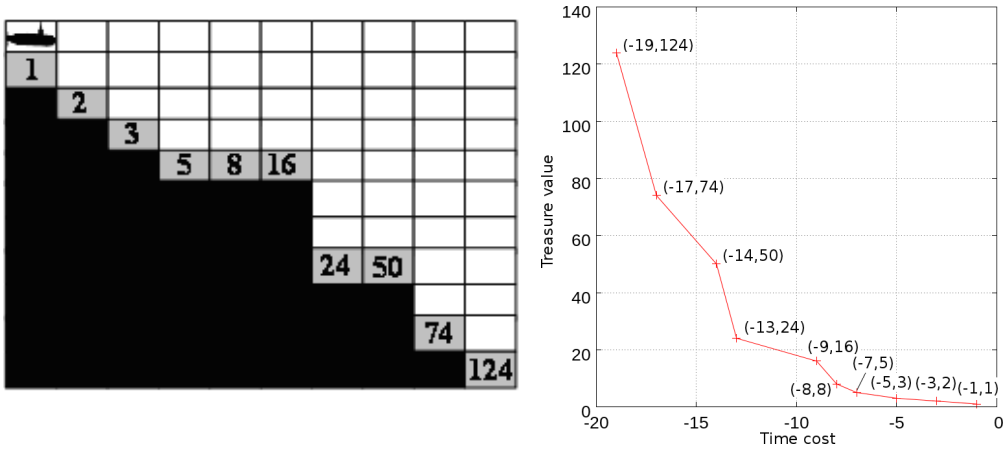
4.1. Goals of experiments

The first goal is to assess the performance of the MOMCTS approaches comparatively to the state of the art in MORL (Vamplew et al., 2010). Two artificial benchmark problems (Deep Sea Treasure and Resource Gathering) with probabilistic transition functions are considered. The Deep Sea Treasure problem has two objectives which define a non-convex Pareto front (section 4.2). The Resource gathering problem has three objectives and a convex Pareto front (section 4.3). The second goal is to assess the performance and scalability of MOMCTS approaches in a real-world setting, that of grid scheduling problems (section 4.4).

All reported results are averaged over 11 runs unless stated otherwise.

Indicators of performance

Two indicators are defined to measure the quality of solution sets in the multi-dimensional space. The first indicator is the hypervolume indicator (section 3.1.1). The second indicator, inspired from the notion of regret, is defined as follows. Let P^* denote the true Pareto front. The empirical Pareto front P defined by a search process is assessed from its generational distance (Van Veldhuizen, 1999) and inverted generational distance w.r.t. P^*. The generational distance (GD) is defined by $GD(P) = \left(\sqrt{\sum_{i=1}^{n} d_i^2}\right)/n$, where n is the size of P and d_i is the Euclidean distance between the i-th point in P and its nearest point in P^*. GD measures the average distance from points in P to the Pareto front. The



(*a*) The state space (*b*) The Pareto front

Figure 2: The Deep Sea Treasure problem. Left: the DST state space with black cells as sea-floor, gray cells as terminal states, the treasure value is indicated in each cell. The initial position is the upper left cell. Right: the Pareto front in the time×treasure plane.

inverted generational distance (IGD) is likewise defined as the average distance of points in P^* to their nearest neighbour in P. For both generational and inverted generational distances, the smaller, the better.

The algorithms are also assessed w.r.t. their computational cost (measured on a PC with Intel dual-core CPU 2.66GHz).

4.2. Deep Sea Treasure

The Deep Sea Treasure (DST) problem, first introduced by Vamplew et al. (2010), is converted into a stochastic sequential decision making problem by introducing noise in the transition function of DST. The state space of Deep Sea Treasure (DST) consists of a 10×11 grid (Fig. 2(*a*)). The action space of DST includes four actions (up, down, left and right), each sending the agent to one adjacent square in the indicated direction with probability $1-\eta$, and in the other three directions with equal probability $\eta/3$, where $0 \leq \eta < 1$ indicates the noise level in the environment. When the selected action would send the agent beyond the grid or the sea borders, the agent stays in the same place. Each policy, with the top left square as initial state, gets a two dimensional reward: the time spent until reaching a terminal state or reaching the time horizon $T = 100$, and the treasure attached to the terminal state (Fig. 2(*a*)). The list of all 10 non-dominated vectorial rewards in the form of (−time, treasure) are depicted in the two-dimensional plane in Fig. 2(*b*). It is worth noting that the Pareto front is non-convex.

4.2.1. Baseline algorithm

As mentioned in the introduction, the state of the art in MORL considers a scalar aggregation (e.g. a weighted sum) of rewards associated to all objectives. Several multiple-policy MORL algorithms have been proposed (Natarajan and Tadepalli, 2005; Tesauro et al., 2007; Barrett and Narayanan, 2008; Lizotte et al., 2012) using the weighted sum of the

objectives (with several weight settings) as scalar reward, which is optimized using standard reinforcement learning algorithms. The differences between the above algorithms are how they share the information between different weight settings and which weight settings they choose to optimize. In the following, MOMCTS-dom is compared to Multi-Objective Q-Learning (MOQL) (Vamplew et al. (2010)). Choosing MOQL as baseline is motivated as it yields all policies found by other linear-scalarisation based approaches, provided that a sufficient number of weight settings be considered.

Formally, in the two objective reinforcement learning case, MOQL optimizes independently m scalar RL problems through Q-learning, where the i-th problem considers reward $r_i = (1-\lambda_i) \times r_a + \lambda_i \times r_b$, where $0 \leq \lambda_i \leq 1, i = 1, 2, \ldots, m$ define the m weight settings of MOQL, and r_a (respectively r_b) is the first (resp. the second) objective reward. In its simplest version, the overall computational effort is equally divided between the m scalar RL problems. The computational effort allocated to the each weight setting is further equally divided into n_{tr} training phases; after the j-th training phase, the performance of the i-th weight setting is measured by the two-dimensional vectorial reward, noted $r_{i,j}$, of the current greedy policy. The m vectorial rewards of all weight settings $\{r_{1,j}, r_{2,j}, \ldots, r_{m,j}\}$ together compose the Pareto front of MOQL at training phase j.

4.2.2. Experimental setting

We use the same MOQL experimental setting as in Vamplew et al. (2010):

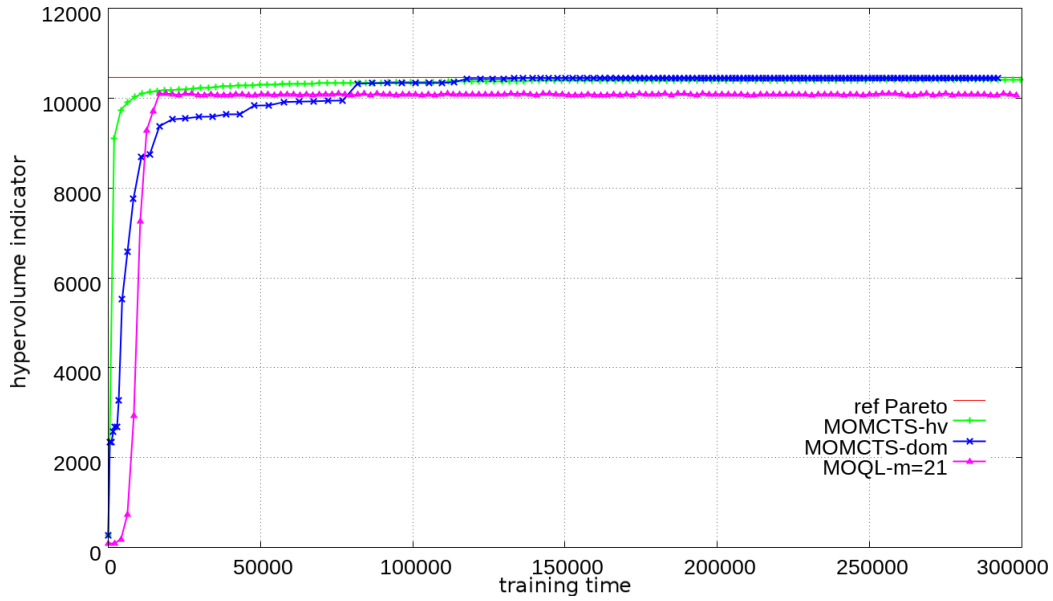
- ϵ-greedy exploration is used with $\epsilon = 0.1$.
- Learning rate α is set to 0.1.
- The state-action value table is optimistically initialized ($time = 0, treasure = 124$).
- Due to the episodic nature of DST, no discounting is used in MOQL($\gamma = 1$).
- The number m of weight settings ranges in $\{3, 7, 21\}$, with $\lambda_i = \frac{i-1}{m-1}, i = 1, 2, \ldots, m$.

After a few preliminary experiments, the progressive widening parameters b is set to 2 in both MOMCTS-hv and MOMCTS-dom. In MOMCTS-hv, the exploration vs exploitation (EvE) trade-off parameters in the time cost and treasure value objectives are respectively set to $c_{time} = 20,000$ and $c_{treasure} = 150$. As the DST problem is concerned with minimizing the search time (maximizing its opposite) and maximizing the treasure value, the reference point used in the hypervolume indicator calculation is set to (-100,0).
In MOMCTS-dom, the EvE trade-off parameter c_e is set to 1, and the discount factor δ is set to 0.999.

