Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence (IJCAI-18)

Methods for Off-line/On-line Optimization under Uncertainty

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

In this work we present two general techniques to deal with multi-stage optimization problems under uncertainty, featuring off-line and on-line decisions. The methods are applicable when: 1) the uncertainty is exogenous; 2) there exists a heuristic for the on-line phase that can be modeled as a parametric convex optimization problem. The first technique replaces the on-line heuristics with an anticipatory solver, obtained through a systematic procedure. The second technique consists in making the off-line solver aware of the on-line heuristic, and capable of controlling its parameters so as to steer its behavior. We instantiate our approaches on two case studies: an energy management system with uncertain renewable generation and load demand, and a vehicle routing problem with uncertain travel times. We show how both techniques achieve high solution quality w.r.t. an oracle operating under perfect information, by obtaining different trade-offs in terms of computation time.

1 Introduction

Dealing with uncertainty in optimization problems is challenging, but also increasingly recognized as necessary to obtain practically relevant results [Powell, 2016]. In many cases, this class of problems features both an off-line "strategic" phase that can be tackled with relative leisure, and an on-line "operational" phase where decisions need to be taken under stringent time constraints. Since optimization under uncertainty is tough, *anticipatory methods such as stochastic optimization* (see [Shapiro and Philpott, 2007; Birge and Louveaux, 1997; Kall and Wallace, 1994]) *have been historically employed for the off-line phase, and simple non-anticipativity heuristics for the on-line phase*.

However, on-line algorithms have the ability to exploit additional information as the uncertainty is slowly revealed. With the aim of tapping into this potential, a growing number of works has been *addressing on-line problems via techniques originally introduced for stochastic programming*, e.g. sampling and the Sample Average Approximation [Shapiro, 2013]. 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 enrich an on-line algorithm with some degree of anticipation. These developments lead to the EXPECTATION [Chang *et al.*, 2000], CONSENSUS [Bent and Van Hentenryck, 2004b] and REGRET [Bent and Van Hentenryck, 2004a] algorithms, and to more advanced methods such as AMSAA [Hentenryck and Bent, 2009; Mercier and Van Hentenryck, 2008]. All such methods are well discussed in [Hentenryck and Bent, 2009].

Intuitively, by *integrating off-line and on-line decision making*, it should be possible to obtain even better results, as it is often the case when a complex problem is partitioned. This paper aims at introducing two such methods that are applicable when: 1) the uncertainty is exogenous; 2) there exists a heuristic for the on-line phase with certain properties. In practice, each method alters either the off-line or the on-line component of the solution process, so that the two play better together. We believe our techniques represent a significant step toward integrated off-line/on-line optimization.

We instantiate our approaches on two case studies: 1) an energy system management problem, where load shifts are planned off-line and power flows must be controlled on-line; and 2) a Vehicle Routing Problem where customers are assigned off-line, but the routes can be chosen on-line. We show how the two methods strike radically different trade-offs in terms of off-line and on-line complexity, but they achieve solutions of similar (high) quality.

2 Formalization

We consider multi-stage optimization problems under uncertainty, where the first stage (indexed with 0) involves off-line decisions, and all subsequent n stages involve on-line decisions. We will start by describing a baseline solution approach, and then improve it by: 1) adding anticipation to the on-line solver through a systematic procedure; and 2) making the off-line solver aware of the on-line heuristic, and capable of controlling its parameters so as to steer its behavior. Generally, a simple heuristic may always choose the action with minimum "cost", i.e. the parameters may be either the costs themselves or constants used for score computation. The first method is closely related to existing on-line anticipatory algorithms; the second relies on the mixed nature of the problem.

Formally, let y represent the off-line decisions; let x^k represent the on-line decisions for stage k; let s^k (resp. ξ^k) rep-

resent the system state (resp. the uncertainty) revealed at the beginning (resp. the end) of stage k. All variables are assumed to be vectors, they can be either continuous or discrete, and have either finite or infinite domain.

Baseline on-line heuristic: We assume the availability of an *on-line heuristic that can be modeled as a parametric convex (and therefore efficient) optimization problem*:

$$\min f(y, x^k, s^k; \alpha^k) \qquad (\mathbf{PH})$$
$$\text{s.t. } e(y, x^k, s^k) = 0 \qquad (1)$$
$$g(y, x^k, s^k) \leq 0 \qquad (2)$$

where f is the cost function with parameter vector α^k, while e and g are the constraint functions (with vector output). We assume the optimization problem to be convex, which means that f and g must be convex and e to be linear. Typically, the convexity requirement will prevent x^k from being integer, but there are important exceptions (e.g. the one in Section 3.2).

