Optimized Lookahead Trees: Extensions to Large and Continuous Action Spaces

Tobias Jung, Damien Ernst, and Francis Maes

Montefiore Institute

University of Liège

{tjung}@ulg.ac.be

Motto: Bridging the gap between lookahead tree search and direct policy search

Motivation / Background

Problem statement

We consider general discrete-time systems with

- X state space (e.g., $\subset \mathbb{R}^{n_x}$, n_x dimensionality of X)
- A action space (e.g., $\subset \mathbb{R}^{n_u}$, n_u dimensionality of A)
- $f : X \times A \to X$ deterministic transition function
- $\varrho : X \times A \to \mathbb{R}$ stepwise performance measure ("rewards")
- no regularity assumptions on (f, ϱ) (e.g., linearity) (this rules out standard MPC)

Let $\pi : X \to A$ denote a stationary deterministic policy. Consider

$$V^\pi(x_0) := \lim_{T\to\infty} \sum_{t=0}^{T} \gamma^t \varrho(x_t, \pi(x_t)) \quad \text{where } x_{t+1} = f(x_t, \pi(x_t))$$

Goal: model based deterministic optimal control

Given (f, ϱ), we want to find the best policy $\pi^* = \operatorname{argmax}_\pi V^\pi(x_0)$

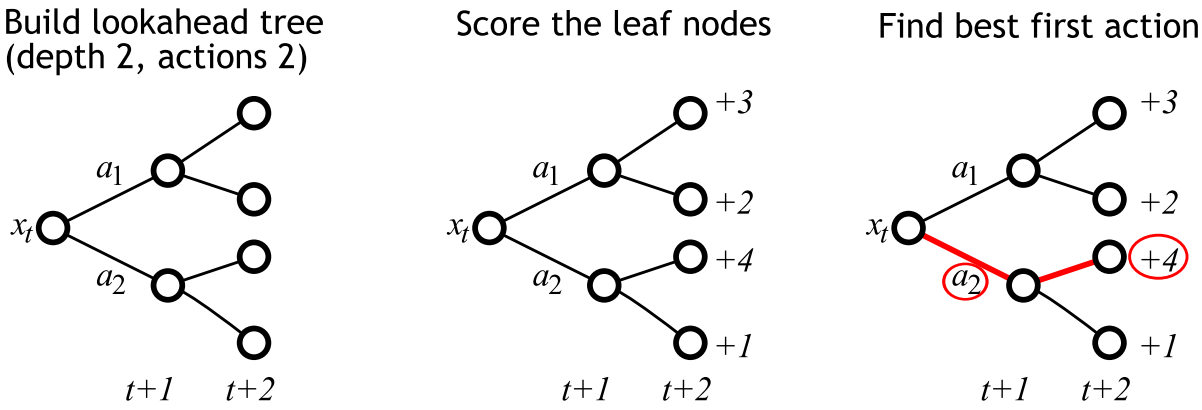
Challenge: as we all know, solving the HJB equation is difficult, in particular if the dimensionality of the state space is large. (And we do not even want to think about high dim action spaces.)

Lookahead tree policies (LT)

(Assume for the moment that the action space is finite and small.)

We consider approaches that locally build a tree of limited depth every time a decision is required:

1. Let x_t be the current state.
2. Build tree (in general non-uniformly!)



(building the tree requires simulation model)

3. Discard tree, execute best first action to make transition $x_{t+1} = f(x_t, \pi_{f,\varrho}(x_t))$. Goto 1.

Note: conceptually, all LT policies can be specified by implementing the two abstract functions

- Node expansion heuristic: determines how the tree is build.
  (Assigns scores to intermediate nodes; the node with the highest score gets expanded next.)
- Action selection heuristic: once the tree is built, determines what the best first action is.
  (Assigns scores to leaf nodes; best first action is the one that lies on the path to the highest scored leaf node.)

