# Multi-objective Monte-Carlo Tree Search

Weijia Wang, Michèle Sebag

JMLR: Workshop and Conference Proceedings 25:1–16, 2012 Asian Conference on Machine Learning

# 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*

**Editor:** Steven C.H. Hoi and Wray Buntine

## Abstract

Concerned with multi-objective reinforcement learning (MORL), this paper presents MO-MCTS, an extension of Monte-Carlo Tree Search to multi-objective sequential decision making. The known multi-objective indicator referred to as hyper-volume indicator is used to define an action selection criterion, replacing the UCB criterion in order to deal with multi-dimensional rewards. MO-MCTS is firstly compared with an existing MORL algorithm on the artificial Deep Sea Treasure problem. Then a scalability study of MO-MCTS is made on the NP-hard problem of grid scheduling, showing that the performance of MO-MCTS matches the non RL-based state of the art albeit with a higher computational cost.

**Keywords:** Monte-Carlo tree search, multi-objective optimization, sequential decision, hypervolume indicator

## 1. Introduction

Reinforcement learning (RL) (Sutton and Barto, 1998; Szepesvári, 2010) addresses sequential decision problems 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 best 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).

The goal of this paper is to extend MCTS to multi-objective sequential decision making. The proposed scheme called MO-MCTS basically aims at discovering several Pareto-optimal policies (decision sequences, or *solutions*) within a single tree. MO-MCTS only 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). The proposed approach relies on the hyper-volume indicator (Zitzler and Thiele, 1998) measuring the MOO quality of a set of solutions. Taking inspiration from (Auger et al., 2009), we use this indicator to define a single optimization objective for the current path being visited in each MCTS tree-walk, *conditioned on the other solutions* previously discovered. MO-MCTS thus handles a single-objective optimization problem in each tree-walk, while eventually discovering several decision sequences pertaining to the Pareto-front.

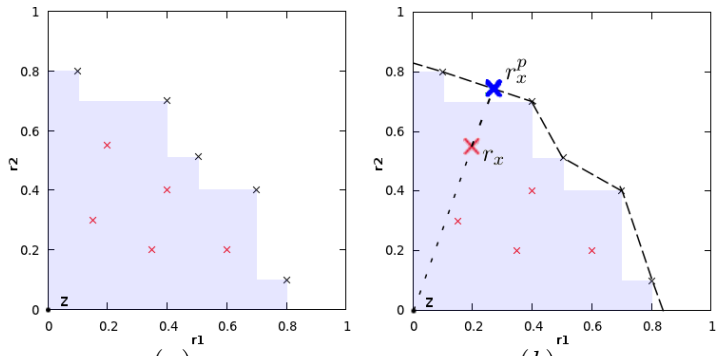
The experimental validation of the MO-MCTS approach considers two problems. Firstly, the performance of MO-MCTS is tested on the artificial Deep Sea Treasure problem for the sake of comparative evaluation with MORL algorithms. Secondly, the performance and scalability of MO-MCTS are also experimentally assessed on a real world application, the NP-hard problem of grid scheduling (Yu et al., 2008).

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

## 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 $\mathcal{A}$. Only deterministic environments will be considered in the following; the transition function ($tr : \mathcal{S} \times \mathcal{A} \mapsto \mathcal{S}$) gives the next state $tr(s, a)$ reached by executing action $a$ in state $s$. The (scalar) reward function is defined from the state $\times$ action space onto $\mathbb{R}$ ($r : \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}$).




(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 crosses. The hyper-volume 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 3.2).

### 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 $i$. 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 $A$.*

Notably, there is no natural total order on sets of points. Still, a total order on the sets of points *relatively to a reference point* has been proposed, based on the hyper-volume indicator (Zitzler and Thiele, 1998).

**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 hyper-volume 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 $\mu$ 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 hyper-volume indicator ($HV(A; z) = HV(P_A; z)$). As shown by Fleischer (2003), the hyper-volume 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 hyper-volume 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 hyper-volume indicator w.r.t $z$.