We will assume that $\mathcal{F}(y, x^k, s^k)$ is the actual cost incurred at stage k for taking decisions x^k, *which may not be the same as the objective function f of the heuristic*. The transition from the state in stage k to the state in stage $k+1$ is defined by means of a *transition function* T, i.e.:

$$s^{k+1} = T(y, x^k, s^k, \xi^k)$$

where it can be seen that the effect of the uncertainty (i.e. the random variable) is encoded in the state.

Flattened Problem: Let Ω be a set of scenarios ω from the sample space of $\xi = (\xi^0, \dots \xi^{n-1})$. Given a single scenario ω, it is possible to collapse the instantiations of **PH** for each stage to obtain a *flattened (on-line) problem*:

$$\min \sum_{k=1}^{n} \mathcal{F}(y, x^k_\omega, s^k_\omega) \qquad (\mathbf{PF})$$
$$\text{s.t. } e(y, x^k_\omega, s^k_\omega) = 0 \qquad \forall k = 1..n \qquad (3)$$
$$g(y, x^k_\omega, s^k_\omega) \leq 0 \qquad \forall k = 1..n \qquad (4)$$
$$s^{k+1}_\omega = T(y, x^k_\omega, s^k_\omega, \xi^k_\omega) \qquad \forall k = 1..n-1 \qquad (5)$$

where $x^k_\omega/s^k_\omega/\xi^k_\omega$ are the on-line decisions/state/realizations for stage k in scenario ω. The only actual adjustment w.r.t. instantiating **PH** multiple times is that the true cost $\mathcal{F}$ is used rather than the heuristic cost function. Since **PF** assumes the availability of all ξ^k_ω values, it is effectively a clairvoyant approach, due to the lack of non-anticipativity constraints. In the on-line optimization literature the flattened problem is known as the off-line problem (see [Hentenryck and Bent, 2009]), but we adopt a different name to avoid ambiguity with the actual off-line phase.

Since the on-line problem can be solved with relative ease, the complexity depends heavily on the properties of the state transition function. If T is linear (e.g. in section 3.1), then the flattened problem will be convex and relatively easy to solve. Non-linear transition functions (e.g. section 3.2) are conversely much harder to handle.

Baseline off-line problem: As a baseline to deal with the off-line decisions we consider a two-stage stochastic optimization problem obtained by instantiating **PF** once per scenario:

$$\min f_o(y) + \frac{1}{|\Omega|} \sum_{\omega \in \Omega} \sum_{k=1}^{n} \mathcal{F}(y, x^k_\omega, s^k_\omega) \qquad (\mathbf{PO})$$
$$\text{s.t. Eq. } (3) - (5) \qquad \forall \omega \in \Omega$$
$$s^1_\omega = T_o(y, \xi^0_\omega) \qquad \forall \omega \in \Omega \qquad (6)$$
$$y \in \mathcal{Y} \qquad (7)$$

where the function $f_o(y)$ represents the cost that depends directly on the off-line decisions. The remainder of the cost function is given by the Sample Average Approximation of the expected cost of the subsequent stages. The function $T_o(y, \xi^0_\omega)$ determines the initial state for the on-line stages, based on the value of y and on the uncertainty revealed at the end of the off-line stage (i.e. ξ^0_ω). Finally, $\mathcal{Y}$ is the feasible space for the off-line decision variables y. We make no special assumption on $\mathcal{Y}$, $f_o(y)$, and $T_o(y, \xi^0_\omega)$, meaning that even when the flattened problem is convex the off-line problem may be NP-hard (or worse). Still, the fact that the problem is solved off-line makes its complexity less critical.

The biggest drawback of this approach is that *using the flattened problem to estimate the effect of the off-line decision on the future stages is equivalent to assuming the availability of an oracle*. In practice, however, an on-line approach can behave much worse than an oracle-powered solver.