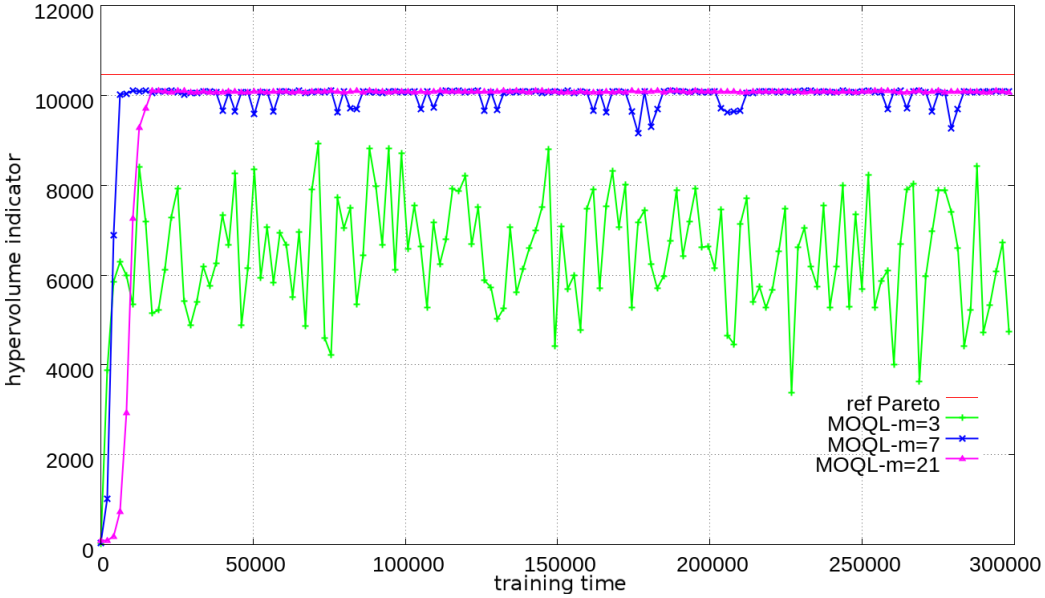
Experiments are carried out in a DST simulator with the η noise level ranging in 0, 1×10^{-3}, 1×10^{-2}, 5×10^{-2} and 0.1. The training time of MOQL, MOMCTS-hv and MOMCTS-dom is limited to 300,000 time steps (ca 37,000 tree-walks in MOMCTS-hv and 45,000 tree-walks in MOMCTS-dom). The entire training process is equally divided into $n_{tr} = 150$ phases. At the end of each training phase, the MOQL and MOMCTS solution sets are tested in the DST simulator, and form the Pareto set P. The performance of algorithms is reported as the hypervolume indicator of P.

4.2.3. Results

Table 1 shows the performance of MOMCTS approaches and MOQL measured by the hypervolume indicator, with reference point $z = (-100, 0)$.



(a)



(b)

Figure 3: The hypervolume indicator performance of MOMCTS-hv, MOMCTS-dom and MOQL versus training time in the deterministic DST problem ($\eta = 0$). For the sake of comparison with MOQL, the training time refers to the number of action selections in MOMCTS approaches, and each tree-walk in MOMCTS carries out on average 7 to 8 action selections in the DST problem. Top: The hypervolume indicator of MOMCTS-hv, MOMCTS-dom and MOQL-m=21. Bottom: The hypervolume indicator of MOQL with $m = 3, 7, 21$.

Table 1: The DST problem: hypervolume indicator results of MOMCTS-hv, MOMCTS-dom and MOQL with m ranging in 3,7 and 21 in with different noise levels η, averaged over 11 independent runs. The optimal hypervolume indicator is 10455. For each η, significantly better results are indicated in bold font (significance value $p < 0.05$ for the Student's t-test).

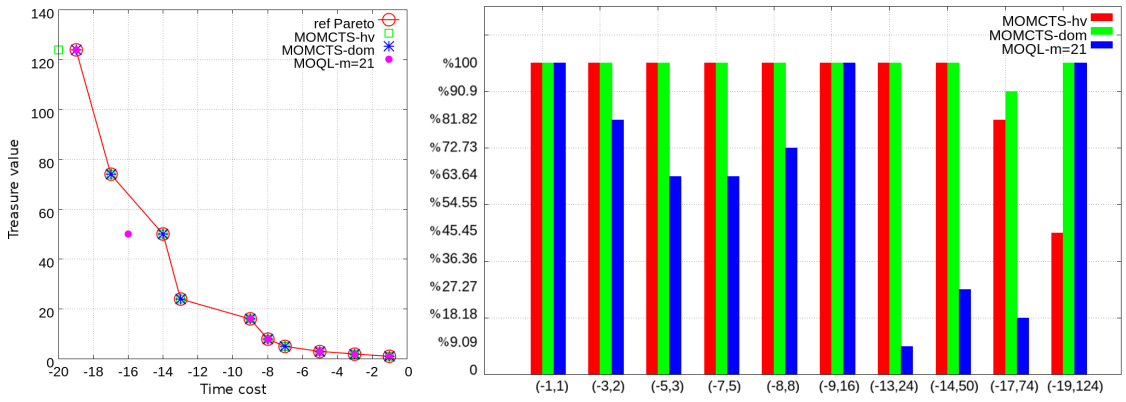
	$\eta = 0$	$\eta = 1 \times 10^{-3}$	$\eta = 0.01$	$\eta = 0.05$	$\eta = 0.1$
MOMCTS-hv	**10416±37**	**10434±31**	10436±32	10205±211	9883±1091
MOMCTS-dom	**10450±4**	**10446±19**	10389±65	9858±1153	9982±360
MOQL-m=3	7099±3926	8116±3194	6422± 4353	7333±4411	6953±3775
MOQL-m=7	10078±34	10049±94	9495±1701	8345±2887	8924±2663
MOQL-m=21	10078±17	10085±129	7806±1933	8744±2070	6744±2355

Deterministic setting

Fig. 3 displays the hypervolume indicator performance of MOMCTS-hv, MOMCTS-dom and that of MOQL for $m = 3, 7, 21$ in the deterministic setting ($\eta = 0$). It is observed that for $m = 7$ or 21, MOQL reaches a performance plateau (10062) within 20,000 time steps. The fact that MOQL does not reach the optimal hypervolume indicator 10455 is explained as the DST Pareto front is not convex (Fig. 2(b)). As widely known (Deb, 2001), linear-scalarisation based approaches of MOO fail to discover solutions in non-convex regions of the Pareto front. In such cases, MOQL is prevented from finding the true Pareto front and thus is inconsistent. Ultimately, MOQL only discovers the extreme points (-19,124) and (-1,1) of the Pareto front (Fig. 4(a)). In the meanwhile, MOMCTS-hv performance dominates that of MOQL throughout the training process. MOMCTS-dom catches up MOQL after 80,000 time steps. The entire Pareto front is found by MOMCTS-hv in 5 out 11 runs, and by MOMCTS-dom algorithm in 10 out 11 runs.

Fig. 3(b) shows the influence of m on MOQL. For $m = 7$, MOQL reaches the performance plateau before $m = 21$ (respectively 8,000 time steps vs 20,000 time steps), albeit with some instability. The instability increases as m is set to 3. The fact that for MOQL-$m = 3$ fails to reach the MOQL performance plateau is explained as the extreme point (-19,124) can be missed in some runs as MOQL uses a discount factor of 1 (after Vamplew et al. (2010)). Therefore the largest 124 treasure might be discovered later than in time step 19.

The percentage of times out of 11 runs that each non-dominated vectorial reward is discovered for at least one test episode during the training process of MOMCTS-hv, MOMCTS-dom and MOQL for $m = 21$ is displayed in Fig. 4(b). This picture shows that MOQL discovers all strategies (lying in the non-convex regions of the Pareto front) during intermediate test episodes. However, these non-convex strategies are eventually discarded as the MOQL solution set gradually converges to extreme strategies. Quite the contrary, MOMCTS approaches discovers all strategies in the Pareto front, and keeps them in the search tree after they have been discovered. The weakness of MOMCTS-hv is that the longest decision sequences corresponding to the vectorial rewards (-17,74) and (-19,124) need more time to be discovered. The MOMCTS-dom successfully discovers all non-dominated vectorial rewards (in 10 out of 11 runs) and reaches an average hypervolume indicator performance slightly higher than that of MOMCTS-hv.



(a) (b)

Figure 4: Left: The vectorial rewards found by representative MOMCTS-hv, MOMCTS-dom and MOQL-$m = 21$ runs. Right: The percentage of times out of 11 runs that each non-dominated vectorial reward was discovered by MOMCTS-hv, MOMCTS-dom and MOQL-$m = 21$, during at least one test episode.