### 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 $a$ in node $s$ defines the child nodes $(s, a)$ of $s$; 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 cumulative 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 $c_e$. In a deterministic setting, the selection of the child node $(s, a)$ yields a single next state $tr(s, a)$, which replaces $s$ as current node. The bandit phase stops upon arriving in a leaf node of the tree.

The **tree building phase** takes place upon arriving in a leaf node $s$; some action $a$ is (uniformly or heuristically) selected and $tr(s, a)$ 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 $tr(s, a)$ 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 cumulative reward estimates in all nodes $(s, a)$ visited during the tree-walk:

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

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, $RAVE(a)$ 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[1], 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, it is clearly desirable to consider the best options as early as possible, despite the asymptotic consistency guarantees of MCTS (Berthier et al., 2010).

## 3. Overview of MO-MCTS

This section presents the MO-MCTS algorithm. The main difference between MCTS and MO-MCTS regards the node selection step. The challenge is to extend the single-objective node selection criterion (Eq. (1)) to the multi-objective setting.

### 3.1. From multi- to single-objective optimization

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. The hyper-volume indicator however makes it feasible to model multi-objective optimization as a set of (interdependent) single-objective optimization problems, as follows. Conditioned on a set $P \subset \mathbb{R}^d$, let us define the value of any $r$ in $\mathbb{R}^d$ as the hyper-volume indicator of $P \cup \{r\}$.

$$V(r) = HV(P \cup \{r\}; z)$$

By construction, any $r$ maximizing $V$ belongs to the Pareto front and offers some diversity w.r.t. $P$.

### 3.2. A scalar multi-objective value function

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.

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:

$$\overline{r}_{s,a} = \left( \hat{r}_{s,a;\ i} + \sqrt{c_i \ln(n_s)/n_{s,a}} \right)_{i=1}^{d} \tag{2}$$

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

Finally, an upper-bound $U(s, a)$ on the value of $(s, a)$ is given by considering the hyper-volume indicator of $\overline{r}_{s,a}$ w.r.t. archive $P$.

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

1. Another option is to use a dynamically weighted combination of the reward $\hat{r}_{s,a}$ and $RAVE(a)$ in Eq. (1), see e.g. (Gaudel and Sebag, 2010).

While $U(s,a)$ 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 $P$ (section 2.1).
A finer-grained value function must thus be defined. A straightforward option is to consider the so-called Pareto-rank of the vectorial rewards (Deb et al., 2000). This option however hardly scales up as it requires one to maintain the archive of all vectorial rewards ever evaluated, and the rank of every point in it with quadratic complexity in the size of the archive, that is the number $N$ of tree-walks.
Instead, 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 $\overline{r}_{s,a}$)[2]. 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:

$$W(s,a) = \begin{cases} U(s,a) & \text{if } \overline{r}_{s,a} \text{ is non-dominated in } P \\ U(s,a) - \| \overline{r}^p_{s,a} - \overline{r}_{s,a} \|_2 & \text{otherwise} \end{cases} \tag{3}$$

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. (3) sets a total order on all vectorial rewards in $\mathbb{R}^d$, where non-dominated points are ranked higher than dominated ones.

It is straightforward to see that the total order based on $W(s,a)$ is consistent with Pareto-domination: if $\overline{r}_{s,a}$ is dominated by $\overline{r}_{s',a'}$ then $W(s,a) < W(s',a')$. In the general case however, the order based on $W(s,a)$ is not necessarily consistent with the Pareto rank: As the Pareto front is not bound to make regular progress in the objective space, a point with low Pareto rank can be closer to the Pareto front in the objective space, than a point with higher Pareto rank.

### 3.3. MO-MCTS algorithm

MO-MCTS differs from MCTS in only three respects (Alg. 1). Firstly, the selected action $a^*$ now is the one maximizing value function $W(s,a)$ (Eq. (3) replacing Eq. (1)). Secondly, MO-MCTS maintains the archive $P$ of all non-dominated vectorial rewards evaluated in previous tree-walks by MO-MCTS. Upon arriving in a terminal state $u$, MO-MCTS evaluates the cumulative reward $r_u$ of the tree-walk. It then updates $\overline{r}_{s,a}$ as well as the (vectorial) $RAVE(a)$ for all nodes $(s,a)$ visited during the tree-walk, and it updates $P$ if $r_u$ is non-dominated. Thirdly, the RAVE vector is used to select the new node in the tree-building phase. Letting $s$ denote the current node and $a$ an admissible action in state $s$, letting $RAVE(a)$ denote the average vectorial reward associated to $a$, letting $RAVE^p(a)$ denote the perspective projection of $RAVE(a)$ on the Pareto front, then the action selected is the one minimizing