Anticipatory On-line Phase: In this context, off-line on-line integration can be obtained by providing the on-line algorithm with something that resembles an oracle, i.e. by making it anticipatory. A simple approach to achieve this is the one employed for the baseline off-line problem, i.e. instantiating **PF** for the remaining stages. Formally, let h be the index of the current stage, then we consider:

$$\min \mathcal{F}(y, x^h, s^h) + \frac{1}{|\Omega|} \sum_{\omega \in \Omega} \sum_{k=h+1}^{n} \mathcal{F}(y, x^k_\omega, s^k_\omega) \qquad (\mathbf{PB})$$
$$\text{s.t. Eq. } (1), (2) \qquad \text{for stage } h$$
$$\text{Eq. } (3), (4) \qquad \text{for } k > h$$
$$\text{Eq. } (5) \qquad \text{for } k \geq h, \text{ with } s^h_\omega = s^h \text{ and } x^h_\omega = x^h$$

The off-line decisions are taken using **PO**. This first approach, nicknamed Boosted On-line OptimizatioN (BOON), improves the accuracy of the on-line component at the expense of its solution time.

PB has the same semantic as the EXPECTATION algorithm, which obtains the same results by enumerating the feasible decisions for the current stage, and evaluating their expected cost by solving the flattened problem on each scenario individually. Since each scenario is considered in isolation, EXPECTATION is arguably much more efficient than BOON whenever the current stage decisions can be enumerated reasonably fast. The same argument applies to the REGRET algorithm, an efficient approximation of EXPECTATION.

However, when the decision space is not enumerable (e.g. for continuous x^k variables, as in section 3.1), EXPECTATION, REGRET (and even CONSENSUS and AMSAA) cannot

be applied directly, while our method is still viable. Moreover, when each on-line stage requires to take *multiple decisions*, enumeration may be expensive and associating an expected cost to each decision in isolation leads to underestimations if the costs are not additive (e.g. as done in [Awasthi and Sandholm, 2009]).

On-line Aware Off-line Phase: An alternative integration approach consists in making the off-line solver aware of the on-line heuristic. Moreover, we can allow the off-line phase to adjust the heuristic parameters so as to steer its behavior.

We start by observing that, since the on-line heuristic problem **PH** is convex, any local minimum must be a global minimum. Local minima can be characterized in terms of the Karush-Kuhn-Tucker optimality conditions [Winston, 2004], which for **PH** in a given scenario ω are given by:

$$-\nabla_{x^k_\omega} f = \sum_{i=1}^{|e|} \lambda^k_{\omega,i} \nabla_{x^k_\omega} e_i + \sum_{i=1}^{|g|} \mu^k_{\omega,i} \nabla_{x^k_\omega} g_i \qquad (8)$$

$$\mu^k_{\omega,i} g_i = 0 \qquad \forall i = 1..|g| \qquad (9)$$

$$\mu^k_{\omega,i} \geq 0 \qquad \forall i = 1..|g| \qquad (10)$$

$$\text{Eq. } (1), (2)$$

where, for sake of readability, $f(y, x^k_\omega, s^k_\omega; \alpha_k)$ has been shortened to f, the i-th component (out of $|e|$) of $e(y, x^k_\omega, s^k_\omega)$ to e_i, and the i-th component (out of $|g|$) of $g(y, x^k_\omega, s^k_\omega)$ to g_i. The $\lambda^k_{\omega,i}$ and $\mu^k_{\omega,i}$ variables represent dual multipliers. Eq. (8) corresponds to the gradient cancellation condition, Eq. (9) to complementary slackness, Eq. (10) to dual feasibility ($\lambda^k_{\omega,i}$ is free), and Eq. (1), (2) to primal feasibility. Note that *here we use the heuristic cost*, parameterized in α^k.

Now, the KKT conditions can be injected as constraints in **PO**. This will force all x^k_ω variables in the off-line problem to take the values that would be actually assigned by the heuristic. This leads to the following problem:

$$\min f_o(y) + \frac{1}{|\Omega|} \sum_{\omega \in \Omega} \sum_{k=1}^{n} \mathcal{F}(y, x^k_\omega, s^k_\omega) \qquad (\mathbf{PM})$$

$$\text{s.t. Eq. } (3) - (5) \qquad \forall \omega \in \Omega$$

$$\text{Eq. } (6), (7)$$

$$\text{Eq. } (8) - (10) \qquad \forall \omega \in \Omega, \forall k = 1 \dots n$$

The decision variables are in this case y, x^k_ω, s^k_ω, $\lambda^k_{\omega,i}$, $\mu^k_{\omega,i}$, and crucially α^k. The on-line decisions are then taken using the original heuristics, but its behavior will be affected by the "parameter schedule" $\alpha^1, \dots \alpha^n$ produced by solving **PM**.

We nicknamed this second appraoch Master Off-line OptimizatioN (MOON): it achieves integration at the cost of off-line solution time, because of the additional variables in **PM** and the presence of non-linearities in Eq. (9).

