The Blind Men and the Elephant: Integrated Offline/Online Optimization Under Uncertainty

Allegra De Filippo, **Michele Lombardi** and **Michela Milano**
DISI, University of Bologna, Italy
{allegra.defilippo, michele.lombardi2, michela.milano}@unibo.it

Abstract

Optimization problems under uncertainty are traditionally solved either via offline or online methods. Offline approaches can obtain high-quality robust solutions, but have a considerable computational cost. Online algorithms can react to unexpected events once they are observed, but often run under strict time constraints, preventing the computation of optimal solutions. Many real world problems, however, have *both offline and online elements*: a substantial amount of time and information is frequently available (offline) before an online problem is solved (e.g. energy production forecasts, or historical travel times in routing problems); in other cases both offline (i.e. strategic) and online (i.e. operational) decisions need to be made. Surprisingly, the interplay of these offline and online phases has received little attention: like in the blind men and the elephant tale, we risk missing the whole picture, and the benefits that could come from *integrated offline/online optimization*. In this survey we highlight the potential shortcomings of pure methods when applied to mixed offline/online problems, we review the strategies that have been designed to take advantage of this integration, and we suggest directions for future research.

1 Introduction

Many constrained decision problems in the real world have elements of uncertainty, e.g. customer demands or travel times in vehicle routing, renewable production in energy systems, resource availability in production scheduling. Indeed, the ability to deal with uncertainty is increasingly regarded as a key requirement for the applicability of optimization methods from the Artificial Intelligence and Operations Research domain (such as Constraint Programming, SAT Modulo Theories, or Mathematical Programming).

All constrained stochastic optimization problems have a stage-based structure: at each stage some decision must be made and then some elements of uncertainty are revealed. At least one stage is always present, but there is no upper limit (in which case the problem has an infinite horizon).

The available solution methods tend to be *polarized between offline and online methods*. Offline approaches are concerned with making decisions before the uncertainty is revealed: they can obtain high quality and robust solutions, but they have a significant computational cost. Online algorithms focus on reacting to unexpected events once they are observed, but often run under strict time constraints, making it challenging to compute high-quality solutions.

Real world problems, however, *often mix offline and online elements*. In many cases, a substantial amount of information about the uncertainty (e.g. in the form of historical solutions, event logs or probability distributions) is available before it is revealed, i.e. before the online execution starts. In other cases we must make both strategic (offline) and operational (online) decisions: in routing problems we may choose a fixed customer assignment and then route dynamically based on the traffic conditions; in production scheduling we may devise an initial plan, to be revised at run time in case of disruptions.

We observe that *offline and online elements are tightly integrated, even if they are typically analyzed in isolation* in the literature. We risk being like the blind men of the tale, trying to make sense of what an elephant is by touching each part of the animal, and missing the whole picture.

Failing to realize that we are dealing with an elephant may lead to undesirable side effects and missed opportunities: for example, the solutions produced by pure offline methods may conflict with the behavior of the approach used for making online decisions; conversely, an online solver may struggle with producing solutions within a reasonable time budget, not realizing that some degree of offline preparation could greatly simplify the issue. At the same time, offline and online aspects have specificities that justify the use of dedicated approaches, which may be combined effectively once a clearer understanding of the problem has been reached. We experienced these shortcomings and observed the benefits of a tighter integration first-hand, and we want to bring them to the attention of the wider scientific community.

In this work, we survey pure offline and online approaches, pointing out their strengths, together with hints of hybrid offline/online aspects in the problems they tackle, and the resulting drawbacks. We also attempt to give greater visibility to the limited number of approaches that integrate both offline and online aspects. We focus on constrained optimization, with an emphasis on combinatorial problems. The goal is to provide a more consistent view of mixed offline/online problems, how their different parts can be tackled, and how a tighter integration can be achieved, with the aim to inform future research: after all, history shows that an elephant can be a useful ally, if properly handled.

The paper is structured as follows: in Section 2 and Section 3 we present and analyze representative approaches, respectively for pure offline and online stochastic optimization. In Section 4 we review solution methods that achieve a tighter degree of integration, with different degrees of clarity in the identification of the offline and online elements. In Section 5 we provide concluding remarks and highlight some promising research directions.