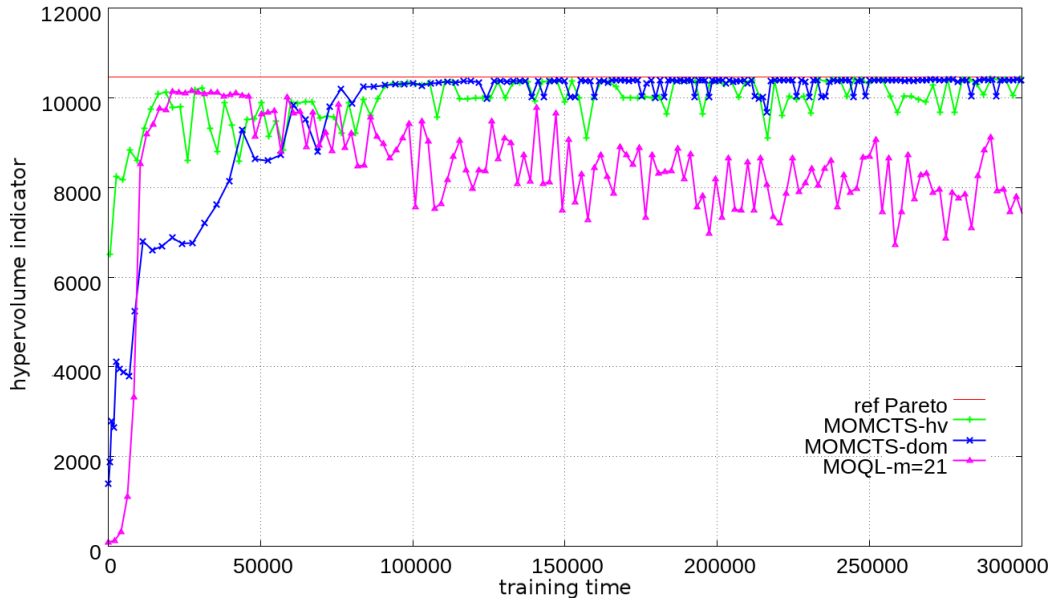
Stochastic setting

Fig. 5 shows the performance of MOMCTS-hv, MOMCTS-dom and MOQL-m=21 in the stochastic environments ($\eta = 0.01, 0.1$). As could have been expected, the performances of MOQL and MOMCTS approaches decrease and their variances increase with noise level η, although their performances improve with training time (except for the MOQL in the $\eta = 0.01$ case). In the low noise case ($\eta = 0.01$), MOQL reaches its optimal performance after time step 40,000, with a high performance variance. It is outperformed by MOMCTS-hv and MOMCTS-dom, with higher average hypervolume indicators and lower variances. When the noise rate increases ($\eta = 0.1$), both performances are degraded while MOMCTS approaches still outperforms MOQL in terms of relative performance and lower variance (as shown in Table 1), showing a good robustness w.r.t. noise.

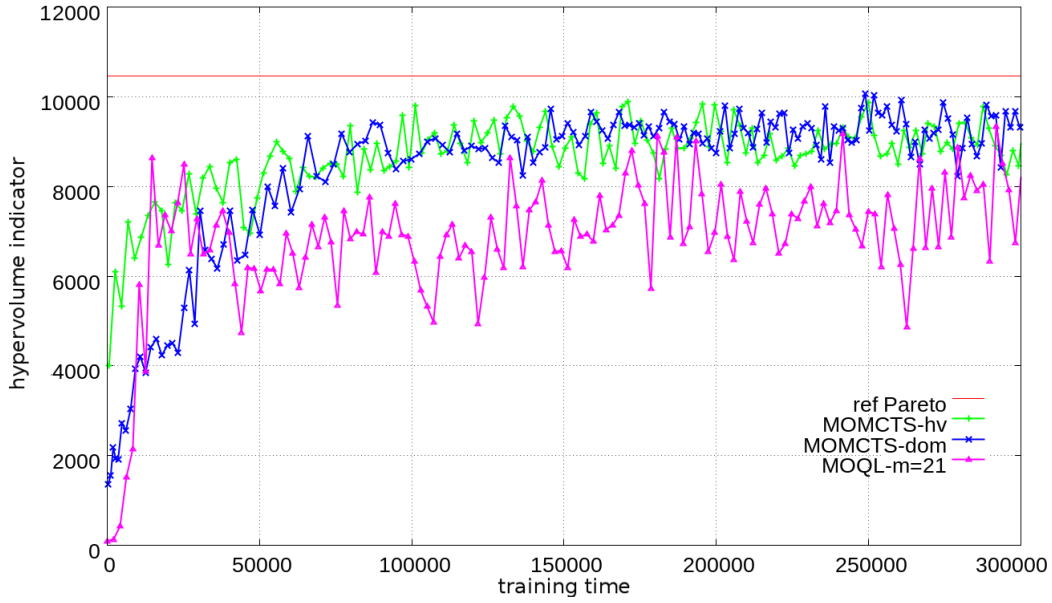
In summary, the empirical validation on the artificial DST problem shows both the strengths and the weaknesses of MOMCTS approaches. On the positive side, MOMCTS approaches show able to find solutions lying in the non-convex regions of the Pareto front, as opposed to linear scalarization-based methods. Moreover, MOMCTS shows a relatively good robustness w.r.t. noise. On the negative side, MOMCTS approaches are more computationally expensive than MOQL (for 300,000 time steps, MOMCTS-hv takes 147 secs, MOMCTS-dom takes 49 secs versus 25 secs for MOQL).

4.3. Resource Gathering

The Resource Gathering (RG) task first introduced in Barrett and Narayanan (2008) is carried out in a 5×5 grid (Fig. 6). The action space of RG include the same four actions (up, down, left and right) as in the DST problem. Starting from the home location, the goal of the agent is to gather two resources (gold and gems) and take them back home. Each time the agent reaches one resource location, the resource is picked up. Both resources can be carried by the agent at the same time. If the agent steps on one of the two enemy cases



(a) $\eta = 0.01$



(b) $\eta = 0.1$

Figure 5: The hypervolume indicator of MOMCTS-hv, MOMCTS-dom and MOQL-m=21 versus training time in the stochastic environment $(\eta = 0.01, 0.1)$.

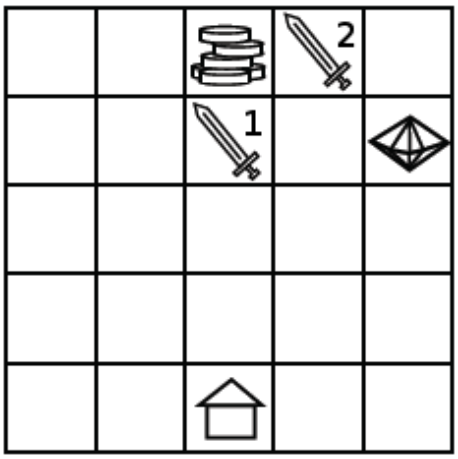


Figure 6: The Resource Gathering problem. The initial position of the agent is marked by the home symbol. Two resources (gold and gems) are located in fixed positions. Two enemy cases (marked by swords) send the agent back home with 10% probability.

(indicated by swords), it may be attacked with 10% probability, in which case the agent loses all resources being carried and is returned to the home location immediately. The agent enters a terminal state when it returns home (including the case of being attacked) or when the time horizon $T = 100$ is reached. Five possible immediate reward vectors ordered as $(enemy, gold, gems)$ will be received upon the termination of a policy:

- $(-1, 0, 0)$ in case of an enemy attack;
- $(0, 1, 0)$ for returning home with only gold;
- $(0, 0, 1)$ for returning home with only gems;
- $(0, 1, 1)$ for returning home with both gold and gems;
- $(0, 0, 0)$ in all other cases.

The RG problem contains a discrete state space of 100 states corresponding to the 25 agent positions in the grid, multiplied by the four possible states of resources currently being held (none, gold only, gems only, both gold and gems). The vectorial reward associated to each policy π is calculated as follows:
Let $r = (enemy, gold, gems)$ be the vectorial reward obtained by policy π after a L-step episode. The immediate reward of π is set to $r_{\pi;L} = r/L = (enemy/L, gold/L, gems/L)$, and the policy is associated its immediate reward averaged over 100 episodes, favoring the discovery of policies with shortest length. Seven policies (Table 2 and Fig. 7) corresponding to the non-dominated average vectorial rewards of the RG problem are identified by Vamplew et al. (2010). The non-dominated vectorial rewards compose a convex Pareto front in the three dimensional space (Fig. 8).

4.3.1. Experimental setting

In the RG problem, the MOMCTS approaches are assessed comparatively with the MOQL algorithm, which independently optimizes the weighted sums of the three objective func-

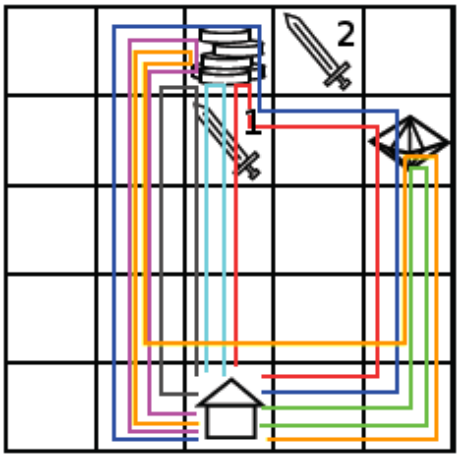


Figure 7: The seven policies in the Resource Gathering problem that correspond to the non-dominated vectorial rewards.

