Planning under Uncertainty, Ensembles of Disturbance Trees and Kernelized Discrete Action Spaces

Boris Defourny, Damien Ernst and Louis Wehenkel

***Abstract*— Optimizing decisions on an ensemble of *incomplete disturbance trees* and aggregating their first stage decisions has been shown as a promising approach to (model-based) planning under uncertainty in large continuous action spaces and in small discrete ones. The present paper extends this approach and deals with large but highly structured action spaces, through a kernel-based aggregation scheme. The technique is applied to a test problem with a discrete action space of 6561 elements adapted from the NIPS 2005 SensorNetwork benchmark.**

I. Introduction

DISTURBANCE (or scenario) tree approaches for *multi-stage stochastic programming* (see e.g. [1] for an introduction to this paradigm for sequential decision making under uncertainty) have been extensively investigated in the operations research community during the last 10 or 20 years [2], [3], [4]. In this approach, *uncertainties* [5] are modeled as a finite number of random trajectories $w_0, w_1, \ldots, w_{T-1}$ organized in a tree (see Figure 1). The branches of the tree are weighted by probabilities and to its internal nodes are attached decision variables. For a given initial state at the root of the tree, and given *system dynamics* and *reward function* [6], the disturbance tree expresses the relation between the choice of decisions and the expected total return, leading to an optimization problem with as many sets of variables as there are internal nodes in the disturbance tree.

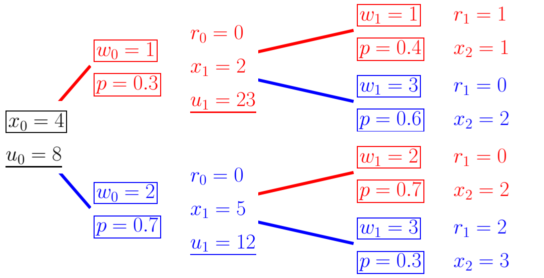


Fig. 1. A 2-stage disturbance tree on a binary disturbance space $w_i \in \{1, 2\}$ with decisions u_i attached to nodes, allowing the computation of states x_i and rewards r_i. The $\boxed{x_0}$, $\boxed{w_i}$, $\boxed{p}$ correspond to problem data; the $\underline{u_i}$ are to be chosen optimally; the x_i, r_i are functions of these two.

The disturbance tree approach has been well studied in the context of *convex* [7] problems (i.e. problems with continuous state and action spaces, linear dynamics and convex return functions) [8], [9], [10] and has already been applied to very large scale applications, such as for example electric power generation scheduling [11], [12] and financial applications [13], [14]. Its main advantage with respect to the dynamic programming framework is primarily that it may be directly applied to very high-dimensional continuous state and action spaces [15] but also that it can cope with a larger class of optimality criteria [16] (in particular non-decomposable ones). Its main drawback stems from the fact that in order to remain tractable, the approach is strongly limited in terms of tree branching factor and depth [17]. In practice, this means that the true (typically continuous, and often high-dimensional) disturbance process needs to be reduced to a rather rough discrete approximation [18]. Much work in this field has therefore been devoted to the construction of "optimal" disturbance trees of limited complexity [19], [20], [21]. Also, in order to mitigate the so introduced suboptimalities, the approach is typically used in a "receding horizon fashion" (just like in model-predictive control [22]) where, at each time step, a new disturbance tree is rebuilt and re-optimized in order to compute the optimal current first-stage decision [23].

To improve (see [24]) the disturbance tree approach, we have proposed in [25] a simple extension inspired by the "perturb and combine" paradigm of supervised learning [26], [27]. Rather than deriving the first-stage decision from a single optimized tree, an ensemble of randomized disturbance trees are solved for their optimal first-stage decisions, and these latter decisions are then aggregated in some fashion. This method is in principle very generic: it may be applied both to large continuous and discrete state and/or action spaces and cope with a large class of performance criteria. Earlier results in continuous action spaces and aggregation by averaging [28], and those in [25] focusing on small discrete action spaces and aggregation by majority vote, illustrate the promising character of the method.

The present paper extends this approach to large discrete action spaces by using kernel-based [29] aggregation schemes exploiting the problem structure. The paper is organized as follows: Section II defines the class of problems that we address and discusses the three main ingredients of our algorithm, namely randomized disturbance tree generation, computation of their optimal first stage decisions by the cross-entropy method [30], and finally the kernelized decision aggregation scheme; Section III shows how to use these ideas in practice on a test problem with a large and structured action space; finally, Section IV discusses our work with respect to related approaches in stochastic programming and planning under uncertainties. Section V concludes and outlines directions of further research.

The authors are with the Department of Electrical Engineering and Computer Science of the University of Liège (Belgium) (Email: {Boris.Defourny, dernst, L.Wehenkel}@ulg.ac.be).

Damien Ernst is a Research Associate of the Belgian FNRS of which he acknowledges the financial support.

II. Methods

In this section we outline the principle of the proposed approach, while briefly discussing the main underlying assumptions. We start by formally describing the class of problems addressed and then provide a schematic overview of the main ingredients of the proposed solution approach, namely the procedure for generating ensembles of disturbance trees, an algorithm for computing optimal first-stage decisions from them based on the cross-entropy method, and finally a discussion of the way for structuring the space of candidate decisions by using kernel-based methods and how to use this for the aggregation of first-stage decisions derived from an ensemble of trees.