2 Offline Optimization Under Uncertainty

Stochastic optimization methods have traditionally focused on strategic planning, under relaxed time constraints. Many such approaches focus on two-stage stochastic problems: these require to determine a single set of decisions for stage-1; after such decisions have been implemented, uncertain outcomes are observed and corrective measures (so-called recourse actions) can be applied to improve the cost or recover feasibility. Anticipating the recourse actions is necessary to identify the best stage-1 decisions.

Here we stress that *recourse actions are simply an online optimization problem in disguise*. This allows us to see from a new perspective all the techniques commonly employed to (e.g.) manage computational cost/quality trade-offs in stochastic optimization.

The Sample Average Approximation. In most practical cases, stochastic optimization requires to take into account large (sometimes infinite) amounts of (possibly rare) events. A common strategy to retain scalability with such a large search space is the Sample Average Approximations (SAA), which is a staple of Stochastic Programming [Shapiro, 2013].

In this approach, the probability distributions of the random variables are approximated via a finite number of samples: this yields a collection of realizations referred to as *scenarios*. Then, a set (i.e. a copy) of deterministic decisions is associated to each scenario, which allows to deal with expected values and chance constraints via summations and averages (see e.g. [Zhang and Li, 2011]).

The SAA is a general method that greatly improves over exact enumeration in terms of scalability, and can be applied even where enumeration is impossible (e.g. uncertain continuous quantities). However, the technique can still be very expensive from a computational point of view.

SAA and Two-Stage Problems. In two-stage problems, instantiating the SAA approach requires to define a single set of decision variables for stage-1, and one set of decision variables per scenario for stage-2.

In this context, the approximation has been proved to converge to the true expected values exponentially fast [Kleywegt *et al.*, 2002] with the number of samples. In [Charikar *et al.*, 2005] the authors go one step further by proving that a polynomial number of samples suffices to obtain a $(1+\varepsilon)$ approximation. The work shows that small variations of the SAA method are enough to obtain a bound on the sample size,

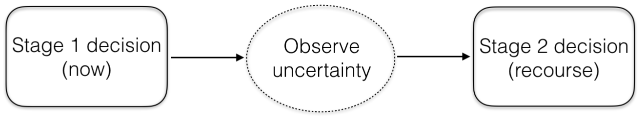

Figure 1: Two-Stage Stochastic Programming

even when the sampled problem is solved by an approximation algorithm, to reduce the computational load.

Indeed, much of the SAA related literature has focused on improving its efficiency. In the L-shaped method [Laporte and Louveaux, 1993], this is done by applying classical Benders decomposition, provided that the recourse actions can be determined by solving a Linear Program.

For example, [Schütz *et al.*, 2009] formulate a supply chain design problem as a two-stage stochastic program, where the stage-1 decisions are strategic location assignments, whereas stage-2 involves operational decisions. The authors solve the problem via the SAA in combination with dual decomposition by considering different sample sizes and different levels of data aggregation in the second stage. In [Verweij *et al.*, 2003], the authors present a detailed computational study of the application of the SAA method to tackle three classes of stochastic routing problems, by using decomposition and branch-and-cut to solve the second-stage estimate. A two-stage stochastic programming model is proposed in [Zhou *et al.*, 2013] for the optimal design of distributed energy systems, considering a vast case study. A decomposition-based solution strategy is used to solve the optimization problem, with a genetic algorithm performing the search on the stage-1 variables and a Monte Carlo method dealing with uncertainty in stage-2.

Two-Stage Approximation of Multi-Stage Problems. The above two-stage formulations can be extended to n-stage stochastic programs, which from a computational point of view are even more challenging. [Shapiro, 2008] has shown that the SAA method cannot be extended efficiently to multistage stochastic optimization problems: the number of required samples (and therefore of decision variables) must grow exponentially with the number of stages.

A common workaround consists in removing all non-anticipativity constraints between stages 2 to n, i.e. treating all future stages as one big second stage. This approach is at the core of the dual decomposition method from [CarøE and Schultz, 1999]: they propose a branch and bound algorithm for multistage stochastic programs in which bounding is achieved using the Lagrangian dual for the anticipatory relaxation, and branching is performed on non-anticipativity constraints. A generalization of the SAA method is proposed in [Shmoys and Swamy, 2004] for a broad class of multistage stochastic linear programs, based on the idea of treating the distribution as a black-box.

