

Lazy Planning under Uncertainty by Optimizing Decisions on an Ensemble of Incomplete Disturbance Trees

Boris Defourny, Damien Ernst, and Louis Wehenkel

University of Liège,
Department of Electrical Engineering and Computer Science,
Grande Traverse, 10, Sart-Tilman, B-4000 Liège, Belgium
{Boris.Defourny, dernst, L.Wehenkel}@ulg.ac.be

Abstract. This paper addresses the problem of solving discrete-time optimal sequential decision making problems having a disturbance space W composed of a finite number of elements. In this context, the problem of finding from an initial state x_0 an optimal decision strategy can be stated as an optimization problem which aims at finding an optimal combination of decisions attached to the nodes of a *disturbance tree* modeling all possible sequences of disturbances $w_0, w_1, \dots, w_{T-1} \in W^T$ over the optimization horizon T . A significant drawback of this approach is that the resulting optimization problem has a search space which is the Cartesian product of $O(|W|^{T-1})$ decision spaces U , which makes the approach computationally impractical as soon as the optimization horizon grows, even if W has just a handful of elements. To circumvent this difficulty, we propose to exploit an ensemble of randomly generated *incomplete* disturbance trees of controlled complexity, to solve their induced optimization problems in parallel, and to combine their predictions at time $t = 0$ to obtain a (near-)optimal first-stage decision. Because this approach postpones the determination of the decisions for subsequent stages until additional information about the realization of the uncertain process becomes available, we call it *lazy*. Simulations carried out on a robot corridor navigation problem show that even for small incomplete trees, this approach can lead to near-optimal decisions.

Keywords: Stochastic dynamic programming, Ensemble methods.

1 Introduction

The discrete-time optimal control paradigm can be used to formalize a broad class of problems arising in a variety of fields such as finance, automatic control, robotics, or operations research. In this paradigm, at each time step t , the decision maker measures the state $x_t \in X$ of the environment and takes a decision $u_t \in U$ according to his decision rule. As a result of his decision, the environment transits to a new state x_{t+1} and the decision maker observes a scalar reward signal r_t which reflects in some way the impact of his decision on his performance

criterion. The environment behavior as perceived by the decision maker can be in general nonlinear and stochastic.

In this paper, we assume that the stochastic nature of the environment is modeled by an *exogenous* disturbance process ($w_t \in W$) which acts as an additional input on the state transitions. We also assume that this process is memoryless and that the optimality criterion J of the decision maker is the expected value of the sum of rewards r_t over a finite number T of stages. Within this context, we focus on the computation of decisions that maximize J when a model of the environment is provided to the decision maker in terms of the initial condition x_0 , a full specification of the stochastic process w_t , and the functions $f_t(\cdot, \cdot, \cdot)$ and $r_t(\cdot, \cdot, \cdot)$ allowing to compute x_{t+1} and r_t from x_t, u_t and w_t .

Different strategies can be thought of for computing decisions in such a context. One of them would be to compute a fixed sequence of decisions u_0, u_1, \dots, u_{T-1} from the available information. Solving the problem in this manner is equivalent to searching in the space $U^T = \times_{t=0}^{T-1} U$ an element that maximizes the optimality criterion. This approach, also known as the *open-loop* approach, is very convenient for solving problems for which the dynamics of the environment is linear and deterministic, and the optimality criterion J convex. Indeed, under these conditions, efficient algorithms for searching for the best element in U^T exist, and it can be shown that among the set of all plausible decision rules, the one deduced by such an approach is optimal within this restricted context [1,2].

Putting aside the difficulty of solving this optimization problem for more general classes of problems, the open-loop approach is intrinsically less attractive for stochastic problems since the sequence of decisions so obtained is generally quite suboptimal with respect to the use of optimal closed-loop decision rules, i.e. decision rules which determine the current action from the current time and current information available about the state of the environment. While the suboptimality of the open-loop approach could to some extent be decreased if at time $t > 0$, the last remaining $T - t$ decisions were reoptimized by taking into account that the system is in state x_t at time t , something it was impossible to predict at times $0, 1, \dots, t - 1$ due to the stochasticity of the system, the inherent suboptimality of such a time receding open-loop approach remains because the stochastic nature of the problem is not explicitly taken into account when computing the sequence of decisions.

