Amsaa: A Multistep Anticipatory Algorithm for Online Stochastic Combinatorial Optimization

Luc Mercier and Pascal Van Hentenryck

Brown University, Box 1910, Providence, RI 02912, USA

Abstract. The one-step anticipatory algorithm (*1s-AA*) is an online algorithm making decisions under uncertainty by ignoring future non-anticipativity constraints. It makes near-optimal decisions on a variety of online stochastic combinatorial problems in dynamic fleet management, reservation systems, and more. Here we consider applications in which the *1s-AA* is not as close to the optimum and propose *Amsaa*, an anytime multi-step anticipatory algorithm. *Amsaa* combines techniques from three different fields to make decisions online. It uses the sampling average approximation method from stochastic programming to approximate the problem; solves the resulting problem using a search algorithm for Markov decision processes from artificial intelligence; and uses a discrete optimization algorithm for guiding the search.
Amsaa was evaluated on a stochastic project scheduling application from the pharmaceutical industry featuring endogenous observations of the uncertainty. The experimental results show that *Amsaa* significantly outperforms state-of-the-art algorithms on this application under various time constraints.

1 Introduction

In recent years, progress in telecommunication and in information technologies has generated a wealth of Online Stochastic Combinatorial Optimization (OSCO) problems. These applications require to make decisions under time constraints, given stochastic information about the future. Anticipatory algorithms have been proposed to address these applications [18]. We call an algorithm *anticipatory* if, at some point, it anticipates the future, meaning that it makes some use of the value of the clairvoyant. These anticipatory algorithms typically rely on two black-boxes: a conditional sampler to generate scenarios consistent with past observations and an offline solver for the deterministic version of the combinatorial optimization problem.

1s-AA is a simple one-step anticipatory algorithm. It works by transforming the multi-stage stochastic optimization problem into a 2-stage one by ignoring all non-anticipativity constraints but those of the current decision. This 2-stage problem is approximated by sampling, and the approximated problem is solved optimally by computing the offline optimal solutions for all pairs (scenario,decision). *1s-AA* was shown to be very effective on a variety of OSCO problems in dynamic fleet management [3, 2], reservation systems [18], resource allocation [15], and jobshop scheduling [17]. Moreover, a quantity called the global anticipatory gap (GAG) was introduced by [14] to measure the stochasticity of the application and that paper showed that *1s-AA* returns high-quality solutions when the GAG is small.

Here we consider OSCO applications with a significant GAG and propose to address them with *Amsaa*, a multi-step anticipatory algorithm which provides an innovative integration of techniques from stochastic programming, artificial intelligence, and discrete optimization. Like *1s-AA*, *Amsaa* samples the distribution to generate scenarios of the future. Contrary to *1s-AA* however, *Amsaa* approximates and solves the multi-stage problem. The SAA problem is solved by an exact search algorithm [4] using anticipatory relaxations as a heuristic to guide the search.

Amsaa was evaluated on a stochastic resource-constrained project scheduling problem (S-RCPSP) proposed in [6] to model the design and testing of molecules in a pharmaceutical company. This problem is highly combinatorial because of precedence and cumulative resource constraints. It is also stochastic: the durations, costs, and results of the tasks are all uncertain. The S-RCPSP features what we call *endogenous observations*: the uncertainty about a task can only be observed by executing it. This contrasts with online stochastic combinatorial optimization (OSCO) problems studied earlier, in which the observations were exogenous, and leads to significant GAGs [8]. More generally, *Amsaa* applies to a class of problems that we call *Stoxuno problems* (STochastic Optimization with eXogenous Uncertainty and eNdogenous Observations). The experimental results indicate that *Amsaa* outperforms a wide variety of existing algorithms on this application.

The rest of the paper is organized as follows. Sections 2 and 3 describe the motivating problem and introduce Stoxuno problems. Section 4 presents the background in Markov Decision Processes and dynamic programming. Section 5 introduces the concept of Exogenous MDPs (X-MDPs) to model Stoxuno and exogenous problems. Section 6 describes *Amsaa*. Section 7 presents extensive experimental results. Section 8 compares *Amsaa* with a mathematical programming approach. Section 9 concludes the paper and discusses research opportunities.

2 A Stochastic Project Scheduling Problem

This section describes the *stochastic resource-constrained project scheduling problem* (S-RCPSP), a problem from the pharmaceutical industry [6]. A pharmaceutical company has a number of candidate molecules that can be commercialized if shown successful, and a number of laboratories to test them. Each molecule is associated to a project consisting of a sequence of tasks to be executed in order. A task is not preemptive and cannot be aborted once started. Its duration, cost, and result (failure, which ends the project, or success, which allows the project to continue) are uncertain. The realization of a task is a triplet (duration, cost, result). A project is successful if all its tasks are successful. A successful project generates a revenue which is a given decreasing function of its completion date. The goal is to schedule the tasks in the laboratories, satisfying the resource constraints (no more running tasks than the number of labs at any given time), to maximize the expected profit. The profit is the difference between the total revenues and the total cost. There is no obligation to schedule a task when a lab is available and there are tasks ready to start. Indeed, it is allowed to drop a project (never schedule a task ready to start), as well as to wait some time before starting a task. Waiting is sometimes optimal, like in dynamic fleet management [2].

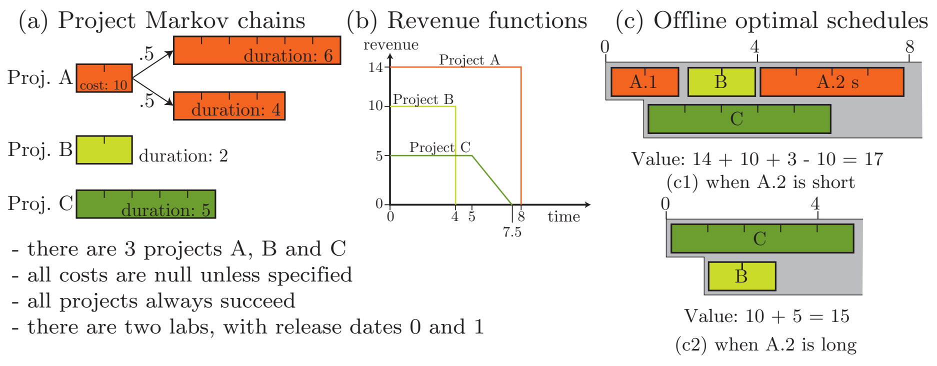
(a) Project Markov chains
(b) Revenue functions
(c) Offline optimal schedules
Proj. A
cost: 10
.5
.5
duration: 6
duration: 4
Proj. B
duration: 2
Proj. C
duration: 5
- there are 3 projects A, B and C
- all costs are null unless specified
- all projects always succeed
- there are two labs, with release dates 0 and 1
revenue
14
10
5
0
Project A
Project B
Project C
0
4
5
8
7.5
time
0
4
8
A.1
B
A.2 s
C
Value: 14 + 10 + 3 - 10 = 17
(c1) when A.2 is short
0
4
C
B
Value: 10 + 5 = 15
(c2) when A.2 is long

Fig. 1. An Instance of the Stochastic Project Scheduling Problem.