3 Case Studies

In this section we present our case studies. The first one (an energy management system) was considered in [De Filippo *et al.*, 2017]: since it features continuous on-line decision variables, it is not amenable to existing approaches such as EXPECTATION or REGRET. The second use case (a Vehicle Routing Problem variant) is meant to provide a realistic, dramatically different, example of how the methods can be instantiated: it features discrete on-line decisions, and allows a quality comparison with classical algorithms because in such cases BOON leads to the same results as EXPECTATION (with no solution time restrictions). For simplicity, we will make use of monolithic models: our methods work with any solution approach, provided that the correct problems are solved.

3.1 Energy Management System

We consider a Virtual Power Plant management system (see [Morales *et al.*, 2013]) with partially shiftable loads, renewable energy generators, storage systems, and grid-connection. The load shifts must be planned off-line and the energy balance should be maintained on-line. The goal is to decide the minimum-cost energy flows at each on-line stage (see [Clavier *et al.*, 2015]), i.e.: 1) how much energy should be bought; 2) which generators should be used; 3) if the surplus energy (if any) should be either stored or sold to the market.

The uncertainty stems from uncontrollable deviations from the planned shifts and from the presence of Renewable Energy Sources (RES) (see [Palma-Behnke *et al.*, 2011; Bai *et al.*, 2015]). We assume that the RES production forecast is good enough that its error in each stage can be considered an independent random variable.

Baseline Heuristic and Transition Function: Based on the shifts produced by the off-line step, and adjusted to take into account the uncertainty, the on-line heuristic minimizes the operational cost and covers the energy demand by manipulating flows between nodes in $g \in G$. We assume the index 0 refers to the storage system and index 1 to the RES generators. The stages represent periods long enough to treat the corresponding flow decisions as independent. The heuristic can be formulated as an LP model:

$$\min \sum_{k=1}^{n} \sum_{g \in G} c^k_g x^k_g \qquad (\mathbf{P1.1})$$

$$\text{s.t. } \tilde{L}^k = \sum_{g \in G} x^k_g \qquad (11)$$

$$0 \leq \gamma_k + \eta x^k_0 \leq \Gamma \qquad (12)$$

$$\underline{x}_g \leq x^k_g \leq \overline{x}_g \qquad (13)$$

where n is the number of on-line stages, and x^k_g represents the flow from g to the VPP (if positive) or in the reverse direction (if negative). All flows must respect the physical bounds $\underline{x}_g$ and $\overline{x}_g$. The flow costs c^k_g correspond to the problem parameters α^k in **PH**. The state variables are the RES energy flow x^k_1, the load to be satisfied $\tilde{L}^k$, and the battery charge γ^k. The battery upper limit is Γ and η is the charging efficiency. The off-line decisions do not appear directly in the heuristic model, but they affect instead the state transition function:

$$\gamma^{k+1} = \gamma^k + \eta x^k_0 \qquad (14)$$

$$x^{k+1}_1 = \hat{R}^k + \xi^k_R \qquad (15)$$

$$\tilde{L}^{k+1} = \hat{L}^k + y^k + \xi^k_L \qquad (16)$$

where $\hat{R}_k$ and $\hat{L}^k$ are the estimated RES production and load, and ξ_R^k and ξ_L^k are the corresponding errors (random variables). We assume that the errors follow roughly a Normal distribution $N(0, \sigma^2)$, and that the variance σ^2 is such that the 95% confidence interval corresponds to $\pm 20\%$ of the estimated value [Gamou *et al.*, 2002]. The y^k variable represents the (off-line planned) shift from the estimated load.

Baseline Off-line Problem: The off-line problem is modeled via Mixed Integer Programming (MILP) and it is given by:

$$\min \frac{1}{|\Omega|} \sum_{\omega \in \Omega} \sum_{g \in G} \sum_{k=1}^{n} c_g^k x_{g,\omega}^k \qquad (\mathbf{P1.2})$$

$$\text{s.t. } \tilde{L}_\omega^k = \sum_{g \in G} x_{g,\omega}^k \qquad \forall \omega \in \Omega, \forall k = 1, \dots n \quad (17)$$

$$\underline{x}_g \leq x_{g,\omega}^k \leq \overline{x}_g \qquad \forall \omega \in \Omega, \forall k = 1, \dots n \quad (18)$$

$$0 \leq \gamma_\omega^k \leq \Gamma \qquad \forall k = 1, \dots n \quad (19)$$

$$\gamma_\omega^{k+1} = \gamma_\omega^k + \eta x_{0,\omega}^k \quad \forall \omega \in \Omega, \forall k = 1, \dots n-1 \quad (20)$$