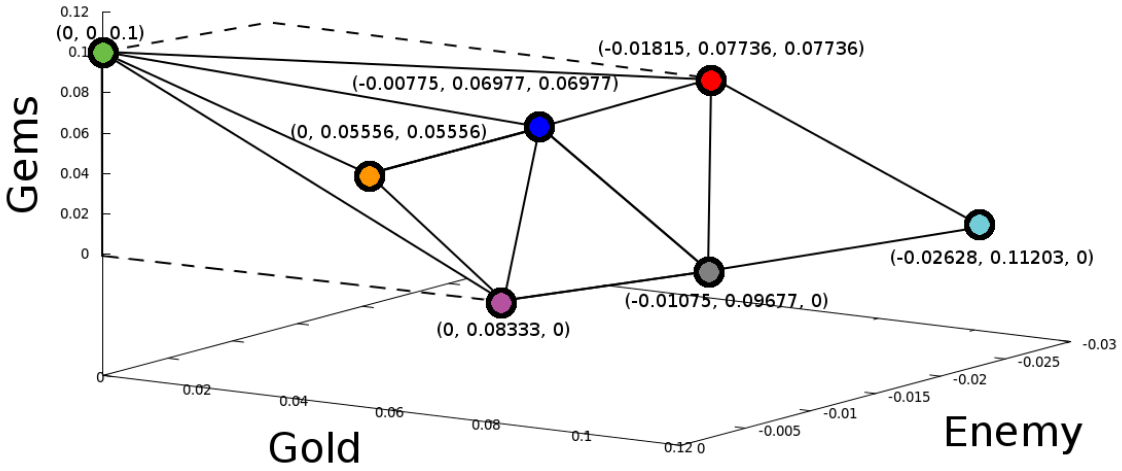


Figure 8: The seven non-dominated vectorial rewards in the Resource Gathering problem identified by Vamplew et al. (2010).

Table 2: The optimal policies for the Resource Gathering problem.

#	policy description	vectorial reward
π_1	Go directly to gems, avoiding enemies	(0,0,0.1)
π_2	Go to both gold and gems, avoiding enemies	$(0, 5.556 \times 10^{-2}, 5.556 \times 10^{-2})$
π_3	Go directly to gold, avoiding enemies	$(0, 8.333 \times 10^{-2}, 0)$
π_4	Go to both gold and gems, through enemy1 or enemy2 once	$(-7.75 \times 10^{-3}, 6.977 \times 10^{-2}, 6.977 \times 10^{-}$
π_5	Go directly to gold, through enemy1 once	$(-1.075 \times 10^{-2}, 9.677 \times 10^{-2}, 0)$
π_6	Go to both gold and gems, through the enemies twice	$(-1.815 \times 10^{-2}, 7.736 \times 10^{-2}, 7.736 \times 10^{}$
π_7	Go directly to gold, through enemy1 twice	$(-2.628 \times 10^{-2}, 1.1203 \times 10^{-1}, 0)$

tions $(enemy, gold, gems)$ under m weight settings. In the three dimensional reward space, one weight setting is defined by a 2D vector (λ_i, λ'_j), with $\lambda_i, \lambda'_j \in [0,1]$ and $0 \leq \lambda_i + \lambda'_j \leq 1$. Let us denote the scalar rewards optimized by MOQL as $r_{i,j} = (1 - \lambda_i - \lambda'_j) \times r_{enemy} + \lambda_i \times r_{gold} + \lambda'_j \times r_{gems}$, where l weights λ_i (respectively λ'_j) are evenly distributed in $[0,1]$ for the gold (resp. gems) objective, subject to $\lambda_i + \lambda'_j \leq 1$, the total number of weight settings thus is $m = \frac{l(l-1)}{2}$.

The parameters of MOQL and MOMCTS approaches have been selected after preliminary experiments, using the same amount of computational resources for a fair comparison. For the MOQL:

- The ϵ-greedy exploration is used with $\epsilon = 0.2$.
- Learning rate α is set to 0.2.
- The discount factor γ is set to 0.95.
- By taking $l = 4, 6, 10$, the number m of weight settings ranges in $\{6, 15, 45\}$.

In MOMCTS-hv, the progressive widening parameter b is set to 2. The exploration vs exploitation (EvE) trade-off parameters associated to each objective are defined as $c_{enemy} = 1 \times 10^{-3}, c_{gold} = 1 \times 10^{-4}, c_{gems} = 1 \times 10^{-4}$. The reference point z used in the hypervolume indicator calculation is set to $(-0.33, -1 \times 10^{-3}, -1 \times 10^{-3})$, where -0.33 represents the maximum enemy penalty averaged in each time step of the episode, and the -1×10^{-3} values in the gold and gems objectives are taken to encourage the exploration of solutions with vectorial rewards lying in the hyper-planes $gold = 0$ and $gems = 0$.
In MOMCTS-dom, the progressive widening parameter b is set to 1 (no progressive widening). The EvE trade-off parameter c_e is set to 0.1. The discount factor δ is set to 0.99.

The training time of all considered algorithms is 600,000 time steps (ca 17,200 tree-walks for MOMCTS-hv and 16,700 tree-walks for MOMCTS-dom). Like in the DST problem, the training process is equally divided into 150 phases. At the end of each training phase, the MOQL and MOMCTS solution sets are tested in the RG simulator. Each solution (strategy) is launched 100 times and is associated the average vectorial reward (which might dominate the theoretical optimal ones due to the limited sample). The vectorial rewards of the solution set provided by each algorithm defines its Pareto archive. The algorithm performance is set to the hypervolume indicator of the Pareto archive with reference point $z = (-0.33, -1 \times 10^{-3}, -1 \times 10^{-3})$. The optimal hypervolume indicator is 2.01×10^{-3}.

4.3.2. Results

Table 3 shows the performance of MOMCTS-hv, MOMCTS-dom and MOQL algorithms after 600,000 times steps of training, measured by the hypervolume indicator. Fig. 9 dis-

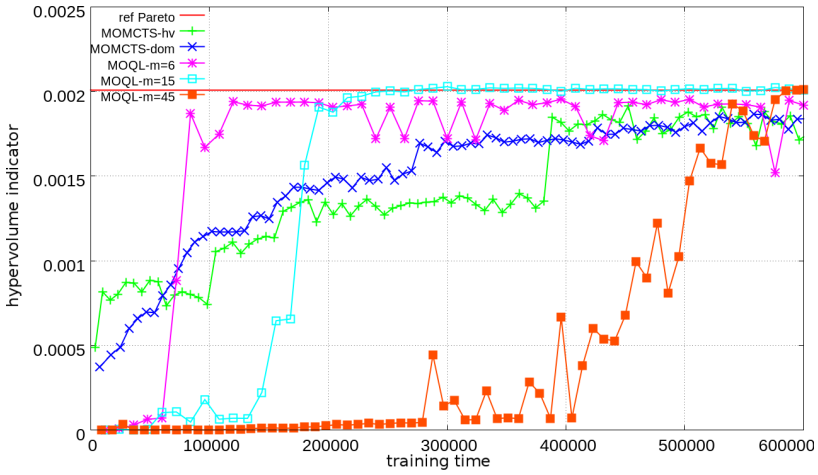
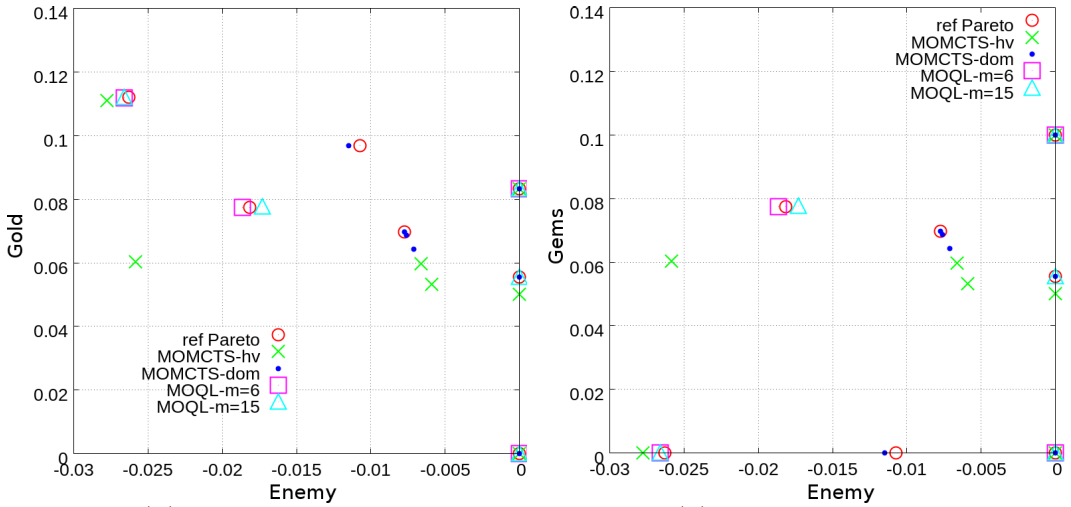


Figure 9: The Resource Gathering problem: Average hypervolume indicator of MOMCTS-hv, MOMCTS-domand MOQL (with $m = 6$, 15 and 45) over 11 runs, versus number of time steps. The optimal hypervolume indicator 2.01×10^{-3} is indicated by the top line.