A. Basic algorithm and assumptions (adapted from [25])

1) Problem formulation – assumptions: 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 an initial state x_0. Its trajectories are controlled by the decisions $u_t \in U$ that are to be optimized, and perturbed by disturbances $w_t \in W$ which are chosen by a memory-less and exogenous process (i.e. independently of each other and of previous states and decisions) from a probability distribution $\mathbb{P}_{t,w}$. A reward process is defined by $r_t = r_t(x_t, u_t, w_t)$ for $0 \leq t < T$. The system dynamics $f_t(\cdot)$, reward function $r_t(\cdot)$, disturbance model $\mathbb{P}_{t,w}$, as well as the initial state x_0 are supposed to be known beforehand, and the goal is to find a non anticipative (see below for a precise definition) decision strategy μ choosing the actions u_t maximizing the expectation of the sum of the rewards over T stages, i.e.

$$J^*(x_0) = \max_{\mu} \mathbb{E}\{\sum_{t=0}^{T-1} r_t(x_t, u_t, w_t)|x_0\}. \tag{1}$$

The candidate strategies μ for selecting the decisions u_t at any time $0 \leq t < T$ consist of a time-indexed set of deterministic mappings μ_t, each one projecting a current history $h_t = [w_0, w_1, \ldots, w_{t-1}]$ of the disturbance process to decisions $u_t = \mu_t(h_t) \in U$ at time t.

Notice that in this paper we make no assumptions about the size or structure (e.g. finiteness) of the state space X, while we restrict the space U of possible actions and the space W of possible disturbances to be finite, but they may possibly be of very large size. On the other hand, the assumption of a memory-less disturbance process is made only to facilitate the connection with the dynamic programming framework (see [25]). This latter assumption, as well as the assumption that the disturbance process is exogenous may indeed be relaxed, provided that at the decision making steps sufficient information is available to predict correctly the probabilities of all future disturbance sequences. Furthermore, the decomposable nature of the performance criterion of Eqn. (1), while fundamental in dynamic programming frameworks, is not essential in the approach proposed in this paper.

2) Exact solution based on a complete disturbance tree: A complete disturbance tree of depth T represents all the possible outcomes of the process $w_0, w_1, \ldots, w_{T-1}$ together with their probabilities of occurrence. In such a tree, the root node (at depth 0) corresponds to $t = 0$ and has an empty disturbance process history and a probability of one associated to it. To each node n of depth $t \in (0; T]$ in the tree corresponds a history $h_n = [w_0, \ldots, w_{t-1}]_n$ of the process, through the unique path from the root to the node n. The disturbance $(w_{t-1})_n$ is directly associated to node n together with its probability of occurrence, while $[w_0, \ldots, w_{t-2}]_n$ and their joint probability can be collected from the disturbances and probabilities associated to the nodes in the path.

Any strategy μ can be mapped on the tree by associating to each node n of depth $0 \leq t < T$ the value $u_n = \mu(h_n)$. Consequently, searching for an optimal strategy becomes equivalent to jointly optimizing the values u_n attached to all internal nodes of the tree.

The performance criterion defined by the expectation in (1) can be computed in a forward way once the vector of node decisions has been chosen. Indeed, from the given value of x_0, $u_0 = \mu_0$ and a particular w_0, one may compute $r_0 = r_0(x_0, u_0, w_0)$ and $x_1 = f_0(x_0, u_0, w_0)$ by exploiting the model. The values r_0 and x_1 can thus be assigned to the node associated to $[w_0]$; its probability $\mathbb{P}_{0,w}(w_0)$ may also be determined from the disturbance process model. From the node decision for $u_1 = \mu_1(w_0)$ and using x_1 and a particular w_1 one gets x_2 and r_1. The value r_1 can be assigned to the node corresponding to $[w_0, w_1]$, and has probability $\mathbb{P}_{0,w}(w_0)\mathbb{P}_{1,w}(w_1)$. The propagation continues up to x_T, r_{T-1}. It can be done through all disturbance paths of the tree. Therefore, for a given decision strategy μ, all the rewards and probabilities in the expectation in (1) are directly computable in a forward way from the system model $(f_t(\cdot), r_t(\cdot), \mathbb{P}_{t,w})$ and the initial state x_0.

Notice that, in a more general setting, the performance criterion would be expressed as a non decomposable function of the vector of variables $[w_n, u_n, x_n]$ of the tree; its computation would also be based on a forward pass used to first determine the values of x_n at each node as a function of the w_n and u_n variables.

The optimization of μ itself may be done directly by searching over a vector of all possible combinations of values of decisions u_n at the internal nodes of the disturbance tree. However, the number of possible values of this vector of decision variables is of $O(|U|^{|W|^{T-1}})$, which means that as soon as $|U|$, $|W|$ or $|T|$ are a bit large, an exact optimization becomes intractable.

3) Approximate solution based on an ensemble of incomplete disturbance trees: Conceptually, an incomplete disturbance tree is obtained by selecting a subset of the nodes of a complete tree, by removing the arcs leading to these nodes as well as the subtrees emanating from them and by adjusting probabilities of successor nodes so that they sum to one (see §II-B), with the restriction that each node of depth $< T$ in the resulting incomplete tree has

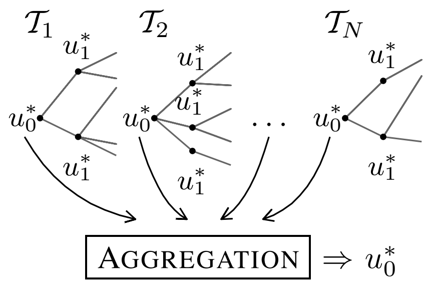


Fig. 2. Sketch of the method. The sequential decision making problem is posed on randomly generated trees $\mathcal{T}_i$. Each tree is solved separately. The first-stage decisions u_0^* extracted from the solutions are aggregated into a definitive first-stage decision u_0^*.

at least one successor. In practice, incomplete trees may be constructed top-down, for example by subsampling the disturbance process according to its probabilities, with the constraint that their total complexity remains manageable in terms of the solution process of the optimization problem.

