Optimized look-ahead tree search policies

Francis Maes, Louis Wehenkel, and Damien Ernst

University of Liège
Dept. of Electrical Engineering and Computer Science
Institut Montefiore, B28, B-4000, Liège - Belgium

Abstract. We consider in this paper look-ahead tree techniques for the discrete-time control of a deterministic dynamical system so as to maximize a sum of discounted rewards over an infinite time horizon. Given the current system state x_t at time t, these techniques explore the look-ahead tree representing possible evolutions of the system states and rewards conditioned on subsequent actions u_t, u_{t+1}, ... When the computing budget is exhausted, they output the action u_t that led to the best found sequence of discounted rewards. In this context, we are interested in computing good strategies for exploring the look-ahead tree. We propose a generic approach that looks for such strategies by solving an optimization problem whose objective is to compute a (budget compliant) tree-exploration strategy yielding a control policy maximizing the average return over a postulated set of initial states.
This generic approach is fully specified to the case where the space of candidate tree-exploration strategies are "best-first" strategies parameterized by a linear combination of look-ahead path features – some of them having been advocated in the literature before – and where the optimization problem is solved by using an EDA-algorithm based on Gaussian distributions. Numerical experiments carried out on a model of the treatment of the HIV infection show that the optimized tree-exploration strategy is orders of magnitudes better than the previously advocated ones.

Keywords: Real-time Control, Look-ahead Tree Search, Estimation of Distribution Algorithms

1 Introduction

Many interesting problems in the field of engineering and robotics can be casted as optimal control problems for which one seeks to find policies optimising a sum of discounted rewards over an infinite time horizon. Among the various approaches that have been proposed in the literature to solve these problems when both the system dynamics and the reward function are assumed to be known, a quite recent one has caught our attention. In this approach, which was first published in [4], a new type of policy relying on look-ahead trees was proposed. The key idea in this paper is to explore, at any control opportunity t, in an optimistic way the look-ahead tree starting from the current state x_t and whose branchings are determined by the possible sequences of control actions

$u_t, u_{t+1}, \ldots$. More specifically, this strategy expands at every iteration a leaf of the tree for which the discounted sum of rewards collected from the root to this leaf (u-value) plus an optimistic estimation of the future discounted rewards is maximal, and when the computing budget is exhausted it returns the action u_t (or the sequence of actions $u_t, u_{t+1}, \ldots$) leading to the leaf with the best u-value. The results reported by the authors are impressing: with rather small trees they managed to control in an efficient way quite complex systems.

We tested this approach on some even more complex benchmark problems than those considered in [4], and the results were still excellent. This raised the following question in our mind: while this *problem independent* tree-exploration strategy yields excellent results, would it be possible to obtain better results by computing automatically a *problem dependent* look-ahead tree-exploration strategy? From there came the idea of looking within a broader family of tree-exploration strategies for a best one for a given optimal control problem and a given real-time computing budget. This led us to the main contribution presented in this paper, namely the casting of the search for an optimal tree-exploration strategy as an optimization task exploiting prior knowledge about the optimal control problem and computing budget available for real-time control.

The present work has also been influenced by other authors which have sought to identify good tree/graph exploration strategies for other decision problems than the one tackled here but for which the solution could also be identified in a computationally efficient way by exploring in a clever way a tree/graph structure. One of the most seminal works in this field is the A* algorithm [3] that uses a best-first search to find a shortest path towards a goal state. In A*, the function used for evaluating the nodes is the sum of two terms: the length of the so far shortest path towards the current node and an optimistic estimate of the shortest path from this node to a goal state (a so-called "admissible" heuristic). Note the close connection that exists between the optimistic planning algorithm of [4] and the A* algorithm even if they apply to different types of planning problems. Several authors have sought to learn good admissible heuristics for the A* algorithm. For example, we can mention the LRTA* algorithm [5] which is a variant of A* which is able to learn over multiple trials an optimal admissible heuristic. It has been shown later on, in the work on real-time dynamic programming from Barto and al. [1], that LRTA* has a behaviour similar to the asynchronous value iteration algorithm. More recent works for learning strategies to efficiently explore graphs have focused on the use of supervised regression techniques using various approximation structures to solve this problem (e.g., linear regression, neural networks, k-nearest neighbours) [7, 8, 10]). To position the contribution of this paper with respect to this large body of work, we may say that, to the best of our knowledge, this paper is the first one where the search for a tree-exploration based control strategy is explicitly casted as an optimization problem which is afterwards solved using a derivative-free optimisation algorithm.