(a) Enemy vs Gold (b) Enemy vs Gems

Figure 10: The vectorial rewards found by representative MOMCTS-hv, MOMCTS-dom and MOQL with $m = 6, 15$ runs. Left: the points projected on the $Gems = 0$ plane. Right: the points projected on the $Gold = 0$ plane. The Pareto optimal points are marked by circles.

Table 3: The Resource Gathering problem: Average hypervolume indicator of MOMCTS-hv, MOMCTS-dom and MOQL (with $m = 6$, 15 and 45) over 11 runs. The optimal hypervolume indicator is 2.01×10^{-3}. Significantly better results are indicated in bold font (significance value $p < 0.05$ for the Student's t-test).

	HV($\times 10^{-3}$)		HV($\times 10^{-3}$)
MOMCTS-hv	1.735±0.304	MOMCTS-dom, $\delta = 0.9$	1.285±0.351
MOQL, $m = 6$	1.933±0.04	MOMCTS-dom, $\delta = 0.98$	1.75±0.38
MOQL, $m = 15$	**2.021±0.033**	MOMCTS-dom, $\delta = 0.99$	1.836±0.175
MOQL, $m = 45$	**2.012±0.041**	MOMCTS-dom, $\delta = 0.999$	1.004±0.26

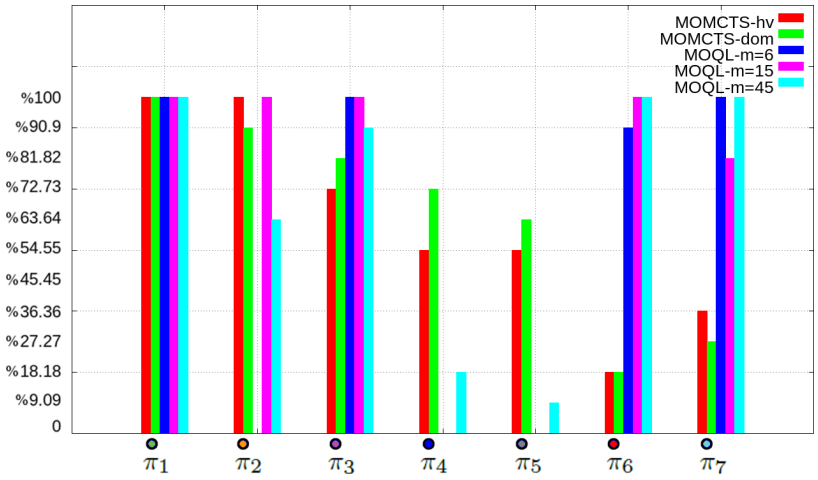


Figure 11: The percentage of of times out of 11 runs that each non-dominated vectorial reward was discovered by MOMCTS-hv,MOMCTS-domand MOQL with $m = 6, 15, 45$, during at least one test period.

plays the evolution of hypervolume indicator in MOMCTS-hv, MOMCTS-dom and MOQL with $m = 6, 15, 45$. The percentage of times out of 11 runs that each non-dominated vectorial reward is discovered for at least one test period during the training process of each algorithm is displayed in Fig. 11. It is observed that with $m = 6$ weight settings, the MOQL performance stops improving after reaching a plateau of 1.9×10^{-3} at 120,000 time steps. Inspecting the Pareto archive, the difference between the performance plateau of and the optimal performance (2.01×10^{-3}) is due to the non-discovery of policies π_2, π_4 and π_5 whose vectorial rewards are not covered by the 6 weight settings. MOQL reaches the optimum when m increases (after 240,000 steps for $m = 15$ and 580,000 steps for $m = 45$).

The MOMCTS approaches are outperformed by MOQL; their average hypervolume indicator reach 1.8×10^{-3} in the end of the training process, which is explained as the MOMCTS approaches rarely find the risky policies (π_6, π_7) (Fig. 11). For example, policy π_6 visits the enemy case twice; the neighbor nodes of this policy thus get the (-1,0,0) reward (more in section 5).

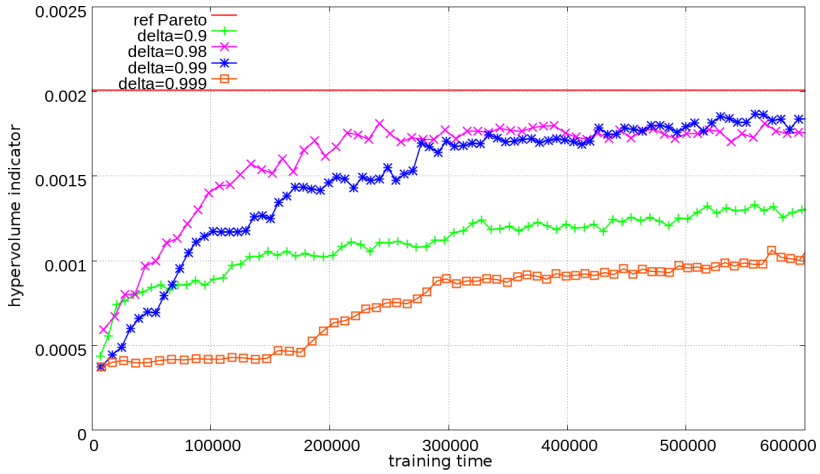


Figure 12: The hypervolume indicator performance of MOMCTS-dom with δ varying in $\{0.9, 0.98, 0.99, 0.999\}$, versus training time steps in the Resource Gathering problem.

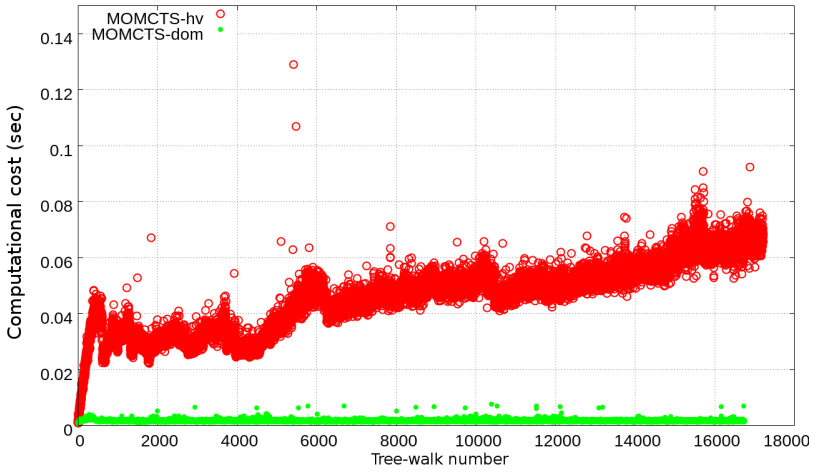


Figure 13: The Resource Gathering problem: average computational cost for one tree-walk for MOMCTS-hv and MOMCTS-dom over 11 independent runs. On average, each tree-walk in MOMCTS is ca. 35 training time steps.

As shown in Fig. 12, the δ parameter governs the MOMCTS-dom performance. A low value ($\delta = 0.9$) leads to quickly forgetting the discovery of non-dominated rewards, turning MOMCTS-dom into pure exploration. Quite the contrary, high values of δ ($\delta = 0.999$) limit the exploration and likewise hinder the overall performance.

On the computational cost side, the average execution time of 600,000 training steps of in MOMCTS-hv, MOMCTS-dom and MOQL are respectively 944 secs, 47 secs and 43 secs. As the size of the Pareto archive is close to 10 in most tree-walks of MOMCTS-hv and

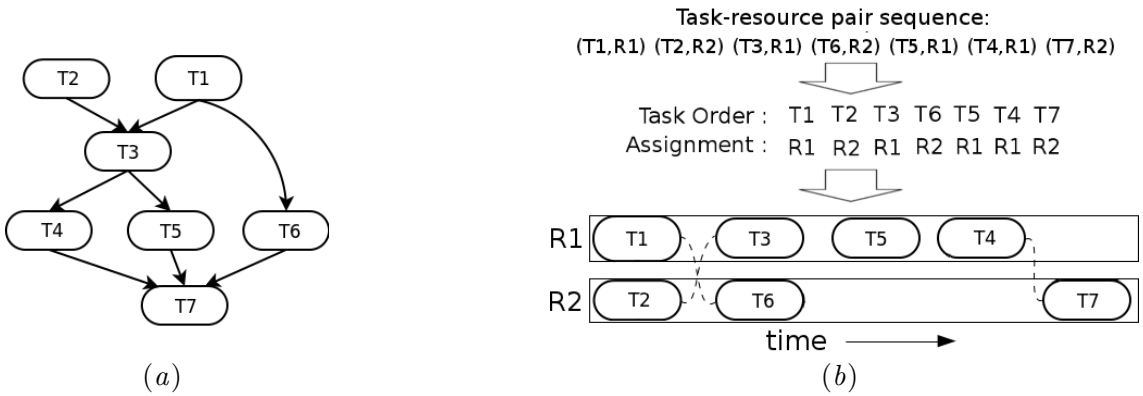


Figure 14: Scheduling a job containing 7 interdependent tasks on a grid of 2 resources. Left: The dependency graph of tasks in the job. Right: The illustration of an execution plan.