$$x_{1,\omega}^{k+1} = \hat{R}_k + \xi_{R,\omega}^k \qquad \forall \omega \in \Omega, \forall k = 1, \dots n \quad (21)$$

$$\tilde{L}_\omega^{k+1} = \hat{L}_k + y_k + \xi_{L,\omega}^k \quad \forall \omega \in \Omega, \forall k = 1, \dots n \quad (22)$$

$$\sum_{k=t}^{t+m} y_k = 0 \qquad \forall t = 1, \dots n-m \quad (23)$$

$$\underline{y}^k \leq y_k \leq \overline{y}^k \qquad \forall k = 1, \dots n \quad (24)$$

where Eq.(17) $-$ (22) define the flattened problem, and Eq. (23) $-$ (24) the feasible space for the off-line variables y. Eq. (23) ensures that the shifts respect a local balance. The initial battery charge γ_ω^0 is identical for all scenarios.

Instantiating BOON for the VPP: A model for the BOON approach can be obtained by applying in an almost straightforward fashion the definitions from Section 2:

$$\min \sum_{g \in G} c_g^h x_g^h + \frac{1}{|\Omega|} \sum_{\omega \in \Omega} \sum_{k=h+1}^{n} \sum_{g \in G} c_g^k x_{g,\omega}^k \qquad (\mathbf{P1.3})$$

$$\text{s.t. Eq. } (11) - (13)$$

$$\text{Eq. } (17) - (19) \qquad \forall k > h$$

$$\text{Eq. } (20) - (22) \qquad \forall k \geq h, \text{ with } s_\omega^h = s^h \text{ and } x_\omega^h = x^h$$

Note that **P1.3**, although potentially large, is a Linear Program and can be solved in polynomial time.

Instantiating MOON for the VPP: We start by formulating the KKT conditions for the on-line heuristic in a single scenario, thus obtaining:

$$-c_g^k = \lambda_\omega^k + \mu_{g,\omega}^k - \nu_{g,\omega}^k \qquad \forall g \in G \quad (25)$$

$$\mu_{g,\omega}^k (x_{g,\omega}^k + \overline{x}_g) = 0 \qquad \forall g \in G \quad (26)$$

$$\nu_{i,\omega}^k (\underline{x}_g - x_{g,\omega}^t) = 0 \qquad \forall g \in G \quad (27)$$

$$\hat{\mu}_\omega^k (\eta x_{0,\omega}^k + \gamma^k - \Gamma) = 0 \quad (28)$$

$$\hat{\nu}_\omega^k (\eta x_{0,\omega}^k + \gamma^k) = 0 \quad (29)$$

$$\mu_{g,\omega}^k, \nu_{g,\omega}^k \geq 0 \qquad \forall g \in G \quad (30)$$

$$\hat{\mu}_\omega^k, \hat{\nu}_\omega^k \geq 0 \quad (31)$$

where $\mu_{g,\omega}^k$ and $\nu_{g,\omega}^k$ are the multipliers associated to the physical flow bounds, while $\hat{\mu}_\omega^k$ and $\hat{\nu}_\omega^k$ are associated to the battery capacity bounds. The multiplier λ_ω^k is associated to the balancing constraint, i.e. Eq. (11), and can be eliminated with a few algebraic transformations. Injecting the conditions in the off-line model yields:

$$\min \frac{1}{|\Omega|} \sum_{\omega \in \Omega} \sum_{g \in G} \sum_{k=1}^{n} c_g^k x_{g,\omega}^k \qquad (\mathbf{P1.4})$$

$$\text{s.t. Eq. } (17) - (24)$$

$$\text{Eq. } (25) - (31) \qquad \forall \omega \in \Omega, \forall k = 1, \dots n$$

where the decision variables are y^k, $x_{g,\omega}^k$, $\mu_{g,\omega}^k$, $\nu_{g,\omega}^k$, $\hat{\mu}_\omega^k$, $\hat{\nu}_\omega^k$. To those, we add the cost c_0^k associated to the flow between the VPP and the storage system (the only parameter that we allow the solver to adjust). Normally, there are neither economic penalties nor incentives for such flow, while there is a profit associated to flows from the VPP to the grid. As a side effect, the naive **P1.1** heuristic will always choose to sell the surplus energy. MOON allows the off-line solver to associate a "virtual profit" to storing energy, which enables addressing the original limitation at no on-line computational cost.