Analysis. In the SAA scheme, recourse actions are viewed just as a tool to obtain a more robust set of stage-1 decisions, which are the only actual output of these algorithms. This is a sound and powerful idea, but in a practical setting there are a few subtle drawbacks.

First, it is very likely that the observed outcomes will not match any of the scenarios used offline. In practice, this will require to run some optimization technique to obtain an actual set of recourse actions. There is *no guarantee that such technique will match that one employed by the offline method*, with some interesting side effects.

For example, let us assume that an exact solution technique (say a single Mathematical Programming model built on the SAA scheme) is used for offline optimization. Within the limits of the sampling approximation, this will lead to optimal stage-1 decisions, but also to optimal (estimated) recourse actions. In practice, however, optimal recourse actions may be unachievable: for example, if strict time constraints exist on the online execution, the only viable approach may be just a greedy heuristic. This creates a discrepancy between the estimated recourses and the real ones, which can significantly affect the solution quality.

The situation is even worse if an approximate method is used to compute the estimated recourse actions (e.g. a continuous relaxation of a discrete problem), since this can make the gap even wider. The same considerations apply to the computational trick of collapsing future stages in multi-stage problems: while very effective at improving scalability, such method introduces an assumption (i.e. perfect knowledge of the future when computing the recourse actions) that is very hard to satisfy online, even in an approximate fashion.

3 Online Optimization

When dealing with a problem with more than one stages, optimizing at run time provides the opportunity to adapt the solutions to unexpected events (since those can be observed): this is at the basis of stochastic online optimization. Since online approaches tackle multi-stage problems one stage at a time, they are not forced to look for policies in a huge search space, potentially reducing their computational cost.

However, online optimization approaches often need to perform under tight time limits on the solution process. This is the chief reason why in many practical settings online problems are tackled with heuristics with no (or very limited) look-ahead capabilities. It is actually possible to apply sampling-based algorithms also in an online setting: these are often referred to as anticipatory algorithms, many of which received excellent coverage in [Hentenryck and Bent, 2009]. These approaches can be very effective in practice, but their computational cost must be carefully controlled so as not to violate the time constraints.

Here, we observe that *sampling is possible only if some degree of information is available prior to the solution process, i.e. offline*. From this point of view, it is natural to wonder how such offline phase could be better exploited.

Myopic vs Anticipatory Algorithms. In many practical online problems, it is customary to consider each stage in isolation [Hentenryck and Bent, 2009]: this retains the ability to adapt to unexpected events, with a limited computational cost. In domains such scheduling, vehicle routing or in Energy Management Systems, this is done via greedy heuristics (list scheduling, shortest arc first), or by solving polynomial problems (e.g. Linear Programs in Energy Management Sys-

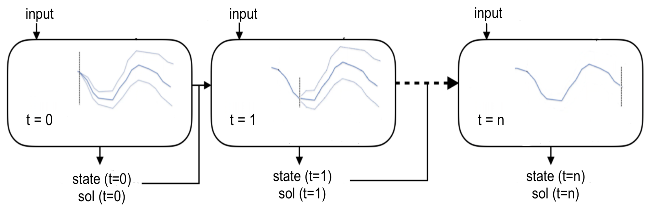

Figure 2: An Example of Online Anticipatory Algorithm

tems for online heuristics considered in [De Filippo *et al.*, 2018]) or even NP-hard problems (e.g. Mixed Integer Linear Programs in the baseline from [Awasthi and Sandholm, 2009]). This kind of approach is effective at controlling the computational cost, but it leads to *myopic* solution methods, lacking any ability to anticipate the future and estimate long-term effects of the current decisions.

Incorporating some mechanisms to estimate these effects leads to *online anticipatory algorithms*. Such mechanisms often consist in using sampling to approximate the probability distribution of future outcomes, for a fixed number of future stages (known as *look-ahead horizon*).

The simpler approaches rely directly on the Sample Average Approximation (e.g. in [Chisca *et al.*, 2018]), while more advanced methods are described in the next section. Anticipatory algorithms enable significant improvements in terms of quality, but this comes with a substantial computational cost that must be carefully managed.