The rest of this paper is organized as follows. In Section 2, we specify the class of control problems that we consider and present the look-ahead tree-exploration approach in this context. Section 3 presents our framework for optimizing look-ahead tree-exploration strategies and fully specifies an algorithm for solving this

problem in a near-optimal way. This algorithm is compared in Section 4 with several other tree-exploration methods. Finally, Section 5 concludes and presents further research directions.

2 Problem formulation

We start this section by describing the type of optimal control problem that will be considered throughout this paper. Afterwards, we describe look-ahead tree exploration based control policies as well as the specific problem we address.

2.1 Optimal control problem

We consider the general class of deterministic time-invariant dynamical systems whose dynamics is described (in discrete-time) by:

$$x_{t+1} = f(x_t, u_t) \quad t = 0, 1, \ldots \tag{1}$$

where for all t, the state x_t is an element of the state space X and the action u_t is an element of the action space U. We assume in this paper that U is finite (and not too large) but we do not restrict X.

To the transition from t to $t+1$ is associated an instantaneous reward $\rho(x_t, u_t) \in \mathbb{R}^+$ and, for every initial state x_0, we define the (infinite horizon, discounted) return of a (stationary) control policy $\mu : X \to U$ as:

$$J^\mu(x_0) = \lim_{T \to \infty} \sum_{t=0}^{T} \gamma^t \rho(x_t, u_t) \tag{2}$$

subject to $u_t = \mu(x_t)$ and $x_{t+1} = f(x_t, u_t)\ \forall t > 0$, and where $\gamma \in [0, 1[$ is a discount factor. Within the set $\mathcal{U}$ of stationary policies, we define optimal policies $\mu^* \in \mathcal{U}$ such that, for every initial state $x_0 \in X$, we have:

$$J^{\mu^*}(x_0) \geq J^\mu(x_0), \forall \mu \in \mathcal{U}. \tag{3}$$

In this paper, we are interested in finding a good approximation of an optimal policy in the sense that its return is close to J^{μ^*}.

2.2 Look-ahead tree exploration based control policies

The control policies considered in this paper are the combination of two components: an *exploration* component that exploits the knowledge of f and ρ to generate a look-ahead tree representing the evolution of the system states and the rewards observed given various sequences of actions $u_t, \ldots, u_{t+n}$, and a *decision* component that exploits this information to select the action u_t. The concept of look-head tree is represented on Figure 1.

In the following, we denote look-ahead trees by τ. A look-ahead tree is composed of nodes $n \in \tau$ that are triplets (u_{t-1}, r_{t-1}, x_t) where $u_{t-1} \in U$ is an

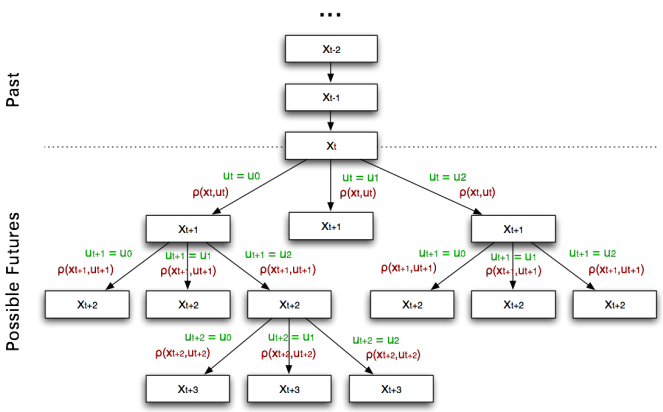

Fig. 1. Example of look-ahead tree in a system with three possible control actions. Each node of the tree is associated to an action u, a reward value ρ and a state x.

action whose application on the parent's node state x_{t-1} leads to the reward $r_{t-1} = \rho(x_{t-1}, u_{t-1})$ and the successor state $x_t = f(x_{t-1}, u_{t-1})$. We denote by $parent(n)$ the parent node of $n \in \tau$. When clear from the context, we use $r(n)$ (*resp.* $x(n)$) to denote the reward (*resp.* state) attached to node n. Internal nodes of a look-ahead tree are said to be *expanded (or closed)* while its leaves are *open*. An expansion of an open node consists in adding to this node a number $|U|$ of children corresponding to the different actions u that can be applied to the state x of this node. We use $path(n)$ to denote the complete sequence of nodes from the root node n_0 to a node n at depth $d(n)$. Such a path corresponds to a trajectory $(x_t, u_t, r_t, x_{t+1}, \ldots, u_{t+d(n)-1}, r_{t+d(n)-1}, x_{t+d(n)})$ of the system.