3.2 Vehicle Routing Problem

We consider a variant of the Capacitated VRP with uncertain travel times (see [Toth and Vigo, 2002; Bertsimas and Simchi-Levi, 1996; Lee *et al.*, 2012; Taş *et al.*, 2013]). The problem consists in establishing the paths of a set of vehicles to serve a set of customers. All vehicles have a finite capacity, and customers have a known demand and can be visited by a single vehicle. There are n fully connected customers/nodes, with node 0 being the (single) depot.

Customer assignments must be done off-line, while the vehicle routes are chosen on-line. We assume that, whenever a node is reached, its binary "state" becomes known, and with that the (uniform) distributions followed by the travel times of all its outgoing arcs. Formally, this results in bimodally distributed, statistically dependent, travel times. The objective is to minimize the total travel time.

Baseline Heuristic and Transition Function: The on-line heuristic consists in simply picking the outgoing arc with the shortest travel time. This can be modeled also as a simple Integer Program. Let h be the current node, then we have:

$$\min \sum_{j \in V_h} c_{hj} x_{hj} \qquad (\mathbf{P2.1})$$

$$\text{s.t. } \sum_{j \in V_h} x_{h,j} = 1 \quad (32)$$

$$x_{h,j} \in \{0, 1\} \qquad \forall j \in V \quad (33)$$

where $x_{hj} = 1$ iff we choose to move from h to j, V_h is the set of nodes that still needs to be visited (and it always include the depot), and the travel times c_{hj} are the heuristic parameters. **P2.1** does not apparently satisfy our assumptions, due to the integer variables. However, *its LP relaxation has always an integer solution, banning degenerate cases* (i.e. arcs with the

same cost). We can therefore relax the integrality requirement without loss of generality. The transition function is given by:

$$V_{h^*} = V_h \setminus \{h^*\} \tag{34}$$

$$c_{h^*,j} = \xi_{h^*,j} \tag{35}$$

where h^* is the index of the next node selected by the heuristic and $\xi_{h^*,j}$ is the travel time from h^* to j (a random variable). Note also that in this case *the index of the on-line stage is implicitly given by* h. We take advantage of this and reduce the notation clutter by moving the ω index to apex position.

Baseline Off-line Problem: We tackle the off-line problem via Mixed Integer Linear Programming, which forbids us to directly embed the non-linear Eq. (34) in the model. In practice, however, the equation states that 1) each vehicle should serve only its assigned customers, and 2) the visit should form a single loop. Both are well known VRP constraints and can be linearized. In particular, we use the model:

$$\min \frac{1}{|\Omega|} \sum_{\omega\in\Omega} \sum_{k\in K} \sum_{i,j\in V} \xi^{\omega}_{i,j} x^{\omega}_{k,i,j} \qquad (\mathbf{P2.2})$$

$$\text{s.t.} \sum_{j\in V} x^{\omega}_{k,i,j} = y_{k,i} \qquad \forall k \in K, \forall i \in V \tag{36}$$

$$\sum_{i\in V} x^{\omega}_{k,i,j} = y_{k,j} \qquad \forall k \in K, \forall j \in V \tag{37}$$

$$y_{k,0} = 1 \qquad \forall k \in K \tag{38}$$

$$t^{\omega}_{k,j} \geq t^{\omega}_{k,i} - M + (M+1)x^{\omega}_{k,i,j} \quad \forall k \in K, \forall i,j \in V, V^+ \tag{39}$$

$$t^{\omega}_{k,0} = 0 \qquad \forall k \in K \tag{40}$$

$$\sum_{i\in V} q_i y_{k,i} \leq C_k \qquad \forall k \in K \tag{41}$$

$$\sum_{k\in K} y_{k,i} = 1 \qquad \forall i \in V^+ \tag{42}$$

where all constraints where an ω apex appears should be posted $\forall\omega \in \Omega$. All x and y variables are binary, and $y_{ki} = 1$ iff customer i should be visited by vehicle k. We have $M = |V|$, and $V^+ = V \setminus \{0\}$. Eq.(36) $-$ (40) define the flattened problem, and Eq. (41) $-$ (42) define the feasible space of the off-line decision variables. For sake of simplicity, we eliminate subloops by keeping track of the visiting order t^{ω}_{ki} of each node for each vehicle: this is a simple, but not particularly effective method, because it relies on big-Ms and reduces the quality of the LP bound [Miller *et al.*, 1960].