Actually, some approaches exist to extend in an optimal way the open-loop philosophy to stochastic problems [3]. For easing subsequent discussions, we will suppose that the disturbance space W of the stochastic optimal control problem is finite. The main philosophy behind these approaches is the following. First, they generate all the $|W|^T$ possible T -stage disturbance scenarios w_0, w_1, \dots, w_{T-1} . Secondly, they associate to each one of these scenarios and each possible sequence of decisions a return. Then they determine by solving in one single optimization problem $|W|^T$ sequences of decisions of length T , one for each scenario, such that the expected return over all the scenarios is maximized. By imposing the constraint that the first t decisions of two such sequences corresponding to two scenarios having the same $t - 1$ first elements w are the same,

these techniques can be used to determine optimal decision rules which associate a decision to each time step t and any partial sequence of disturbances w_0, \dots, w_{t-1} . An immediate variant of this approach which represents implicitly the equality constraints over the decisions (called non-anticipativity constraints) is obtained by reformulating the optimization problem on a disturbance tree.

One main drawback of these disturbance tree approaches for solving stochastic optimal control problems is related to the size of the optimization problem they require to solve. Indeed, the search space for these optimization problems is the Cartesian product of n decision spaces U where n is $O(|W|^{T-1})$. This exponential growth makes the approach rapidly computationally impractical as soon as T grows, even if W has only a few elements.

In order to circumvent this difficulty, we propose in this paper an alternative approach exploiting an ensemble of randomly generated *incomplete* disturbance trees of controlled complexity. In this approach, the incomplete disturbance trees are built by developing each node only partially, that is by not necessarily associating to a non-terminal node $|W|$ children nodes. More specifically, our strategy for selecting the disturbances for developing a node is nondeterministic and tends to develop less the nodes as the tree depth increases. This strategy depends on parameters that control the expected number of nodes of a tree, and, consequently, which influence the size of the search space for the optimization problem since it is equal to the Cartesian product of n decision spaces U , where n is the number of nonterminal nodes of the tree. The optimization problems induced by these trees can be solved in parallel. Their predictions at time $t = 0$ are combined in order to compute a (near-)optimal first-stage decision for the problem at hand. We call this approach *lazy*, in order to stress the fact that it postpones the determination of the decisions for subsequent stages until additional information about the realization of the uncertain process becomes available¹. We provide simulation results carried out on a robot navigation problem which suggest that the proposed framework can strongly improve the computational performances of the original disturbance tree approach for planning under uncertainty while being only slightly suboptimal.

The rest of the paper is structured as follows. Section 2 discusses our “ensemble of incomplete disturbance trees” approach with respect to different works in planning under uncertainty and machine learning. In Section 3, we specify the type of planning under uncertainty problem considered in this paper and show how the problem of finding an optimal decision strategy can be reformulated as an optimization problem on a disturbance tree. This section also describes an example of application of this technique to a robot navigation problem when solving the optimization problem by using a Cross-Entropy based algorithm. Section 4 describes our ensemble of incomplete disturbance trees approach for deriving the first-stage decision and evaluates its performances on the robot navigation benchmark problem. Finally, Section 5 concludes.

¹ The term “lazy” is used in compiler theory and machine learning in order to qualify algorithms which delay computations until enough information is available to decide that they indeed need to be carried out [4,5].

2 Related Work

The paradigm of optimizing decisions on a disturbance tree has already been studied by several authors. Most of their works have been carried out by assuming that the system dynamics is linear and by supposing as disturbance spaces compact subsets of \mathbb{R}^n . One central theoretical result in this field is that as the discretization of the disturbance space becomes finer, the suboptimality of the approach decreases to zero [6,7]. Approaches proposed for building the disturbance trees have been diverse but the central motivation behind them is always the same: having a small disturbance tree which leads to a near-optimal decision rule. As different strategies used for building these trees, one can mention those which interlace the building of the tree with the optimization process [8] or those which optimize decision variables once the tree is built. For these latter ones, one distinguishes methods based on Monte Carlo sampling [9], on the preservation of the statistical properties [10], and on the minimization of probability metrics between the target and the approximate distribution [11,12].

When the disturbance space is composed of a single element (deterministic environment), these disturbance tree approaches degenerate into the computation of an open-loop sequence of decisions. The computation of such sequences of decisions is at the heart of the vastly successful Model Predictive Control techniques which combine their computation with some time receding horizon strategies [1,2,13]. These techniques have also been extended to stochastic environments but rather by using some min-max approaches aimed at finding a solution which is optimal with respect to the worst-case “disturbance sequence” [14], rather than by trying to maximize a compound return function J .

The approach proposed in this paper for alleviating the computational burdens related to the disturbance tree paradigm is the first one which proposes to build an ensemble of models and to aggregate their solution, with the exception perhaps of the work of Nesterov and Vial, who develop in [15] similar ideas for highly structured environments. However, the idea of aggregating the individual outputs of an ensemble of models has already been vastly exploited in machine learning, and especially in supervised learning (classification and regression). As way of example, one can mention the boosting method [16] which builds models sequentially and refines the output regions where the errors are important or the bagging method which builds the models in parallel from some randomized sets of data [17]. These ideas of aggregating the predictions of an ensemble of models have also already been used for planning under uncertainty but in the context where closed-loop decision rules are computed [18,19].