Each project is modeled by its own finite heterogeneous first-order Markov chain. That is, for each task, the set of possible realizations is known. The Markov chain, which is given, provides the distribution of the realization of the first task, and the probability transition matrices that, for any realization of the *i*-th task, gives the distribution of the realization of the $(i+1)$-th task.

Figure 1 depicts a small instance to illustrate these concepts. In this instance, there are 3 projects and 4 tasks, and all the projects always succeed. In this instance, the offline optimal schedules for the two possible realizations, which are shown in Figure 1(c), differ at the first decision when the uncertainty is not yet resolved. Hence the optimal online policy is necessarily inferior to a perfect clairvoyant decision maker. The schedule in Figure 1(c2) is the optimal online solution.

3 Exogeneity and Endogeneity: Problem Classification

Traditionally, stochastic optimization problems were separated into two classes according to the exogenous or endogenous nature of their uncertainty. To delineate precisely the scope of *Amsaa*, we need to refine this classification.

Purely exogenous problems are those in which the uncertainty, and the way it is observed, is independent of the decisions. Customers and suppliers behavior is considered exogenous [18], as well as nature (e.g., water inflow in hydroelectric power scheduling), and prices in perfect markets. In this class, there is a natural concept of scenario (e.g., the sequence of customer requests) and, given two scenarios, it is possible to compute when they become distinguishable.

Purely endogenous problems are those for which there is no natural concept of scenarios. Most benchmark problems for Markov Decision Processes are of this nature. Problems in robotics where the uncertainty comes from the actuators are endogenous.

Stoxuno Problems (STochastic Optimization problems with eXogenous Uncertainty and eNdogenous Observations) are applications like the S-RCPSP, for which the underlying uncertainty is exogenous, but observations depend on the decisions. In these problems, the concept of scenario is natural. However, given two scenarios, it is not possible to decide when a decision maker will be able to distinguish them. Many scheduling problems with uncertainty on tasks should belong to this category. The lot sizing problem in [10] also falls into that category.

Amsaa applies to both purely exogenous and Stoxuno problems.

4 Background in Stochastic Dynamic Programming

Stochastic Dynamic Programming is a field of research that aims at solving stochastic optimization problems modelled as Markov Decision Processes (MDPs). MDPs can model purely endogenous problems, purely exogenous, and Stoxuno problems. We only consider finite horizon MDPs with no reward discounting and no transition costs.

Markov Decision Processes. An MDP $(S, s_0, F, X, \perp, \mathscr{X}, f, \mathscr{P})$ consists of:

- a state space S, an initial state $s_0 \in S$, and a set of final states $F \subseteq S$.
- a decision space X containing a decision $\perp$ (denoting no action) and a function $\mathscr{X} : S \to X$ returning the set of feasible decisions in a given state such that $\forall s \in S, 0 < \#\mathscr{X}(s) < \infty$ and that $\forall s \in F, \mathscr{X}(s) = \{\perp\}$.
- a bounded reward function $f : F \to \mathbb{R}$.
- a transition function $\mathscr{P} : S \times X \to \text{prob}(S)$, where $\text{prob}(S)$ is the set of probability distributions over S, satisfying $\forall s \in F, \mathscr{P}(s,\perp)(\{s\}) = 1$.

For convenience, we write $\mathscr{P}(\cdot|s,x)$ instead of $\mathscr{P}(s,x)(\cdot)$. A run of an MDP $(S, s_0, F, X, \perp, \mathscr{X}, f, \mathscr{P})$ starts in the initial state s_0. In a given state s, the decision maker selects a decision $x \in \mathscr{X}(s)$ which initiates a transition to state $s' \in A \subseteq S$ with probability $\mathscr{P}(A|s,x)$. The resulting sequence of states and decisions, i.e. $s_0 \xrightarrow{x_0} s_1 \xrightarrow{x_1} \dots \xrightarrow{x_{t-1}} s_t \xrightarrow{x_t} \dots$, is called a trajectory. This is a Markovian process: conditionally on s_i and x_i, the distribution of s_{i+1} is independent of the past trajectory.

We assume horizon finiteness: there exists an integer T such that all trajectories starting in s_0 are such that s_T is final. As a corollary, the state space graph has to be acyclic. The objective of the decision maker is to maximize $\mathbb{E}[f(\boldsymbol{s}_T)]$.

Policies, Value functions, and Optimality. A (deterministic) Markovian policy $\pi : S \to X$ is a map from states to feasible decisions, i.e., that satisfies $\forall s \in S,\ \pi(s) \in \mathscr{X}(s)$. The value $v_\pi(s)$ of policy π in state s is the expected value obtained by running policy π from state s. A policy π is optimal if the value $v_\pi(s_0)$ is maximal among all policies.

A value function v is a map $S \to \mathbb{R}$. The Q-value function canonically associated to v is the mapping $S \times X \to \mathbb{R}$ defined by $Q(s,x) = \mathbb{E}_{\mathscr{P}}[v(\boldsymbol{s}')|s,x]$, which, in the case of finite state space, becomes $Q(s,x) = \sum_{s' \in S} v(s')\mathscr{P}(s'|s,x)$. Given a value function v and a state s, a decision $x \in \mathscr{X}(s)$ is *greedy* if $Q(s,x) = \max_{x' \in \mathscr{X}(s)} Q(s,x')$. We assume that there is a rule to break ties, so we can talk about "the" greedy decision even though it is not unique. The *greedy policy* π_v associated with a value function v is the policy defined by taking the greedy decision in every state. A value function is optimal if the associated greedy policy is optimal. A necessary and sufficient condition for v to be optimal is that, for all state s reachable under π_v, we have $v(s) = f(s)$ if s is final, and $Res_v(s) = 0$ otherwise, where $Res_v(s) = v(s) - \max Q(s,x)$ is called the *Bellman residual* of v at s. Under our assumptions, there is always an optimal value function $v^\star$.

5 Exogenous Markov Decision Processes

Section 3 discussed the *nature* of the uncertainty. MDPs can model problems of any nature, but *represents* the uncertainty endogenously. For exogenous problems, it is better to use a model that represents the uncertainty exogenously. Stochastic programs are an example of such models, but they cannot model Stoxuno problems. Therefore we introduce exogenous MDPs (X-MDPs) that allow the modeling of purely exogenous and of Stoxuno problems. They are neither more nor less expressive than traditional MDPs [18], but have computational advantages discussed at length in Section 6.

Model and Definitions. An exogenous Markov decision process (X-MDP) $(S, s_0, F, X, \perp, \mathscr{X}, f, \boldsymbol{\xi}, \mu_{\boldsymbol{\xi}}, \tau)$ consists of:

- a state space S, an initial state $s_0 \in S$, and a set of final states $F \subseteq S$.
- a decision space X containing a decision $\perp$ (denoting no action) and a function $\mathscr{X} : S \to X$ returning the set of feasible decisions in a given state such that $\forall s \in S, 0 < \#\mathscr{X}(s) < \infty$ and that $\forall s \in F, \mathscr{X}(s) = \{\perp\}$.
- a bounded reward function $f : F \to \mathbb{R}$.
- a random variable $\boldsymbol{\xi}$, with values in a scenario space Ξ, and distribution $\mu_{\boldsymbol{\xi}}$.
- a (deterministic) transition function $\tau : S \times X \times \Xi \to S$ satisfying $\forall s \in S$, $\forall \xi \in \Xi$, $\tau(s, \perp, \xi) = s$.