MOMCTS-dom, the fact that MOMCTS-hv algorithm is 20 times slower than MOMCTS-dom matches their computational complexities.
As shown in Fig. 13, the cost of tree-walks in MOMCTS-hv increases up to 20 times higher than that of MOMCTS-dom within the first 500 tree-walks, during which period the Pareto archive size $|P|$ grows. Afterwards, the cost of MOMCTS-hv gradually increases with the depth of the search tree ($\mathcal{O}(\log N)$). On the contrary, the computational cost of each tree-walk in MOMCTS-dom remains stable (between 1×10^{-3} secs and 2×10^{-3} secs) throughout the training process.

4.4. Grid scheduling

Pertaining to the domain of autonomic computing (Tesauro et al., 2007), the problem of grid scheduling has been selected to investigate the scalability of MOMCTS approaches. The presented experimental validation considers the problem of grid scheduling, referring the reader to Yu et al. (2008) for a comprehensive presentation of the field. Grid scheduling at large is concerned with scheduling the different tasks involved in the jobs on different computational resources. As tasks are interdependent and resources are heterogeneous, grid scheduling defines an NP-hard combinatorial optimization problem (Ullman (1975)).

Grid scheduling naturally aims at minimizing the so-called makespan, that is the overall job completion time. But other objectives such as energy consumption, monetary cost, or the allocation fairness w.r.t. the resource providers become increasingly important. In the rest of section 4.4, two objectives will be considered, the makespan and the cost of the solution.

In grid scheduling, a job is composed of J tasks $T_1 \dots T_J$, partially ordered through a dependency relation; $T_i \to T_j$ denotes that task T_i must be executed before task T_j (Fig. 14(a)). Each task T_i is associated with its unitary load L_i. Each task is assigned one out of M resources $R_1, \dots R_M$; resource R_k has computational efficiency $speed_k$ and unitary cost $cost_k$. Grid scheduling achieves the task-resource assignment and orders the tasks executed on each resource. A grid scheduling solution called execution plan is given as a sequence σ of (task-resource) pairs (Fig. 14(b)).

Let $\rho(i) = k$ denote the index of the resource R_k on which T_i is executed. Let $\mathcal{B}(T_i)$ denote the set of tasks T_j which must either be executed before T_i ($T_j \rightarrow T_i$) or which are scheduled to take place before T_i on the same resource $R_{\rho(i)}$. The completion time of a task T_i is recursively computed as:

$$end(T_i) = \frac{L_i}{speed_{\rho(i)}} + max\{end(T_j), T_j \in \mathcal{B}(T_i)\}$$

where the first term is the time needed to process T_i on the assigned resource $R_{\rho(i)}$, and the second term expresses the fact that all jobs in $\mathcal{B}(T_i)$ must be completed prior to executing T_i.
Finally, grid scheduling is the two-objective optimization problem aimed at minimizing the overall scheduling makespan and cost:

$$\begin{array}{l}\text{Find } (\sigma) = argmin\ \{max\{end(T_j),\ j = 1 \ldots J\}\ ; \\ \sum_{k=1\ldots M} \frac{cost_k}{speed_k} \times \sum_{i\ s.t.\ \rho(i)=k} L_i\}\end{array}$$

4.4.1. Baseline algorithms

The state of the art in grid scheduling is achieved by stochastic optimization algorithms (Yu et al., 2008). The two prominent multi-objective variants thereof are NSGA-II (Deb et al., 2000) and SMS-EMOA (Beume et al., 2007).
Both algorithms can be viewed as importance sampling methods. They maintain a population of solutions, initially defined as random execution plans. Iteratively, the solutions with best Pareto rank and best crowded distance (a density estimation of neighboring points in NSGA-II) or hypervolume indicator (in SMS-EMOA) are selected and undergo unary and binary stochastic perturbations.

4.4.2. Experimental setting

A simulated grid environment containing 3 resources with different unit time costs and processing capabilities ($cost_1 = 20, speed_1 = 10; cost_2 = 2, speed_2 = 5; cost_3 = 1, speed_3 = 1$) is defined. We firstly compare the performance of MOMCTS approaches and baseline algorithms on a realistic bio-informatic workflow *EBI_ClustalW2*, which performs a ClustalW multiple sequence alignment using the EBI's WSClustalW2 service[3]. This workflow contains 21 tasks and 23 precedence pairs (graph density $q = 12\%$), assuming that all workloads are equal. Secondly, the scalability of MOMCTS approaches is tested through experiments based on artificially generated workflows containing respectively 20, 30 and 40 tasks with graph density $q = 15\%$.

As evidenced from the literature (Wang and Gelly (2007)), MCTS performances heavily depend on the so-called random phase (section 2.2). Preliminary experiments showed that a uniform action selection in the random phase was ineffective. A simple heuristic was thus used to devise a better suited action selection criterion in the random phase, as follows.
Let EFT_i define the expected finish time of task T_i (computed off-line):

$$\text{EFT}_i = L_i + max\{\text{EFT}_j\ s.t.\ T_j \rightarrow T_i\}$$

The heuristic action selection uniformly selects an admissible task T_i. It then compares EFT_i to all EFT_j for T_j admissible. If EFT_i is maximal, T_i is allocated to the resource

3. The complete description is available at *http://www.myexperiment.org/workflows/203.html.*

which is due to be free at the earliest; if EFT_i is minimal, T_i is allocated to the resource which is due to be free at the latest. The random phase thus implements a default policy, randomly allocating tasks to resources, except for the most (respectively less) critical tasks that are scheduled with high (resp. low) priority.

The parameters of all algorithms have been selected after preliminary experiments, using the same amount of computational resources for a fair comparison. The progressive widening parameter b is set to 2 in both MOMCTS-hv and MOMCTS-dom. In MOMCTS-hv, the exploration vs. exploitation (EvE) trade-off parameters associated to the makespan and cost objectives, c_{time} and c_{cost} are both set to 5×10^{-3}. In MOMCTS-dom, the EvE trade-off parameters c_e is set to 1, and the discount factor δ is set to 0.99. The parameters used for NSGA-II (respectively SMS-EMOA) involve a population size of 200 (resp. 120) individuals, of which 100 are selected and undergo stochastic unary and binary variations (resp. one-point re-ordering, and resource exchange among two individuals). For all three algorithms, the number N of tree-walks a.k.a. evaluation budget is set to 10,000. The reference point in each experiment is set to (z_t, z_c), where z_t and z_c respectively denote the maximal makespan and cost.

Due to the lack of the true Pareto front in the considered problems, we use a reference Pareto front P^* gathering all non-dominated vectorial rewards obtained in all runs of all three algorithms to replace the true Pareto front. The performance indicators are defined by the generational distance (GD) and inverted generational distance (IGD) between the actual Pareto front P found in the run and the reference Pareto front P^*. In the grid scheduling experiment, the IGD indicator plays a similar role as the hypervolume indicator in DST and RG problems.

4.4.3. Results

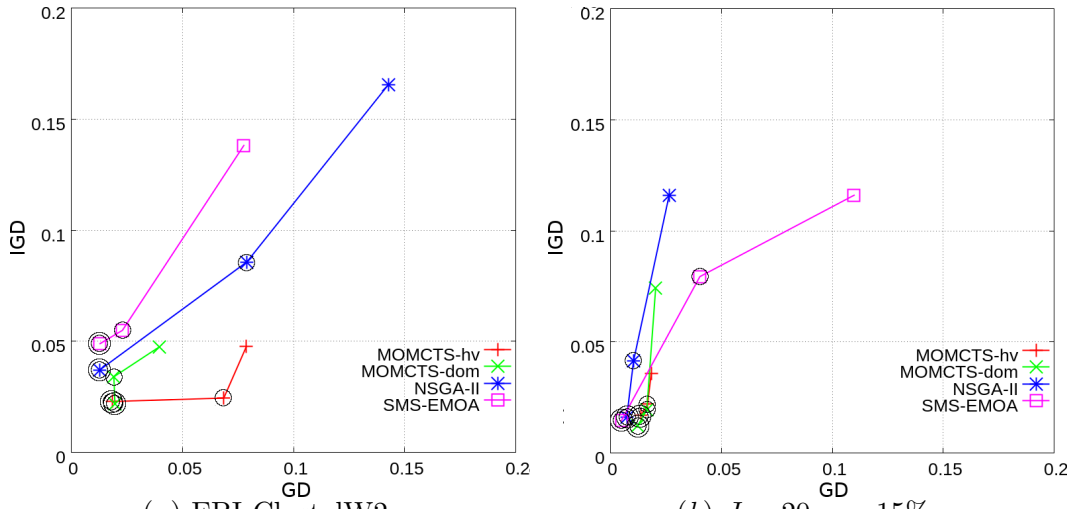
Fig. 15 displays the GD and IGD of MOMCTS-hv, MOMCTS-dom, NSGA-II and SMS-EMOA on EBI_ClustalW2 workflow scheduling and on artificial jobs with a number J of tasks ranging in $20, 30$ and 40 with graph density $q = 15\%$. Fig. 16 shows the Pareto front discovered by MOMCTS-hv, MOMCTS-dom, NSGA-II and SMS-EMOA on the EBI_ClustalW2 workflow after $N = 100,\ 1000$ and 10000 policy evaluations (tree-walks), comparatively to the reference Pareto front. In all considered problems, the MOMCTS approaches are outperformed by the baselines in terms of the GD indicator, while they quickly find good solutions, they fail to discover the reference Pareto front. In the meanwhile, they yield a better IGD performance than the baselines, indicating that on average a single run of MOMCTS approaches reaches a better approximation of the true Pareto front.