Finally, we mention the work of Kearns et al. [20] who propose to solve stochastic planning problems by developing a tree where each branch corresponds to a decision-disturbance pair. They apply the dynamic programming principle on the tree to compute decisions and show under some particular conditions that sparse sampling of the disturbances suffices to compute near-optimal decisions. As in our approach, the complexity of their tree does not depend on the number of states. However, and contrary to the disturbance tree approach, their tree size grows (rapidly) with the cardinality of the set of possible actions U .

3 Planning over a Disturbance Tree

3.1 Formalization

Our first task will be to formulate a sequential decision problem over a time horizon T , and to explain how a decision making strategy can be evaluated on a complete disturbance tree of depth T .

We consider a system that evolves according to a state transition function $x_{t+1} = f_t(x_t, u_t, w_t)$ starting from a fixed initial state x_0 . Its trajectories are controlled by the decisions $u_t \in U$ and perturbed by disturbances $w_t \in W$, which are drawn independently from a finite time-invariant probability distribution \mathbb{P}_w ². A reward process r_0, r_1, \dots, r_T is defined by $r_t = r_t(x_t, u_t, w_t)$ for $0 \leq t < T$ and $r_T = r_T(x_T)$. The goal is to find a strategy μ maximizing the expectation of a discounted sum of the rewards, with $0 < \gamma \leq 1$ the discount factor³:

$$J^*(x_0) = \max_{\mu} \mathbb{E} \left\{ \sum_{t=0}^{T-1} \gamma^t r_t(x_t, u_t, w_t) + \gamma^T r_T(x_T) \right\}. \quad (1)$$

In the disturbance tree framework, the strategy μ for selecting a decision u_t at time $0 \leq t < T$ consists in deterministic mappings μ_t from current histories $h_t = [w_0, w_1, \dots, w_{t-1}]$ of the disturbance process to decisions u_t at time t . The mapping at time 0 degenerates into a fixed decision u_0 , the history at time 0 being empty. This class of strategies is in principle more general than the class of time-dependent strategies mapping the state x_t to a decision u_t , since the state x_t can always be recovered by the procedure described in Table 1. However, when the disturbance process is memoryless, these two classes of strategies are equivalent in terms of optimality.

Table 1. How to recover states from disturbance histories

Input: An initial state x_0 , a history of the disturbance process $h_t = [w_0, w_1, \dots, w_{t-1}]$, a strategy $\mu_0, \mu_1, \dots, \mu_{t-1}$ for computing decisions up to time $t - 1$.

Output: The state x_t .

1. Initialization: Set x to x_0 and τ to 0.
2. While $\tau < t$, set u to $\mu_{\tau}(w_0, w_1, \dots, w_{\tau-1})$, set x to $f_{\tau}(x, u, w_{\tau})$, and increment τ .
3. Return x .

The complete disturbance tree represents all the possible outcomes of the process w_0, w_1, \dots, w_{T-1} . Its construction is given in Table 2. To each node n of depth $0 < t \leq T$ in the tree corresponds a history $h_n = [w_0, \dots, w_{t-1}]_n$ of the process, through the unique path from the root to the node n . The disturbance

² Independence of the w_t is imposed only to simplify the presentation and facilitate the parallel with the dynamic programming framework.

³ More general objective functions could also be considered in this framework.

Table 2. How to build a complete disturbance tree

Input: The horizon T , the disturbance space $W = \{W_1, \dots, W_m\}$, the probabilities $P_j = \mathbb{P}(w_t = W_j)$.

Output: A complete disturbance tree over the finite horizon T .

1. Initialization : Create a root node of depth 0.
Associate the probability 1 to the root. Set $t = 0$.
2. While $t < T$:
For each node n of depth t , create m successor nodes (of depth $t+1$) with associated values W_1, \dots, W_m and probabilities P_1, \dots, P_m .
Increment t .
3. Return the tree structure and the values w_n and probabilities p_n associated to its nodes n . (Now w_n denotes a value of w_t , where t is the depth of node n .)

Table 3. Evaluation of the expected value of a strategy on a disturbance tree

Input: A disturbance tree, an initial state x_0 , a strategy μ represented by decisions u_n associated to the nodes n of depth $0 \leq t < T$.

Output: The expected value of the decision making strategy.

