How to Tame Your Anticipatory Algorithm

Allegra De Filippo, Michele Lombardi and **Michela Milano**
Department of Computer Science and Engineering, University of Bologna
allegra.defilippo@unibo.it, michele.lombardi2@unibo.it, michela.milano@unibo.it

Abstract

Sampling-based anticipatory algorithms can be very effective at solving online optimization problems under uncertainty, but their computational cost may be prohibitive in some cases. Given an arbitrary anticipatory algorithm, we present three methods that allow to retain its solution quality at a fraction of the online computational cost, via a substantial degree of *offline* preparation. Our approaches are obtained by combining: 1) a simple technique to identify likely future outcomes based on past observations; 2) the (expensive) offline computation of a "contingency table"; and 3) an efficient solution-fixing heuristic. We ground our techniques on two case studies: an energy management system with uncertain renewable generation and load demand, and a traveling salesman problem with uncertain travel times. In both cases, our techniques achieve high solution quality, while substantially reducing the online computation time.

1 Introduction

Optimization problems under uncertainty arise in many important application domains, such as production scheduling or energy system management. These problems often benefit from making all or part of their decisions online, reacting and adapting to external events. In this context, stochastic online anticipatory algorithms have proved particularly effective (see e.g. [Hentenryck and Bent, 2009]). However, many of such algorithms have a considerable computational cost, which may be problematic if (as it is often the case) online decisions must be taken within a short time frame.

In most practical settings, however, a substantial amount of time and information is available before the online problem is solved, in an *offline phase*. For example, one may have access to energy production forecasts, historical travel times in routing problems, results from test runs in cyber-physical systems. We refer to this sort of data as *offline information*. Usually, it is employed to characterize the uncertain elements and for sampling likely outcomes (i.e. scenarios). We will show how to exploit this information at a much deeper level.

We propose three hybrid offline/online methods that build over a given, sampling-based, anticipatory algorithm, and allow to match its solution quality at a fraction of the online computational cost. One of them can even rely on a deterministic algorithm, thus providing state-of-the art performance in problems for which no anticipatory approach is available. All our methods work by shifting part of the computation to the offline phase, where time limits are more relaxed and the costs can be better amortized (e.g. via parallelization).

We obtain our methods by combining three basic contributions: 1) a technique to estimate the probability of future outcomes, given past observations; 2) a scheme for building a "contingency table", with precomputed solutions to guide the online choices; and 3) an efficient fixing heuristic for adapting the precomputed solutions to run-time conditions.

We ground our approaches on a (numeric) energy management problem with uncertain loads and generation from Renewable Energy Sources (RES), and on a (discrete) Traveling Salesman Problem with uncertain travel times. We show how our methods reach a solution quality comparable with the anticipatory algorithm, with lower (or dramatically lower) online computational cost.

The paper is structured as follows: in Section 2 we cover background and related works. We show our contributions in Section 3 and their grounding in Section 4. Experimental results are in Section 5 and concluding remarks in Section 6.

2 Background and Related Work

Historically, the literature on optimization under uncertainty has focused on offline problems [Powell, 2016; Sahinidis, 2004]. These methods usually rely on sampling (yielding a number of *scenarios*) to obtain a statistical model of future uncertainty. Robust solutions can be obtained by building one copy of the decision variables per scenario, and linking them via *non-anticipativity constraints* (decisions based on the same observations should be identical). This is known as the Sample Average Approximation method [Kleywegt *et al.*, 2002]: it provides convergence guarantees under reasonable assumptions, and can substantially outperform myopic optimization.

More recently, improvements in the solution techniques and computational power have enabled the application of similar methods to online optimization. This lead to so-called *online anticipatory algorithms*, which proved very effective at finding robust, high quality, solutions as uncertainty slowly reveals itself. A very good overview of this line of research is provided by [Hentenryck and Bent, 2009], while notable algorithms and results are described in [Chang *et al.*, 2000; Bent and Van Hentenryck, 2004; 2005; Mercier and Van Hentenryck, 2008; De Filippo *et al.*, 2018].

Online anticipatory algorithm typically rely on scenario sampling to estimate the possible developments for a fixed number of future steps, known as *look-ahead horizon*. Larger sample sizes result in higher accuracy, but also in more and bigger (possibly NP-hard) problems to be solved. This is a strong limitation, since in many practical cases online decision must be produced within strict time limits. Considerable research effort has therefore focused on improving the efficiency of these algorithms. For example, the CONSENSUS and REGRET algorithms from [Bent and Van Hentenryck, 2004] both attempt to reduce the number of problems w.r.t. the earlier EXPECTATION approach. Computational studies such as [Borodin and El-Yaniv, 2005; Mercier and Van Hentenryck, 2007] aim at characterizing the algorithm sensitivities to their design parameters (such as the number of sampled scenarios and the look-ahead horizon). The approaches from [John and Langley, 1996; Philpott and De Matos, 2012; Lin *et al.*, 2013] attempt instead to reduce the number of scenarios by increasing their relevance, and in particular by taking into account past observations while sampling.