While the optimization of the decision variables associated to the nodes of an incomplete disturbance tree will only permit an incomplete description of a decision strategy (because the nodes are representative only of a subset of the possible histories), such a strategy always provides a value for the first stage decision u_0. Thus, one could replace the original problem of solving a complete disturbance tree (generally intractable), by the combined problems of determining an optimal incomplete tree and optimally solving the latter.

Instead of this approach, we propose here to subsample the ensemble of incomplete trees and compute u_0 by aggregating their first stage decisions. We thus assume that the information about the optimized first stage decisions of incomplete trees is valuable, and that we know how to aggregate them (see §II-D) and can use this aggregated value as a useful approximation of the optimal first stage decision that would have been determined from the complete tree (see Figure 2).

Thus, we suggest that the solution of the original sequential decision problem may be approached by a combination of three ideas, namely the generation of an ensemble of incomplete trees according to some approppriate subsampling scheme of successors combined with an appropriate node reweighting scheme and optimization method, the proper aggregation of first stage decisions obtained by optimally solving each one of these incomplete trees, and the receeding horizon approach which consists in re-optimizing (according to the same approach) the optimal value of first stage decisions as time flows.

The main ingredients of this approach are further discussed below, while focusing on decision problems with large discrete action and disturbance spaces.

B. Generation of an ensemble of incomplete trees

In our proposal, the generation of an ensemble of incomplete disturbance trees is based on random sampling of a small number of successors, in a top-down fashion. Given the complexity constraint on incomplete trees, their internal nodes have usually only a few children nodes, representing explicitly only a small subset V of W. Since we attempt to optimize an expected reward-to-go at each node, say Q_V instead of Q_W, in the hope that the decision maximizing Q_V is also near-optimal for Q_W, it is important to attach probabilities to the tree branches in the best possible way.

Without any a priori knowledge about the good decisions and state trajectories, we want to limit the approximation error uniformly over the possible decisions and states. In the proposed framework, V is determined by sampling. But how to wisely choose the probabilities of the elements W_i in V? Indeed, V now stands for the support of a new distribution that approximates the original distribution of larger support W. Not much can be said on the approximation errors (and on the justification of the approximation scheme in the first place) without assuming that disturbances that are close in a certain sense also yield close rewards-to-go. Now, under this assumption, if we put a metric over the disturbance space and redistribute the probability of a disturbance not in V among the closest disturbances present in V, then in the computation of an expected reward-to-go, this amounts to replace some rewards-to-go by (presumably) close rewards-to-go, and thus reduce the approximation error between Q_V and Q_W.

In particular, we can exploit a kernel $k(w, w') : W \times W \rightarrow \mathbb{R}$ to define the distance between disturbances by $d(w, w') = (k(w, w) + k(w', w') - 2k(w, w'))^{1/2}$, so as to associate every $w \in W \setminus V$ to its nearest neighbor(s) in V, and compute the probability associated to elements of V by

$$\hat{P}_n(v) = \mathbb{P}_{t,w}(v) + \textstyle\sum_{w' \in C^{-1}(v)} \mathbb{P}_{t,w}(w')/|C(w')| \ , \quad (2)$$

where $|C(w')|$ denotes the size of the subset of V of nearest neighbors of w' and where $C^{-1}(v)$ denotes the subset of elements of $W \setminus V$ of which v is a nearest neighbor. These ideas will be illustrated in Section III.

C. Optimization with the Cross-Entropy method

Because we want to address a class of non-convex combinatorial optimization problems, we propose to use the Cross-Entropy method [30] as a tool to solve the optimization problem associated to each incomplete tree. This algorithm is based on two components: first, a random generator $G(\theta)$ with parameters θ for sampling candidate solutions (the parametrization is such that the distribution can be uniform or degenerate into any deterministic solution); second, a scoring device $S(\mu)$ for scoring a candidate solution μ, represented here by a vector of node variables u_n.

Starting from a uniform distribution, one generates N samples of candidate solutions, scores them, and tags as elite solutions those with a score at least equal to the $\lceil N\rho \rceil$-th largest score, with ρ small, say 10^{-2}. The parameter θ is then updated to maximize the probability of generating the elite solutions, and a new set of N samples is drawn by using the update value of θ in the random generator. The steps are repeated until the distribution degenerates or the elite scores have ceased to improve, and the best candidate solution is then returned. Usually one chooses N proportional to the number of parameters in θ, itself being a function of the size of the search space. Also, the update of θ is often smoothed as $\theta_{t+1} = \alpha\tilde{\theta} + (1-\alpha)\theta_t$ where $\tilde{\theta}$ is the elite probability maximizer, and $0 < \alpha \leq 1$ the smoothing parameter.

A key to the successful application of the Cross-Entropy method lies in finding a parametrization of the solution generator such that those near-optimal parts of a solution that have a positive effect on the score can be made more likely to occur. Often one tries to exploit some decomposability property of the score. This will be illustrated in Section III.

D. Aggregation of optimal first stage decisions

Let $u_i^* \in U$ be a near-optimal *first stage* decision obtained from an incomplete tree i. The tree i approximates the intractable complete tree having an optimal first stage decision $u^* \in U$. Consider a set $S^* = \{u_1^*, \dots, u_m^*\}$ of near-optimal first stage decisions from m incomplete trees. We ask the question: can we find an estimate $\hat{u}^*$ of some $u^* \in U$, on the basis of the set S^*?