Efficient Anticipatory Algorithms. In general, large sample sizes or look-ahead horizons result in better estimates, but also in more and/or bigger (possibly NP-hard) problems to be solved. Considerable research effort has therefore focused on improving the efficiency of these algorithms.

Sampling refers to obtaining realizations (scenarios) of the random variables used to model the uncertainty; by solving deterministic optimization problems over multiple scenarios and by computing averages, it is possible to imbue an online algorithm with some degree of anticipativity. This leads to the EXPECTATION algorithm [Chang *et al.*, 2000; Hentenryck and Bent, 2009; Bent and Van Hentenryck, 2004] which attempts to reduce the solution time by optimizing each scenario independently, for all possible decisions; the method then selects the decision which maximizes the expected profit. The CONSENSUS algorithm [Hentenryck and Bent, 2009] improves over this scheme by solving a deterministic problem per scenario. Every time a decision for the current stage is picked as optimal in one of those problems it receives a “vote”; once the process is over, the algorithm chooses the decision with the most votes. The technique employed by CONSENSUS has some adverse effects on the solution quality, which are addressed in the REGRET algorithm [Hentenryck and Bent, 2009] by extracting more information from each solved problem; this leads to a more reliable selection of the optimal decision for the current stage. These anticipatory algorithms are applicable only to problems with discrete, enumerable, decisions.

There is always a trade-off between the computation cost and the quality and robustness of the provided solution. This trade-off is the primary object of investigation in [Mercier and Van Hentenryck, 2007], and can be tuned by adjusting the number scenarios and the look-ahead horizon. In general, the method for generating the scenarios can be adapted to the given problem and the user goals, as described e.g. in [Kaut and Wallace, 2003]. Some approaches like [Lin *et al.*, 2013; De Filippo *et al.*, 2019] attempt to reduce the number of scenarios by increasing their relevance, and in particular by taking into account past observations while sampling.

In a few cases, improved scalability is achieved by replacing samples with a different form of approximation for the probability distribution. This can be done by either using a parametric distribution (e.g. [Terekhov *et al.*, 2014]) or ad-hoc representations.

Analysis. The key idea in anticipatory online algorithm is to take advantage of information about the possible future outcomes; often, this is achieved either by relying on a collection of historical samples, or by making assumption on the probability distribution of the uncertain elements. This kind of approach actually implies the existence of an offline phase, where some useful information is available.

However, if this is the case, it makes sense to exploit this phase as much as possible. For example, a substantial amount of preparation could be done (contingency plans, tuning of probabilistic models...) before the online solution process starts, since time constraints in the offline phase tend to be fairly relaxed.

Moreover, if the online problem is defined over the output of some kind of offline optimization (e.g. routing after facility assignments, energy supply management after production scheduling), it may be possible to improve the effectiveness of the online algorithm by adjusting instead the offline decisions. For example, more convenient energy prices could be obtained by adjusting a pre-defined production plan.

A particularly clear example is given by rescheduling approaches: in these case, once a schedule is generated, manufacturing/production operations begin, and then, at runtime, operators may deviate from the schedule. Ideally, the initial schedule is followed as closely as possible but small or larger deviations to the sequence occur when unexpected events can disrupt it (see examples in [Herrmann, 2006]). A clear need of strategic decisions to plan the production and of correctives runtime measures is evident but not totally exploited in terms of integration also in these problems.

4 Integrated Offline/Online Optimization

In addition to the situations described in Section 2 and Section 3, many real world use cases that are typically solved via either offline or online models are in fact *integrated offline/online problems*. For example, in *Energy Management Systems* the electrical load should be planned the day ahead, while power flow balance should be maintained hour by hour; in many *transportation systems*, (e.g. Vehicle Routing Problems), customer assignments are fixed, but routes can be adjusted online (e.g. based on traffic conditions); in *project scheduling* it is possible to plan project activities offline and then to use online algorithms to improve the solution as the elements of uncertainty reveal themselves; and in *reservation systems* a base reservation plan is usually devised offline, but it then needs to be integrated with an dynamic system to cope with unexpected disruptions.

Whenever distinct offline and online phases are present, a tighter integration can lead to substantial improvements in terms of both solution quality and computational costs. In this section, we will discuss works that focus on such a research direction.