3 "Taming" an Anticipatory Algorithm

Our goal is reducing the online computational cost of a given sampling-based anticipatory algorithm, referred to as A, by exploiting the existence of an offline phase. Such A algorithm is the main input for all our methods.

Similarly to [Hentenryck and Bent, 2009], we view online optimization under uncertainty as a stochastic n-stage problem. At each stage some uncertainty is resolved, and some decision must be made. A stage k is associated to a decision variable x_k (e.g. the power flows between loads and generators) and a state variable s_k (summarizing the effect of past decisions). All variables may be vector-valued.

We assume that uncertainty is *exogenous*, i.e. not affected by the decisions (e.g. the RES generation does not depend on how we choose to route it), and modeled via a set of random variables ξ_i. Which variables are observed at each stage depends on the state, and is controlled by a *peek* function:

$$O = peek(s_k) \tag{1}$$

which returns a set O with the indices of the observed variables. We will use the notation ξ_O to denote the observed ξ variables, and $\xi_{\bar{O}}$ for the unobserved ones.

Base Anticipatory Algorithm

Sampling-based algorithms rely on scenarios to estimate future outcomes. Formally, a scenario ω specifies a value ξ_i^ω for all the random variables. Given a set Ω of scenarios, the system state s_k, and values for ξ_O corresponding to the observed uncertainty, we assume that A can compute the decisions for stage k:

$$x_k = A(s_k, \xi_O, \{\xi^\omega\}_{\omega\in\Omega}) \tag{2}$$

Once the decisions are computed, the next state can be determined. This is controlled via a state transition function *next*

Algorithm 1 ANTICIPATE (s_1, ξ)

```
Sample a set of scenarios $\Omega$
for $k = 1 \dots n$ do
    $O = O \cup peek(s_k)$                                  # observe uncertainty
    $x_k = A(s_k, \xi_O, \{\xi^\omega\}_{\omega\in\Omega})$      # obtain decisions
    $s_{k+1} = next(s_k, x_k, \xi_O)$                       # determine next state
return $s, x$
```

that, based on the current state, decisions, and observed uncertainty, computes:

$$s_{k+1} = next(s_k, x_k, \xi_O) \tag{3}$$

Given the initial state s_1, a set of scenarios Ω, and a set of values sampled from ξ_k (which represent the online observations), we can simulate the execution of the method via Algorithm 1. The O set is assumed to be initially empty. This generic anticipatory algorithm will be referred to as ANTICIPATE in the remainder of the paper.

Offline Information

Defining a representative set of scenarios Ω is critical for the approach effectiveness and it is usually done by exploiting the available *offline information*. Here, we assume that the such offline information is a collection of observed uncertain values. This definition captures many practical cases (e.g. forecasts or predictions, historical data, data from test runs). More importantly, this means that *the offline information is in fact a collection of scenarios*. We will denote the offline information as I, index its element with ω, and assume (as it is usual) that I is representative of the true probability distribution of the random variables. A set Ω of scenarios for ANTICIPATE can be obtained by sampling a number of elements uniformly at random from I.

3.1 Basic Contributions

All our methods rely on three techniques, which will be described in this section.

Probability Estimation for Scenario Sampling

Using a fixed set of scenarios (as in ANTICIPATE) is beneficial when the ξ_i variables are statistically independent. When they are not, however, the set of scenarios may loose relevance as uncertainty is resolved. For example, a scenario based on a cloudy day forecast becomes less likely if fair weather is observed at the beginning of online execution.

Formally, at stage k we wish to sample scenarios that are likely to occur given the past observations, i.e. to sample the unobserved variables $\xi_{\bar{O}}$ according to the conditional distribution $P(\xi_{\bar{O}} \mid \xi_O)$. If we draw the scenarios from the offline information (which guarantees physically meaningfulness), then sampling requires to estimate the conditional probabilities *of the elements in I*. From basic probability theory, this is given by the ratio of two joint probabilities:

$$\forall \omega \in I, \quad P(\xi_{\bar{O}}^\omega \mid \xi_O) = \frac{P(\xi_O \xi_{\bar{O}}^\omega)}{P(\xi_O)} \tag{4}$$

where $P(\xi_O \xi_{\bar{O}}^\omega)$ is the probability that observed values occur together with the remaining predictions from the scenario, and $P(\xi_O)$ is the probability that the values are observed. The