Motivation

Various ideas for implementing the two heuristics have been proposed (Hren & Munos, EWRL'08)

The good and the bad: LT policies

1. completely ignore the dimensionality of the state space ...
   ... they are, however, "vulnerable" to large action spaces

2. can produce high-performance policies for difficult nonlinear control problems ...
   ... if they are allowed to build trees of "sufficient" depth
   (which sometimes means many million nodes at each decision step)

The problem is: limited computational resources

- In some applications simulating a transition from a model is very expensive
- Moreover, assume we have a constrained online budget
  (=number of transitions we can simulate before we have to return a decision)

Thus: our goal is to provide an answer to the following question:

> **"Assume we have a model of a system (f, ϱ). What is the best (LT-based) policy $\pi_{f,\varrho}$ one can find if the policy is allowed to spend no more than K resources to output a decision at each step?"**

Motivation II: LT+DPS=OLT

Point is: LT policies require *large* trees because they rely on **generic** heuristics.

Hence our idea: optimized lookahead tree policies (OLT)

- Parameterize the heuristics, e.g.,

$$h(\underbrace{\mathtt{n}}_{\text{node}};\theta) = \sum_i \underbrace{\theta_i}_{\text{parameter}} \underbrace{g_i(\mathtt{n})}_{\text{feature}}$$

- **Offline:** optimize θ via global optimization for given domain *and* budget.
  (e.g., stochastic search, genetic algorithm, or my favorite: Gaussian process optimization)
- **Online:** use this θ and deploy the policy $\pi_{f,\varrho}(\cdot;\theta)$
- OLT can be seen as standard direct policy search with a nonstandard form of implicit policy representation.

Previously: OLT for finite and small action spaces
(Jung et al. IJACSP, 2013), (Maes et al., EWRL, 2011)

Goal now:

- **Extend OLT to continuous action spaces.**

OLT for Continuous Action Spaces

Basic algorithm

We start by considering finite actions

Notation:

- Let x_t be the current state for which we want to compute action $\pi_{f,\varrho}(x_t;\theta)$.
- Let $\mathcal{T}$ be the list of open/unexplored nodes.
- Every node in corresponds to a state-action pair and encodes a path from the root.
- Every node $\mathtt{n} \in \mathcal{T}$ is a `struct` object of type $\aleph$ with members:
  - $\mathtt{n}.x$ the underlying state
  - $\mathtt{n}.a$ the underlying action
  - $\mathtt{n}.d$ the depth from the root
  - $\mathtt{n}.r$ cumulative reward on path root$\longrightarrow (\mathtt{n}.x, \mathtt{n}.a)$.
  - $\mathtt{n}.\pi$ first action on path

Algorithm:

- While curr_expansions $<$ budget
  1. Find node with highest expansion score: $\mathtt{n}^* := \text{argmax}_{\mathtt{n}\in\mathcal{T}}\, \text{exp\_score}(\mathtt{n};\theta)$.
  2. Simulate transition from $(\mathtt{n}^*.x, \mathtt{n}^*.a)$: $x' = f(\mathtt{n}^*.x, \mathtt{n}^* a)$.
  3. For each action, add new node from x' to $\mathcal{T}$. Remove $\mathtt{n}^*$.
- Return policy action: $\pi_{f,\varrho}(x_t;\theta)$:
  1. Find node with highest action selection score: $\mathtt{n}^* := \text{argmax}_{\mathtt{n}\in\mathcal{T}}\, \text{act\_score}(\mathtt{n};\theta)$.
  2. Return first action: $\pi_{f,\varrho}(x_t;\theta) = \mathtt{n}^*.\pi$

Recursive action splitting

We now turn the continuous-action problem into a discrete-action problem. Cf. (Pazis & Lagoudakis, ICML'09)

Transformed problem: $(f, \varrho) \longrightarrow (f', \varrho')$

- New state space = old state space + a partition of the action space
- New action space = $\{split\_left, split\_right, go\}$
  (which refine the partition of the action space of the current state)
- The center of a partition (in 1D an interval) encodes the action under 'go'
- New transition model:

$$\begin{aligned} f'((x,\alpha), split\_left) &= (x, left(\alpha)) \\ f'((x,\alpha), split\_right) &= (x, right(\alpha)) \\ f'((x,\alpha), go) &= (f(x, center(\alpha)), A) \end{aligned}$$