1. (Computation of the rewards r_n associated to the nodes of the tree.)
Associate the initial state x_0 to the root node. Set t to 1.
While $t \leq T$:
For each node n of depth t : Identify the parent node n' and associate $x_n = f_{t-1}(x_{n'}, u_{n'}, w_n)$ and $r_n = r_{t-1}(x_{n'}, u_{n'}, w_n)$ to node n .
Increment t .
2. (Computation of the expected discounted sum of rewards by backpropagation.)
Associate $J_n = r_T(x_n)$ to each node n of depth T . Set t to $T - 1$.
While $t \geq 0$:
For each node n of depth t : Identify the set of successor nodes $S(n)$, and associate $J_n = \sum_{n' \in S(n)} p_{n'} \cdot (r_{n'} + \gamma J_{n'})$ to node n .
Decrement t .
3. Return the value J_n associated to the root node.
Clearly, this value is equal to $\mathbb{E}\{\sum_{t=0}^{T-1} \gamma^t r_t(x_t, u_t, w_t) + \gamma^T r_T(x_T)\}$
where $u_t = \mu_t(w_0, \dots, w_{t-1})$.

$(w_{t-1})_n$ is directly associated to node n , while $[w_0, \dots, w_{t-2}]_n$ can be collected from the disturbances associated to the nodes in the path. The root, at $t = 0$, has an empty history. The strategy μ can thus be represented on the tree by associating to each node n of depth $0 \leq t < T$ the value $u_n = \mu(h_n)$.

Alternatively, searching for an optimal strategy becomes equivalent to optimizing the values u_n . To this end, states x_n , rewards r_n and partial sums J_n are associated to nodes n . This helps to compute the expected value of an arbitrary strategy μ represented by particular values for u_n . Table 3 describes the full process. The optimization itself is done directly over the node variables u_n . The algorithm of Table 3 will serve as an oracle for scoring the strategy.

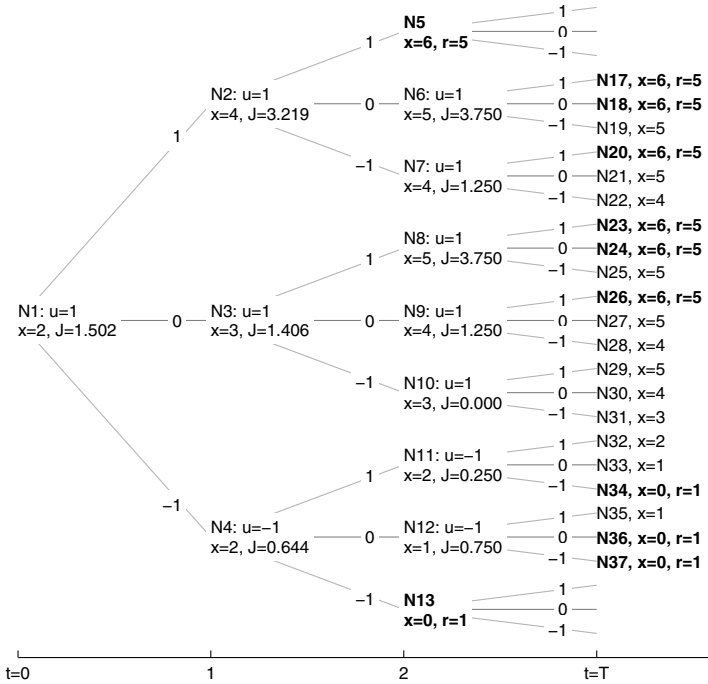


Fig. 1. A toy corridor problem on $T = 3$ with terminal states $\{0, 6\}$, starting from $x_0 = 2$, solved on a complete disturbance tree. The node disturbances w_n are written on the branches and the corresponding node probabilities ($p_n = 0.25$ for $w_n = \pm 1$, $p_n = 0.5$ for $w_n = 0$) are omitted. The values u_n have been optimized, and this fully specifies an optimal strategy μ . For instance, $u = -1$ at node N11 means that $\mu_2(w_0 = -1, w_1 = +1) = -1$. A terminal state is reached at node N5, among other cases. This makes the subtree beyond N5 useless, but only once the decisions are optimized. The expected value of the strategy, $J = 1.502$, is read from the root node N1. Reported values for x_n, r_n, J_n are those obtained with the last invocation of the algorithm of Table 3, which served to score decisions u_n 's generated by the Cross-Entropy method. For instance, the value $J = 0.75$ at node N12 comes from $p_{N36}r_{N36} + p_{N37}r_{N37}$, in accordance with the step 2 of Table 3.

3.2 Illustration

The following corridor navigation problem illustrates the search for an optimal strategy on a complete disturbance tree.

Let $X = \{1, 2, \dots, S-1\} \cup X_{\text{term}}$, where $X_{\text{term}} = \{0, S\}$ is a set of terminal states. Let $W = \{-1, 0, 1\}$, with probabilities $\mathbb{P}_w = \{0.25, 0.50, 0.25\}$.