$$R(a) = \| RAVE^p(a) - RAVE(a) \|_2 \tag{4}$$

**Remark**. MO-MCTS theoretical analysis is hindered as value function $W$ dynamically depends on archive $P$ (Eq. (3)). However, the reward estimates do not undergo any

2. Another possibility would be to define $\overline{r}^p_{s,a}$ as the orthogonal projection of $\overline{r}_{s,a}$ onto $\mathcal{P}$. However, the use of orthogonal projection might have induced discontinuities of value function $W$ w.r.t. $\overline{r}_{s,a}$, since $\mathcal{P}$ is not necessarily convex.

**Algorithm 1** MO-MCTS

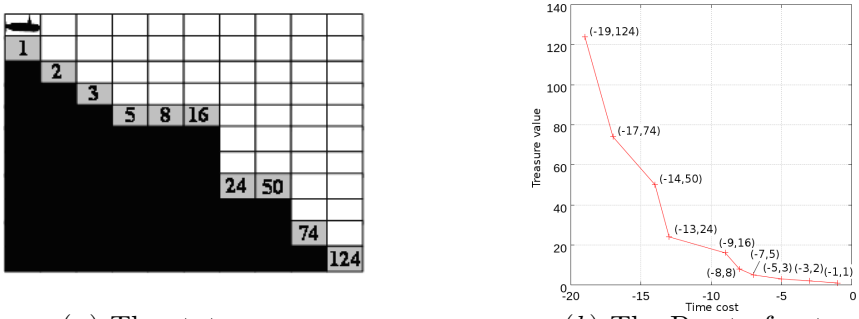
**MoMCTS**
**Input:** number $N$ of tree-walks
**Output:** search tree $\mathcal{T}$
Initialize $\mathcal{T} \leftarrow$ initial state, $P \leftarrow \{\}$
**for** $i = 1$ **to** $N$ **do**
  $r_u \leftarrow \text{TreeWalk}(\mathcal{T}, P,$ initial state $)$
  **if** $r_u$ is not dominated by any point in $P$ **then**
    Eliminate all points dominated by $r_u$ in $P$
    $P \leftarrow P \cup \{r_u\}$
  **end if**
**end for**

---

**TreeWalk**
**Input:** search tree $\mathcal{T}$, archive $P$, state $s$
**Output:** reward vector $r_u$
*//Test of the Progressive Widening condition, section 2.2*
**if** $s$ is not a leaf node, and $(\lfloor (n_s+1)^{1/b} \rfloor == \lfloor (n_s)^{1/b} \rfloor)$ **then**
  Select $a^* = \text{argmax}\ \{W(s,a), tr(s,a) \in \mathcal{T}\}$ // Eq. (3)
  $r_u \leftarrow \text{TreeWalk}(\mathcal{T}, P, tr(s,a^*))$
**else**
  $\mathcal{A}_s = \{$ admissible actions not yet visited in $s\}$
  Select $a^* = \arg\min\{R(a), a \in \mathcal{A}_s\}$ // Eq. (4)
  Add $tr(s,a^*)$ as child node of $s$
  $r_u \leftarrow \text{RandomWalk}(tr(s,a^*))$
**end if**
Update $n_s$, $n_{s,a^*}$, $RAVE(a^*)$ and $\hat{r}_{s,a^*}$
**return** $r_u$

---

**RandomWalk**
**Input:** state $u$
**Output:** reward vector $r_u$
$\mathcal{A}_{rnd} \leftarrow \{\}$ // store the set of actions visited in the random phase
**while** $u$ is not final state **do**
  Uniformly select an admissible action $a$ for $u$
  $\mathcal{A}_{rnd} \leftarrow \mathcal{A}_{rnd} \cup \{a\}$
  $u \leftarrow \text{tr}(u,a)$
**end while**
$r_u = evaluate(u)$ //obtain the reward vector of the tree-walk
Update $RAVE(a)$ for $a \in \mathcal{A}_{rnd}$
**return** $r_u$




(*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, grey 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.

distribution change *per se*; they are updated and the confidence increases with the number of tree-walks as in UCT. The only change affects the value function $W(s,a)$, which needs to be recomputed for all child nodes of every visited node when the Pareto front $P$ is updated.