Running an X-MDP consists of first sampling a realization ξ of the random variable $\boldsymbol{\xi}$. The decision maker doesn't know ξ, but it makes inferences by observing transition outcomes. Starting in s_0, it makes a decision, observes of the outcome of the transition, and repeats the process. For a state s and a decision x, the next state becomes $\tau(s, x, \xi)$. The alternation of decisions and state updates defines a trajectory $s_0 \xrightarrow[\xi]{x_0} s_1 \xrightarrow[\xi]{x_1} \dots \xrightarrow[\xi]{x_{t-1}} s_t$ satisfying (i) for all i, $x_i \in \mathscr{X}(s_i)$ and (ii) for all i, $s_{i+1} = \tau(s_t, x_t, \xi)$.

Like for MDPs, we assume finite horizon: there is a T such that s_T is final regardless of the decisions made and of $\boldsymbol{\xi}$. The objective also consists of maximizing $\mathbb{E}[f(\boldsymbol{s_T})]$, which is always defined if f is bounded. We will also restrict attention to Markovian policies; in this order, we need to introduce a new concept before specifying the property that ensures their dominance.

In an X-MDP, scenario ξ is *compatible* with a trajectory $s_0 \xrightarrow{x_0} s_1 \xrightarrow{x_1} \dots \xrightarrow{x_{t-1}} s_t$ if $\tau(s_i, x_i, \xi) = s_{i+1}$ for all $i < t$. $\mathscr{C}\left(s_0 \xrightarrow{x_0} \dots \xrightarrow{x_{t-1}} s_t\right)$ is the set of such scenarios. A scenario is *compatible* with a state s if it is compatible with a trajectory from s_0 to s, and $\mathscr{C}(s)$ is the set of such scenarios.

The *Markov property for X-MDPs*, which ensures the dominance of Markovian policies, then reads:

$$\text{for all trajectory } s_0 \xrightarrow{x_0} \dots \xrightarrow{x_{t-1}} s_t, \quad \mathscr{C}\left(s_0 \xrightarrow{x_0} \dots \xrightarrow{x_{t-1}} s_t\right) = \mathscr{C}(s_t). \tag{1}$$

It will be easy to enforce this property in practice: simply include all past observations into the current state. An elementary but important corollary of this assumption is that conditional probabilities on the past trajectory are identical to conditional probabilities on the current state, i.e.,

$$\forall A \subseteq \Xi, \quad \mathbb{P}\left(\boldsymbol{\xi} \in A \,\middle|\, \boldsymbol{\xi} \in \mathscr{C}\left(s_0 \xrightarrow{x_0} \dots \xrightarrow{x_{t-1}} s_t\right)\right) = \mathbb{P}\left(\boldsymbol{\xi} \in A \mid \boldsymbol{\xi} \in \mathscr{C}\left(s_t\right)\right),$$

Hence, sampling scenarios conditionally on the current state is equivalent to sampling scenarios conditionally on the past trajectory.

X-MDPs naturally exhibit an underlying *deterministic* and *offline* problem that has no counterpart in MDPs. The *offline value* of state s under scenario ξ, denoted by $\mathscr{O}(s,\xi)$, is the largest reward of a final state reachable from state s when $\boldsymbol{\xi} = \xi$. It is defined recursively by:

$$\mathscr{O}(s,\xi) = \begin{cases} f(s) & \text{if } s \text{ is final,} \\ \max_{x \in \mathscr{X}(s)} \mathscr{O}(\tau(s,x,\xi),\xi) & \text{otherwise.} \end{cases}$$

Consider the instance shown in Figure 1. If ξ_s and ξ_l denote the scenarios in which A.2 is short and long respectively, then $\mathscr{O}(s_0,\xi_s) = 17$ and $\mathscr{O}(s_0,\xi_l) = 15$.

Policies and Optimality for X-MDPs. Like for MDPs, it is possible to define the value of a policy for an X-MDP. Let A be an X-MDP and $\pi : S \to X$ be a policy for A. Consider a past trajectory $s_0 \xrightarrow{x_0} \dots \xrightarrow{x_{t-1}} s_t$, not necessarily generated by π. Remember that for any sequence of decisions s_T is final. Therefore the expected value obtained by following π after this past trajectory is well defined and is denoted by $v_\pi\left(s_0 \xrightarrow{x_0} \dots \xrightarrow{x_{t-1}} s_t\right)$. By the Markov property, this quantity only depends on s_t, so we denote it $\pi(s_t)$. A policy π is optimal if the value $v_\pi(s_0)$ is maximal among all policies.

Modelling the Stochastic RCPSP as an X-MDP. It is easy to model the S-RCPSP as an X-MDP. A state contains: (1) the current time, (2) the set of currently running tasks with their start times (but without lab assignment), and (3) the set of all past observed task realizations. Thanks to (3) the Markov property for X-MDPs is satisfied.

6 Amsaa: an Algorithm for Decision Making in X-MDPs

Overview of *Amsaa*. This section presents a high-level overview of *Amsaa*, the Anytime Multi-Step Anticipatory Algorithm, which aims at producing high-quality decisions for X-MDPs. Its pseudo-code follows.

Because we want an *anytime* algorithm, that is, one that can be stopped and return something at any time, there is an outer loop for which the condition can be anything. In an operational setting, it will most likely be a time constraint (e.g., "make a decision within a minute"), and in a prospective setting, it could be a stopping criteria based on some accuracy measure (for example, the contamination method [9]).

Amsaa's first step is to approximate the X-MDP to make it more tractable. It then converts it to an MDP in order to apply standard search algorithm for MDPs. This search is guided by an upper bound that exploits the existence of offline problems due

Function *Amsaa* (*X-MDP A*)

```
1 while some condition do
2     Approximate the X-MDP A by replacing $\boldsymbol{\xi}$ with a random variable $\boldsymbol{\xi}'$ whose support is smaller,
      or refine the current approximation.
3     Convert the resulting X-MDP to a standard MDP.
4     Solve the resulting MDP with a search algorithm for MDPs, using the offline upper bound
      $h_{\mathbb{E},\max}(s) = \mathbb{E}\left[\mathscr{O}(s,\boldsymbol{\xi}') \mid \boldsymbol{\xi}' \in \mathscr{C}(s)\right]$.
5 return the greedy decision at the root node of the MDP.
```

to the exogenous nature of the uncertainty. For efficiency, lines 3–4 are incremental, so that when the approximation is refined (line 1), the amount of work to be done is small if the refinement does not change the approximated problem too much.

We will now present the details of the approximation, of the convertion to an MDP, of the MDP solving, and, finally, of the incrementality.