A typical example of look-ahead tree search algorithm is the "uniform tree search" (or breadth-first) algorithm, that first develops a tree representing the evolutions of the system for all sequences of actions of length d, and then selects as action u_t the first action of a path of the tree along which the highest sum of discounted cumulated rewards ($\sum_{t'=t}^{t+d-1} \gamma^{t'} r_{t'}$) is collected. The computing budget necessary for developing a uniform tree of depth d is $\mathcal{O}(|U|^d)$, which may often lead to the case where for a given computational budget the tree cannot be developed up to a depth which is sufficient to obtain a well performing policy. This is often true in the context of real-time control of a system and is a reason for looking for other types of look-ahead tree search policies.

2.3 Budget constrained path-scoring based tree exploration

From now on, we will mainly focus on a particular subclass of look-ahead tree search policies which is easy to implement, works well in an anytime setting and is sufficiently rich. Every policy of this subclass is fully defined by a scoring function $h(\cdot)$ (with values $\in \mathbb{R}$) that assigns scores to look-ahead search tree paths. These policies develop the tree incrementally in such a way that they always fully expand the leaf whose corresponding path has the largest score.

They stop the tree development when the computational budget is exhausted, in which case they output as action u_t, the first action taken along a path in the

Input: the current state of the system x_t
Input: path scoring function $h(\cdot) \in \mathbb{R}$
Output: an action u_t
Set $n_0 = (\emptyset, \emptyset, x_t)$ *(root node)*
Set $\tau = \{n_0\}$ *(initial tree)*
Set $v(n_0) = 0$
repeat
  Select a terminal node $n = \operatorname{argmax} v(n)$
  foreach $u \in U$ **do**
    Add child node $n' = \big(u, \rho(x(n), u), f(x(n), u)\big)$ to n
    Set $v(n') = h(path(n'))$
  end
until *computational budget is exhausted*
return *the first action of a path towards an open node n and that maximizes the discounted cumulated rewards $\sum_{t'=t}^{t+d(n)-1} \gamma^{t'-t} r_{t'}$ (u-value)*

Algorithm 1: Generic budget constrained tree exploration based control

tree along which the highest sum of discounted rewards (u-value) is observed. A generic tabular representation of this path-scoring look-ahead tree control policy is given by Algorithm 1, where $v(n)$ denotes the score associated to node n using the scoring function $h(\cdot)$.

Note that the uniform tree search algorithm discussed previously is a particular case of Algorithm 1, using heuristic $h^{mindepth}(path(n)) = -d(n)$. The approach proposed by [4] for problems with bounded rewards is also a particular case of path-scoring based look-ahead tree search with heuristic $h^{optimistic}(path(n)) = \sum_{t'=t}^{t+d(n)-1} \gamma^{t'-t} r_{t'}(n) + B_r \frac{\gamma^d}{1-\gamma}$, where $\{r_{t'}(n)\}_{t'=t}^{t+d(n)-1}$ are the rewards collected along the path leading to n.

3 Optimized look-ahead tree exploration based control

In this section we present a generic approach for constructing optimal budget constrained look-ahead tree exploration based control policies, and we then describe the fully specified instance of this approach that is tested in Section 4.

3.1 Generic optimized look-ahead tree exploration algorithm

Let $\mu^h(x)$ denote the look-ahead control policy described by Algorithm 1 with a given scoring function $h \in \mathcal{H}$. For identifying a good scoring function $h(\cdot)$, we propose to proceed as follows. First, we define a rich enough set of candidate scoring functions $\mathcal{H}$. Second, we select a subset X_0 of X in a way that it "covers well" the area from which the system is likely to be controlled. Finally we pose the following optimization problem:

$$\operatorname*{argmax}_{h \in \mathcal{H}} R_{X_0}(\mu^h)$$

where $R_{X_0}(\mu^h)$ is the mean return of policy μ^h when starting from states in X_0, *i.e.* $R_{X_0}(\mu^h) = \frac{1}{|X_0|}\sum_{x_0 \in X_0} J^{\mu^h}(x_0)$. An algorithm solving this problem yields optimized look-ahead tree policies adjusted to the problem at hand (dynamics, rewards, and budget constraints).