Let $B$ denote the average branching factor in the MO-MCTS tree, and let $N$ denote the number of tree-walks. As each tree-walk adds a new node in the search tree, the number of nodes in the tree is $N+1$. The average length of a tree-path thus is in $\mathcal{O}(\log N)$. Depending on the number $d$ of objectives, the hyper-volume indicator is computed with complexity $\mathcal{O}(|P|^{d/2})$ 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.

MO-MCTS parameters are i) the total number of tree-walks $N$, ii) the exploration vs exploitation trade-off parameter $c_i$ for every $i$-th objective ; iii) the $b$ parameter used in the progressive widening heuristic; and iv) the reference point $z$.

## 4. Experimental validation

This section presents the experimental validation of MO-MCTS, with two goals in mind. The first goal of experiments is to assess the MO-MCTS performance comparatively to the state of the art in MORL (Vamplew et al., 2010). The study will consider the artificial Deep Sea Treasure (DST) problem (section 4.1).

The second goal of experiments is to assess the performance and scalability of MO-MCTS in a real-world setting, that of grid scheduling problems (section 4.2).

All reported results are averaged over 11 runs unless stated otherwise. The computational times are measured on a PC with Intel dual-core CPU 2.66GHz.

### 4.1. Deep Sea Treasure

The Deep Sea Treasure (DST) problem was firstly introduced by Vamplew et al. (2010). The state space of DST consists of a $10 \times 11$ grid (Fig. 2($a$)). The action space of DST includes four actions (up, down, left and right), each deterministically sending the agent to one adjacent square in the indicated direction, except when the agent would cross the border line of the grid or touch the sea floor, in which case the agent keeps 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 (depicted in Fig. 2($a$)). The 10 non-dominated vectorial rewards in the form of (-time, treasure) are depicted in the two-dimensional plane in Fig. 2($b$), forming a non-convex Pareto front.

#### 4.1.1. 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. Numerous multiple-policy MORL algorithms have been proposed (Natarajan and Tadepalli (2005); Tesauro et al. (2007); Barrett and Narayanan (2008)) using the weighted sum of the objectives (with several weight settings) as scalar reward, which is optimized with 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, MO-MCTS is compared to Multi-Objective Q-Learning (MOQL) (Vamplew et al. (2010)). Choosing MOQL as baseline is motivated as it is able to yield 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 $r_i = (1 - \lambda_i) \times r1 + \lambda_i \times r2$, where $0 \leq \lambda_i \leq 1, i = 1, 2, \ldots, m$ define the $m$ weight settings of MOQL, and $r1$ (respectively $r2$) 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 $\{r_{1,j}, r_{2,j}, \ldots, r_{m,j}\}$ all together compose the Pareto front of MOQL at training phase $j$.

#### 4.1.2. Experimental setting

The DST problem is concerned with minimizing the search time (maximizing its opposite) and maximizing the treasure value. Accordingly, the reference point used in the hypervolume indicator calculation is set to (-100,0).