Contingency Based Approaches. The two-stage stochastic formulation can be extended to multistage stochastic programs: the resulting policy can be encoded as a tree where nodes represent decisions (made at different stages) and arcs correspond to all the potential uncertain outcomes. The resulting *policy tree* can encode the responses to all possible unexpected events: hence, in this case the online problem is solved *entirely* in the offline phase and perfect integration is achieved.

Unfortunately, this kind of approach is often computationally intractable; [Shapiro, 2008] has shown that the SAA method cannot be extended efficiently to multistage problems: the number of required samples must grow exponentially with the number of stages.

Exact policy trees are computed in multi-stage stochastic optimization [Birge and Louveaux, 2011]. Approximate trees are instead considered by [Chiralaksanakul and Morton, 2004; Defourny *et al.*, 2012], to reduce the computational cost. By doing so, one no longer pre-computes all possible online policies; however, the quality of the offline estimation is improved without the adverse effects of discarding non-anticipativity constraints in future stages (i.e. collapsing stages 2 to n into a single stage).

A related idea is pursued in [De Filippo *et al.*, 2019], where the authors take advantage of information and computation time, available in an offline phase, to build a contingency table. This consists of historical instances of the online problem, pre-solved via a given, sampling-based, anticipatory algorithm. Such solutions are used to guide an efficient online heuristic that can guarantee the satisfaction of operational constraints. The approach nearly matches the solution quality of the anticipatory algorithm at a fraction of the online computational cost.

The idea of preparing plans to handle the possible disruptions is also at the base of conditional planning approaches, which try to identify all possible contingencies to create a conditional plan. The resulting huge search space, often limits these methods to the contingencies that contribute the most to a plan overall utility. The idea can be extended by taking into account the probabilities (when known) of the possible outcomes of each action [Onder and Pollack, 1999], often assuming that such outcomes will not be directly observable.

As a further example, in most practical scheduling environments (e.g. Partial Order Scheduling) offline solutions can be very limited and scheduling has to consider the online process of responding to unexpected and evolving circumstances. Unfortunately, the lack of guidance that might be provided by a schedule often leads to myopic, sub-optimal decision-making. To overcome this problem, in [Policella *et al.*, 2007] the authors use an approach relying on less knowledge approach to generate robust schedules offline: temporally flexible schedules that possess good robustness properties to retain flexibility where problem constraints allow. The flexibility is able to absorb unexpected events to promote both high reactivity and solution stability as execution proceeds in the online scheduling.

Markov Decision Processes and Dynamic Programming. The idea of computing policy trees is related to Dynamic Programming in Markov Decision Processes [Puterman, 2014]. In this case, the goal is determine a policy for an n-stage stochastic problem: the policy is defined via a value function that maps states (which encode all past history) to actions. Such value function is obtained by repeatedly simulating executions, and making updates with the aim to minimize an additive cost.

If 1) the probability distribution of the elements of uncertainty is known at the beginning of the process and remains unchanged; 2) both states and actions can be reasonably enumerated, then approximate dynamic programming [Powell, 2007] can be used to compute in an offline phase a policy, which can be efficiently executed online. Again, near-perfect integration is achieved. Unfortunately, the basic approach is viable only for relatively small problems and simple use cases, and hence practical methods need to trade approximation quality for improved scalability.

In [Ulmer *et al.*, 2019] the authors tackle the Traveling Salesman Problem with stochastic service requests, and correctly realize that extensive offline information is available in the form of historical customer transactions. They exploit this fact to use Approximate Dynamic Programming to yield dynamic routing policies: in detail, they combine offline value function approximation based on a highly compressed state representation (current location plus remaining time), with online rollout algorithms (to improve adaptability). The resulting approximate approach is scalable and worked well on the considered benchmarks.

The specific case where uncertainty is exogenous, i.e. it does not depend on the actions taken at each stage (renewable energy production, traffic conditions), is particularly frequent in practice. For such a situation, [Hentenryck and Bent, 2009] shows that it is possible to use a fixed set of samples during the whole optimization process, leading to more statistically reliable rankings of the possible actions. In [Mercier and Van Hentenryck, 2008], this strategy is coupled with an initialization of the value function via an anticipatory algorithm based on the Sample Average Approximation, to accelerate convergence.