3.2 A particular instance

We now instantiate our generic approach for the particular case of linear functions of path features optimized by an estimation of distribution algorithm.

Set of candidate path scoring functions. In our study, we consider a generic set of candidate path scoring functions $\mathcal{H}^\phi$ that are using a vector function $\phi(\cdot) \in \mathbb{R}^p$ of *path features*. Such features are chosen beforehand and they may describe any aspect of a path such as cumulated rewards, depth or states. In the next section, we give an example of a generic $\phi(\cdot)$ that can be applied to any optimal control problem with continuous state variables.

Scoring functions $h^\phi_\theta \in \mathcal{H}^\phi$ are defined as parameterized linear functions: $h^\phi_\theta(path(n)) = \langle \theta, \phi(path(n)) \rangle$, where $\theta \in \mathbb{R}^p$ is a vector of parameters and $\langle \cdot, \cdot \rangle$ is the dot-product operator. Each value of θ defines a candidate tree-exploration strategy h^ϕ_θ; we thus have $\mathcal{H}^\phi = \{h^\phi_\theta\}_{\theta \in \mathbb{R}^p}$.

Optimization problem. Given our candidate set of tree-exploration strategies, the optimization problem can be reformulated as follows:

$$\underset{\theta \in \mathbb{R}^p}{\text{argmax}}\, R_{X_0}(\theta) \text{ with } R_{X_0}(\theta) = \frac{1}{|X_0|} \sum_{x_0 \in X_0} J^{\mu^{h^\phi_\theta}}(x_0) \tag{4}$$

Since $J^{\mu^{h^\phi_\theta}}(x_0)$ involves an infinite sum, it has to be approximated. This can be done with arbitrary precision by truncating the sum with a sufficiently large *horizon* limit. Given a large enough horizon H, we hence approximate it by:

$$J^\mu(x_0) \approxeq \sum_{t=0}^{H-1} \gamma^t \rho(x_t, u_t) | u_t = \mu(x_t), x_{t+1} = f(x_t, u_t). \tag{5}$$

Optimization algorithm. To solve Equation 4, we suggest the use of derivative-free global optimization algorithms, such as those provided by metaheuristics. In this work, we used a powerful, yet simple, class of metaheuristics known as *Estimation of Distribution Algorithms* (EDA) [6]. EDAs rely on a probabilistic model to describe promising regions of the search space and to sample good candidate solutions. This is performed by repeating iterations that first *sample* a population of N candidates using the *current* probabilistic model and then *fit* a *new* probabilistic model given the $b < N$ best candidates. Any kind of probabilistic model may be used inside an EDA. The simplest form of EDAs uses one marginal distribution per variable to optimize and is known as the *univariate marginal distribution algorithm* [9]. We have adopted this approach that, although simple, proves to be quite effective to solve Equation 4.

Our EDA algorithm proceeds as follows. There is one Normal distribution per parameter $f \in \{1, 2, \ldots, p\}$. At first iteration, there distributions are initialized as standard Normal distributions. At each iteration, N candidates are sampled using the current distributions and evaluated. At the end of the iteration, the p distributions are re-estimated using the $b < N$ best candidates of current iteration. The policy that is returned corresponds to the θ parameters that led to the highest observed value of $R_{X_0}(\theta)$ after a number i_{max} of EDA iterations.

4 Experiments

We describe numerical experiments comparing optimized look-ahead tree policies against previously proposed look-ahead tree policies on a toy problem and on a much more challenging one that is of some interest to the medical community.

4.1 Path features function

Our optimization approach relies on a path feature function $\phi(\cdot) \in \mathbb{R}^p$, that, given a path in the look-ahead search tree, outputs a vector of features describing this path. The primary goal of the $\phi(\cdot)$ function is to project paths of varying length into a fixed-dimension description. Both global properties (e.g. depth, cumulated rewards, actions histogram) and local properties (e.g. state contained in the terminal node, last actions) may be used inside $\phi(\cdot)$.