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

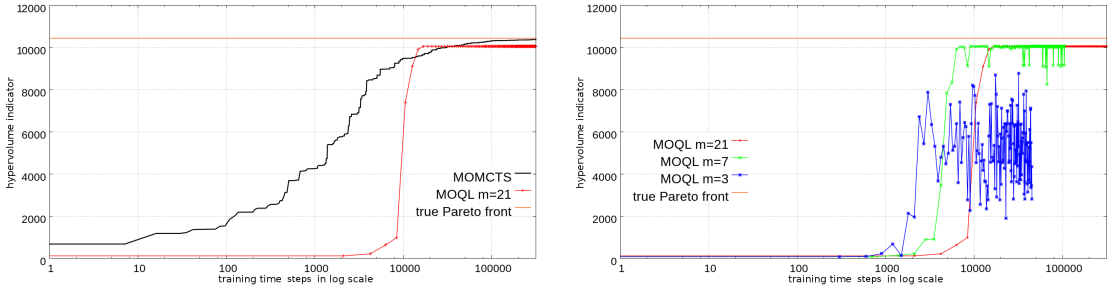
- $\epsilon$-greedy exploration is used with $\epsilon = 0.1$.
- Learning rate $\alpha$ is set to 0.1.
- The state-action value table is optimistically initialized ($time = 0, treasure = 124$).
- Due to the episodic nature of DST, no discounting is used in MOQL ($\gamma = 1$).
- The number $m$ of weight settings ranges in $\{3, 7, 21\}$, with $\lambda_i = \frac{i-1}{m-1}, i = 1, 2, \ldots, m$.
- Training time is set to 15,000 time steps for each weight setting, totaling 45,000 time steps for $m$ = 3-MOQL, 105,000 times steps for $m$ = 7-MOQL, and 315,000 time steps for $m$ = 21-MOQL.
- The number of training phases $n_{tr}$ is set to 150.

MO-MCTS parameters include the parameter $b$ used in progressive widening, and the exploration vs. exploitation trade-off parameters associated to the time cost objective and

the treasure value objective. After a few preliminary experiments, $b$ is set to 2, $c_{time}$ is set to 20,000 and $c_{treasure}$ is set to 150. The overall training time for MO-MCTS is 315,000 time steps, ca 15,600 tree-walks.

The performance is reported as the hyper-volume indicator (averaged over 11 independent runs with the same experimental setting), and the Pareto front of the MO-MCTS run with median hyper-volume indicator, in the time×treasure plane.

#### 4.1.3. Results



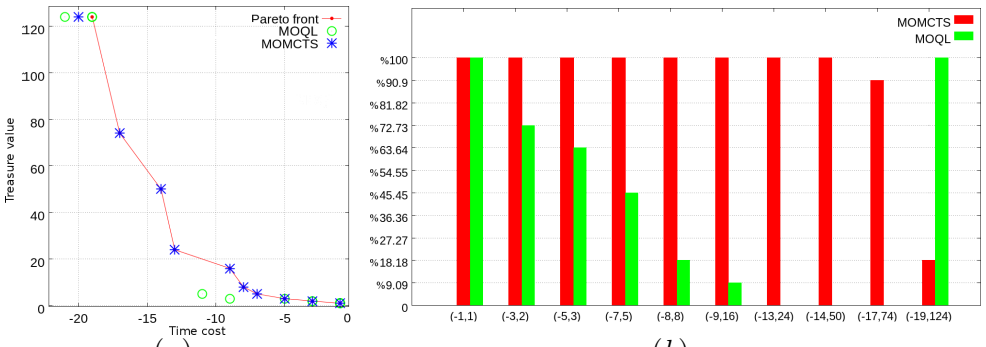

$(a)$ $(b)$

Figure 3: Left: The hyper-volume indicator of MO-MCTS and $m = 21$-MOQL. Right: The hyper-volume indicator of MOQL for $m = 3, 7, 21$. All results, averaged over 11 independent runs, are reported vs the number of training time steps.

Fig. 3 displays the hyper-volume indicator performance of MO-MCTS and that of MOQL for $m = 3, 7, 21$. It is observed that for $m = 7$ or 21, MOQL reaches a performance plateau (10062) within 20,000 time steps. MOQL does not reach the optimal hyper-volume indicator (10455) as it is not asymptotically consistent. This is explained by the fact that the Pareto front of DST is not convex. As is widely known (Deb, 2001), linear-scalarisation based approaches of MOO fail to discover solutions in non-convex regions of the Pareto front. Actually, the plateau of MOQL corresponds to the discovery of the extreme points (-19,124) and (-1,1) of the Pareto front, as confirmed by Fig. 4($a$). In the meanwhile, MO-MCTS improves its performance as time increases, and it reaches a better hyper-volume indicator than MOQL after 40,000 time steps. Note that each time step in MO-MCTS is computationally heavier than for MOQL (for 315,000 time steps, MO-MCTS takes 82 secs versus 27 secs for MOQL).