Instantiating BOON for the VRP: The BOON method can be instantiated for each vehicle k separately, by first restricting the focus to the set of nodes V_h, and then by applying the definition from Section 2 and linearizing Eq. (34) in the baseline off-line problem, we get:

$$\min \sum_{j\in V_h} c_{h,j} x_{k,i,j} + \frac{1}{|\Omega|} \sum_{\omega\in\Omega} \sum_{i,j\in V_h} \xi^{\omega}_{i,j} x^{\omega}_{k,i,j} \qquad (\mathbf{P2.3})$$

$$\text{s.t. Eq. (32)}$$

$$\text{Eq. (36) restricted to } V_h \setminus \{0\} \tag{43}$$

$$\text{Eq. (37)} - \text{(39) restricted to } V_h$$

$$t^{\omega}_{k,h} = 0 \tag{44}$$

where Eq. (44) means that the vehicle path should start from the current node h (and end as usual in the depot).

Instantiating MOON on the VRP: As usual, the first step in the MOON is formulating the KKT conditions for **P2.1**. In this case after some algebraic transformations, for a given vehicle k, node h, and scenario ω we obtain:

$$(c_{hj} + \lambda^{\omega}_{k,h}) x^{\omega}_{k,h,j} = 0 \qquad \forall j \in V_h \tag{45}$$

$$(c_{hj} + \lambda^{\omega}_{k,h}) \geq 0 \qquad \forall j \in V_h \tag{46}$$

where $\lambda^{\omega}_{k,h}$ is the multiplier for Eq. (32), and all other multipliers have been eliminated. The main difficulty is again dealing with the set V_h, which is part of the state and should be constructed dynamically in the off-line problem. Here, we handle V_h by introducing fresh variables r^{ω}_{kji} such that $r^{\omega}_{kji} = 1$ iff node i has been visited when node j is reached. The semantic is enforced via additional non-linear constraints in the off-line model. The latter is given by:

$$\min \frac{1}{|\Omega|} \sum_{\omega\in\Omega} \sum_{k\in K} \sum_{i,j\in V} \xi^{\omega}_{i,j} x^{\omega}_{k,i,j} \qquad (\mathbf{P2.4})$$

$$\text{s.t. Eq. (36)} - \text{(42)}$$

$$(c_{ij} + \lambda^{\omega}_{k,i}) x^{\omega}_{k,i,j} (1 - r_{kji}) = 0 \qquad \forall k \in K, \forall i,j \in V$$

$$(c_{ij} + \lambda^{\omega}_{k,i})(1 - r_{kji}) \geq 0 \qquad \forall k \in K, \forall i,j \in V$$

$$r^{\omega}_{k,i,i} = y_{k,i} \qquad \forall i \in V$$

$$r^{\omega}_{k,j,i} = r^{\omega}_{k,h,i} x^{\omega}_{k,h,j} \qquad \forall i \in V, \forall h \in V, \forall j \in V$$

$$\underline{c}_{ij} \leq c_{ij} \leq \overline{c}_{ij} \qquad \forall i,j \in V$$

The decision variables are y_{ki}, x^{ω}_{kij}, λ^{ω}_{ki}, r^{ω}_{kji}, plus the "virtual travel times" c_{ij}, i.e. the parameters for the on-line heuristic. The constraints on the r^{ω}_{kji} variables enforce the transitive property on the set of visited nodes. Bounding the virtual travel times is necessary to prevent the solver from building degenerate parameterizations for **P2.1** on purpose, which would trivially satisfy all KKT constraints and make the approach boil down to the baseline off-line solver.

4 Experiments

We performed an experimentation to compare the solution quality and run times of our methods. As references for comparison we use the baseline approaches, plus an optimal solver operating under perfect information. Additionally, on the VRP, BOON obtains the same solution quality as EXPECTATION, which gives a third term of comparison.

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 model, and for the travel times in the VRP model. We consider a sample of 100 realizations for each of the six different instances of each problem. We then run each approach on each relization and measure the cost and run time. The scenarios in our models, conversely, are not sampled, but programmatically chosen: for the VPP we consider four "extreme" scenarios where (resp.) the load and the RES generation are at

Instance	Oracle (k€)	Baseline (k€)	BOON (k€)	MOON (k€)
I1	**331.362**	404.622	**342.061**	344.604
I2	**247.213**	311.145	265.326	**263.808**
I3	**393.818**	462.577	**404.322**	408.721
I4	**798.388**	923.243	819.249	**811.119**
I5	**565.607**	684.197	580.174	**573.934**
I6	**856.955**	984.904	874.585	**868.764**

Table 1: Cost values for the different VPP models

low/high values. For the VRP, each scenario corresponds to the mean travel times in one mode of the distribution. The VPP has 24 on-line stages, while in the VRP the number depends on how many customers are assigned to each vehicle.