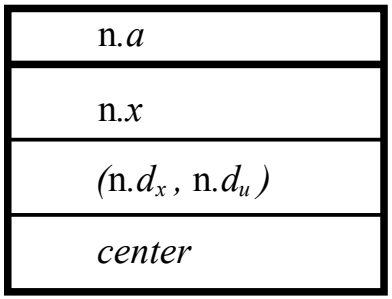
- New reward model:

$$\varrho'((x,\alpha), a) = \begin{cases} \varrho(x, center(\alpha)) & \text{if } a = go \\ 0 & \text{otherwise.} \end{cases}$$

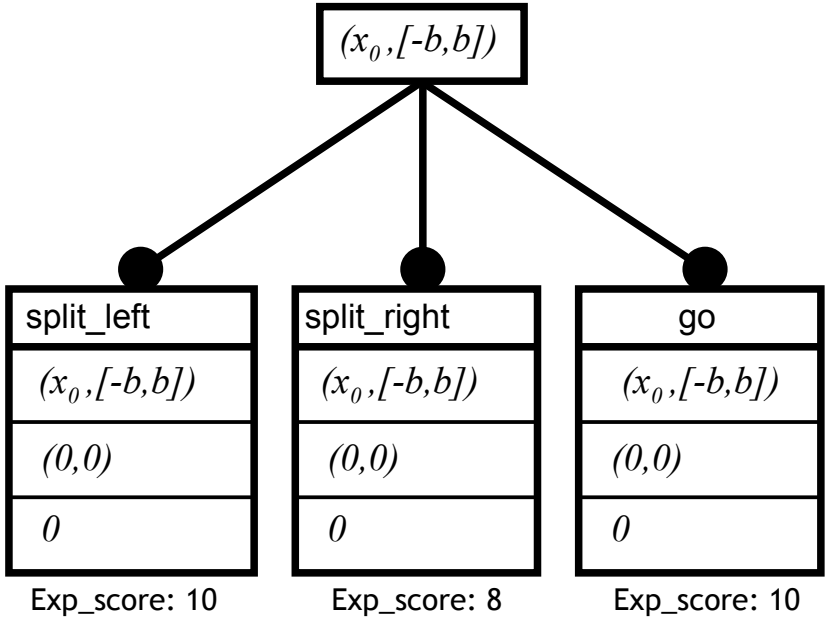
Illustration

Illustration

Nodes:

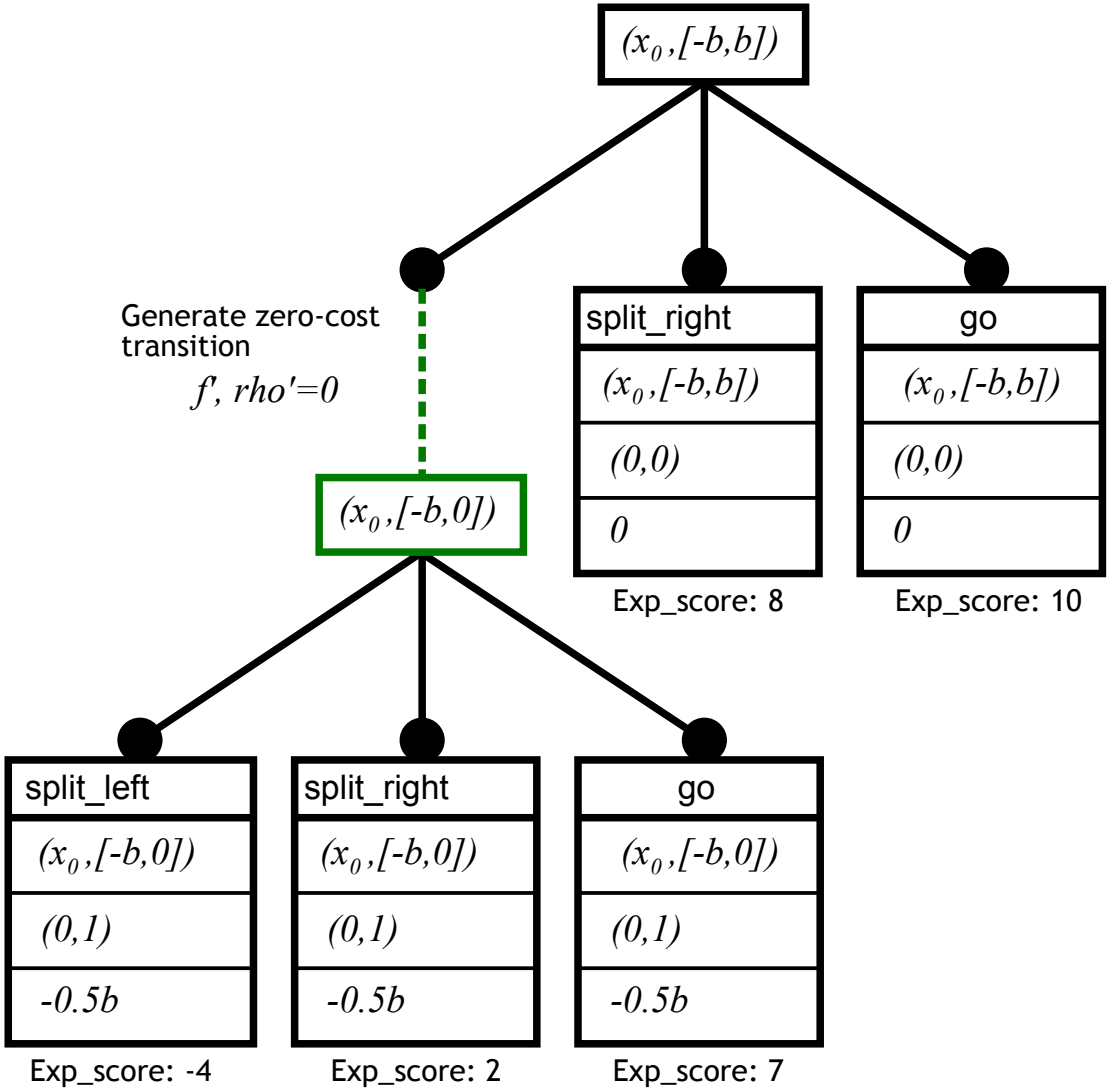


Initial look-ahead tree:



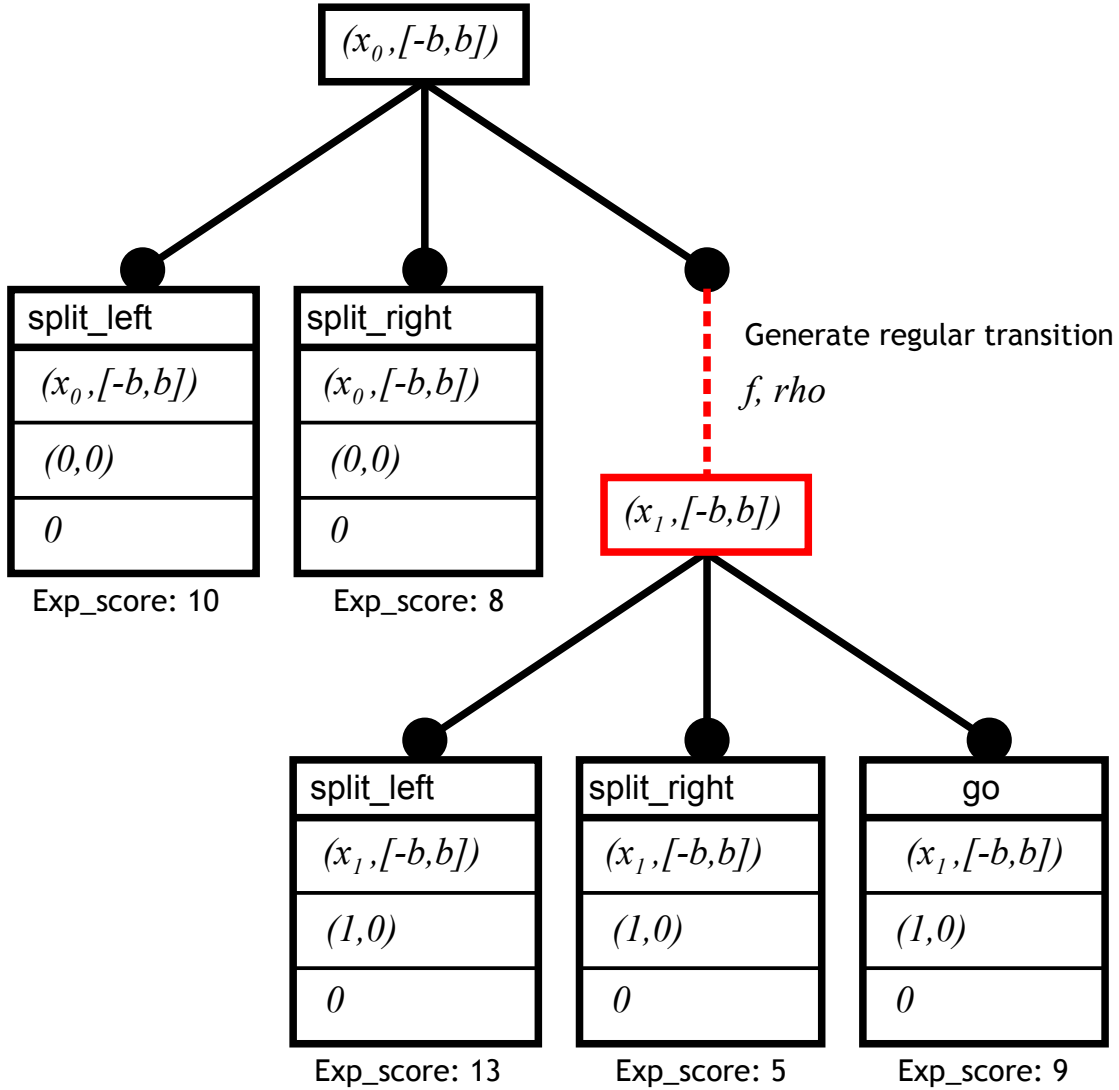
Illustration

Expanding the "split_left" node:



Illustration

Expanding the "go" node:



Experiments

Experimental Setup

Domain: HIV multidrug treatment (Adams et al., Math. Biosciences and Engin.,Vol. 1, 2004)

- Goal is to administer drug cocktail over a period of 1500 days (1 decision = 5 days) to reduce number of infected cells and bring patient into unstable healthy equilibrium
- Challenging optimal control problem with nonlinear dynamics
- 6 dimensional state space; 2 dimensional action space

Methods examined:

- 4-action scenario (baseline)
  - FQI with random trees (req. millions of sample transitions!)
  - LT (Hren & Munos, EWRL, 2008)
  - OLT (Jung et al., IJACSP, 2013)
- continuous action scenario
  - OLT+rec.act.splt
  - OLT+act.sampl. (only described in our paper)

Experimental protocol:

- Each LT based method tested for various settings of the allowed online budget.
- Global optimization implemented through cross entropy (∼5000 pe) and GPO (∼500 pe).

How did we parameterize the heuristics?

- **Action selection heuristic:** as in (Hren & Munos, EWRL'08)

$$\text{act\_score}(\mathbf{n};\theta) \equiv \text{act\_score}(\mathbf{n}) = \mathbf{n}.r + \gamma^{\mathbf{n}.d}\underline{B}/(1-\gamma)$$

where $\underline{B}$ is minimum attainable reward in domain.
(Thus does not depend on any parameters.)