We propose below a particular $\phi(\cdot)$ function that can be used for any control problem with continuous state variables, *i.e.* for which $X \subset \mathbb{R}^m$. Despite its simplicity, this function lead to optimized look-ahead tree policies performing remarkably well in our test-beds.

Our feature function is motivated by the fact that two highly relevant variables to look-ahead tree search are *depth* and *reward*. In a sense, penalizing or favoring *depth* enables to trade-off between breadth-first and depth-first search and acting on *reward* enables to trade-off between greedy search and randomized search. Since the best way to perform search may crucially depend on the current state variables, we propose to combine *depth* and *reward* with *state* variables. This is performed in the following way:

$$\begin{aligned}\phi^{simple}(n_0, \ldots, n_d) = \big(&x_1(n_d), \ldots, x_m(n_d),\\ &dx_1(n_d), \ldots, dx_m(n_d),\\ &r(n_d)x_1(n_d) \ldots, r(n_d)x_m(n_d)\big)\end{aligned}$$

where d is the depth, *i.e.* the length of path $n_0, \ldots, n_d$, and $r(n_d)$ is the last reward perceived on the path $n_0, \ldots, n_d$. If the dimension of state space is m, ϕ^{simple} computes vectors of $p = 3m$ features. Note that these features mostly depend on the leaf n_d and only capture global information of the path through the depth d. Of course, several other feature functions that better exploit global information of the path could be used, but ϕ^{simple} already provides a basis to construct very well performing policies as shown below.

4.2 Baselines and parameters

In the following, we consider the look-ahead tree policies defined by the following path scoring tree-exploration strategies:

$$h^{mindepth}(n_0,\ldots,n_d) = -d \qquad h^{optimistic}(n_0,\ldots,n_d) = \sum_{i=0}^{d} \gamma^i r(n_i) + B_r \frac{\gamma^d}{1-\gamma}$$

$$h^{greedy1}(n_0,\ldots,n_d) = r(n_d) \qquad h^{greedy2}(n_0,\ldots,n_d) = \gamma^d r(n_d)$$

$$h^{optimized}(n_0,\ldots,n_d) = \left\langle \theta^*, \phi^{simple}(n_0,\ldots,n_d) \right\rangle$$

$h^{mindepth}$ corresponds to *uniform tree search* (a.k.a. breadth-first search). $h^{optimistic}$ is the heuristic proposed by [4]. $h^{greedy1}$ and $h^{greedy2}$ are greedy look-ahead tree exploration strategies *w.r.t.* immediate rewards.

To find the θ^* parameter, we use the EDA algorithm with $i_{max} = 50$ iterations, $N = 100$ candidates per iteration and $b = 10$ best candidates; one full optimization thus involves 5000 look-ahead tree exploration strategy evaluations.

4.3 Synthetic problem

To compare optimized look-ahead tree policies with previously proposed ones, we adopt the synthetic optimal control problem also used in [4]. In this problem, a state is composed of a position y and a velocity v and there are two possible control actions: $U = \{-1, +1\}$. The dynamics are as follows:

$$(y_{t+1}, v_{t+1}) = (y_t, v_t) + (v_t, u_t)\Delta t \qquad \rho((y_t, v_t), u_t) = \max(1 - y_{t+1}^2, 0)$$

where $\Delta t = 0.1$ is the time discretization step. In this problem, the reward is obviously upper bounded by $B_r = 1$, and this bound is therefore used in the optimistic heuristic in our simulations, as in [4].

The set of initial states X_0 used for optimizing the exploration policy is composed of 10 initial states randomly sampled from the domain $[-1, 1] \times [-2, 2]$, and the discount factor is $\gamma = 0.9$. The evaluation of the average return by the EDA algorithm is truncated at horizon $H = 50$.

We are interested in the mean return values $R_{X_0}(\mu^h)$ for different amounts of computational resources. The *budget* is the number of node expansions allowed to take one decision. For a given budget B, the policy expands B nodes and computes $B|U|$ transitions and values for the nodes. As in [4], we consider values $B = 2^{d+1} - 1$ with $d \in \{1, \ldots, 18\}$, corresponding to complete trees of depth d. We optimize one tree-exploration strategy per possible value of budget B, *i.e.* tree-exploration strategies are specific to a given computational budget.