Algorithm 2 BUILDTABLE $(\mathrm{s}_1, \mathrm{AA})$

```
for $\omega \in I$ do
    $s^\omega, x^\omega = AA(s_1, \xi^\omega)$
return $\{\xi^\omega, s^\omega, x^\omega\}_{\omega \in T}$
```

joint probability at the numerator can be approximated via any density estimation method, such as Kernel Density Estimation [Silverman, 2018], Gaussian Mixture Models [Gauvain and Lee, 1994], or recent Deep Learning techniques such as Normalizing Flows [Rezende and Mohamed, 2015] and Real NVP [Dinh *et al.*, 2016]. *Any such method can be trained on the offline information to obtain an estimator $\tilde{P}(\xi)$ for the joint distribution of the random variables*.

An estimator for the distribution $P(\xi_O)$ at the denominator can then be derived from $\tilde{P}(\xi)$ via marginalization, i.e. by averaging the contribution of the unobserved variables. We perform this step over all possible completions of the observed values *in the offline information*. Overall, we have:

$$\forall \omega \in I, \quad \tilde{P}(\xi_{\bar{O}}^\omega \mid \xi_O) = \frac{\tilde{P}(\xi_O \xi_{\bar{O}}^\omega)}{\sum_{\omega' \in T} \tilde{P}(\xi_O \xi_{\bar{O}}^{\omega'})} \tag{5}$$

This estimator defines a discrete distribution over the offline information I. The chosen marginalization technique guarantees an estimate that is approximately proportional (not approximately equal) to the true $P(\xi_O)$. Hence, we have that:

$$\forall \omega \in I, \quad P(\xi_{\bar{O}}^\omega \mid \xi_O) \propto \tilde{P}(\xi_{\bar{O}}^\omega \mid \xi_O) \tag{6}$$

Sampling from I according to Equation (5) yields scenarios with a distribution that takes into account the observed values.

Building a Contingency Table

If a significant amount of time is available in the offline phase, we can exploit the offline information more aggressively, by trying to prepare for each likely future development. Intuitively, we can *treat each scenario $\omega \in I$ as if it were an actual sequence of online observations, and process it via some anticipatory algorithm*. By doing this, we build a pool of solutions that can then be used to guide an online method.

The process is outlined in Algorithm 2, which requires as input the initial state s_1 of the system, and a solution algorithm AA, accepting the same parameters as ANTICIPATE. The result is an augmented version of the offline information, where each scenario ω is additionally associated to the sequence of states s^ω visited by the algorithm and its sequence of decisions x^ω. We refer to this data structure as *contingency table*, and to its elements as *traces*. We denote the table as T.

Fixing Heuristic

We use the traces from T to guide an efficient *fixing heuristic*, which tries to choose decisions having the largest chance of being optimal. Formally, it solves:

$$\arg\max\{P^*(x_k \mid s_k \xi_O) : x_k \in X_k\} \tag{7}$$

where P^* is the probability that the chosen x_k is optimal, given the state s_k and the observed uncertainty. The X_k set represents the feasible decision space, which is defined via problem-dependent constraints and auxiliary variables.

Algorithm 3 FIXING (s_1, ξ, T)

```
for $k = 1 \dots n$ do
    $O = O \cup peek(s_k)$
    $\Omega$ = top elements in $T$ by descending Equation (10)
    Compute $p_{jv}$ and/or $p_\omega$ based on $\Omega$
    Solve Equation (8)/(11) to obtain $x_k$
    $s_{k+1} = next(s_k, x_k, \xi_O)$
return $s, x$
```

Closed-forms for P^* can be obtained separately for discrete and numeric problems, based on the contingency table. The process is described in detail in Appendix A, and relies on several approximations. Overall, *in case of discrete decisions, the problem from Equation* (7) *translates into*:

$$\arg\min\left\{-\sum_{j=1}^{m}\sum_{v \in D_j} \log p_{jv} [\![x_{kj} = v]\!] : x_k \in X_k\right\} \tag{8}$$

where $[\![\cdot]\!]$ denotes the truth value of a predicate, D_j is the domain of x_{kj}, and:

$$p_{jv} = \frac{\sum_{\omega \in T, x_{kj}^\omega = v} P(\omega)}{\sum_{\omega \in T} P(\omega)} \tag{9}$$