- **Node expansion heuristic:**

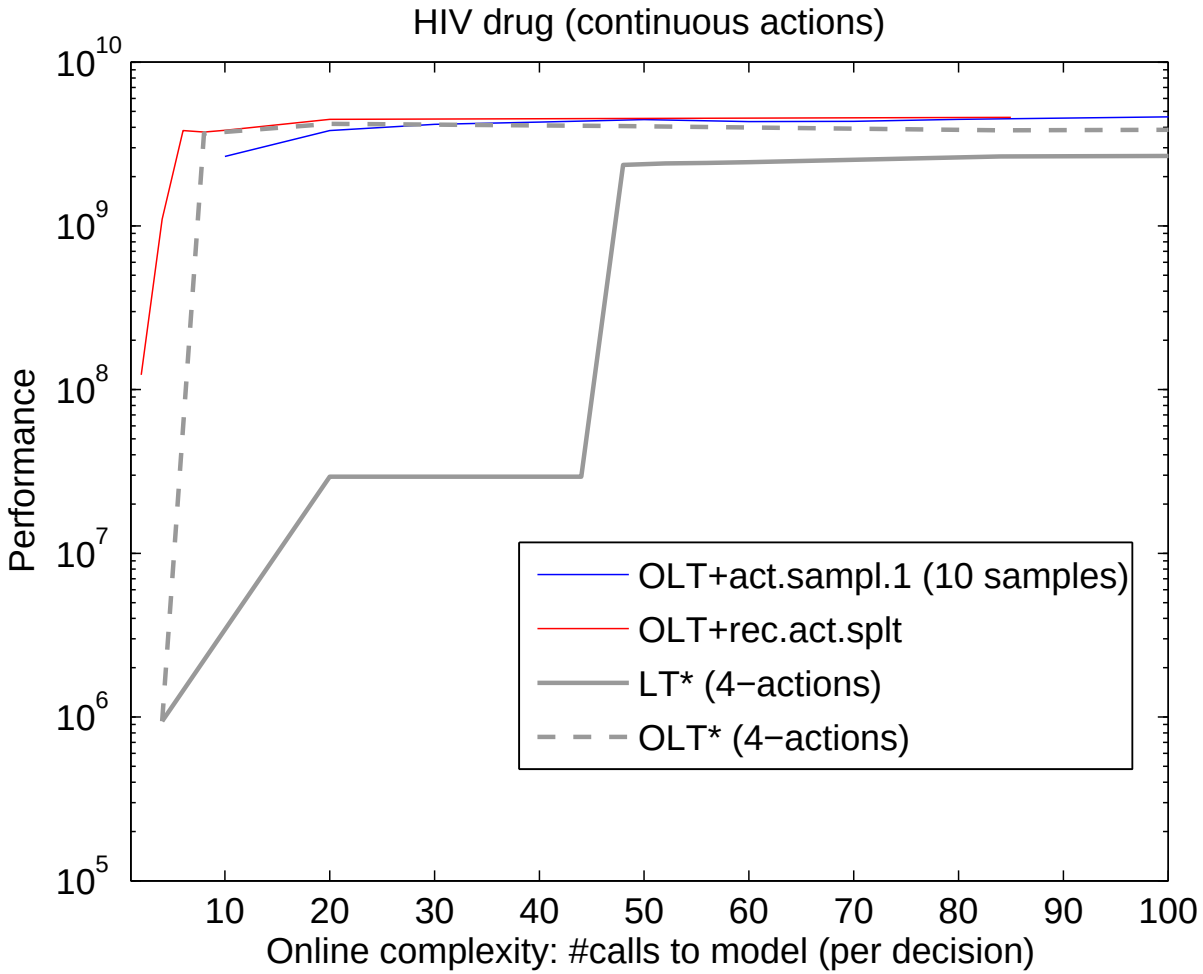
$$\text{exp\_score}(\mathbf{n};\theta) = \sum_{i=1}^{5}\sum_{j=1}^{n_x}\theta_{ij}g_{ij}(\mathbf{n})$$

with

$$\begin{aligned}
g_{1j}(\mathbf{n}) &= \mathbf{n}.x_j \\
g_{2j}(\mathbf{n}) &= \mathbf{n}.x_j \cdot \mathbf{n}.r \\
g_{3j}(\mathbf{n}) &= \mathbf{n}.x_j \cdot \mathbf{n}.d \\
g_{4j}(\mathbf{n}) &= \mathbf{n}.x_j \cdot number_splits(\mathbf{n}.\alpha) \\
g_{5j}(\mathbf{n}) &= \mathbf{n}.x_j \cdot center(\mathbf{n}.\alpha)
\end{aligned}$$

(features/weights duplicated for each discrete action)

Results for HIV drug treatment



Some numbers:

Method	Perf.
4 actions	
FQI	4.22e9
LT	4.21e9
OLT	4.22e9
continuous actions	
OLT+rec.act.splt	4.78e9

Summary

3 Ways of Solving Optimal Control Your Mother Could Understand

	Performance	Online cost	Offline cost
Direct Policy Search	good – very good	very low	very high
Lookahead Tree Policies	very good	very high	zero
Optimized Lookahead Tree Policies	very good	low	medium – high