Fig. 3($b$) shows the influence of $m$ on MOQL. For $m = 7$, MOQL reaches the performance plateau before MOQL for $m = 21$ (respectively 8,000 time steps vs 20,000 time steps). However, Fig. 3($b$) shows the instability of the average hyper-volume indicator for $m = 7$, and shows that the instability increases as $m$ decreases to 3. The fact that for $m = 3$ MOQL fails to reach the MOQL performance plateau is explained as the extreme point (-19,124) can be missed in some runs as the discount parameter $\gamma$ is 1 (by consistency with Vamplew et al. (2010)). Therefore the 124 treasure might be discovered later than in time 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 MO-MCTS and MOQL for $m = 21$ is displayed in Fig. 4($b$). This picture shows that MOQL can discover




(a) (b)

Figure 4: Left: The vectorial rewards found by representative MO-MCTS and MOQL runs. For MOQL, the non-extreme vectorial rewards were only temporarily present (see text). Right: The percentage of times out of 11 runs that each non-dominated vectorial reward was discovered by MO-MCTS and MOQLfor $m = 21$, during at least one test episode.

non-dominated vectorial rewards during intermediate test episodes, but eventually discards them. Quite the contrary, MO-MCTS discovers all points in the Pareto front, and keeps them in the search tree after they have been discovered. The weakness of MO-MCTS is that the longest decision sequences corresponding to the vectorial rewards (-17,74) and (-19,124) need more time to be discovered. Nevertheless, MO-MCTS discovers all non-dominated vectorial rewards. The consistency study of MO-MCTS is left for further work.

In summary, on the artificial DST problem, the empirical validation reveals the limitations of both MOQL and MO-MCTS. As was expected, MOQL does not and can not find vectorial rewards lying in the non-convex regions of the Pareto front. The experimental validation also reveals MO-MCTS limitations in discovering the longest decision sequences in competitive time.

### 4.2. Grid scheduling

Pertaining to the domain of autonomic computing (Tesauro et al., 2007), the problem of grid scheduling has been selected to establish a proof on the scalability of MO-MCTS. 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 is concerned with scheduling $J$ tasks on $m$ distinct computational resources with different unit time costs and processing capabilities. As tasks are interdependent and resources are heterogeneous, job scheduling is 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 4.2, two objectives will be considered, the makespan and the cost of the solution [3].

3. As both objectives in the grid scheduling problem are to be minimized, the *plus* sign in the r.h.s. of Eq. (2) is changed to *minus* sign, in order to provide duly optimistic estimates of the objective rewards.

#### 4.2.1. 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 hyper-volume indicator (in SMS-EMOA) are selected and undergo unary or binary stochastic perturbations.

#### 4.2.2. Experimental setting

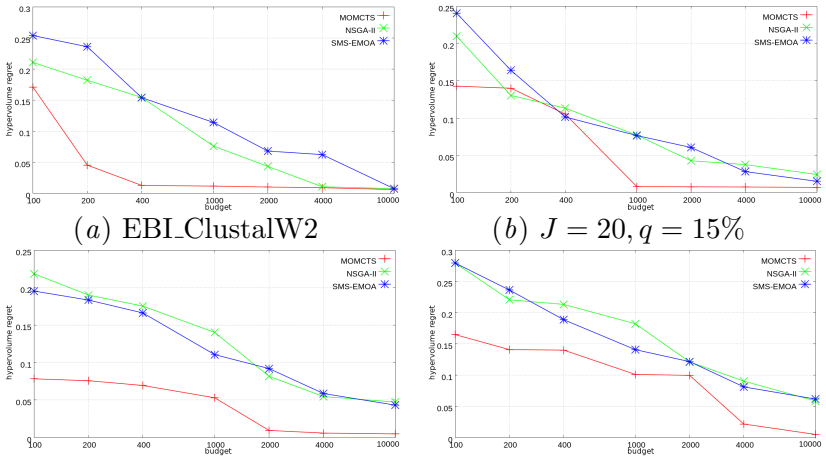
A simulated grid environment containing 3 resources with different unit time costs and processing capabilities ($cost_1 = 20, speed_1 = 10; cost_2 = 2, speed_2 = 5; cost_3 = 1, speed_3 = 1$) is defined. We firstly compare the performance of MO-MCTS and baseline algorithms on a realistic bio-informatic workflow *EBI_ClustalW2*, which performs a ClustalW multiple sequence alignment using the EBI's WSClustalW2 service[4]. This workflow contains 21 tasks and 23 precedence pairs (graph density $q = 12\%$), assuming that all workloads are equal. Secondly, the scalability of MO-MCTS 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 on grid scheduling problem. A simple heuristic which allocates the task with maximum (respectively minimum) expected finish time to the fastest (resp. slowest) resource is implemented in the random phase.