The results are reported on the left part of Figure 2. It can be seen that the optimized tree-exploration strategy performs significantly better on X_0 than all other tree-exploration strategies, in the whole range of budget values. In particular, it reaches a mean discounted return of 7.198 with a budget $B = 63$, which is still better than the return of 7.179 of the best-performing tree-exploration strategies with the maximal budget $B = 524287$. In other words,

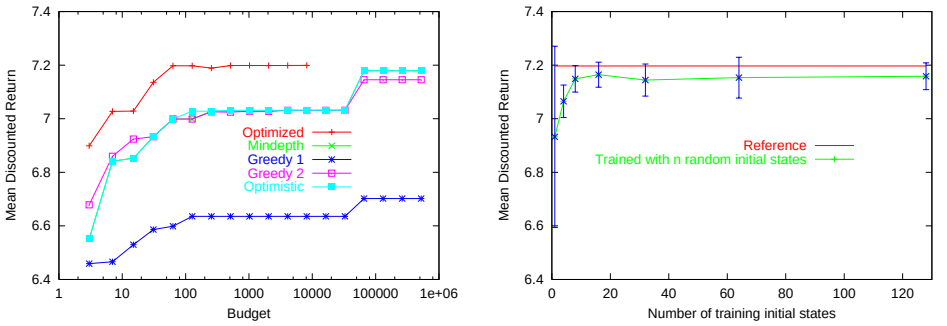

Fig. 2. Left: mean discounted return of five look-ahead tree policies given different amounts of computational resources. Right: Comparison of the tree-exploration strategy optimized on X_0 with the one optimized on randomly sampled initial states.

with a budget $B = 63$, the optimized tree-exploration strategy outperforms exhaustive breadth-first search with a maximal depth of 18.

As an example, the strategy optimized for a budget $B = 63$ is equal to:

$$h^{optimized-63}(n_0, \ldots, n_d) = x_1(n_d)(0.3249 + 1.3368r(n_d) - 2.9695d) + x_2(n_d)(0.9078 - 0.2566r(n_d) + 0.4561d),$$

where $x_1(n_d)$ is the current position and $x_2(n_d)$ is the current velocity.

Notice that for the moment, we evaluated the optimized tree-exploration strategy on the same sample X_0 that was used for optimization. These results are thus "in-sample" results for our method. In order to assess out-of sample behavior and robustness of our algorithm, we performed an additional set of experiments reported on the right of Figure 2. In these experiments, we optimized our strategy with training samples X_0' of growing sizes n and generated independently from the sample X_0 used for the purpose of evaluating the average return (the same as the one in the left part of the figure). We carried out these evaluations by averaging the results over 10 runs for a computing budget $B = 63$ and report the mean and standard deviation of the average returns. When optimized with samples X_0' and evaluated on X_0, the optimized strategy is slightly inferior to the one obtained when optimizing on X_0, but it still outperforms all the other tree-exploration strategies (for all the runs when $n \geq 4$).

4.4 HIV infection control

We now consider the challenging problem described in [2]. The aim is to control the treatment of a simulated HIV infection. Prevalent HIV treatment strategies involve two types of drugs that will generically be called here "drug 1" and "drug 2". The negative side effects of these drugs in the long term motivate the investigation of optimal strategies for their use. The problem is represented by a six-dimensional nonlinear model and has four actions: "no drugs", "drug 1", "drug 2", "both drugs". The system is controlled with a sampling time of 5 days and we seek for an optimal strategy over a horizon of a few years.

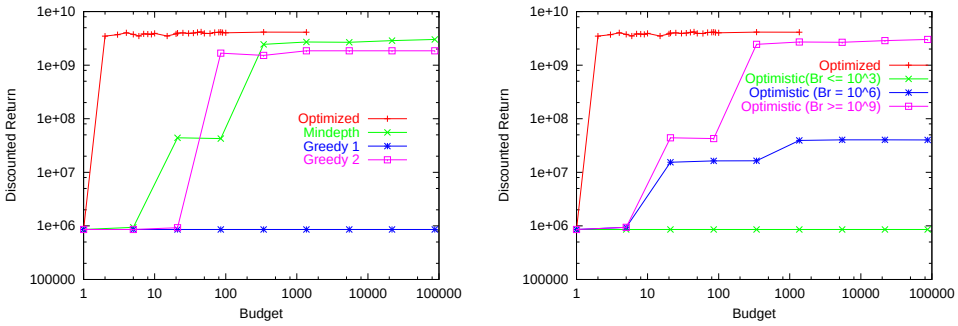

Fig. 3. Left: Discounted returns when starting from the *unhealthy* initial state given different amounts of computational resources. Right: Comparison of the optimized tree-exploration strategy with various optimistic heuristics.