Overall, the main weakness of MOMCTS approaches is their computational runtime. The computational cost of MOMCTS-hv and MOMCTS-dom are respectively 5 and 2.5 times higher than that of NSGA-II and SMS-EMOA[4]. This weakness should have been relativized, noting that in real-world problems, the evaluation cost dominates by several orders of magnitude the search cost.

5. Discussion

As mentioned, the state of the art in MORL is divided into single-policy and multiple policy algorithms (Vamplew et al., 2010). In the former case, the authors use a set of preferences between objectives which are user-specified or derived from the problem domain

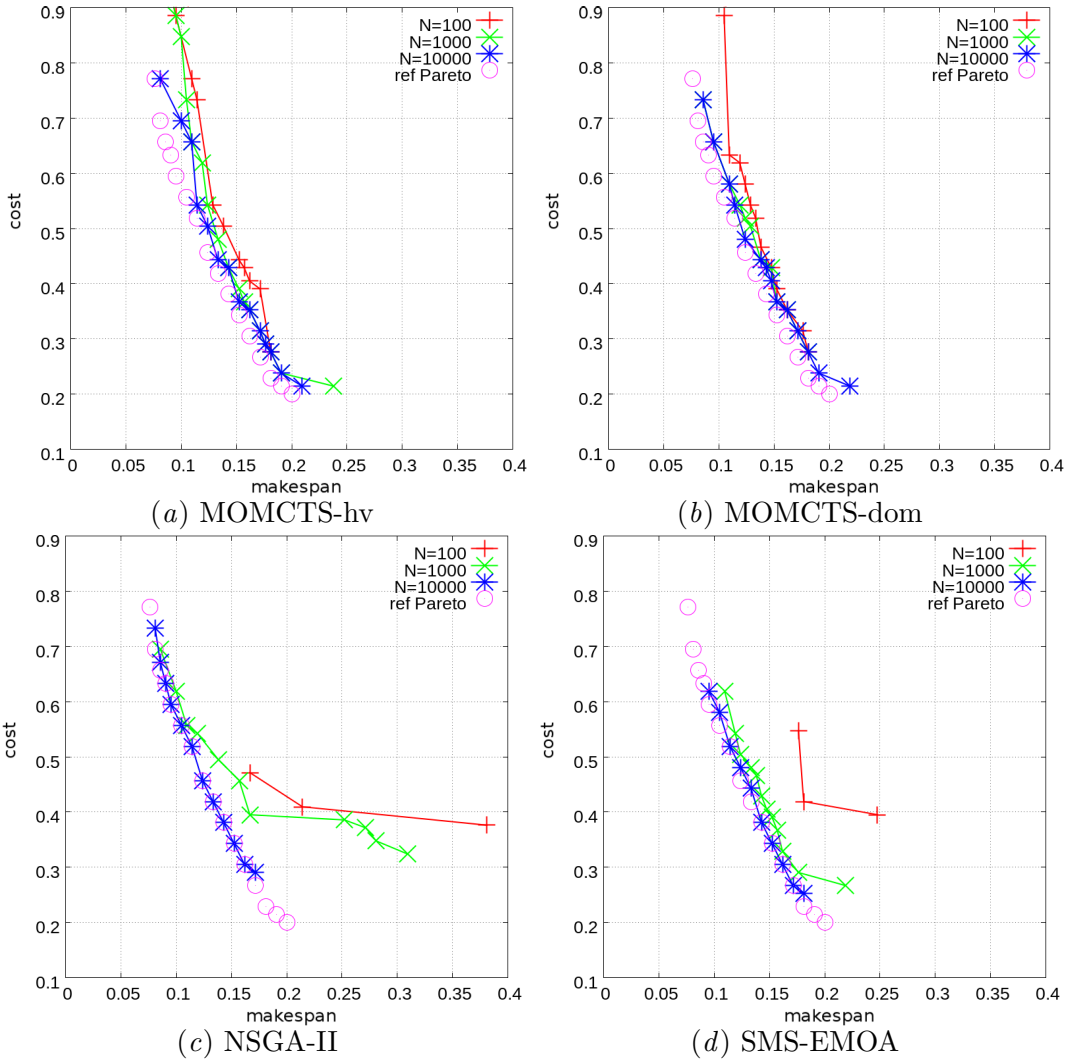
4. On workflow EBI_ClustalW2, the average execution time of MOMCTS-hv, MOMCTS-dom, NSGA-II and SMS-EMOA are respectively 142 secs, 74 secs, 31 secs and 32 secs.



(a) EBI_ClustalW2 (b) $J = 20, q = 15\%$

(c) $J = 30, q = 15\%$ (d) $J = 40, q = 15\%$

Figure 15: The generational distance (GD) and inverted generational distance (IGD) for $N = 100$, 1000 and 10000 of MOMCTS-hv, MOMCTS-dom, NSGA-II and SMS-EMOA on (a): EBI_ClustalW2; (b)(c)(d): artificial problems with number of tasks J and graph density q. Each performance point after 1000 and 10 000 evaluations are respectively marked by single and double cycles.



(*a*) MOMCTS-hv (*b*) MOMCTS-dom

(*c*) NSGA-II (*d*) SMS-EMOA

Figure 16: Progression of the Pareto-optimal solutions found for $N = 100$, 1000 and 10000 for MOMCTS-hv, MOMCTS-dom, NSGA-II and SMS-EMOA on the EBI_ClustalW2 workflow. The reference Pareto front is indicated by circles.

(e.g. defining preferred regions (Mannor and Shimkin, 2004) or setting weights on the objectives (Tesauro et al., 2007)) to aggregate the multiple objectives in a single one. The strength of the single-policy approach is its simplicity; its long known limitation is that it cannot discover a policy in non-convex regions of the Pareto front (Deb, 2001).

In the multiple-policy case, multiple Pareto optimal vectorial rewards can be obtained by optimization of different scalarized RL problems under different weight settings. Natarajan and Tadepalli (2005) show that the efficiency of MOQL can be improved by sharing information between different weight settings. A hot topic in multiple-policy MORL is how to design the weight settings and share information among the different scalarized RL problems. In the case where the Pareto front is known, the design of the weight settings is made easier − provided that the Pareto front is convex. When the Pareto front is unknown, an alternative provided by Barrett and Narayanan (2008) is to maintain Q-vectors instead of Q-values for each pair (state, action). Through an adaptive selection of weight settings corresponding to the vectorial rewards on the boundary of the convex set of the current Q-vectors, this algorithm narrows down the set of selected weight settings, at the expense of an higher complexity of value iteration in each state: the $\mathcal{O}(|S||A|)$ complexity of standard Q-learning is multiplied by a factor $O(n^d)$, where n is the number of points on the convex hull of the Q-vectors and d is the number of objectives. While the approach provides optimality guarantees (n converge toward the number of Pareto optimal policies), the number of intermediate solutions can be huge (in the worst case, $\mathcal{O}(|A|^{|S|})$). Based on the convexity and piece-wise linearity assumption on the shape of the convex hull of Q-vectors, Lizotte et al. (2012) extends (Barrett and Narayanan, 2008) by narrowing down the range of points locating on the convex hull, thus keeping the n value under control.

In the MOMCTS-hv approach, each tree node is associated its average reward w.r.t. each objective, and the selection rule involves the scalar associated reward based on the hypervolume indicator (Zitzler and Thiele, 1998), with complexity $\mathcal{O}(B|P|^{d/2}logN)$. On the one hand, this complexity is lower than that of a value iteration in Barrett and Narayanan (2008) (considering that the size of archive P is comparable to the number n of non-dominated Q vectors). On the other hand, this complexity is higher than that of MOMCTS-dom, where the dominance test only needs be computed at the end of each tree-walk, thus with linear complexity in the number of objectives and tree-walks. The MOMCTS-dom complexity thus is $\mathcal{O}(BlogN + d|P|)$. The price to pay for the improved scalability of MOMCTS-dom is that the dominance reward might less favor the diversity of the Pareto archive than the hypervolume indicator: any non-dominated point has the same dominance reward whereas the hypervolume indicator of non-dominated points in sparsely populated regions of the Pareto archive is higher.

As shown in the Resource Gathering problem, the MOMCTS approaches have difficulties in finding "risky" policies, visiting nodes with many low reward nodes in their neighborhood. A tentative explanation for this fact is given as, as already noted by Coquelin and Munos (2007), it may require an exponential time for the UCT algorithm to converge to the optimal node if this node is hidden by nodes with low reward.