A similar approach is considered in [Lorenzen *et al.*, 2017] in a sampling-based Stochastic Model Predictive Control algorithm. In this case, offline sampling enables a significant speed up of the online computation; the authors provide also rigorous bounds on the number of samples needed to guarantee chance constraint satisfaction, thus allowing them to tighten the constraints and guarantee robust feasibility when bounds on the uncertain variables are provided.

Deep Reinforcement Learning. In the last few years, Deep Reinforcement Learning (DRL) has emerged as a powerful tool for solving complex sequential decision-making problems [Arulkumaran *et al.*, 2017]. The approach is tied to approximate dynamic programming and shares some of its best properties: provided that extensive offline information is available, with a substantial computational cost DRL enables obtaining a policy that can then be efficiently deployed online. Intuitively, the main idea in DRL is to encode either the policy or the value function using a (Deep) Neural Network, which makes it applicable to large-scale and complex dynamic optimization problems.

DRL has been mostly developed and applied to problems that lack a strong constraint structure, but recently there has been some progress toward the application of DRL method to combinatorial problems, e.g. [Liu and Zeng, 2009] and [Bello *et al.*, 2016]. However, these approaches remain far from the state of the art and require access to very significant amounts of data, whereas (e.g.) anticipatory algorithms can work with as few as 100 samples. Overall, this seems a very promising line of research, but further advancements are needed before DRL can be profitably applied to practical problems of the class considered in this paper.

Online-Aware Offline Optimization. In many application domains, efficient suboptimal algorithms for online optimization are already available or easy to design (e.g. greedy heuristics or myopic declarative models). If some degree of offline information is available, it is possible to rely on a (typically expensive) parameter tuning phase to improve the behavior of the online solver, maintaining its original efficiency.

For example, [Dickerson *et al.*, 2012] consider a dynamic matching application, (Kidney Exchange Problem), and the need to take distributional information about possible future outcomes. Rather than relying on the Sample Average Approximation, they propose to learn “potentials” of elements (e.g. adjusted vertex weights) for a myopic problem. Then, at run time, they simply run a deterministic matching algorithm at each time period, with a modified objective that includes the potentials.

The approaches in [Wilder *et al.*, 2019] and [Donti *et al.*, 2017] work in a similar fashion: in this case the goal is improving a Machine Learning model, which feed information to an online problem. Both approaches train the ML model in an unconventional fashion, so that it outputs values leading to high quality online solutions, rather than accurate predictions. In this case, the training set represents the offline information, while the ML model can be assimilated to a particularly flexible parameter tuner: specifically, the trained model can identify approximately optimal parameter for a given vector of observations. As a main downside, the training process for these approaches can be very expensive.

In [De Filippo *et al.*, 2018], the authors rely on an idea from the Game Theory domain to inject knowledge of a (convex) online approach into an offline solver. This is achieved by formulating the KKT optimality conditions for the online solver and adding them as constraints in a (offline) Mathematical program. The resulting model can be used to perform parameter tuning, but also to adjust strategic (i.e. offline) decisions so that they play well with the limitations of the online heuristic.

5 Conclusions and Open Questions

Offline and online optimization under uncertainty have so far been addressed in many cases in relative isolation, despite in practice both offline and online elements are present in the same problem. In other domains (e.g. Dynamic Programming) the two phases are well integrated, but they are sometimes not clearly distinguished. We believe that the resulting lack of clarity has generally hindered the research progress.

In this work, we have provided a cross-disciplinary survey to highlight these issues and point them out as promising directions for future research. We have tried to identified common pitfalls of pure offline/online methods, and to provide a simple classification of integrated approaches, as a guide to promote cross-fertilization efforts.

Our analysis brings forth many open questions and possible hybridizations. For example, many online approaches can be seen as attempts to manage the delicate trade off between solution quality and computational effort, either via approximations or by shifting part of the computational load to the offline phase. Based on this, it may be appealing to devise automatic methods to handle these decisions.This could be done by relying on ideas from the algorithm selection domain: for example one could use ML techniques to build a model of the algorithm runtime as a function of problem-specific instance features [Hutter *et al.*, 2014], then rely on an optimization approach to either select or configure an online algorithm.