Here, $P(\omega)$ is a compact notation for the probability that *we reach the same state as trace ω, and then everything goes according to plan*. It can be approximated using:

$$\forall \omega \in T, \quad P(\omega) \propto \tilde{P}(s_{sk+1}^\omega \mid s_k)\tilde{P}(\xi_{\bar{O}}^\omega \mid \xi_O) \tag{10}$$

where $\tilde{P}(\xi_{\bar{O}} \mid \xi_O)$ is the estimator from Equation (5), and $\tilde{P}(s_{sk+1} \mid s_k)$ is a second estimator obtained via similar means. The cost function in Equation (8) is linear if a one-hot encoding is adopted for x_{kj}, and the size of T affects only the computation of the p_{jv} values. Overall, the problem is efficient to solve. *In case of numeric decisions, we have instead*:

$$\arg\min\left\{\sum_{j=1}^{m}\sum_{\omega \in T} p_\omega \frac{1}{2\sigma_j}(x_{kj} - x_{kj}^\omega)^2 : x_k \in X_k\right\} \tag{11}$$

with:

$$p_\omega = \frac{P(\omega)}{\sum_{\omega' \in T} P(\omega')} \tag{12}$$

The cost function is quadratic and convex, and the problem size is small due to the same arguments as Equation (8).

Intuitively, the discrete version of the heuristic is related to minimizing weighted discrepancies w.r.t. the traces in T, i.e. to weighted Hamming distances. The numeric version is instead related to weighted Euclidean distances. The pseudocode for the heuristic is provided in Algorithm 3. The only difference with the process described so far is that the p_{jv} and p_ω probabilities may be computed based on a subset Ω of the full contingency table. This may be useful to bias the choice of the online decision according to the most relevant traces.

3.2 Hybrid Offline/Online Methods

Our three solution methods can now be defined with relative ease, by combining the techniques just described.

Algorithm 4 ANTICIPATE-D (s_1, ξ)

```
Train the $\tilde{P}(\xi)$ estimator on $I$
for $k = 1 \dots n$ do
    $O = O \cup peek(s_k)$
    Sample $\Omega$ from $T$, according to Equation (5)
    $x_k = A(s_k, \xi_O, \{\xi^\omega\}_{\omega \in \Omega})$
    $s_{k+1} = next(s_k, x_k, \xi_O)$
return $s, x$
```

Algorithm 5 CONTINGENCY (s_1, ξ)

```
Train the $\tilde{P}(\xi)$ estimator on $I$
$T = $ BUILDTABLE$(s_1, $ ANTICIPATE$)$
Train the $\tilde{P}(s_k s_{k+1})$ estimators on $T$, for all $k$
$s, x = $ FIXING$(s_1, \xi, T)$
return $s, x$
```

ANTICIPATE-D. Our first hybrid method is obtained from ANTICIPATE by simply replacing the static set of samples with a dynamically adjusted one. The dynamic set can be populated according to the estimated probabilities from Equation (5), so as not to loose relevance: this may enable to reach similar solution qualities with fewer scenarios, at the cost of training an estimator offline. We refer to this approach as ANTICIPATE-D, and its pseudo-code is in Algorithm 4

CONTINGENCY. The second method is based on the idea of computing robust solutions for the scenarios in the offline information, and then use them as guidance for the FIXING heuristic. Robust solutions are obtained by using ANTICIPATE, so that hopefully the (fast) fixing heuristic will be able to match their quality: the price to pay is a hefty offline computational effort. We refer to this approach as CONTINGENCY, and its pseudo-code is reported in Algorithm 5.

CONTINGENCY-D. Our final method is similar to the previous one, except that the contingency table is populated with *non-robust* solutions. This is done by using ANTICIPATE *with a single scenario, given by the values of ξ^ω (i.e. the pretend online observations)*. This technique (referred to as ANTICIPATE$_1$) provides perfect information about the future, so that achieving robustness is entirely delegated to the FIXING heuristic. The approach is likely to loose reliability, but has two important advantages: 1) lower offline computational costs; and 2) while ANTICIPATE is a stochastic algorithm, ANTICIPATE$_1$ is deterministic. So, this method may provide anticipatory-like results even when no anticipatory algorithm is available. We refer to this method as CONTINGENCY-D, and its pseudo-code is reported in Algorithm 6.

4 Grounding the Approaches

Grounding our approaches requires to specify: 1) the x, s and ξ variables, 2) the $peek$ and $next$ functions, 3) the sampling-based algorithm A, and 4) the feasible space X_k for the FIXING heuristic. Additionally, evaluating the solution quality requires to define 5) a cost metric.