Approximating the X-MDP by Sampling. The first step of *Amsaa* is to approximate the original X-MDP by replacing the distribution of the scenarios by one with a finite and reasonably small support. The simplest way of doing so is by sampling. For stochastic programs, this idea is called the Sample Average Approximation (SAA) method [16], and it can be extended to X-MDPs. Suppose we want a distribution whose support has cardinality at most n: just sample $\boldsymbol{\xi}$ n times, independently or not, to obtain $\boldsymbol{\xi}^1,\dots,\boldsymbol{\xi}^n$ and define $\hat{\boldsymbol{\mu}}_n$ as the empirical distribution induced by this sample, that is, the distribution that assigns probability $1/n$ to each of the sampled scenarios. Some results of the SAA theory translate to X-MDPs. In particular, if Ξ is finite and the sampling iid, then the SAA technique produces almost surely optimal decisions with enough scenarios.

Benefits of Exterior Sampling for X-MDPs. Sampling can be used either to compute an optimal policy for an approximated problem (The SAA method, used in *Amsaa*); or to compute an approximately optimal policy for the original problem, like in [12], who proposed an algorithm to solve approximately an MDP by sampling a number of outcomes at each visited state (interior sampling). Their algorithm was presented for discounted rewards but generalizes to finite horizon MDPs. We argue that the SAA method is superior because sampling internally does not exploit a fundamental advantage of problems with exogenous uncertainty: *positive correlations*.

Indeed, in a state s, the optimal decision maximizes $Q^\star(s,x)$, where $Q^\star$ is the Q-value function associated to the optimal value function $v^\star$. However, estimating this value precisely is not important. What really matters is to estimate the sign of the difference $Q^\star(s,x_1) - Q^\star(s,x_2)$ for each pair of decisions $x_1,x_2 \in \mathscr{X}(s)$. Now, consider two functions g and h mapping scenarios to reals, for example the optimal policy value obtained from a state s after making a first decision. That is, $g(\xi) = v^\star(\tau(s_0,x_1,\xi))$ and $h(\xi) = v^\star(\tau(s_0,x_2,\xi))$ for two decisions $x_1,x_2 \in \mathscr{X}(s_0)$. If $\boldsymbol{\xi}^1$ and $\boldsymbol{\xi}^2$ are iid scenarios:

$$\begin{aligned}
&\operatorname{var}\left(g(\boldsymbol{\xi}^1) - h(\boldsymbol{\xi}^2)\right) = \operatorname{var}\left(g(\boldsymbol{\xi}^1)\right) + \operatorname{var}\left(h(\boldsymbol{\xi}^2)\right), \\
&\operatorname{var}\left(g(\boldsymbol{\xi}^1) - h(\boldsymbol{\xi}^1)\right) = \operatorname{var}\left(g(\boldsymbol{\xi}^1)\right) + \operatorname{var}\left(h(\boldsymbol{\xi}^1)\right) - 2\operatorname{cov}\left(g(\boldsymbol{\xi}^1), h(\boldsymbol{\xi}^1)\right),
\end{aligned}$$

and therefore $\text{var}\left(g(\boldsymbol{\xi}^1)-h(\boldsymbol{\xi}^1)\right) = \left(1-\text{acorr}\left(g(\boldsymbol{\xi}^1),h(\boldsymbol{\xi}^1)\right)\right)\cdot \text{var}\left(g(\boldsymbol{\xi}^1)-h(\boldsymbol{\xi}^1)\right)$, where $\text{acorr}(X,Y) = \frac{\text{cov}(X,Y)}{1/2(\text{var}(X)+\text{var}(Y))}$ is a quantity we call *arithmetic correlation*. Note that $\text{acorr}(X,Y) \approx \text{corr}(X,Y)$ when $\text{var}(X)$ and $\text{var}(Y)$ are close. Now consider an infinite iid sample $\boldsymbol{\xi}^1, \boldsymbol{\xi}^{1\prime}, \boldsymbol{\xi}^2, \boldsymbol{\xi}^{2\prime}, \ldots$, and a large integer n. By the central limit theorem, the distributions of $\frac{1}{n}\sum_{i=1}^{n} g(\boldsymbol{\xi}^i) - h(\boldsymbol{\xi}^i)$ and of $\frac{1}{n\gamma}\sum_{i=1}^{n\gamma} g(\boldsymbol{\xi}^i) - h(\boldsymbol{\xi}^{i\prime})$ are almost the same when $1/\gamma = 1-\text{acorr}\left(g(\boldsymbol{\xi}^1),h(\boldsymbol{\xi}^1)\right)$. Therefore, for some specified accuracy, the number of required scenarios to estimate the expected difference between $g(\boldsymbol{\xi})$ and $h(\boldsymbol{\xi})$ is reduced by this factor γ when the same scenarios (exterior sampling) are used instead of independent scenarios (interior sampling).

This argument is not new, and can be found for example in [16]. However, no empirical evidence of high correlations were given, which we now report. Consider an SAA problem approximating the standard instance of the S-RCPSP application with 200 scenarios generated by iid sampling, and consider the optimal policy values in the initial state for the 6 possible initial decisions (the first is to schedule nothing, the others are to schedule the first task of each project). Associating a column with each decision, the values for the first five scenarios are:

$$OptPolicyValue = 1e4 \times \begin{pmatrix} 0 & 2.110 & 2.038 & 1.910 & 1.893 & 2.170 \\ 0 & -0.265 & -0.275 & -0.275 & -0.275 & -0.225 \\ 0 & -0.205 & -0.230 & -0.230 & -0.170 & -0.170 \\ 0 & 1.375 & 1.405 & 1.279 & 1.345 & 1.365 \\ 0 & 1.045 & 1.070 & 1.015 & 1.105 & 1.160 \end{pmatrix}$$

The correlation is evident. Excluding the first decision (which is uncorrelated to the others), the arithmetic correlations range from 94% to 99%, computed on the 200 scenarios. Moreover, the minimal correlation is 98.7% among the second, third, and fourth decisions, which are the three good candidates for being selected as the initial decision.

It remains to see whether these correlations are a characteristic of the problem or even of the instance. In most OSCO problems, some scenarios are more favorable than others regardless of the decisions, causing these correlations: in the S-RCPSP, scenarios with many successful projects bring more money than scenarios with many failures, and this is very visible on the matrix above. As a result we conjecture that, for most OSCO problems, exterior sampling converges with far fewer scenarios than interior sampling.

Converting the X-MDP into an MDP. This is the second step of *Amsaa*.

Definition 1 *Given an X-MDP A with state-space S and final states set F, the* trimmed *X-MDP B induced by A is the X-MDP that is in all equal to A, except:*

1. *its state space is $S' = \{s \in S | \mathscr{C}(s) \neq \varnothing\}$;*
2. *its set of final states is $F' = F \cup \{s \in S' | \#\mathscr{C}(s) = 1\}$, and the function $\mathscr{X}$ is modified accordingly;*
3. *its reward function f' is defined, for states $s \in F' \setminus F$, by $f(s) = \mathscr{O}(s,\xi)$, where ξ is the unique scenario compatible with s.*

A trimmed X-MDP is equivalent to the original one, in the sense that an optimal policy in A induces an optimal policy in B and vice versa.