If $x_t \notin X_{\text{term}}$, then $U = \{-1, +1\}$ and

$$\begin{cases} x_{t+1} = x_t + u_t + w_t, & r_t = 0 & \text{if } 0 < x_t + u_t + w_t < S \\ x_{t+1} = 0, & r_t = 1 & \text{if } x_t + u_t + w_t \leq 0 \\ x_{t+1} = S, & r_t = 5 & \text{if } x_t + u_t + w_t \geq S. \end{cases}$$

If $x_t \in X_{\text{term}}$, then $x_{t+1} = x_t$, $r_t = 0$, and U is irrelevant.

The terminal rewards of equation (1) are set to $r_T(\cdot) = 0$.

In this section, $T = 3$, $S = 6$, $\gamma = 0.9$, and $x_0 = 2$.

The complete disturbance tree is represented on Fig. 1, along with the optimized decisions u_n and the corresponding states x_n , rewards r_n , and partial sums J_n . The Cross-Entropy method was used to find a global optimal solution for u_n . The reader may refer to [21] for more details on the Cross-Entropy method, which was well adapted for the particular problem at hand. Simply put, the Cross-Entropy method samples candidate solutions, using importance sampling oriented towards candidate solutions with the highest scores. If the method succeeds, then the output of Table 3 corresponds to (1). Otherwise, the suboptimal solution that has been found may still be acceptable.

The decision making strategy reported on the figure is indeed optimal. The value $J = 1.502$ corresponds to the value returned by the well-known value iteration algorithm from dynamic programming, stopped after $T = 3$ iterations.

One can check on Fig. 1 that the mapping from states x_t to decisions u_t (which is the kind of mapping considered in the value iteration algorithm) corresponding to the history-to-decision mapping found on the disturbance tree is time-variant⁴.

4 Lazy Approach Using an Ensemble of Incomplete Disturbance Trees to Derive a First-Stage Decision

4.1 Principle

Our second task will be to explain how we can advantageously work on an ensemble of incomplete disturbance trees to determine an optimal first-stage decision.

The representation of a strategy μ by decisions u_n defined on the nodes of the complete disturbance tree shows that an incomplete disturbance tree will only permit an incomplete description of a strategy. However, we restrict our attention to the first-stage decision u_0 , and assume that the optimization of incomplete strategies will give some valuable information on u_0 . We thus propose to aggregate several first-stage decisions obtained from several incomplete strategies optimized on several incomplete disturbance trees. The full process, based on a majority vote for the decision u_0 and described in Table 4, can be restarted at each time step, with the current state as the initial condition.

An incomplete disturbance tree can be built using the randomized algorithm given in Table 5. Starting from the root, the algorithm creates for each node a set of successor nodes by sampling disturbances. Distinct samples form the successor nodes, while the sample multiplicities serve to define node probabilities. The growth of the tree is controlled by choosing a random number m of samples to draw.

The probability distribution of m is an input for the algorithm. For building deeper trees while preserving the branching structure allowed on shorter time

⁴ For instance the nodes N1 ($t = 0$) and N4 ($t = 1$) share the state $x = 2$ but differ in the decision.

Table 4. Computing a decision at time $t = 0$ with an ensemble of M incomplete trees

Input: The initial state x_0 , the optimization horizon T , the desired number M of trees, a specification of the disturbance process $w_{t=0}^{T-1}$.

Output: A decision u_0 to be applied at time 0.

1. Build in parallel M incomplete disturbance trees of depth T .
2. Optimize in parallel a decision making strategy on each tree.
3. Return the decision obtained by a majority vote over the M first-stage decisions u_0 gathered from the roots of the M trees.

Table 5. How to build an incomplete disturbance tree for a discrete, time-invariant, and memoryless disturbance process

Input: The disturbance space W , the probabilities \mathbb{P}_w , the tree depth T .

Parameters: Probability distributions $\mathbb{Q}_A, \mathbb{Q}_B$, both of support in $\{1, 2, \dots, q\}$ where q is the maximal allowed number of samples.

Output: An incomplete disturbance tree over the horizon T .