We show how this can be done in two case studies: 1) a Virtual Power Plant energy management problem with numerical

Algorithm 6 CONTINGENCY-D (s_1, ξ)

```
Train the $\tilde{P}(\xi)$ estimator on $I$
$T = $ BUILDTABLE$(s_1, $ ANTICIPATE$_1)$
Train the $\tilde{P}(s_k s_{k+1})$ estimators on $T$, for all $k$
$s, x = $ FIXING$(s_1, \xi, T)$
return $s, x$
```

decisions; and 2) a combinatorial Traveling Salesman Problem with uncertain travel times. In both cases, the input anticipatory algorithm A is given by a Mathematical Programming model, based on the Sample Average Approximation. The models are slight improvements over those by [De Filippo *et al.*, 2018], whose work brought to attention the interplay between offline and online phases. Both approaches are serviceable, but not necessarily representative of the state-of-the-art (especially for the TSP). The full useful information for grounding our approaches are in a companion repository[1], while here we focus on the minimal information needed.

4.1 The VPP Case Study.

A Virtual Power Plant aggregates different sources of power generation and consumption to offer a predictable power envelope. Managing a VPP requires to route power flows so as to satisfy the demand, to obey physical limits, and to minimize the operating costs [Palma-Behnke *et al.*, 2011; Bai *et al.*, 2015]. Both the demand and the RES generation are uncertain. Formally, the decision vector x_k specifies the power flow x_{kj} to/from each node (demand, generator, storage...). In particular, we assume that x_{kS} refers to flow for the storage system. The state component s_{kS} corresponds to the storage charge level, while s_{kD} to its flow direction. The random variable ξ_{kL} corresponds to the load, while ξ_{kR} to the RES generation. The $peek$ function simply returns the pairs (k, L) and (k, R). The $next$ function is given by:

$$s_{k+1,S} = s_k + \eta x_{kS} \tag{13}$$

$$s_{k+1,D} = 0 \text{ if } x_{kS} \leq 0 \text{ and } 1 \text{ otherwise} \tag{14}$$

where η is the charging efficiency of the storage system. The feasible space X_k is given by the Mathematical Program:

$$\xi_{kL} = \sum_{j=1}^{m} x_{kj} + \xi_{kR} \tag{15}$$

$$l_j \leq x_{kj} \leq u_j \qquad \forall j \in [1..m] \tag{16}$$

$$0 \leq s_k + \eta x_{kS} \leq \Gamma \tag{17}$$

$$x_k \in \mathbb{R}^m \tag{18}$$

where Eq. (15) enforces power balance, Eq. (16) states the physical limits for the power flows, and Eq. (17) those for the storage charge. The cost incurred at each stage is given by:

$$\sum_{j=1}^{m} c_{kj} x_{kj} + \alpha |s_{kD} - s_{k+1,D}| \tag{19}$$

where c_{kj} is a cost associate to each flow. Unlike the model from [De Filippo *et al.*, 2018], we include a cost term α related to storage wear-off, which increases each time the corresponding flow switches direction. Due to this term, the input

[1]https://github.com/alleDe/OffOn

algorithm A needs to solve an NP-hard problem, while the fixing heuristic has no such need.

4.2 The TSP Case Study.

As a second case study, we consider a TSP over an asymmetric, fully connected, graph with uncertain, exogenous, travel times (e.g. the visits have a negligible impact on traffic). In this case, the x_k vector includes m components x_{kj}, each equal to 1 iff j is the next node to be visited. There is no x_{kj} variable for the depot, which is reached by default once all other nodes are visited. The state vector contains a component s_{kj} equal to 1 iff node j (excluding the depot) has been visited at stage k, plus a s_{kC} component specifying the index of the current node. The uncertainty is modeled via random variables ξ_{ij}, each associated to the travel time between nodes i and j. The travel times for all outgoing arcs from i are observed when the node is visited, i.e. the $peek$ function returns the pairs (s_{kC}, j) with $j \in [1..m]$. The $next$ function is:

$$s_{k+1,C} = \sum_{j=1}^{m} j\, x_{kj} \tag{20}$$

$$s_{k+1,j} = \max(x_{kj}, s_{k+1,j}) \qquad \forall j \in [1..m] \tag{21}$$

where Equation (20) makes sure that the value $s_{k+1,C}$ matches the index of the next node to be visited. The feasible space X_k is given by the Mathematical Program:

$$\sum_{j=1}^{m} x_{kj} = 1 \tag{22}$$

$$x_{kj} \leq 1 - s_{kj} \qquad \forall j \in [1..m] \tag{23}$$

which forces moving to a single, unvisited node. The cost incurred at each stage is the travel time to the next node, i.e.:

$$\sum_{j=1}^{m} \xi_{s_{kC},j}\, x_{kj} \tag{24}$$

The final cost is obtained by summing the cost of each stage, plus the distance from the last visited node to the depot. Once again, while the anticipatory algorithm A needs to solve an NP-hard problem (stochastic TSP), the fixing heuristic has no such need and is therefore much faster.

5 Experimentation

We empirically evaluated the three hybrid offline/online methods on realistic instances for the two case studies. The baseline in both cases are myopic heuristics, obtained by running ANTICIPATE with an empty set of scenarios.