Each element in S^* is near-optimal with respect to an objective defined by a complex process, and satisfies possibly complex constraints. So in that sense the elements in S^* are structured, inherit optimality properties from the underlying approximate problems, and through these approximations (or *weak models*), optimality properties from the original problem. In the spirit of “perturb and combine” methods, $\hat{u}^*$ could be formed by aggregating the decisions in S^*. But the challenge lies in the discovery or at least the preservation of properties of u^* present in the elements of S^*.

To do so, we propose to take for $\hat{u}^*$ the decision in S^* closest to the *centroid* of the decisions in S^* with respect to a given metric induced on U by a kernel $k(\cdot,\cdot)$ measuring the similarities of pairs of decisions over $U \times U$. Formally, let $\varphi : U \rightarrow H$ be the feature-map from the decision space to the Hilbert space induced by the kernel $k(\cdot,\cdot)$. Define the centroid of S^* in that Hilbert space as $\varphi(\bar{u}) = m^{-1}\sum_{i=1}^m \varphi(u_i^*)$. With $k(u,u') = \langle\varphi(u), \varphi(u')\rangle$ denoting the inner product, the squared distances between the centroid and some solution u are thus evaluated as

$$||\varphi(u) - \varphi(\bar{u})||_2^2 = \langle\varphi(u), \varphi(u)\rangle - \\ 2m^{-1}\sum_{i=1}^m\langle\varphi(u), \varphi(u_i^*)\rangle + m^{-2}\sum_{i=1}^m\sum_{j=1}^m\langle\varphi(u_i^*), \varphi(u_j^*)\rangle \ .$$

Consequently, the squared distance from $u_k^* \in S^*$ to the centroid $\bar{u}$ may be expressed directly in terms of the elements of the Gram matrix by

$$||\varphi(u_k^*) - \varphi(\bar{u})||_2^2 = \\ K_{kk} - 2m^{-1}\sum_{i=1}^m K_{ik} + m^{-2}\sum_{i=1}^m\sum_{j=1}^m K_{ij} \ , \tag{3}$$

where the element K_{ij} of the Gram matrix $K \in \mathbb{R}^{m\times m}$ is defined by $k(u_i^*, u_j^*) = \langle\varphi(u_i^*), \varphi(u_j^*)\rangle$.

The aggregated solution

$$\hat{u}^* \in \arg\min_{u_k^* \in S^*} ||\varphi(u_k^*) - \varphi(\bar{u})||_2^2 \tag{4}$$

may thus also be computed directly from the Gram matrix. Let us also notice that the kernelized variance $V_{S^*} \triangleq m^{-1}\sum_{k=1}^m ||\varphi(u_k^*) - \varphi(\bar{u})||_2^2$, which could serve as a measure of discrepancy between candidate decisions in S^*, may also be directly computed from the Gram matrix.

Discussion: First consider the case where U only possesses a handful of elements. There is no complex structure there, and thanks to the small cardinality of U, optimal elements u^* should be present in the set S^* of candidate solutions. Therefore a simple majority vote among the elements of S^* can be used as the estimate $\hat{u}^*$ of an optimal decision. Notice that the majority vote can be obtained from the general formulation (4) by setting $K_{ij} = \delta\{u_i^* = u_j^*\}$, where $\delta\{\cdot\}$ denotes the 0-1 indicator function. Indeed, the squared distances $||\varphi(u_k^*) - \varphi(\bar{u})||_2^2$ will differ by the term $-2m^{-1}\sum_{i=1}^m K_{ik}$, where the sum of indicators will amount to count the number of solutions in S^* identical to u_k^*.

Now consider the case where U is still finite but has a number of elements much larger than m, the size of S^*. It is then very likely that a clear majority will not be attained in S^*, especially if there are many quasi-equivalent decisions in terms of optimality. However, a large U is likely to be formed from the combination of several elementary decisions, e.g. $U = U_1 \times \ldots \times U_d$. One could thus combine kernels on the elementary decision spaces U_l, e.g. by summing majority vote kernels made of indicator functions on each elementary decision : $K_{ij} = \sum_{l=1}^d \delta\{\Pi_l(u_i^*) = \Pi_l(u_j^*)\}$ where Π_l acts as a projection operator from U to the subspace U_l.

Besides other variations along this line, the kernelization of the decision space provides a way of injecting knowledge on the structure of the decision space. Typically, one can define kernels which are invariant with respect to classes of decisions which are known to be invariant in terms of their impact on the problem performance.

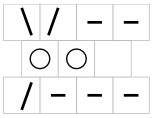
III. Experiment

In this section we illustrate the proposed approach on a benchmark problem which has a large, structured, discrete action space. We explain in detail how the action space is kernelized, how incomplete disturbance trees are generated and optimized, and provide an assessment of the decision strategy obtained with our approach in comparison to the exact solution of this benchmark problem.

A. Description of the benchmark

The benchmark *SensorNetwork* originates from [31] and was part of the NIPS 2005 benchmarking event in reinforcement learning. The problem consists in two arrays of sensors bracketing an array of three target cells. Two targets float over the target cells (Fig. 3). The targets start at energy level 3. At each step, a sensor can focus on the cell to its left or to its right, or be idle. Once the action of each sensor is set, the targets can move; in turn (from left to right), a target that is adjacent to at least one empty cell uniformly randomly chooses to try to move left, to try to move right, or to stay in its current cell – if the cell it tries to move into is empty, it succeeds. At that moment the focused sensors are activated.

A target in a cell hit by 3 sensors or more loses one energy point. A target is killed when its energy falls to zero. The goal is to eliminate the targets as soon as possible.