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 $b$ in MO-MCTS is set to 4. The exploration vs exploitation trade-off parameters associated to the makespan and the cost objectives, $c_{time}$ and $c_{cost}$, are both set to $5 \times 10^{-3}$. 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.

The performance indicator is the difference between the actual Pareto front found in the run, and the reference Pareto front $P^*$ gathering all non-dominated vectorial rewards obtained in all runs of all three algorithms. The run with Pareto front $P$ is assessed from its hyper-volume indicator regret w.r.t. the reference Pareto front (the smaller, the better):

$$F(P) = HV(P^*; z) - HV(P; z)$$




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

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

Figure 5: Regret of the hyper-volume indicator versus number $N$ of evaluations of MO-MCTS, NSGA-II and SMS-EMOA on (a): EBI_ClustalW2; (b)(c)(d): artificial problems with number of tasks $J$ and graph density $q$.

#### 4.2.3. Results

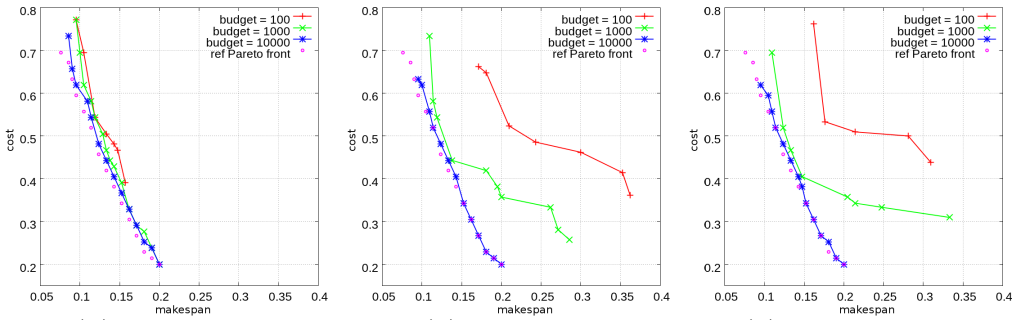
Fig. 5 displays the comparative hyper-volume indicator regrets of MO-MCTS, NSGA-II and SMS-EMOA on EBI_ClustalW2 workflow scheduling and on artificial jobs with a number $J$ of tasks ranging in $20, 30$ and $40$ with graph density $q = 15\%$. Fig. 6 shows the Pareto front discovered by MO-MCTS, NSGA-II and SMS-EMOA after $N = 100, 1000$ and $10000$ policy evaluations (tree-walks), comparatively to the reference Pareto front. On all considered problems regardless of their size, MO-MCTS outperforms both NSGA-II and SMS-EMOA throughout the search process. The good performance of MO-MCTS is explained as MO-MCTS discovers solutions in the low cost and small makespan regions, much earlier than NSGA-II and SMS-EMOA. More generally, MO-MCTS tends to explore more widely the Pareto front than NSGA-II and SMS-EMOA. In counterpart, MO-MCTS performances are slightly worse in the lower region of the Pareto front, than that of NSGA-II and SMS-EMOA. Overall, the main weakness of MO-MCTS is its computational runtime, ca 5 times higher than that of NSGA-II and SMS-EMOA[5]. Such disadvantage can be alleviated when the evaluation time of one strategy is far more than the time of one tree-walk in MO-MCTS.