For problems that feature both strategic and operational decisions, a rather underinvestigated idea is that of controlling offline decisions so that they synergize with the online solver. To the best of our knowledge, this has been attempted only in [De Filippo *et al.*, 2018], which is however limited to convex online optimizers. Using ML-based approximations may lift this limitation and make the approach more general.

Similarly, ML predictors could conceivably replace sampling-based estimates (e.g. build the Sample Average Approximation); provided the ML model is simple enough, this could results in a computation gain, at the price of an offline training phase.

As one last example, most current Deep Reinforcement Learning approaches have trouble dealing with combinatorial structures: this issue could be addressed by injecting knowledge of the online solver in the policy itself, either by making the solver part of the environment, or by using Differentiable Programming to embed the online solver in the structure of the Deep Neural Network model. The resulting hybrid approach would benefit from powerful learning algorithms, and be well suited to deal with operational constraints.

Acknowledgments

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

References

[Arulkumaran *et al.*, 2017] Kai Arulkumaran, Marc Peter Deisenroth, Miles Brundage, and Anil Anthony Bharath. Deep reinforcement learning: A brief survey. *IEEE Signal Processing Magazine*, 34(6):26–38, 2017.

[Awasthi and Sandholm, 2009] Pranjal Awasthi and Tuomas Sandholm. Online stochastic optimization in the large: Application to kidney exchange. In *IJCAI*, volume 9, pages 405–411, 2009.

[Bello *et al.*, 2016] Irwan Bello, Hieu Pham, Quoc V Le, Mohammad Norouzi, and Samy Bengio. Neural combinatorial optimization with reinforcement learning. *arXiv preprint arXiv:1611.09940*, 2016.

[Bent and Van Hentenryck, 2004] Russell Bent and Pascal Van Hentenryck. Scenario-based planning for partially dynamic vehicle routing with stochastic customers. *Operations Research*, 52(6):977–987, 2004.

[Birge and Louveaux, 2011] John R Birge and François Louveaux. Evaluating and approximating expectations. In *Introduction to Stochastic Programming*, pages 341–387. Springer, 2011.

[CarøE and Schultz, 1999] Claus C CarøE and Rüdiger Schultz. Dual decomposition in stochastic integer programming. *Operations Research Letters*, 24(1-2):37–45, 1999.

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

[Charikar *et al.*, 2005] Moses Charikar, Chandra Chekuri, and Martin Pál. Sampling bounds for stochastic optimization. In *Approximation, Randomization and Combinatorial Optimization. Algorithms and Techniques*, pages 257–269. Springer, 2005.

[Chiralaksanakul and Morton, 2004] Anukal Chiralaksanakul and David P Morton. Assessing policy quality in multi-stage stochastic programming, 2004.

[Chisca *et al.*, 2018] Danuta Chisca, Michele Lombardi, Michela Milano, and Barry O'Sullivan. From offline to online kidney exchange optimization. In *2018 IEEE 30th International Conference on Tools with Artificial Intelligence (ICTAI)*, pages 587–591. IEEE, 2018.

[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.

[De Filippo *et al.*, 2019] Allegra De Filippo, Michele Lombardi, and Michela Milano. How to tame your anticipatory algorithm. In *IJCAI-19*, pages 1071–1077. IJCAI, 7 2019.

[Defourny *et al.*, 2012] Boris Defourny, Damien Ernst, and Louis Wehenkel. Multistage stochastic programming: A scenario tree based approach to planning under uncertainty. In *Decision theory models for applications in artificial intelligence: concepts and solutions*, pages 97–143. IGI Global, 2012.

[Dickerson *et al.*, 2012] John P Dickerson, Ariel D Procaccia, and Tuomas Sandholm. Dynamic matching via weighted myopia with application to kidney exchange. In *Twenty-Sixth AAAI Conference*, 2012.

[Donti *et al.*, 2017] Priya Donti, Brandon Amos, and J Zico Kolter. Task-based end-to-end model learning in stochastic optimization. In *Advances in Neural Information Processing Systems*, pages 5484–5494, 2017.

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

[Herrmann, 2006] Jeffrey W Herrmann. Rescheduling strategies, policies, and methods. In *Handbook of production scheduling*, pages 135–148. Springer, 2006.

[Hutter *et al.*, 2014] Frank Hutter, Lin Xu, Holger H Hoos, and Kevin Leyton-Brown. Algorithm runtime prediction: Methods & evaluation. *Artificial Intelligence*, 206:79–111, 2014.