(configuration written as

$x = [3\ 3\ 0]$

$u = \begin{array}{l}\backslash/\text{--}\,\text{-}\\ /\text{--}\,\text{-}\,\text{-}\end{array}$

for targets with energy 3 each)

Fig. 3. SensorNetwork with its targets (o), focused sensors (/ or \) and idle sensors (-).

The state space is $X = \{0, 1, 2, 3\}^3$ for the target energy level (0 to 3) of the 3 cells. When a target moves from one cell to another, these cells swap their energy level. The initial state is either [3 3 0], [3 0 3], [0 3 3], i.e. 2 targets of energy 3. The state [0 0 0] with no remaining targets is terminal. The decision space is $U = \{0, 1, 2\}^8$ for the 3 possible actions (0: idle, 1: focus left, 2: focus right) for the 8 sensors, totalling 6561 possible actions. There is a reward -1 for each sensor focus, and one of $+30$ for killing a target. We put a discount factor $\gamma = 0.95$, and limit the time horizon to $T = 10$.

B. Disturbance tree node probabilities

The disturbances are the joint tried moves of the two targets: $W = \{W_1, \ldots, W_9\}$ and $\mathbb{P}_{w,t}(W_i) = 1/9$. A disturbance has the form $[\Delta_L\ \ \Delta_R]$, with Δ_L the tried move of the leftmost target (3 same possibilities: left, right, stay), and Δ_R is the tried move of the rightmost target (3 possibilities). The part Δ_R is ignored when there is only one target left. It is convenient to define the operator Π_L (resp. Π_R) that extracts the part Δ_L (resp. Δ_R) of the disturbance.

We will assume that if W_i and W_j have the same Δ_L or the same Δ_R, they are closer in terms of induced state transitions and rewards compared to the case of disturbances with no move in common. Thus we will use a kernel on the disturbance space that induces such distances:

$$k(w, w') = \delta\{\Pi_L(w) = \Pi_L(w')\} + \delta\{\Pi_R(w) = \Pi_R(w')\}\ .$$

C. Cross-entropy based optimization

The sampling distribution for candidate solutions μ for a given disturbance tree of $\#n$ decision nodes is factored into $\#n$ independent components, one per node. The $\#n$ components are themselves decomposed into 8 independent parts corresponding to the 8 sensors. Each part defines the distribution over $\{0, 1, 2\}$ of the action a_{ij} of a sensor j at a node i:

$$\begin{aligned}&\mathbb{P}\{a_{ij} = 0\} = p_{ij} \quad \mathbb{P}\{a_{ij} = 1\} = q_{ij}\\ &\mathbb{P}\{a_{ij} = 2\} = 1 - p_{ij} - q_{ij}\ .\end{aligned}$$

The distribution of the solution $\mu \in \mathbb{R}^{\#n}$ (actions at node i are packed into a single code for the resulting node decision, hence the dimension of μ) is thus specified by $2 \cdot 8 \cdot \#n$ parameters. At the beginning, $p_{ij} = q_{ij} = 1/3$ (uniform distribution). In fact the decomposition in elementary actions is coherent with the kernelization of the decision space U described in Section III-D.

Once elite samples $\mu^{(1)}, \mu^{(2)}, \ldots, \mu^{(L)}$ are identified on the basis of the associated expected discounted sum of rewards computed on the disturbance tree, the parameters are updated by the rule

$$\begin{aligned}&p_{ij} \leftarrow \alpha\, \tilde{p}_{ij} + (1 - \alpha)\, p_{ij} && q_{ij} \leftarrow \alpha\, \tilde{q}_{ij} + (1 - \alpha)\, q_{ij}\\ &\tilde{p}_{ij} \triangleq L^{-1} \textstyle\sum_{l=1}^{L} \delta\{a_{ij}^{(l)} = 0\} && \tilde{q}_{ij} \triangleq L^{-1} \textstyle\sum_{l=1}^{L} \delta\{a_{ij}^{(l)} = 1\}\ .\end{aligned}$$

Note that $\tilde{p}_{ij}$ and $\tilde{q}_{ij}$ are such that a so-updated distribution would match the empirical frequencies of the elementary actions in the elite samples.

The Cross-Entropy optimization is stopped when the sampling distributions corresponding to the actions relative to the root node are almost deterministic, because ultimately only these actions are extracted and used in the aggregation step.

D. Aggregation of optimal first stage decisions

The aggregation scheme will exploit the decomposition of the decision space into separate sensor actions. The action of sensor l in the centroid decision $\bar{u}$ is chosen by a majority vote over the action of sensor l in the first stage decisions collected in S^*. Following Section II-D, the Gram matrix of a kernel inducing such a vote is (with $\Pi_l(u)$ denoting the action of sensor l extracted from a first stage decision u, and $u_i^*, u_j^* \in S^*$)

$$K_{ij} = \textstyle\sum_{l=1}^{8} \delta\{\Pi_l(u_i^*) = \Pi_l(u_j^*)\}\ . \tag{5}$$

Notice that even more knowledge could have been injected, for example by considering classes of equivalent decisions based on the target cells hit by 3 or more focused sensors. To this end, one could use $\tilde{K}_{ij} = \sum_{l=1}^{3} \delta\{\tilde{\Pi}_l(u_i^*) = \tilde{\Pi}_l(u_j^*)\}$ with $\tilde{\Pi}_l(u)$ the indicator function for the event "The cell l is hit by at least 3 sensors" according to decision u.

E. Numerical simulations

Table I reports typical outcomes of the approach with five incomplete trees (reporting their number of decision nodes $\#n$) and 3 initial states. For each decision node, 3^8 actions are possible. The optimization follows Section II-C, using $N = 32\ \#n$ samples per iteration, i.e. twice the dimension of θ. The smoothing is set to $\alpha = 0.6$. The optimization stops as soon as the 8 actions at the root have their distribution concentrated on an action with probability 0.99.