Definition 2 *Let $B = (S, s_0, F, X, \bot, \mathscr{X}, f, \boldsymbol{\xi}, \mu_{\xi}, \tau)$ be the trimmed X-MDP induced by an X-MDP A. Define $\mathscr{P}$ from $S \times X$ to the set of probability distributions on S by:*

$$\forall s \in S, x \in X, U \subseteq S, \quad \mathscr{P}(U|s,x) = \mathbb{P}\left(\tau(s,x,\boldsymbol{\xi}) \in U \mid \boldsymbol{\xi} \in \mathscr{C}(s)\right).$$

Then $C = (S, s_0, F, X, \bot, \mathscr{X}, f, \mathscr{P})$ is the MDP induced *by X-MDP A.*

The induced MDP is equivalent to the original problem in the following sense.

Theorem 1 *Let A be an X-MDP, C the induced MDP, and π be a policy that is optimal in A for states in $F' \setminus F$. Then, for all states $s \in S'$, $v_\pi^A(s) = v_\pi^C(s)$.*

This theorem is a consequence of the Markov property for X-MDPs, which implies that, following π in B or C, for all t the distribution of $\boldsymbol{s}_t$ is the same in B and in C.

Solving MDPs. Once the approximated X-MDP is converted into an MDP, it is possible to apply existing algorithms for solving the MDP exactly. We use a*heuristic search algorithm*, which, despite its name, is an exact algorithms.

Heuristic Search Algorithms for MDPs. Heuristic search algorithms for MDPs perform a partial exploration of the state space, using a — possibly monotone — upper bound to guide the search. A value function $h : S \to \mathbb{R}$ is an *upper bound* if $\forall s \in S,\ h(s) \geq v^\star(s)$, and is a *monotone upper bound* if, in addition, $Res_h(s) \geq 0$ for all state s. A monotone upper bound is an optimistic evaluation of a state that cannot become more optimistic if a Bellman update is performed.

Function `findRevise(MDP` A`)`

```
precondition: $h$ is a upper bound for $A$, $h(s) = f(s)$ if $s$ is final
1 foreach $s \in S$ do $v(s) \leftarrow h(s)$
2 repeat
3     Pick a state $s$ reachable from $s_0$ and $\pi_v$ with $|Res_v(s)| > 0$
4     $v(s) \leftarrow \max_{x \in \mathscr{X}(s)} Q(s,x)$
5 until no such state is found
6 return $v$
```

Function `findAndRevise`, introduced by [4], captures the general schema of heuristic search algorithm for MDPs and returns an optimal value function upon termination. At each step, the algorithm selects a state reachable with the current policy π_v whose Bellman residual is non-zero and performs a Bellman update. When h is monotone, only strictly positive (instead of non-zero) Bellman residuals must be considered. Different instantiations of this generic schema differ in the choice of the state to reconsider. They include, among others, HDP [4], Learning Depth-First Search (LDFS) [5], Real-Time Dynamic Programming (RTDP) [1], Bounded RTDP [13], and LAO* [11]. These algorithms only manipulate partial value functions defined only on the states visited so far, performing the initialization $v(s) \leftarrow h(s)$ on demand. We chose to use the acyclic version of Learning Depth-First Search (a-LDFS). It applies to acyclic problems (ours are), and requires a monotone upper bound, which we have.

The Upper Bound $h_{\mathbb{E},\max}$. The performance of heuristic search algorithms strongly depends on the heuristic function h. For MDPs induced by X-MDPs, a good heuristic function can be derived from the deterministic offline problems. More precisely, for a state s, the heuristic consists of solving the deterministic offline problems for the scenarios compatible with s in the original X-MDP and taking the resulting expected offline value, i.e., $h_{\mathbb{E},\max}(s) = \mathbb{E}_\mu\left[\mathscr{O}(s,\boldsymbol{\xi}) \mid \boldsymbol{\xi} \in \mathscr{C}(s)\right]$, where μ is $\boldsymbol{\xi}$'s distribution. Function $h_{\mathbb{E},\max}$ is a monotone upper bound. It is attractive for guiding the search because it leverages the combinatorial structure of the application (black-box offline solver) and can be computed efficiently because the sets $\mathscr{C}(s)$ are finite and small. $h_{\mathbb{E},\max}$ provides a significant computational advantage to X-MDPs over MDPs.

Incrementality and Anytime Decision Making. Incrementality is the ability to resolve the MDP quickly after a small change in the approximated problem. Incrementality enables fine-grained refinement, providing for efficient anytime decision making and openning the door to sequential sampling [7]. It is based on the following theorem.

Theorem 2 *Let $\mathscr{A}$, $\mathscr{B}$ and $\mathscr{C}$ be three X-MDPs that differ only by their respective distributions μ, ν, and ρ and let $\rho = \lambda\mu + (1-\lambda)\nu$ for some $0 < \lambda < 1$. Let h_μ and h_ν be monotone upper bounds for $\mathscr{A}$ and $\mathscr{B}$ respectively. Define $h : S \rightarrow \mathbb{R}$ by $h(s) = -\infty$ if $\rho(\mathscr{C}(s)) = 0$, and otherwise by*

$$h(s) = \frac{1}{\rho(\mathscr{C}(s))}\Big(\lambda\mu(\mathscr{C}(s))\, h_\mu(s) \;\; + \;\; (1-\lambda)\nu(\mathscr{C}(s))h_\nu(s) \Big).$$

Then h is a monotone upper bound for the induced MDP of $\mathscr{C}$.

This theorem is used in the following setting. μ is the old sample distribution, and we have solved $\mathscr{A}$ optimally with `findAndRevise()`. The optimal value function it returned is the monotone upper bound $h_\mu(s)$. ν is the distribution of the newly added scenarios, and h_ν is the $h^\nu_{\mathbb{E},\max}$, the offline upper bound for $\mathscr{B}$. ρ is the new sample distribution, and includes the old sample and the newly added scenarios. λ is the weight of the old sample in the new sample. Our experiments showed adding the scenarios one-by-one instead of all at once produced only a 20% slowdown on 500-scenario problems.

7 Experimental Results on Anytime Decision Making

Experimental Setting. The benchmarks are based on the collection of 12 instances for the S-RCPSP from [8]. The reference instance, Reg, is similar to the one in [6]. It has 2 laboratories, 5 projects, and a total of 17 tasks. The number of realizations for each tasks range from 3 to 7, giving a total of 10^9 possible scenarios. The 11 other instances are variant of Reg, scaling the costs, scaling the time axis of the revenue functions, or chaning the structure of the Markov chains for each molecule.

For each instance, we generated 1,000 realizations of the uncertainty. A *run* of an algorithm on one of these realizations consists of simulating one trajectory in the X-MDP. At each encountered state, the online algorithm takes a decision with hard time constraints. If the online algorithm has not enough time to decide, a default decision, closing the labs, is applied. The algorithms were tested on all the realizations and various time limits. With 4 tested algorithms and time limits of 31 ms, 125 ms, 500 ms, 2s, 8s, 32s, this gives a total of 288,000 runs taking more than 8,000 hours of cpu time.