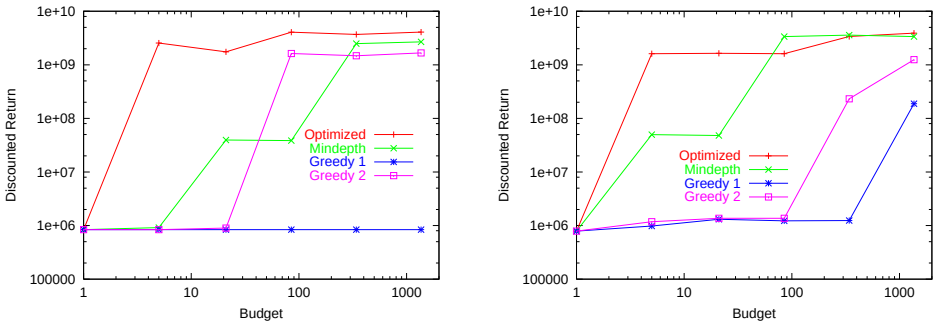

Fig. 4. Discounted returns when starting from two different initial states.

We consider a single initial state $x_0 = (163573, 5, 11945, 46, 63919, 24)$ which corresponds to an *unhealthy* stable equilibrium and which represents the state of a patient with a very low immune response. The maximum horizon is $H = 300$ and the discount factor is $\gamma = 0.98$.

We use the same heuristics as previously except $h^{optimistic}$ which cannot be applied directly since no exact bound on the reward is known for this problem. Since state and reward variables may have very large numerical values (these variables typically range from 10^0 to 10^6), we slightly modified ϕ^{simple} by applying the $log(\cdot)$ function on state variables $x_d(n_d)$ and on rewards $r(n_d)$. Except for this difference, we used the same parameters as in the previous example.

Figure 3 reports the results for growing budget values B. Since optimized look-ahead tree policies worked very well for small values of B, we performed a more fine sampling of its performance for small values of B. The gain of using optimized look-ahead tree policies is striking on this domain: with a budget of only $B = 2$, the optimized tree-exploration strategy outperforms the best heuristic (including uniform tree search) with $B = 87381$ (3.48e9 against 3.02e9). Note that, the root node (corresponding to current state) being always expanded first, the policy with $B = 2$ only chooses which node to expand at the second step, based on the successor states $f(x_t, u_t)$ and on the immediate rewards $\rho(x_t, u_t)$. Therefore, in this case, the depth d is not relevant in the feature function.

Besides the particular case where $B = 2$, the optimized tree-exploration strategy significantly outperforms all other strategies on the whole range of budget values. With $B = 85$, the optimized policy gives similar results to the best results reported by [2] with a reinforcement learning approach (4.13e9 *v.s.* 4.16e9).

To analyze the behavior of an "optimistic" approach on this problem, since B_r is not known exactly, we have tried the $h^{optimistic}$ heuristic with all values $B_r = 10^k$ with $k \in \{-9, -6, -3, \ldots, 12, 15, 18\}$. The scores of these heuristics are reported on the right of Figure 3. We observed three different behaviors. When $B_r <= 10^3$, the optimistic term is very small and the heuristic acts as $h^{greedy1}$. When $B_r >= 10^9$, the optimistic term takes all the importance and the heuristic acts as $h^{mindepth}$. When $B_r = 10^6$, we obtain an intermediate heuristic.