The first stage decision extracted for each tree k ($\hat{u}_k^*$) is reported pictorially. Note that the centroid reported pictorially at the left of $\hat{u}^*$ need not be computed in practice, as $\hat{u}^*$ directly comes from (4). The value $\hat{V}_k$ denotes the optimized expected discounted sum of rewards. Incomplete trees were grown by sampling with replacement m disturbances at each node. Distinct samples are taken as the children of the node, and assigned a probability (cf. Section II-B). The number m is random. For a node of depth d, $m = 3$ samples are drawn with probability $1/(1 + d)$, and $m = 1$ samples with probability $1 - 1/(1 + d)$. This reduces the expected tree size. Also, trees of more than 150 nodes were rejected. In

x_0		Disturbance trees					Centroid	$\hat{u}^*$
[3 3 0]	$\#n$	27	55	95	85	80	`4424`	
	u_k^*	`\//-` `//\-`	`\/--` `/---`	`\\/-` `/\\-`	`-/\/` `/\/-`	`\/\-` `/-/\`	`\//-` `/-\-`	`\//-` `//\-`
	$\hat{V}_k$	35.19	34.10	33.96	32.93	34.92	`5224`	
[3 0 3]	$\#n$	78	94	47	76	24	`4444`	
	u_k^*	`-/\-` `/\/\`	`\/\/` `-\/-`	`\/\-` `-\/\`	`\-\-` `/\/\`	`\//-` `//\-`	`\/\-` `/\/\`	`\/\-` `-\/\`
	$\hat{V}_k$	34.10	37.03	34.53	33.53	34.03	`3443`	
[0 3 3]	$\#n$	116	100	93	71	58	`3335`	
	u_k^*	`\/-/` `/-/\`	`-\\/` `-/\\`	`-\\/` `-/\\`	`\-\/` `/\-\`	`-\//` `-//\`	`-\\/` `-/\\`	`-\\/` `-/\\`
	$\hat{V}_k$	36.31	33.16	34.21	32.28	34.95	`3443`	

Tests on 3 initial conditions x_0.
For each tree, $\#n$: number of decision nodes; u_k^* : root decision; $\hat{V}_k$: root value.
Last 2 columns: centroid with the 8 counts (2 to 5) of dominant actions,
and $\hat{u}^*$: implemented decision.

TABLE I

TYPICAL RESULTS WITH AN ENSEMBLE OF 5 TREES.

	`--\-` `--/\`	`-\--` `-/\-`	`-\\/` `-/\\`	`-/--` `/\--`	`\-\-` `/\/\`	`\\/-` `/\\-`
[3 3 0]	26.33	27.82	26.28	28.57	27.19	$\boxed{30.08}$
[3 0 3]	27.72	27.13	$\boxed{27.88}$	27.52	27.78	27.50
[0 3 3]	28.36	28.62	$\boxed{30.20}$	26.60	26.89	27.30

Optimal values are boxed.
Equivalent decisions are those hitting the same 1 or 2 cells with 3 sensors per cell.
States [3 3 0] and [0 3 3] are not exactly symmetric due to the priority rule in the movements of the targets.

TABLE II

REFERENCE: EXACT EXPECTED VALUE OF (x_0,u_0) PAIRS (FOR $T = 10$, γ = 0.95).

contrast, the complete tree on the $|W| = 9$ disturbances has $3.9 \cdot 10^9$ nodes.

The benchmark can be solved exactly using the value iteration algorithm [6]. It turns out that 6 useful classes of sensor configurations suffice for optimally targeting the cells. Table II gives the exact expected value of a decision from each class, as if it were used as first stage decision and that at subsequent steps optimal decisions were selected. The comparison with Table I shows that the proposed approach has missed for $x_0 = [3\ 0\ 3]$ the exact u^*. However the selected decision is the second-best and is still near-optimal (value 27.78 instead of 27.88).

In general, a direct majority vote over the sensor actions destroys the structure of optimal, or at least efficient, decisions: Table I shows centroids with 2 or 4 sensors targeted to a same cell, instead of 0 or 3. This is not the case of the projection of the centroid on the set of decisions in the ensemble.

Other tests with larger trees or ensembles, or other initial conditions were consistent with the reported results. We repeated 10 times the experiment of building an ensemble of 5 trees and computing the aggregated decision. An optimal decision was found: 7 times for $x_0 = [3\ 3\ 0]$, 9 times for $x_0 = [0\ 3\ 3]$, and 5 times for $x_0 = [3\ 0\ 3]$ with a second-best decision found in the 5 other cases.

F. Benefits of the kernelization

The approach using an ensemble of incomplete disturbance trees of [25] relied on a majority vote for aggregating the optimal first stage decisions computed from these trees. By applying the kernel-based decision aggregation scheme to the SensorNetwork Benchmark, we found out that we could obtain performances similar to those obtained with the majority voting approach, while using a set of incomplete disturbance trees which was several orders of magnitude smaller.

The kernel-based probability imputation over disturbances allows to produce incomplete trees which represent in a better way the original disturbance process than those built with the approach of [25] which uses empirical sampling frequencies to determine node probabilities. We have indeed observed that to obtain near-optimal solutions, much smaller trees can be built with the proposed kernel-based probability imputation scheme than with the empirical weighting scheme of [25].