## 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 (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. Natara-

4. The complete description is available at *http://www.myexperiment.org/workflows/203.html*.
5. On workflow EBI_ClustalW2, the average execution time of MO-MCTS, NSGA-II and SMS-EMOA are respectively 142 secs, 31 secs and 32 secs.




(a) MoMCTS (b) NSGA-II (c) SMS-EMOA

Figure 6: Progression of the Pareto-optimal solutions found for $N = 100,\ 1000$ and 10000 for MO-MCTS, NSGA-II and SMS-EMOA on the EBI_ClustalW2 workflow. The reference Pareto front is indicated by circles.

jan 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 setting 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, one 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 a higher complexity in each value iteration: the $\mathcal{O}(|S|^2|A|)$ 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.

Interestingly, MO-MCTS likewise maintains vectorial reward estimates for all visited nodes with complexity $\mathcal{O}(B|P|^{d/2}logN)$ in the $N$-th tree-walk, which is lower than that of one value iteration in (Barrett and Narayanan, 2008), considering that the size of $P$ is of the same order of magnitude as the number $n$ of non-dominated $Q$ vectors.

In summary, the main strength of MO-MCTS is its ability to discover policies lying in the non-convex regions of the Pareto front. To our knowledge[6], this feature is unique in the MORL literature. On the negative side, MO-MCTS requires some domain knowledge when solving large scale problems (e.g. grid scheduling). Further, it suffers from a higher computational cost compared to stochastic multi-objective optimization algorithms NSGA-II and SMS-EMOA. The average complexity of generating one solution in NSGA-II is $\mathcal{O}(d\lambda)$, where $\lambda$ is the population size; the complexity of SMS-EMOA is $\mathcal{O}(\lambda^{d/2})$. Both algorithms use a fixed size Pareto archive. On the contrary, MO-MCTS with a complexity of $\mathcal{O}(B|P|^{d/2}logN)$ pays a logarithmic price in the number $N$ of tree-walks.

6. 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 $\epsilon$-approximated in time polynomial in $\epsilon$. 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 $\epsilon$-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 $\epsilon$-Pareto-dominated (Thm. 2, page 4, Papadimitriou and Yannakakis (2000)).

## 6. Conclusion and perspectives

The multi-objective extension of MCTS presented in this paper has been validated on two problems : Deep Sea Treasure (DST) and grid scheduling. Compared to the linear-scalarisation based MORL approaches, which fail to discover vectorial rewards in the non-convex regions of the Pareto front, MO-MCTS solves DST and experimentally converges to the Pareto front. In the real-world grid scheduling problem, MO-MCTS yields comparable performances to the state of the art at the price of a higher computational cost. As grid scheduling is an extensively studied problem with major industrial impacts, it might thus be considered promising that MO-MCTS first results match the state of the art of the domain.

Only deterministic settings have been considered in the paper. On-going experiments, investigating the impact of stochastic transition models, show that the MO-MCTS performance degrades gracefully as the stochasticity of the environment increases.

As mentioned, a major perspective for further work is to establish the consistency of MO-MCTS. Another perspective is to refine the RAVE heuristics (e.g. considering the context of task allocation in grid scheduling), and reduce the computational cost, through systematically pruning the archive. Other measures similar to the hyper-volume indicator, e.g. the logarithmic hyper-volume indicator investigated by Friedrich et al. (2011) will also be considered.

## Acknowledgments

We wish to thank Ilya Loshchilov, Jean-Baptiste Hoock, Dawei Feng, Romaric Gaudel, and Julien Perez for many discussions on UCT, MOO and MORL. We also warmly thank the anonymous reviewers for their many remarks and critiques on an earlier 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 $\mu$-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.

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.

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.

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.

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

T. Friedrich, K. Bringmann, T. Voß, and C. Igel. The logarithmic hypervolume indicator. In *FOGA'11*, pages 81–92, 2011.

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

R. Gaudel and M. Sebag. Feature selection as a one-player game. In *ICML'10*, pages 359–366. Omnipress, 2010.

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

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.

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.

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.