5.1 Experimental Setup

Our methods are evaluated over different uncertainty realizations, obtained by sampling the random variables for the loads and RES generation in the VPP, and for the travel times in the TSP. We use models of uncertainty that ensure realistic statistical dependence between the variables. This process yields the offline information I and the sequences of observations for the experiments.

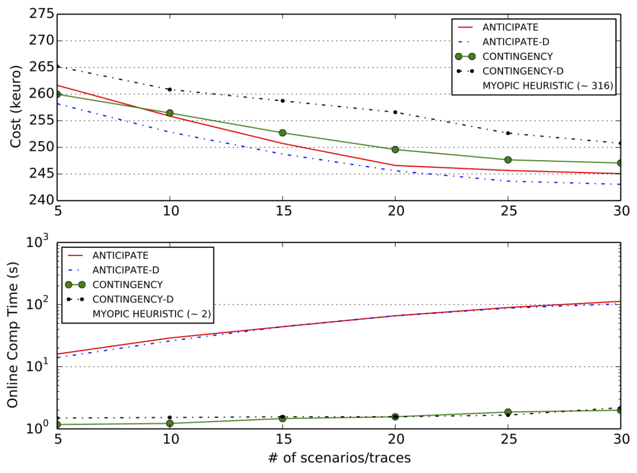

Figure 1: Methods solution/quality comparison for the VPP

For the VPP, grid electricity prices change every 15 minutes, which is also the duration of our online stages. New offline information (e.g. market prices) becomes available every day, hence our horizon corresponds to $24 \times 4 = 96$ stages. We use (real) physical bounds for power generation from [Bai *et al.*, 2015; Espinosa and Ochoa, 2015]. The initial battery state, efficiency, and power flow limit, etc. are also based on real data [Bai *et al.*, 2015; Espinosa and Ochoa, 2015]. Different instances have then been obtained by manually scaling load and RES generation. For the TSP we use classical benchmarks[2], by including problems from 10 to 40 nodes. In the TSP each stage represents a visit, hence our horizon corresponds to the total number of nodes.

We use Kernel Density Estimation (with Gaussian Kernels) to obtain all approximate distributions. As an underlying solver we use Gurobi [3], which can handle both MILPs and Quadratic Programs. Each evaluated algorithm and configuration is run 50 times, with the same 50 sequences of realizations. We use a time limit of 300 seconds for both the VPP and the TSP. For each run we record both the time required by each approach and the corresponding solution cost, and we report their average values over the 50 realizations. In all cases, $|I| = |T| = 100$, and for the CONTINGENCY method, the contingency table is built using ANTICIPATE with 20 scenarios. Due to space constraints, we report results only for a few representative instances.

5.2 Discussion

The offline training times of the KDE models are roughly the same for all the three hybrid methods (~ 65 sec for the VPP and ~ 32 sec for the TSP). Building the contingency tables for CONTINGENCY takes $\sim 6{,}000$ sec in the VPP and $\sim 15{,}000$ sec in the TSP, but only ~ 400 and $\sim 2{,}000$ sec for CONTINGENCY-D.

In Figure 1 we show the cost/quality tradeoff of the proposed methods and of ANTICIPATE for the VPP (base instance) and in Figure 2 for the TSP (a representative 20 customers instance). The use of a dynamic set of scenarios al-

[2]http://myweb.uiowa.edu/bthoa/TSPTWBenchmarkDataSets.htm
[3]Available at http://www.gurobi.com

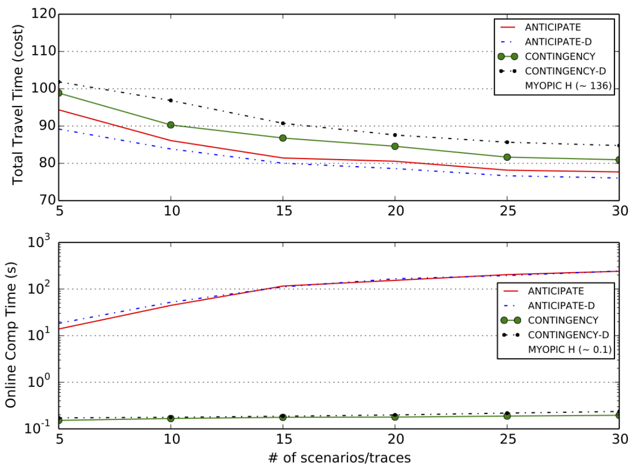

Figure 2: Methods solution/quality comparison for the TSP

Method	VPP (σ)	TSP (σ)
Myopic H	8.499	7.106
ANTICIPATE	4.994	1.889
ANTICIPATE-D	5.730	2.846
CONTINGENCY	5.557	3.788
CONTINGENCY-D	7.017	5.934

Table 1: Standard deviation comparison for the VPP and the TSP

lows ANTICIPATE-D to work better than ANTICIPATE. The CONTINGENCY method is surprisingly close in terms of quality to the original anticipatory algorithm, especially considered its dramatically smaller online computational cost (up to two orders of magnitude). CONTINGENCY-D performs slightly worse than CONTINGENCY, but it still much better than the myopic heuristic. Increasing the number of guiding traces is beneficial in particular for CONTINGENCY-D.