The Four Compared Algorithms. *Amsaa* was used with iid sampling and sample sizes growing by increments of 10%. Its performance relies on a fast offline solver. We used the branch and bound algorithm from [8] whose upper bound relaxes the resource constraints for the remaining tasks. This branch and bound is very fast thanks to a good preprocessing step: it takes on average less than 1ms for the reference instance. *1s-AA* is the one-step anticipatory algorithm with iid sampling. It uses the same offline solver than *Amsaa*. BRTDP is the Bounded Real Time Dynamic Programming algorithm [13]. The lower bound $h^-(s)$ correspond to not scheduling anything after state s. The upper bound is $h^+(s)$ is a very slight relaxation of $h_{\max,\max}$, using the offline solver on an hypothetical best scenario. Like in RTDP, and as opposed to B-RTDP, decisions are taken greedily with respect to the upper bound value function v^+: Indeed experimental results showed that making decisions with respect to v^- provides very poor decisions. HC-DP is the Heuristically Confined Dynamic Programming algorithm from [6] enhanced into an anytime algorithm. The offline learning phase is removed and performed within the given time limits. A full Bellman update is performed at increasing larger intervals, so that the decision can be updated. Less than half the computation time is spent doing updates, the rest being spent exploring the state-space. Its results outperform those of the original HC-DP algorithm in [6].

The Performance of *Amsaa*. Figure 2 summarizes the results for anytime decision making. It contains a table for each of the 12 instances. The first line of this table contains the empirical mean value obtained by running *Amsaa*. The three lines below report the relative gap between the expected value of the considered algorithm and *Amsaa* with the same time constraint. In addition, the background color carries information about the statistical significance of the results, at the 5% level, as indicated by the legend of the figure. It indicates whether the considered algorithm is better than *Amsaa*-32s (no occurrence here); not worse than *Amsaa*-32s (e.g., *Amsaa*-500ms on Cost2); significantly worse than *Amsaa*-32s, but better than *Amsaa*-31ms (e.g., *1s-AA*-31 ms on P3); worse than *Amsaa*-32s, but not than *Amsaa*-31ms (e.g., B-RTDP-2s on Agr); or worse than *Amsaa*-31ms (e.g., HC-DP-32s on Reg).

Overall *Amsaa* exhibits excellent performance. The solution quality of *Amsaa*-32s is often higher by at least 10% than *1s-AA*-32s, HC-DP-32s, and B-RTDP-32s and *Amsaa* is robust across all instances. With 32s, *Amsaa* is significantly better than all other algorithms on 11 instances and as good as any other algorithm on Cost5. Moreover, the remaining three algorithms lacks robustness with respect to the instances: They all rank second and last at least once. Note that, on Cost5, the optimal policy is not to schedule anything. HC-DP is able to realize that quickly, with only 125 ms, because it uses very fast heuristics. *Amsaa*-32s and HC-DP with at least 125ms are optimal on this problem.

Amsaa is also robust with respect to the available computation time. On most instances, the rankings of the four algorithms do not vary much with respect to the computation times. One might think that with very strong time constraints, *1s-AA* is preferable to *Amsaa*, because *1s-AA* can use more scenarios in the same amount of time. Yet,

Instance Reg (Regular)

	31 ms	125 ms	500 ms	2 s	8 s	32 s
Amsaa	7.86 e+3	8.11 e+3	8.31 e+3	8.41 e+3	8.42 e+3	8.45 e+3
1s-AA	-5.52%	-5.15%	-7.12%	-8.07%	-7.86%	-8.28%
HC-DP	-5.09%	-8.59%	-10.26%	-11.04%	-10.97%	-11.35%
B-RTDP	-100.00%	-100.00%	-37.54%	-33.23%	-25.15%	-22.99%

Instance Agr (agregated outcomes)

	31 ms	125 ms	500 ms	2 s	8 s	32 s
Amsaa	1.22 e+4	1.24 e+4	1.26 e+4	1.27 e+4	1.27 e+4	1.28 e+4
1s-AA	1.14%	-0.60%	-1.45%	-2.17%	-2.49%	-2.49%
HC-DP	-4.12%	-6.29%	-7.54%	-8.23%	-8.51%	-8.56%
B-RTDP	-100.00%	-100.00%	-4.28%	-4.47%	-2.44%	-1.74%

Instance Cost2 (Costs x 2)

	31 ms	125 ms	500 ms	2 s	8 s	32 s
Amsaa	4.34 e+3	4.50 e+3	4.82 e+3	4.90 e+3	4.85 e+3	4.89 e+3
1s-AA	-17.66%	-20.13%	-24.74%	-26.26%	-25.27%	-26.05%
HC-DP	0.41%	-1.71%	-10.02%	-9.80%	-9.20%	-10.06%
B-RTDP	-100.00%	-100.00%	-66.81%	-54.79%	-45.34%	-43.05%

Instance Cost5 (Costs x 5)

	31 ms	125 ms	500 ms	2 s	8 s	32 s
Amsaa	-3.23 e+3	-2.64 e+3	-1.71 e+3	-3.38 e+2	6.50 e+0	0.00 e+0
1s-AA	-3.15 e+3	-3.14 e+3	-3.16 e+3	-3.13 e+3	-3.14 e+3	-3.15 e+3
HC-DP	-2.55 e+1	0.00 e+0	0.00 e+0	0.00 e+0	0.00 e+0	0.00 e+0
B-RTDP	0.00 e+0	-9.42 e+3	-9.10 e+3	-8.49 e+3	-7.86 e+3	-6.94 e+3

Instance D.6 (.66 less time before revenues start decreasing)

	31 ms	125 ms	500 ms	2 s	8 s	32 s
Amsaa	6.14 e+3	6.71 e+3	6.89 e+3	6.89 e+3	6.89 e+3	6.89 e+3
1s-AA	-11.53%	-19.62%	-22.71%	-22.34%	-22.31%	-21.43%
HC-DP	6.79%	-2.21%	-4.87%	-5.41%	-5.07%	-5.08%
B-RTDP	-77.04%	-86.00%	-78.56%	-72.46%	-62.12%	-52.53%

Instance D1.5 (1.5 more time before revenues start decreasing)

	31 ms	125 ms	500 ms	2 s	8 s	32 s
Amsaa	1.04 e+4	1.06 e+4	1.07 e+4	1.07 e+4	1.07 e+4	1.08 e+4
1s-AA	-12.31%	-14.29%	-14.61%	-15.11%	-15.16%	-14.87%
HC-DP	-2.54%	-3.94%	-4.22%	-4.87%	-4.89%	-4.93%
B-RTDP	-24.29%	-20.94%	-17.05%	-13.17%	-11.04%	-8.98%

Instance P1 (no failure at task 4)

	31 ms	125 ms	500 ms	2 s	8 s	32 s
Amsaa	1.42 e+4	1.46 e+4	1.45 e+4	1.46 e+4	1.47 e+4	1.47 e+4
1s-AA	-3.61%	-5.96%	-6.26%	-6.83%	-7.27%	-7.72%
HC-DP	-11.84%	-13.93%	-13.72%	-13.99%	-14.41%	-14.80%
B-RTDP	-100.00%	-100.00%	-25.41%	-20.46%	-15.97%	-13.57%