1. Initialization : Create a root node of depth 0.
Associate the probability 1 to the root. Set $t = 0$.
2. While $t < T$:
Set α to $1/(1+t)$. Set \mathbb{Q}_t to $\alpha\mathbb{Q}_A + (1-\alpha)\mathbb{Q}_B$.
For each node n of depth t :
Draw a random number m according to \mathbb{Q}_t . Draw m samples in W according to \mathbb{P}_w .
Obtain $m' \leq m$ distinct samples $[w^{(1)}, \dots, w^{(m')}]$ of multiplicity $[k^{(1)}, \dots, k^{(m')}]$.
Create m' successor nodes (of depth $t+1$) with associated values $w^{(1)}, \dots, w^{(m')}$ and probabilities $k^{(1)}/m, \dots, k^{(m')}/m$.
Increment t .
3. Output the incidence structure of the tree, and the values w_n and probabilities p_n associated to each one of its nodes n .

horizons, it might be advantageous to let the distribution of m evolve with the depth of the tree. A simple way to do that consists in mixing 2 probability distributions \mathbb{Q}_A and \mathbb{Q}_B with relative weights depending on the depth. More specifically, the probability that $m = j$ for a node of depth t is set to

$$\mathbb{Q}_t(m = j) = \alpha\mathbb{Q}_A(m = j) + (1 - \alpha)\mathbb{Q}_B(m = j) \quad (2)$$

with $\alpha \triangleq 1/(1+t)$ moving progressively from 1 to 0 as the depth t increases. Trees are likely to feature sequences of disturbances with few branchings beyond a certain depth when \mathbb{Q}_B is concentrated on small values of m such as 1 or 2. Of course, the algorithm can be run with fixed branching probabilities by setting $\mathbb{Q}_A \equiv \mathbb{Q}_B$. However, if \mathbb{Q}_B has its mass concentrated on 1, the expected number of nodes is asymptotically linear with the depth T of the tree. This strongly contrasts with the exponential growth of the size of the complete tree with T .

It is easy to estimate by Monte Carlo simulation the expected size of an incomplete tree for given depth T , number of disturbances $|W|$ and probabilities \mathbb{Q}_A ,

\mathbb{Q}_B . Therefore, it is possible to choose \mathbb{Q}_A and \mathbb{Q}_B so that the expected size of a tree is in line with the size of the optimization problem one is able to deal with.

4.2 Illustration

The corridor navigation problem of Section 3 will illustrate the clear benefit in terms of computational complexity of working on an ensemble of incomplete disturbance trees instead of a unique complete tree. A set of simulations is run with $S = 10$, $T = 6$, $\gamma = 0.7$, on different initial conditions x_0 .⁵

The problem is solved for $x_0 = 3$ on a complete tree (composed of 1093 nodes) in a reasonable time. This permits to report relative time savings for the solution on incomplete trees. A majority vote over $M = 100$ incomplete trees gives the first decision u_0 . Subsequent decisions are ignored, assuming that simpler problems on shorter time horizons will be solved to obtain u_1, u_2, \dots as soon as x_1, x_2, \dots are known.

Table 6 gives the 4 representative choices for the distributions \mathbb{Q}_A and \mathbb{Q}_B that have been considered as parameters of the tree generation algorithm, along with their macroscopic effect on the size of a tree of depth $T = 6$.

Table 7 shows that with the first choice (Test I), the trees reduce to T -length sequences of disturbances. Wrong decisions for $x_0 = 4$ and $x_0 = 5$ indicate that this structure is too weak. Test IV leads to moderately dense trees and right decisions. Beside these two extremes, Test II uses fixed probabilities for the number of disturbances to sample and Test III use decaying probabilities. The setting for Test III dominates the one for Test II, because it produces smaller trees while these trees prove more reliable on the initial condition $x_0 = 4$: they advocate the optimal decision more often.

Table 6. Representative settings for the probability distributions of the number of samples per node, and their effect on the size of the incomplete trees


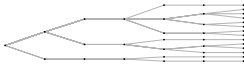
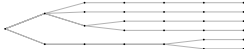
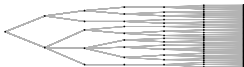
	Q_A †	Q_B	Short description	Number of nodes ‡	
				Mean	Std dev.
I	[1 0 0]	[1 0 0]	Always draw 1 sample.	7	0
II	[1 1 1]/3	[1 1 1]/3	1,2,3 samples with prob. 1/3.	39.31	22.31
III	[0 0 1]	[1 0 0]	Deeper, draw 1 instead of 3.	29.87	13.10
IV	[0 0 1]	[0 0 1]	Always draw 3 samples.	136.25	53.28

† $Q_A = [1 0 0]$ means that the probability of drawing 1, 2, 3 samples is respectively 1, 0, 0 under the distribution \mathbb{Q}_A that parameterizes the algorithm of Table 5. The number of children nodes may be less, depending on the number of distinct samples.

‡ Estimated by building 400 trees of depth $T = 6$. A complete tree has 1093 nodes.

⁵ We used a larger optimization horizon T than in Section 3 to better highlight the performances of our approach. The value of γ differs also from the one previously chosen (now 0.7 rather than 0.9) to have an optimal first-stage decision that varies with the initial state, so as to make the example more interesting.