In Table 1 we show the standard deviation for the solution quality over the 50 realizations with 20 scenarios/traces for both the VPP and the TSP (on the same instances as Figure 1 and Figure 2). All values are significantly lower than the quality gap with the myopic heuristic. As expected CONTINGENCY-D tends to be less stable than the other methods due to its reliance on non-robust guiding traces.

6 Conclusion

We have presented three methods that can be applied *to a generic anticipatory algorithm* to reduce its online computational effort by exploiting offline information. In particular, both CONTINGENCY and CONTINGENCY-D are dramatically faster than ANTICIPATE during online operation; CONTINGENCY is significantly more reliable in terms of quality, but may require a substantial amount of offline computation. The ANTICIPATE-D technique provides a modest advantage in terms of solution time, but can match and even surpass ANTICIPATE in terms of quality.

The ability to shift part of the computational cost to an offline stage provides a significant degree of flexibility to stochastic anticipatory algorithms, and is likely to increase their applicability. We believe there is room for improving the scalability and efficiency of our methods, and achieving this goal is part of our current research directions. We also plan to apply our approaches to different application problems.

Acknowledgments

This work has been partially supported by the H2020 Project AI4EU (g.a. 825619), the VIRTUS Project (CCSEB-00094) and the Emilia-Romagna Region.

A Deriving the FIXING Heuristic

Our main goal will be to obtain a closed-form for P^* in Equation (7), which will require several approximations. We start by treating all components in x_k as statistically independent. This allows to state P^* as a product of $P^*(x_{kj} \mid s_k \xi_O)$ probabilities, related to individual components of x_k. Applying a log transformation then leads to the equivalent problem:

$$\arg\min \left\{ -\sum_{j=1}^{m} \log P^*(x_{kj} \mid s_k \xi_O) : x_k \in X_k \right\} \tag{25}$$

where m is the cardinality of x_k. We then assume that a decision x_{kj} is optimal if the current optimization process is similar to a trace in the contingency table, and x_{kj} is similar to the decision made in that circumstance. Formally, we can obtain $P^*(x_{kj})$ via marginalization:

$$P^*(x_{kj} \mid s_k \xi_O) = \frac{\sum_{\omega \in T} P(\omega) P^*(x_{kj} \mid \omega)}{\sum_{\omega \in T} P(\omega)} \tag{26}$$

where $P(\omega)$ is compact notation for $P(s^\omega_{k+1} \xi^\omega_{\bar{O}} \mid \xi_O s_k)$. By assuming independence between the s and ξ variables, and applying the techniques used for Equation (5), we get:

$$P(\omega) \propto \tilde{P}(\xi^\omega_{\bar{O}} \mid \xi_O) \frac{\tilde{P}(s_k s^\omega_{k+1})}{\sum_{\omega' \in T} \tilde{P}(s_k s^{\omega'}_{k+1})} \tag{27}$$

where the estimator for $\tilde{P}(s_k s_{k+1})$ can be trained over data from the contingency table. We now need a way to estimate $P^*(x_{kj} \mid \omega)$. *In the discrete case*, we assume that x_{kj} is optimal iff it matches the value from the contingency table, i.e. $P^*(x_{kj} \mid \omega)$ is equal to the truth value of the predicate $x_{ki} = x^\omega_{kj}$. Hence, Equation (26) becomes:

$$P^*(x_{kj} \mid s_k \xi_O) = \sum_{v \in D_j} p_{jv} [\![x_{kj} = v]\!] \tag{28}$$

with p_{jv} as in Equation (9). By applying the log transformation, and using the fact that values in D_j are mutually exclusive, we get the discrete formulation from Equation (8).

In the numeric case, we assume that decisions close to the one in the trace have a chance of being optimal, which follows a Normal distribution. Formally, we have that:

$$P^*(x_{kj} \mid \omega) = \frac{1}{\sqrt{2\pi}\sigma_j} e^{-\frac{1}{2\sigma_j}(x_{kj} - x^\omega_{kj})^2} \tag{29}$$

where σ_j is the standard deviation of the value of x_{kj} in the contingency table. By applying the log transformation to Equation (26), then Jensen's inequality, and by getting rid of offset terms (which have no impact on optimization), we get the numeric formulation from Equation (11). Note that, due to the use of Jensen's inequality, the resulting cost function is actually an approximated upper bound for the original probability.