Instance P1 (no failure at task 3 & 4)

	31 ms	125 ms	500 ms	2 s	8 s	32 s
Amsaa	1.79 e+4	1.85 e+4	1.87 e+4	1.88 e+4	1.90 e+4	1.90 e+4
1s-AA	0.54%	-1.01%	-0.49%	-0.70%	-1.64%	-1.65%
HC-DP	-9.87%	-13.07%	-13.79%	-14.43%	-15.26%	-15.22%
B-RTDP	-100.00%	-100.00%	-19.68%	-13.87%	-12.83%	-10.60%

Instance P3 (no failure at task 2, 3, & 4)

	31 ms	125 ms	500 ms	2 s	8 s	32 s
Amsaa	2.31 e+4	2.60 e+4	2.69 e+4	2.69 e+4	2.69 e+4	2.70 e+4
1s-AA	4.47%	1.10%	-0.28%	-0.44%	-0.49%	-0.50%
HC-DP	-0.55%	-11.63%	-14.11%	-14.22%	-14.18%	-14.29%
B-RTDP	-100.00%	-100.00%	-13.39%	-8.64%	-12.22%	-9.42%

Instance P4 (no failures)

	31 ms	125 ms	500 ms	2 s	8 s	32 s
Amsaa	1.91 e+4	2.71 e+4	2.89 e+4	2.90 e+4	2.91 e+4	2.91 e+4
1s-AA	2.33%	1.57%	0.06%	-0.27%	-0.48%	-0.46%
HC-DP	11.35%	-21.75%	-26.64%	-26.97%	-27.24%	-27.08%
B-RTDP	-100.00%	-100.00%	-13.87%	-15.14%	-11.84%	-10.12%

Instance R.6 (revenues x .66)

	31 ms	125 ms	500 ms	2 s	8 s	32 s
Amsaa	3.67 e+3	3.88 e+3	4.03 e+3	3.97 e+3	4.12 e+3	4.13 e+3
1s-AA	-3.55%	-9.89%	-13.44%	-11.91%	-15.63%	-15.58%
HC-DP	-0.62%	-4.29%	-7.82%	-6.20%	-9.24%	-9.72%
B-RTDP	-100.00%	-100.00%	-50.17%	-40.49%	-38.22%	-30.97%

Instance R1.5 (revenues x 1.5)

	31 ms	125 ms	500 ms	2 s	8 s	32 s
Amsaa	1.45 e+4	1.52 e+4	1.54 e+4	1.53 e+4	1.56 e+4	1.55 e+4
1s-AA	-5.98%	-11.81%	-12.93%	-12.87%	-13.94%	-14.01%
HC-DP	-6.20%	-9.99%	-10.89%	-10.15%	-11.85%	-11.71%
B-RTDP	-100.00%	-100.00%	-31.16%	-24.84%	-22.42%	-18.75%

■	Not worse than Amsaa 32 s
▩	Worse than Amsaa 32 s, but better than Amsaa 31 ms
▨	Worse than Amsaa 32 s, but not than Amsaa 31 ms
□	Worse than Amsaa 31 ms

Fig. 2. Experimental Results for Anytime Decision Making on the S-RCPSP.

there are only two instances on which *1s-AA*-31ms beats *Amsaa*-31ms (Agr and P3) and 3 on which they exhibit similar results. Note that B-RTDP-31ms has a zero score on many instances due to the fact that even a single B-RTDP trial has to go deep in the state space and compute the bounds h^+ and h^- for many states. Under such strict time constraints, B-RTDP cannot even perform one trial before the deadline.

Empirical Complexity of *Amsaa*. Figure 3(a) shows how the sample size grows with the available runtime on instance Reg, measured on the making of the initial decision. Because *Amsaa* is exponential in the worst case, one might fear that the number of scenarios grows logarithmically with the runtime. Yet, a power model for the expected sample size $\mathbb{E}[n]$ as a function of the computation time t fits almost perfectly the empirical data. The fitted model is $\mathbb{E}[n] = 105 \times t^{0.61}$, which indicates that *Amsaa*'s execution time grows subquadratically in the number of scenarios ($1/0.61 = 1.64 < 2$)

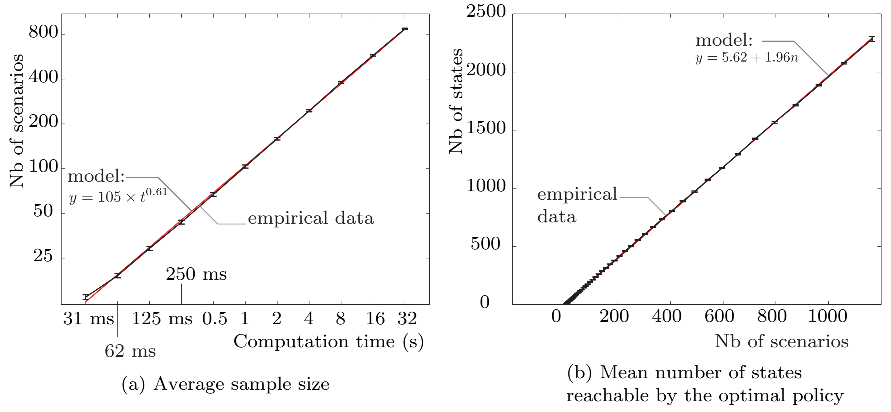

Fig. 3. Empirical complexity of *Amsaa*

However, one may argue that this behavior may be a consequence of iid sampling and is not a convincing evidence that *Amsaa* performs well. Indeed, in the case of a continuous distribution of the uncertainty, all the scenarios would almost surely be dispatched to different states after the first observation and *Amsaa* with iid sampling would have a linear complexity. The stochastic RCPSP has finite distributions but a similar behavior, i.e., a fast divergence of the scenarios, could explain its good performance.

To test whether this is the case, we measured the number of states in the trimmed approximated X-MDP that are reachable by an optimal policy, as depicted on figure 3(b). With a continuous distribution, the number of reachable states would almost surely be $n+1$ for n scenarios: the root node and n leaves. If observations were Bernoulli random variables with parameter $1/2$, the solution state space would be a roughly balanced binary tree with $2n-1$ nodes. These two extreme cases suggest to fit a linear model of the form $(\textit{nb reachable states}) = a + bn$. Such a model fits perfectly the experimental results with a slope of 1.96, making it much closer to a Bernoulli case than a continuous distribution. This provides evidence that scenarios do not diverge too quickly with iid sampling and that the SAA problems become harder with the number of scenarios.

Comparison with Gap Reduction Techniques. The following table reports the relative gap (in %) between [8]'s best algorithm, called $\mathscr{A}_{TEPR}$, based on gap reduction techniques, and *Amsaa*-32s. The background color provides significance information: on Cost2 and R.6, $\mathscr{A}_{TEPR}$ beats *Amsaa*-32s at the 5% significance level. On Reg, Cost5, and R1.5, none is better than the other. On D.6 gap reduction is worse than *Amsaa*-31ms, and on the others gap reduction is worse than *Amsaa*-32s but better than *Amsaa*-31ms.