Since we optimized the tree-exploration strategy from a single initial state, it is of interest to assess to what extent it generalizes well when starting from other initial states. The left part of Figure 4 reports the returns when starting from an initial state which is close to the one used during optimization: $(163500, 4, 12000, 50, 63000, 20)$. The results here are quite similar to those of Figure 3, which shows that small differences in the initial state do not significantly degrade the quality of the optimized strategy on this problem. To experiment the robustness of our approach, we then tried with a completely different initial state that corresponds to a healthy patient that just got infected by the virus: $(10^6, 3198, 0, 0, 1, 10)$. Although this initial state is totally different from the one that was used during optimization, our optimized strategy still works reasonably well and nearly always outperforms the other tree-exploration strategies.

5 Conclusion and further work

In this paper, we have proposed a new approach to design look-ahead tree search policies for optimally controlling a system over an infinite horizon. It departs from the existing techniques by the fact that it optimizes the tree exploration technique in an off-line stage, by tuning it to the system dynamics, the reward function, and to the on-line computing budget. The resulting optimized look-ahead tree policy has been tested on two-benchmark problems and simulation results show that for similar or better performances it requires significantly less on-line computational resources than other look-ahead policies.

While the optimized look-ahead tree exploration strategies tested in this paper where performing very well, we believe that there is still room for improving their performances. This could be done for example by using other search spaces for candidate tree-exploration algorithms as well as other optimization techniques for looking for the best one. In particular, we conjecture that using incremental tree-exploration algorithms which do not score anymore a terminal node based only on the information contained in the path connecting this node to the top one, may be a more relevant technical choice.

We have assumed in this paper that we were dealing with optimal control problems having a finite, and not too large, action space. However, many interesting problems have very large or even continuous action spaces for which it is not possible for look-ahead tree policies to develop a node so that it has

successors for every possible action. One way to extend our approach to such a setting would be to use a set of candidate tree-exploration algorithms that do not only point to the nodes that should be preferably developed but also to a subset of control actions that should be used to expand these nodes.

While we have in this paper considered a deterministic and fully observable setting, related types of policies have also been proposed in the literature for stochastic, adversarial and/or partially observable settings many of them belonging to the class of Monte-Carlo tree search techniques. A key issue for these techniques to work well is to have good tree-exploration strategies. Investigating whether the systematic approach proposed in this paper for designing such strategies could be used in such settings would be very relevant.

Acknowledgements

Damien Ernst acknowledges the financial support of the Belgian National Fund of Scientic Research (FNRS) of which he is a Research Associate. This paper presents research results of the European excellence network PASCAL2 and of the Belgian Network BIOMAGNET, funded by the Interuniversity Attraction Poles Programme, initiated by the Belgian State, Science Policy Office.

References

1. A.G. Barto, S.J. Bradtke, and S.P. Singh. Learning to act using real-time dynamic programming. *Artificial Intelligence*, 72:81–138, 1995.
2. D. Ernst, G. Stan, J. Goncalves, and L. Wehenkel. Clinical data based optimal STI strategies for HIV; a reinforcement learning approach. In *Proceedings of the 45th IEEE Conference on Decision and Control*, 2006.
3. P. Hart, N. Nilsson, and B. Raphael. A Formal Basis for the Heuristic Determination of Minimum Cost Paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2):100–107, February 1968.
4. J-F. Hren and R. Munos. Optimistic planning of deterministic systems. In European Workshop on Reinforcement Learning Springer LNAI 5323, editor, *Recent Advances in Reinforcement Learning*, pages 151–164, 2008.
5. Richard E. Korf. Real-time heuristic search. *Artificial Intelligence*, 42(2-3):189–211, 1990.
6. J A Lozano, J A Lozano, and P Larra Naga. *Estimation of Distribution Algorithms. A New Tool for Evolutionary Computation*, pages 99–124. Kluwer Academic Publishers, 2002.
7. F. Maes. *Learning in Markov Decision Processes for Structured Prediction*. PhD thesis, Pierre and Marie Curie University, Computer Science Laboratory of Paris 6 (LIP6), October 2009.
8. S. Minton. *Machine Learning Methods for Planning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1994.
9. M. Pelikan and H. Mühlenbein. Marginal distributions in evolutionary algorithms. In *Proceedings of the International Conference on Genetic Algorithms Mendel '98*, pages 90–95, Brno, Czech Republic, 1998.
10. S. W. Yoon, A. Fern, and R. Givan. Learning heuristic functions from relaxed plans. In *International Conference on Automated Planning and Scheduling (ICAPS'06)*, pages 162–171, 2006.