References

[Bai *et al.*, 2015] Hao Bai, Shihong Miao, Xiaohong Ran, and Chang Ye. Optimal dispatch strategy of a virtual power plant containing battery switch stations in a unified electricity market. *Energies*, 8(3):2268–2289, 2015.

[Bent and Van Hentenryck, 2004] Russell Bent and Pascal Van Hentenryck. Regrets only! online stochastic optimization under time constraints. In *AAAI*, volume 4, pages 501–506, 2004.

[Bent and Van Hentenryck, 2005] Russell Bent and Pascal Van Hentenryck. Online stochastic optimization without distributions. In *ICAPS*, volume 5, pages 171–180, 2005.

[Borodin and El-Yaniv, 2005] Allan Borodin and Ran El-Yaniv. *Online computation and competitive analysis*. cambridge university press, 2005.

[Chang *et al.*, 2000] Hyeong Soo Chang, Robert Givan, and Edwin KP Chong. On-line scheduling via sampling. In *AIPS*, pages 62–71, 2000.

[De Filippo *et al.*, 2018] Allegra De Filippo, Michele Lombardi, and Michela Milano. Methods for off-line/on-line optimization under uncertainty. In *IJCAI*, pages 1270–1276, 2018.

[Dinh *et al.*, 2016] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real nvp. *arXiv preprint arXiv:1605.08803*, 2016.

[Espinosa and Ochoa, 2015] Alejandro N. Espinosa and Luis N. Ochoa. Dissemination document "low voltage networks models and low carbon technology profiles". Technical report, University of Manchester, June 2015.

[Gauvain and Lee, 1994] Jean-Luc Gauvain and Chin-Hui Lee. Maximum a posteriori estimation for multivariate gaussian mixture observations of markov chains. *IEEE transactions on speech and audio processing*, 2(2):291–298, 1994.

[Hentenryck and Bent, 2009] Pascal Van Hentenryck and Russell Bent. *Online stochastic combinatorial optimization*. The MIT Press, 2009.

[John and Langley, 1996] George H John and Pat Langley. Static versus dynamic sampling for data mining. In *KDD*, volume 96, pages 367–370, 1996.

[Kleywegt *et al.*, 2002] Anton J Kleywegt, Alexander Shapiro, and Tito Homem-de Mello. The sample average approximation method for stochastic discrete optimization. *SIAM Journal on Optimization*, 12(2):479–502, 2002.

[Lin *et al.*, 2013] Minlong Lin, Ke Tang, and Xin Yao. Dynamic sampling approach to training neural networks for multiclass imbalance classification. *IEEE Transactions on Neural Networks and Learning Systems*, 24(4):647–660, 2013.

[Mercier and Van Hentenryck, 2007] Luc Mercier and Pascal Van Hentenryck. Performance analysis of online anticipatory algorithms for large multistage stochastic integer programs. In *IJCAI*, pages 1979–1984, 2007.

[Mercier and Van Hentenryck, 2008] Luc Mercier and Pascal Van Hentenryck. Amsaa: A multistep anticipatory algorithm for online stochastic combinatorial optimization. *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems*, pages 173–187, 2008.

[Palma-Behnke *et al.*, 2011] Rodrigo Palma-Behnke, Carlos Benavides, E Aranda, Jacqueline Llanos, and Doris Sáez. Energy management system for a renewable based microgrid with a demand side management mechanism. In *Computational intelligence applications in smart grid (CIASG), 2011 IEEE symposium on*, pages 1–8. IEEE, 2011.

[Philpott and De Matos, 2012] Andrew B Philpott and Vitor L De Matos. Dynamic sampling algorithms for multi-stage stochastic programs with risk aversion. *European Journal of Operational Research*, 218(2):470–483, 2012.

[Powell, 2016] Warren B. Powell. *A Unified Framework for Optimization Under Uncertainty*, chapter 3, pages 45–83. 2016.

[Rezende and Mohamed, 2015] Danilo Jimenez Rezende and Shakir Mohamed. Variational inference with normalizing flows. *arXiv preprint arXiv:1505.05770*, 2015.

[Sahinidis, 2004] Nikolaos V. Sahinidis. Optimization under uncertainty: state-of-the-art and opportunities. *Computers & Chemical Engineering*, 28(6):971 – 983, 2004. FOCAPO 2003 Special issue.

[Silverman, 2018] Bernard W Silverman. *Density estimation for statistics and data analysis*. Routledge, 2018.