IV. RELATED WORK

The ensemble of disturbance trees approach is inspired by the "perturb and combine" paradigm of supervised learning [26], [27], already employed in the search for closed-loop decision rules [32], [33]. Works on game tree evaluation tracing back to Monte Carlo Go [34] suggest that information on optimal decisions can be obtained from suboptimal or random sequences of decisions. Several other works support the view that subsampling schemes can perform well. In [35], the authors build trees with nodes corresponding to disturbance-decision pairs in a Markov Decision Process, and analyze a sparse sampling strategy of the disturbance space. Random [9] or deterministic [19], [36], [37], [4] subsampling schemes for building scenario trees in the context of stochastic programming have been studied, along with their consistency, mostly for continuous distributions and convex problems. Also for convex problems, authors studied arithmetic means of solutions based on a unique scenario [38] or a scenario tree [39], [28].

Kernelization of output spaces is currently a very active research topic in the context of supervised learning. Various approaches for learning with output space kernels have been recently proposed which could possibly be exploited in the context of the proposed approach [40], [41].

V. Conclusions

This paper has proposed an approach for leveraging the ensemble of disturbance trees framework [25] to problems having complex discrete action spaces. The proposal relies on a kernelization of the action space which is used for the purpose of aggregating decisions derived from different disturbance trees generated by random sampling. It has been illustrated on the so-called SensorNetwork [31] benchmark, demonstrating its applicability in the context of large structured discrete action spaces.

Given the excellent practical performances of this approach, we believe that it would be wishful to characterize its theoretical properties. To begin, the ability of optimized decisions to “overfit” a subset of disturbance paths suggests that the average of the optimal values obtained from incomplete disturbance trees is an optimistically biased estimator of the true optimal value. Thus, it would be useful to study how the bias and variance of this estimator depend on the specificities of the algorithm used for generating the ensemble of incomplete trees. It would also be useful to obtain a pessimistically biased estimator, so as to bracket the true optimal value.

Kernelization of action spaces seems to be a rather powerful approach to incorporate problem domain knowledge in the context of ensemble based multistage stochastic programming. We thus plan to further investigate this approach in the context of a broader variety of practical problems, concerning complex discrete as well as continuous action spaces.

Further analysis is also needed in order to more precisely delineate the features of this approach with respect to stochastic dynamic programming and other reinforcement learning approaches. We believe that these analyses should be more specifically oriented towards the specifications of the basic assumptions underlying these frameworks, such as their information structure as well as the mathematical forms of their performance criteria.

While we have up to now focused on extracting only an approximation of the optimal first stage decision from the ensemble of optimized incomplete trees, it is in principle straightforward to extend the proposed algorithm so as to exploit the full set of history-action pairs at all internal nodes of the trees to extract a “complete” approximate decision policy applicable to the successive stages of the planning problem. This could for example be achieved by using supervised learning methods using kernelized output spaces [41] in order to map histories to decisions obtained at the convergence of the cross-entropy optimization process, or by applying batch-mode reinforcement learning algorithms [33] to subsets of the intermediate solutions produced by the cross-entropy optimization process. This would for example allow one to use Monte Carlo simulation on an independent set of scenarios so as to efficiently compute estimates of optimality of any such “complete” decision strategy produced by the different variants of the approach. In particular, this would allow one to compare in a systematic way different types of kernels, and different values of the other parameters of the method, such as incomplete tree sampling schemes and the size of the tree ensemble, so as to adapt automatically their choice to the characteristics of the problem at hand.

Acknowledgments

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] J. Birge and F. Louveaux, *Introduction to Stochastic Programming*. New York: Springer-Verlag, 1997.

[2] J. Dupacova, “Stability and sensitivity analysis for stochastic programming,” *Annals of Operations Research*, vol. 27, pp. 115–142, 1990.

[3] J. Dupacova, G. Consigli, and S. Wallace, “Scenarios for multistage stochastic programs,” *Annals of Operations Research*, vol. 100, pp. 25–53, 2000.

[4] R. Hochreiter and G. Pflug, “Financial scenario generation for stochastic multi-stage decision processes as facility location problems,” *Annals of Operations Research*, vol. 152, pp. 257–272, 2007.

[5] R. Rockafellar, “Optimization under uncertainty,” Lecture Notes, University of Washington, 2001.

[6] D. Bertsekas, *Dynamic Programming and Optimal Control*, 3rd ed. Belmont, MA: Athena Scientific, 2005.

[7] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, 2004.

[8] M. Dempster, “Sequential importance sampling algorithms for dynamic stochastic programming,” *Annals of Operations Research*, vol. 84, pp. 153–184, 1998.

[9] A. Shapiro, “Monte Carlo sampling methods,” in *Stochastic Programming. Handbooks in Operations Research and Management Science*, A. Ruszczyński and A. Shapiro, Eds. Amsterdam: Elsevier, 2003, vol. 10, pp. 353–425.

[10] H. Heitsch, W. Römisch, and C. Strugarek, “Stability of multistage stochastic programs,” *SIAM Journal on Optimization*, vol. 17, no. 2, pp. 511–525, 2006.

[11] P. Carpentier, G. Cohen, and J. Culioli, “Stochastic optimization of unit commitment: A new decomposition framework,” *IEEE Transactions on Power Systems*, vol. 11, no. 2, pp. 1067–1073, 1996.

[12] J. Kallrath, P. Pardalos, S. Rebennack, and M. Scheidt, Eds., *Optimization in the Energy Industry*, ser. Energy Systems. Springer, 2008.

[13] D. Carino, R. Myers, and W. Ziemba, “Concepts, technical issues and uses of the Russel-Yasuda Kasai financial planning model,” *Operations Research*, vol. 46, pp. 450–462, 1998.