Reg	Agr	Cost2	Cost5	D.6	D1.5	P1	P2	P3	P4	R.6	R1.5
-0.24	-1.11	+9.96	0.00	-16.8	-0.43	-1.98	-2.80	-0.57	-0.62	+5.40	+0.39

Gap reduction techniques are an attractive alternative to *Amsaa*. Nethertheless, *Amsaa* outperforms them on most instances here, sometimes with a large gap (17% on D.6), and converges to the optimal decisions (gap reduction techniques do not).

8 Comparison with Mathematical Programming

Stochastic programming traditionally focuses on purely exogenous problems. However, [10] proposed an integer programming (IP) formulation for SAA problems of a Stoxuno lot-sizing problem. We investigated a similar approach for the solving of SAA problems for S-RCPSP using an IP closely following model $(P2)$ in [10]. In this model, the number of binary variables is quadratic in the number of scenarios and linear in the time horizon. A 20-scenario problem generated by iid sampling had, after CPLEX's presolve, $47 \cdot 10^3$ binary variable and $20 \cdot 10^6$ non-zeros. On this problem, CPLEX 10.1 runs out of memory before finding the first integer solution, while *Amsaa* solves it in 0.2s, and solves 1,000-scenario problems within minutes. With 1,000 scenarios, the IP model would have about 10^8 binary variables ($(10^3)^2 \times 100$: there are about 100 time steps), which is outside the scope of today's IP solvers.

[10] proposed to solve this IP using a branch-and-bound algorithm based on a Lagrangian relaxations of the non-anticipativity constraints. Yet, with 1,000 scenarios, their algorithm would relax 10^9 constraints (10 non-anticipatory constraints for each binary variable), so there would be a billion Lagrange multipliers to optimize at each node of the tree, which is not reasonable either.

Why is *Amsaa* so much more scalable on this problem? The main difference is the way non-anticipativity constraints are handled in the two approaches. In Grossman's approach, these are *relaxed by Lagragian duality* whereas, in *Amsaa*, they are enforced *lazily*. The lazy approach has two major advantages. First, the presence of Lagrangian multipliers alter the structure of the problem, precluding the use of a highly optimized ad-hoc solver like in *Amsaa*. Second, it makes *Amsaa* able to exploit the discrete nature of the decisions, using states and transitions instead of discretizing time.

9 Conclusion and Research Opportunities

We proposed *Amsaa*, the Anytime Multi-Step Anticipatory Algorithm, designed to address the limitations of the one-step anticipatory algorithm on very stochastic applications. *Amsaa* applies to online combinatorial stochastic optimization problems with exogenous uncertainty and exogenous or endogenous observations. Experimental results on stochastic resource-constraint project scheduling indicate that *Amsaa* significantly outperforms existing algorithms under a variety of time constraints and of instances.

The essence of *Amsaa* lies in the integration of three ideas from different fields: the SAA method from stochastic optimization to exploit positive correlations between decisions, search algorithms from AI to solve MDPs exactly without time discretization, and the use of black-box offline solvers from online stochastic combinatorial optimization to compute good upper bounds quickly.

There are many research avenues to improve *Amsaa*. They include the use of lower bounds like in B-RTDP (recall that we are maximizing) and of weaker but faster upper bounds. Other research questions concern the generation of the approximated problems. The stochastic programming literature include a few techniques to produce better sample than by iid sampling [9]. It is not yet clear which of these techniques could be applied to Stoxuno problems.

Acknowlegments Many thanks to Grégoire Dooms for his help. This research is partially supported by NSF awards DMI-0600384 and ONR award N000140610607.

References

1. Andrew G. Barto, S. J. Bradtke, and Satinder P. Singh. Learning to act using real-time dynamic programming. *Artificial Intelligence*, 72(1):81–138, 1995. rtdp.
2. R. Bent and P. Van Hentenryck. Waiting and Relocation Strategies in Online Stochastic Vehicle Routing. *Proceedings of the 20th Int. Joint Conf. on A.I., (IJCAI'07)*, January 2007.
3. R. Bent and P. Van Hentenryck. Scenario-Based Planning for Partially Dynamic Vehicle Routing Problems with Stochastic Customers. *Operations Research*, 52(6), 2004.
4. Blai Bonet and Hector Geffner. Faster heuristic search algorithms for planning with uncertainty and full feedback. In *IJCAI*, 1233–1238, 2003.
5. Blai Bonet and Hector Geffner. Learning depth-first search: A unified approach to heuristic search in deterministic and non-deterministic settings, and its application to mdps. In *ICAPS*, 2006.
6. J. Choi, M. J. Realff, and J. H. Lee. Dynamic prog. in a heuristically confined state space: A stochastic resource-constrained project scheduling appplication. *Computers and Chemical Engineering*, 2004.
7. M.A.H. Dempster. Sequential Importance Sampling Algorithms for Dynamic Stochastic Programming. *Journal of Mathematical Sciences*, 133:1422–1444, 2006.
8. G. Dooms, and P. Van Hentenryck. Gap Reduction Techniques for Online Stochastic Project Scheduling. CPAIOR'08.
9. J. Dupacova, G. Consigli, and S.W. Wallace. Scenarios for multistage stochastic programs. Annals of Operations Research (2000)
10. Vikas Goel and Ignacio E. Grossmann. A class of stochastic programs with decision dependent uncertainty. *Math. Program*, 108(2-3):355–394, 2006.
11. Eric A. Hansen and Shlomo Zilberstein. LAO: A heuristic-search algorithm that finds solutions with loops. *Artificial Intelligence*, 129(1–2):35–62, 2001.
12. M. Kearns, Y. Mansour, and A. Ng. A Sparse Sampling Algorithm for Near-Optimal Planning in Large Markov Decision Processes. In *IJCAI'99*, 1324–1231, 1999.
13. H. Brendan McMahan, Maxim Likhachev, and Geoffrey J. Gordon. Bounded real-time dynamic programming: RTDP with monotone upper bounds and performance guarantees. In *ICML*, 569–576, 2005.
14. L. Mercier and P. Van Hentenryck. Performance Analysis of Online Anticipatory Algorithms for Large Multistage Stochastic Programs. *Proceedings of the 20th Int. Joint Conf. on AI, (IJCAI)*, 2007.
15. D. Parkes and A Duong. An Ironing-Based Approach to Adaptive Online Mechanism Design in Single-Valued Domains. In *AAAI'07*, pages 94–101, Vancouver, Canada, 2007.
16. A. Ruszczynski and A. Shapiro, editors. *Stochastic Programming*, volume 10 of *Hanbooks in Operations Research and Management Series*. Elsevier, 2003.
17. M. Thomas and H. Szczerbicka. Evaluating Online Scheduling Techniques in Uncertain Environments. In *Proceedings of the 3rd Multidisciplinary International Scheduling Conference (MISTA'07)*, 2007.
18. P. Van Hentenryck and R. Bent. *Online Stochastic Combinatorial Optimization*. The MIT Press, Cambridge, Mass., 2006.