Table 7. Decisions obtained with incomplete trees on the corridor problem with terminal states $\{0, 10\}$, $T = 6$, and $\gamma = 0.7$. The 4 settings of Table 6 are tested. Basically, decisions are correct when the trees are not too small.

	Decision u_0^\dagger				Time [‡]	Typical tree
	$x_0 = 2$	$x_0 = 3$	$x_0 = 4$	$x_0 = 5$		
<i>Optimal:</i>	-1	-1	+1	+1		
Test I	-1 (93%)	-1 (78%)	-1 (66%)	-1 (51%)	0.04%	
Test II	-1 (96%)	-1 (82%)	+1 (54%)	+1 (81%)	0.42%	
Test III	-1 (95%)	-1 (80%)	+1 (63%)	+1 (88%)	0.15%	
Test IV	-1 (98%)	-1 (91%)	+1 (71%)	+1 (100%)	2.41%	

[†] Initial decision obtained by a majority vote on $M = 100$ trees, with the share in parenthesis. Separate tests are conducted for the 4 mentioned initial states x_0 .

[‡] Mean computation time for solving 1 incomplete tree (case $x_0 = 3$), in percentage of the time required to solve the complete disturbance tree for the same problem. Time savings are huge.

Beyond the huge time savings, the tests have also revealed an unexpected advantage of using the Cross-Entropy method on incomplete trees. Suboptimal solutions indeed arise more frequently on the complete tree than on the smaller incomplete ones, by a simple scale effect. A wrong decision can thus be inferred from a suboptimal solution on the complete tree, while the majority vote from solutions on incomplete trees yields a correct decision in a more reliable way.

5 Conclusions

This paper has proposed an approach for alleviating the computational burdens of the original disturbance tree paradigm for planning under uncertainty. The approach builds an ensemble of small trees, solves in parallel the small size optimization problems which correspond to these trees and aggregates their first-stage decisions. Applying this algorithm in a time receding fashion results in a *lazy* decision making strategy which computes at each time step the decision to apply given all the available information, thereby focusing only on the particular subproblem to solve rather than trying to pre-compile once and for all a decision strategy that would be applicable to all kinds of realizations of the environment under control. This approach has been evaluated on a robot navigation problem and the results obtained are encouraging. Indeed, with an ensemble of small trees (especially with respect to the fully developed one), it has been possible to obtain in a reliable way accurate predictions of the optimal first-stage decision.

While the strategy adopted in this paper for building the incomplete trees is giving good results, we do not exclude that it could still be significantly improved by for example relying on other sampling procedures than a pure Monte Carlo one for developing a disturbance tree node. It would also be interesting to study to which extent some specific optimization tools could be customized to the structure of the optimization problem defined by a disturbance tree for leveraging their performances. Besides the study of the algorithmic improvements that could be brought to our multi-tree framework, it would also be interesting to establish its theoretical properties. For example, it would be informative to know (even under some highly restrictive assumptions on the environment structure) some upper bounds (in probability) on the suboptimality of the decision rules with respect to the size of the ensemble, the variance of the computed decisions or the computational complexity of these multi-tree based algorithms.

Also, while we have in this paper worked out the approach in the context of a memoryless disturbance process, in which case the state of the system under control is a sufficient statistic to take decisions, our approach extends in a straightforward way to general disturbance processes, or in other words to partially observable environments. Further work should thus be carried out to study in more depth the pros and cons of this approach with respect to the literature on partially observable Markov decision processes [22,23].

Another important direction of research concerns the extension of this approach to continuous disturbance processes. Some preliminary work [24,25] along these lines shows that in this context it is important to properly choose the way in which the continuous disturbance process is discretized in order to generate the disturbance trees.

Overall, the approach presented in this paper adds to the arsenal of methods for planning under uncertainty. However, it is not yet clear how it would compete for some specific classes of problems with algorithms exploiting other paradigms, such as the dynamic programming paradigm [26] or the direct closed-loop policy search one [27]. All these paradigms have pros and cons, and establishing which one, or which combination of paradigms, to exploit for a specific problem certainly deserves further research.

Acknowledgments. Damien Ernst acknowledges the financial support of the Belgian National Fund of Scientific Research (FNRS) of which he is a Research Associate. This paper presents research results of the Belgian Network DYSCO (Dynamical Systems, Control, and Optimization), funded by the Interuniversity Attraction Poles Programme, initiated by the Belgian State, Science Policy Office. The scientific responsibility rests with its authors.