[14] A. Zenios and W. Ziemba, Eds., *Applications and Case Studies*, ser. Handbook of Asset and Liability Management. North Holland, 2007, vol. II.

[15] J. Gondzio and A. Grothy, “Solving nonlinear financial planning problems with 10^9 decision variables on massively parallel architectures,” in *Computational Finance and its Applications II, WIT Transactions on Modelling and Simulation*, M. Costantino and C. Brebbia, Eds., vol. 43, 2006.

[16] A. Eichhorn and W. Römisch, “Polyhedral risk measures in stochastic programming,” *SIAM Journal on Optimization*, vol. 16, no. 1, 2005.

[17] L. Cheng, E. Subrahmanian, and A. Westerberg, “A comparison of optimal control and stochastic programming from a formulation and computation perspective,” *Computers and Chemical Engineering*, vol. 29, no. 1, pp. 149–164, 2004.

[18] D. Kuhn, “Aggregation and discretization in multistage stochastic programming,” *Mathematical Programming, Ser. A*, vol. 113, pp. 61–94, 2008.

[19] K. Høyland and S. Wallace, “Generating scenario trees for multistage decision problems,” *Management Science*, vol. 47, no. 2, pp. 295–307, 2001.

[20] J. Dupacova, N. Gröwe-Kuska, and W. Römisch, “Scenario reduction in stochastic programming, An approach using probability metrics,” *Math. Program., Ser. A*, vol. 95, pp. 493–511, 2003.

[21] S. Vigerske and I. Nowak, “Adaptive discretization of convex multistage stochastic programs,” *Math. Meth. Oper. Res.*, vol. 65, pp. 365–383, 2007.

[22] M. Morari and J. Lee, “Model predictive control: past, present and future,” *Computers and Chemical Engineering*, vol. 23, pp. 667–682, 1999.

[23] R. Kouwenberg, “Scenario generation and stochastic programming models for asset liability management,” *European Journal of Operational Research*, vol. 134, no. 2, pp. 279–292, 2001.

[24] N. Sahidinis, “Optimization under uncertainty: state-of-the-art and opportunities,” *Computers and Chemical Engineering*, vol. 28, no. 6-7, pp. 971–983, 2004.

[25] B. Defourny, D. Ernst, and L. Wehenkel, “Lazy planning under uncertainty by optimizing decisions on an ensemble of incomplete disturbance trees,” in *Recent Advances in Reinforcement Learning, 8th European Workshop, EWRL’08*, LNCS (LNAI) 5323. Springer, 2008.

[26] R. Schapire, “The strength of weak learnability,” *Machine Learning*, vol. 5, no. 2, pp. 197–227, 1990.

[27] L. Breiman, “Bagging predictors,” *Machine Learning*, vol. 24, no. 2, pp. 123–140, 1996.

[28] B. Defourny and L. Wehenkel, “Averaging decisions from an ensemble of scenario trees: a validation on newsvendor problems,” 2008, submitted.

[29] J. Shawe-Taylor and N. Cristianini, *Kernel Methods for Pattern Analysis*. Cambridge University Press, 2004.

[30] R. Rubinstein and D. Kroese, *The Cross-Entropy Method. A Unified Approach to Combinatorial Optimization, Monte-Carlo Simulation, and Machine Learning*, ser. Information Science and Statistics. Springer, 2004.

[31] S. Ali, S. Koenig, and M. Tambe, “Preprocessing techniques for accelerating the DCOP algorithm ADOPT,” in *AAMAS*, 2005.

[32] R. Sutton, “Generalization in reinforcement learning: successful examples using sparse coarse coding,” *Advances in Neural Information Processing Systems*, vol. 8, pp. 1038–1044, 1996.

[33] D. Ernst, P. Geurts, and L. Wehenkel, “Tree-based batch mode reinforcement learning,” *Journal of Machine Learning Research*, vol. 6, pp. 503–556, April 2005.

[34] B. Brügmann, “Monte Carlo Go,” Syracuse University, Tech. Rep., 1993.

[35] M. Kearns, Y. Mansour, and A. Ng, “A sparse sampling algorithm for near-optimal planning in large Markov decision processes,” *Machine Learning*, vol. 49, no. 2-3, pp. 193–208, 2002.

[36] S. Rachev and W. Römisch, “Quantitative stability in stochastic programming: The method of probability metrics,” *Mathematics of Operations Research*, vol. 27, no. 4, pp. 792–818, 2002.

[37] W. Römisch, “Stability of stochastic programming problems,” in *Stochastic Programming. Handbooks in Operations Research and Management Science*, A. Ruszczyński and A. Shapiro, Eds. Amsterdam: Elsevier, 2003, vol. 10, pp. 483–554.

[38] R. Rockafellar and R.-B. Wets, “Scenarios and policy aggregation in optimization under uncertainty,” *Mathematics of Operations Research*, vol. 16, pp. 119–147, 1991.

[39] Y. Nesterov and J.-P. Vial, “Confidence level solutions for stochastic programming,” *Automatica*, vol. 44, no. 6, pp. 1559–1568, 2008.

[40] J. Weston, B. Schoelkopf, and O. Bousquet, “Joint kernel maps,” in *Proc. of the 8th International Work-Conference on Artificial Neural Networks (Computational Intelligence and Bioinspired System)*, J. Cabestany, A. Prieto, and F. Sandoval, Eds., vol. LNCS 3512, 2005, pp. 176–191.

[41] P. Geurts, L. Wehenkel, and F. d’Alché-Buc, “Gradient boosting for kernelized output spaces,” in *ACM International Conference Proceeding Series (Proceedings of the 24th International Conference on Machine Learning)*, vol. 227, 2007, pp. 289–296.