6. Conclusion and perspectives

This paper has pioneered the extension of MCTS to multi-objective reinforcement learning, based on two scalar rewards measuring the merits of a policy relatively to the non-dominated policies in the search tree. These rewards, respectively the hypervolume indicator and the dominance reward, have complementary strengths and weaknesses: the hypervolume indicator is computationally expensive but explicitly favors the diversity of the MOO policies, enforcing a good coverage of the Pareto front. Quite the contrary, the dominance test is linear in the number of objectives; it is further invariant under the

monotonous transformation of the objective functions, a robust property much appreciated when dealing with ill-posed optimization problems.

These approaches have been validated on three problems : Deep Sea Treasure (DST), Resource Gathering (RG) and grid scheduling.

The experimental results on DST confirm a main merit of the proposed approaches, their ability to discover policies lying in the non-convex regions of the Pareto front. To our knowledge[5], this feature is unique in the MORL literature.

In counterpart, MOMCTS approaches suffer from two weaknesses. Firstly, as shown on the grid scheduling problem, some domain knowledge is required in complex problems to enforce an efficient exploration in the random phase. Secondly, as evidenced in the Resource Gathering problem, the presented approaches hardly discover "risky" policies which lie in an unpromising region (the proverbial needle in the haystack).

These first results however provide a proof of concept for the MOMCTS approaches, noting that these approaches yield comparable performances to the (non RL-based) state of the art albeit at the price of a higher computational cost.

This work opens two perspectives for further studies. The main theoretical perspective concerns the properties of the cumulative discounted reward mechanism in the general (single-objective) dynamic optimization context. On the applicative side, we plan to refine the RAVE heuristics used in the grid scheduling problem, e.g. to estimate the reward attached to task allocation paired ordering.

Acknowledgments

We wish to thank Jean-Baptiste Hoock, Dawei Feng, Ilya Loshchilov, Romaric Gaudel, and Julien Perez for many discussions on UCT, MOO and MORL. We are grateful to the anonymous reviewers for their many comments and suggestions on a previous version of the paper.

References

P. Auer, N. Cesa-Bianchi, and P. Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine Learning*, 47(2):235–256, 2002.

A. Auger, J. Bader, D. Brockhoff, and E. Zitzler. Theory of the hypervolume indicator: optimal μ-distributions and the choice of the reference point. In *FOGA'09*, pages 87–102. ACM, 2009.

L. Barrett and S. Narayanan. Learning all optimal policies with multiple criteria. In W. W. Cohen, A. McCallum, and S. T. Roweis, editors, *ICML'08*, pages 41–47. ACM, 2008.

V. Berthier, H. Doghmen, and O. Teytaud. Consistency modifications for automatically tuned Monte-Carlo Tree Search. In C. Blum and R. Battiti, editors, *LION4*, pages 111–124. LNCS 6073, Springer-Verlag, 2010.

N. Beume, B. Naujoks, and M. Emmerich. SMS-EMOA: Multiobjective selection based on dominated hypervolume. *European Journal of Operational Research*, 181(3):1653 – 1669, 2007.

5. A general polynomial result of MOO has been proposed by Chatterjee (2007), which claims that for all irreducible MDP with multiple long-run average objectives, the Pareto front can be ϵ-approximated in time polynomial in ϵ. However this claim relies on the assumption that *finding some Pareto optimal point can be reduced to optimizing a single objective: optimize a convex combination of objectives using as set of positive weights* (page 2, Chatterjee (2007)), which does not hold for non-convex Pareto fronts. Furthermore, the approach relies on the ϵ-approximation of the Pareto front proposed by Papadimitriou and Yannakakis (2000), which assumes the existence of an oracle telling for each vectorial reward whether it is ϵ-Pareto-dominated (Thm. 2, page 4, Papadimitriou and Yannakakis (2000)).

N. Beume, C. M. Fonseca, M. Lopez-Ibanez, L. Paquete, and J. Vahrenhold. On the complexity of computing the hypervolume indicator. *IEEE Transactions on Evolutionary Computation*, 13(5):1075–1082, 2009.

G. Chaslot, L. Chatriot, C Fiter, S. Gelly, J.B. Hoock, J. Perez, A. Rimmel, and O. Teytaud. Combining expert, offline, transient and online knowledge in monte-carlo exploration. 2008.

K Chatterjee. Markov decision processes with multiple long-run average objectives. *FSTTCS 2007 Foundations of Software Technology and Theoretical Computer Science*, 4855:473–484, 2007.

P. Ciancarini and G. P. Favini. Monte-Carlo Tree Search techniques in the game of kriegspiel. In C. Boutilier, editor, *IJCAI'09*, pages 474–479, 2009.

P.A. Coquelin and R. Munos. Bandit algorithms for tree search. *arXiv preprint cs/0703062*, 2007.

R. Coulom. Efficient selectivity and backup operators in Monte-Carlo Tree Search. In *Proc. Computers and Games*, pages 72–83, 2006.

K. Deb. *Multi-objective optimization using evolutionary algorithms*, pages 55–58. Chichester, 2001.

K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II. In Schoenauer, M. et al., editor, *PPSN VI*, pages 849–858. LNCS 1917, Springer Verlag, 2000.

K. Deb, L. Thiele, M. Laumanns, and E. Zitzler. Scalable multi-objective optimization test problems. In *Proceedings of the Congress on Evolutionary Computation (CEC-2002),(Honolulu, USA)*, pages 825–830. Proceedings of the Congress on Evolutionary Computation (CEC-2002),(Honolulu, USA), 2002.

M. Fleischer. The measure of Pareto optima. applications to multi-objective metaheuristics. In *EMO'03*, pages 519–533. LNCS 2632, Springer Verlag, 2003.

Z. Gábor, Z. Kalmár, and C. Szepesvári. Multi-criteria reinforcement learning. In *ICML'98*, pages 197–205. Morgan Kaufmann, 1998.

S. Gelly and D. Silver. Combining online and offline knowledge in UCT. In Z. Ghahramani, editor, *ICML'07*, pages 273–280. ACM, 2007.

N. Hansen. The cma evolution strategy: a comparing review. *Towards a new evolutionary computation*, pages 75–102, 2006.

L. Kocsis and C. Szepesvári. Bandit based Monte-Carlo planning. In J. Fürnkranz, T. Scheffer, and M. Spiliopoulou, editors, *ECML'06*, pages 282–293. Springer Verlag, 2006.

D. J. Lizotte, M. Bowling, and S. A. Murphy. Linear fitted-q iteration with multiple reward functions. *Journal of Machine Learning Research*, 13:3253–3295, 2012.

F. Maes, L. Wehenkel, and D. Ernst. Automatic discovery of ranking formulas for playing with multi-armed bandits. In Scott Sanner and Marcus Hutter, editors, *Recent Advances in Reinforcement Learning - 9th European Workshop, EWRL 2011*, volume 7188 of *Lecture Notes in Computer Science*, pages 5–17. Springer, 2011.

S. Mannor and N. Shimkin. A geometric approach to multi-criterion reinforcement learning. *Journal of Machine Learning Research*, pages 325–360, 2004.

H. Nakhost and M. Müller. Monte-Carlo exploration for deterministic planning. In C. Boutilier, editor, *IJCAI'09*, pages 1766–1771, 2009.

S. Natarajan and P. Tadepalli. Dynamic preferences in multi-criteria reinforcement learning. In *ICML'05*. ACM, 2005.

C. H. Papadimitriou and M. Yannakakis. On the approximability of trade-offs and optimal access of web sources. In *FOCS*, pages 86–92. IEEE Computer Society, 2000.

R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 1998.

C. Szepesvári. *Algorithms for Reinforcement Learning*. Morgan & Claypool Publishers, 2010.

G. Tesauro, R. Das, H. Chan, J. Kephart, D. Levine, F. Rawson, and C. Lefurgy. Managing power consumption and performance of computing systems using reinforcement learning. In J. C. Platt, D. Koller, Y. Singer, and S. T. Roweis, editors, *NIPS'07*, pages 1–8, 2007.

J. D. Ullman. NP-complete scheduling problems. *Journal of Computer and System Sciences*, 10(3):384–393, 1975.

P. Vamplew, R. Dazeley, A. Berry, R. Issabekov, and E. Dekker. Empirical evaluation methods for multiobjective reinforcement learning algorithms. *Machine Learning*, 84:51–80, 2010.

D.A. Van Veldhuizen. Multiobjective evolutionary algorithms: classifications, analyses, and new innovations. Technical report, DTIC Document, 1999.

W. Wang and M. Sebag. Multi-objective Monte-Carlo Tree Search. In *Asian Conference on Machine Learning*, 2012.

Y. Wang and S. Gelly. Modifications of UCT and sequence-like simulations for Monte-Carlo Go. In *CIG'07*, pages 175–182. Ieee, 2007.

Y. Wang, J. Audibert, and R. Munos. Algorithms for infinitely many-armed bandits. In D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, editors, *NIPS'08*, pages 1–8, 2008.

J. Yu, R. Buyya, and K. Ramamohanarao. *Workflow Scheduling Algorithms for Grid Computing*, volume 146 of *Studies in Computational Intelligence*, pages 173–214. Springer, 2008.

E. Zitzler and L. Thiele. Multiobjective optimization using evolutionary algorithms - a comparative case study. In A. E. Eiben, T. Bäck, M. Schoenauer, and H. Schwefel, editors, *PPSN V*, pages 292–301. LNCS 1498, Springer Verlag, 1998.