References

1. Maciejowski, J.: Predictive Control with Constraints. Prentice Hall, Englewood Cliffs (2001)
2. Morari, M., Lee, J.: Model predictive control: past, present and future. Computers and Chemical Engineering 23, 667–682 (1999)

3. Birge, J., Louveaux, F.: *Introduction to Stochastic Programming*. Springer, New York (1997)
4. Launchbury, J.: A natural semantics for lazy evaluation. In: *POPL 1993: Proceedings of the 20th ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, pp. 144–154. ACM, New York (1993)
5. Friedman, J., Kohavi, R., Yun, Y.: Lazy decision trees. In: *Proc. of 13th National Conference on Artificial Intelligence, AAAI 1996. Part 1(of 2)*, pp. 717–724 (1996)
6. Heitsch, H., Römisch, W., Strugarek, C.: Stability of multistage stochastic programs. *SIAM Journal on Optimization* 17(2), 511–525 (2006)
7. Römisch, W.: Stability of stochastic programming problems. In: Ruszczyński, A., Shapiro, A. (eds.) *Stochastic Programming. Handbooks in Operations Research and Management Science*, vol. 10, pp. 483–554. Elsevier, Amsterdam (2003)
8. Dempster, M.: Sequential importance sampling algorithms for dynamic stochastic programming. *Annals of Operations Research* 84, 153–184 (1998)
9. Shapiro, A.: Monte Carlo sampling methods. In: Ruszczyński, A., Shapiro, A. (eds.) *Stochastic Programming. Handbooks in Operations Research and Management Science*, vol. 10, pp. 353–425. Elsevier, Amsterdam (2003)
10. Høyland, K., Wallace, S.: Generating scenario trees for multistage decision problems. *Management Science* 47(2), 295–307 (2001)
11. Hochreiter, R., Pflug, G.: Financial scenario generation for stochastic multi-stage decision processes as facility location problems. *Annals of Operations Research* 152, 257–272 (2007)
12. Rachev, S., Römisch, W.: Quantitative stability in stochastic programming: The method of probability metrics. *Mathematics of Operations Research* 27(4), 792–818 (2002)
13. Ernst, D., Glavic, M., Capitanescu, F., Wehenkel, L.: Reinforcement learning versus model predictive control: a comparison on a power system problem. *IEEE Transactions on Systems, Man and Cybernetics - Part B (to appear, 2008)*
14. Kothare, M., Balakrishnan, V., Morari, M.: Robust constrained model predictive control using matrix inequalities. *Automatica* 32, 1361–1379 (1996)
15. Nesterov, Y., Vial, J.P.: Confidence level solutions for stochastic programming. *Automatica* 44(6), 1559–1568 (2008)
16. Schapire, R.: The strength of weak learnability. *Machine Learning* 5(2), 197–227 (1990)
17. Breiman, L.: Bagging predictors. *Machine Learning* 24(2), 123–140 (1996)
18. Ernst, D., Geurts, P., Wehenkel, L.: Tree-based batch mode reinforcement learning. *Journal of Machine Learning Research* 6, 503–556 (2005)
19. Sutton, R.: Generalization in reinforcement learning: successful examples using sparse coarse coding. *Advances in Neural Information Processing Systems* 8, 1038–1044 (1996)
20. Kearns, M., Mansour, Y., Ng, A.: A sparse sampling algorithm for near-optimal planning in large Markov decision processes. *Machine Learning* 49(2-3), 193–208 (2002)
21. Rubinstein, R., Kroese, D.: The Cross-Entropy Method. A Unified Approach to Combinatorial Optimization, Monte-Carlo Simulation, and Machine Learning. In: *Information Science and Statistics*. Springer, Heidelberg (2004)
22. Cassandra, A., Kaelbling, L., Littman, M.: Acting optimally in partially observable stochastic domains. In: *Proceedings of the Twelfth National Conference on Artificial Intelligence (AAAI 1994)*, Seattle, Washington, USA, vol. 2, pp. 1023–1028. AAAI Press/MIT Press, Menlo Park (1994)

23. Ng, A., Jordan, M.: PEGASUS: a policy search method for large MDPs and POMDPs. In: Proceedings of the Sixteenth Conference on Uncertainty in Artificial Intelligence, pp. 406–415 (1999)
24. Defourny, B.: Approximate solution to multistage stochastic programs with ensembles of randomized scenario trees. Master’s thesis, University of Liège, Department of Electrical Engineering and Computer Science (2007)
25. Defourny, B., Wehenkel, L.: Averaging decisions from an ensemble of scenario trees: a validation on newsvendor problems (submitted, 2008)
26. Bellman, R.: Dynamic Programming. Princeton University Press, Princeton (1957)
27. Sutton, R., McAllester, D., Singh, S., Mansour, Y.: Policy gradient methods for reinforcement learning with function approximation. Advances in Neural Information Processing Systems 12, 1057–1063 (2000)