[Kaut and Wallace, 2003] Michal Kaut and Stein W. Wallace. *Evaluation of scenario-generation methods for stochastic programming*. Humboldt-Universität zu Berlin, Mathematisch-Naturwissenschaftliche Fakultät II, Institut für Mathematik, 2003.

[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.

[Laporte and Louveaux, 1993] Gilbert Laporte and François V Louveaux. The integer l-shaped method for stochastic integer programs with complete recourse. *Operations research letters*, 13(3):133–142, 1993.

[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.

[Liu and Zeng, 2009] Fei Liu and Guangzhou Zeng. Study of genetic algorithm with reinforcement learning to solve the tsp. *Expert Systems with Applications*, 36(3):6995–7001, 2009.

[Lorenzen *et al.*, 2017] Matthias Lorenzen, Fabrizio Dabbene, Roberto Tempo, and Frank Allgower. Stochastic mpc with offline uncertainty sampling. *Automatica*, 81:176 – 183, 2017.

[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.

[Onder and Pollack, 1999] Nilufer Onder and Martha E Pollack. Conditional, probabilistic planning: A unifying algorithm and effective search control mechanisms. In *AAAI/IAAI*, pages 577–584, 1999.

[Policella *et al.*, 2007] Nicola Policella, Amedeo Cesta, Angelo Oddi, and Stephen F Smith. From precedence constraint posting to partial order schedules. *Ai Communications*, 20(3):163–180, 2007.

[Powell, 2007] Warren B Powell. *Approximate Dynamic Programming: Solving the curses of dimensionality*, volume 703. John Wiley & Sons, 2007.

[Puterman, 2014] Martin L Puterman. *Markov Decision Processes.: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, 2014.

[Schütz *et al.*, 2009] Peter Schütz, Asgeir Tomasgard, and Shabbir Ahmed. Supply chain design under uncertainty using sample average approximation and dual decomposition. *European Journal of Operational Research*, 199(2):409–419, 2009.

[Shapiro, 2008] Alexander Shapiro. Stochastic programming approach to optimization under uncertainty. *Mathematical Programming*, 112(1):183–220, Mar 2008.

[Shapiro, 2013] Alexander Shapiro. Sample average approximation. In *Encyclopedia of Operations Research and Management Science*, pages 1350–1355. Springer, 2013.

[Shmoys and Swamy, 2004] D. B. Shmoys and C. Swamy. Stochastic optimization is (almost) as easy as deterministic optimization. In *45th Annual IEEE Symposium on Foundations of Computer Science*, pages 228–237, Oct 2004.

[Terekhov *et al.*, 2014] Daria Terekhov, Tony T. Tran, Douglas G. Down, and J. Christopher Beck. Integrating queueing theory and scheduling for dynamic scheduling problems. *J. Artif. Intell. Res.*, 50:535–572, 2014.

[Ulmer *et al.*, 2019] Marlin W Ulmer, Justin C Goodson, Dirk C Mattfeld, and Marco Hennig. Offline–online approximate dynamic programming for dynamic vehicle routing with stochastic requests. *Transportation Science*, 53(1):185–202, 2019.

[Verweij *et al.*, 2003] Bram Verweij, Shabbir Ahmed, Anton J Kleywegt, George Nemhauser, and Alexander Shapiro. The sample average approximation method applied to stochastic routing problems: a computational study. *Computational Optimization and Applications*, 24(2-3):289–333, 2003.

[Wilder *et al.*, 2019] Bryan Wilder, Bistra Dilkina, and Milind Tambe. Melding the data-decisions pipeline: Decision-focused learning for combinatorial optimization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 1658–1665, 2019.

[Zhang and Li, 2011] Hui Zhang and Pu Li. Chance constrained programming for optimal power flow under uncertainty. *IEEE Transactions on Power Systems*, 26(4):2417–2424, 2011.

[Zhou *et al.*, 2013] Zhe Zhou, Jianyun Zhang, Pei Liu, Zheng Li, Michael C Georgiadis, and Efstratios N Pistikopoulos. A two-stage stochastic programming model for the optimal design of distributed energy systems. *Applied Energy*, 103:135–144, 2013.