We solve our LPs and MILPs using Gurobi, while for the non-linear problems we use BARON via the GAMS modeling system on the Neos server for optimization. The time limit is 100 seconds for the VPP, and 500 seconds for the VRP. For the VPP, we use data from two public datasets to define problem instances for a residential [Espinosa and Ochoa, 2015] and industrial plant[1]. For the VRP we use modified versions of classical instances[2], by including problems from 10 to 30 customers with one depot and different numbers of vehicles.

Discussion: In Table 1 and Table 2 we show the average costs and run time over the 100 input realizations for each approach for the VPP use case. On-line times refer to the sum of the stages. The baseline model (being an LP) appears to be rather efficient in terms of computation time, but yields solutions of limited quality. The BOON method *comes much closer to the oracle solver, at the cost of a higher, but still reasonable, on-line run time*. The MOON method incurs substantially larger off-line solution times, but it *manages to either beat or match the* BOON *solution quality by making use of the original, straightforward, on-line heuristic*.

Table 3 and 4 report the same results for the VRP. Here the on-line times are summed over all the vehicles. The original on-line heuristic is very efficient, but coupled with the baseline off-line model it does not come close to the oracle quality. The off-line model (a Mixed Integer Linear Program) takes also considerably more time to be solved. BOON, which in this case yields the same results as EXPECTATION with no time limit, yields substantially better solutions, but, being also MILP-based, it takes non-negligible time during the on-line phase. The MOON results follow the same trend as the VPP: the solution quality either matches or beats that of BOON, at the cost of a higher off-line computation time, though the gap wrt the baseline is now much smaller.

5 Conclusion

This paper makes a first step toward generic integrated off-line/on-line optimization. We propose two alternative approaches, based on the idea of making the off-line and on-line solvers operate synergistically. In the BOON method this is done by providing the on-line solver with the approximation of an oracle. In the MOON method, we instead make the off-line solver aware of the limitations of the on-line one, and capable of controlling its behavior by adjusting parameters. In general, our approaches work best for problems with numeric on-line decisions, but important classes of on-line heuristics are also covered (e.g. $\arg\min$ of parametric scores).

Both techniques yield substantially improved solutions: BOON matches the quality level of EXPECTATION, but it is applicable under more general assumptions. Unfortunately, the method is also less efficient. MOON often manages to beat BOON (and therefore EXPECTATION) in terms of solution quality. While this comes at the price of a substantially increased off-line computation time, *the method achieves these results by using naive and very efficient on-line heuristics*.

We believe there is room for improving the efficiency of our methods (similarly to how EXPECTATION was improved in REGRET), and achieving this goal is part of our current research directions. We also plan to apply our approaches to different problems, such as resource allocation and scheduling with Simple Temporal Networks under Uncertainty.

Instance	Off-line part (sec) Baseline	Off-line part (sec) MOON	On-line part (sec) Heuristic	On-line part (sec) BOON
I1	0.184	27.884	0.778	5.011
I2	0.190	31.992	0.772	5.017
I3	0.185	30.772	0.775	5.009
I4	0.346	58.913	0.839	5.430
I5	0.341	59.184	0.832	5.423
I6	0.348	57.777	0.835	5.420

Table 2: Computation time for the different VPP model parts

Instance	Oracle (t)	Baseline (t)	BOON (t)	MOON (t)
I1	**146.109**	165.838	151.238	**148.845**
I2	**278.376**	347.282	298.673	**290.433**
I3	**372.823**	561.664	**477.162**	507.803
I4	**321.571**	381.453	342.949	**340.857**
I5	**503.659**	670.867	559.227	**543.923**
I6	**448.537**	971.876	**470.995**	504.827

Table 3: Travel time (cost) values for the different VRP models

Instance	Off-line part (sec) Baseline	Off-line part (sec) MOON	On-line part (sec) Heuristic	On-line part (sec) BOON
I1	1.699	6.255	0.255	7.134
I2	2.477	17.445	0.169	15.222
I3	2.532	25.938	0.554	18.024
I4	186.798	338.998	3.444	255.932
I5	243.330	357.543	5.248	313.656
I6	361.537	490.856	5.342	416.645

Table 4: Computation time for the different VRP model parts

[1] https://data.lab.fiware.org/dataset/

[2] http://myweb.uiowa.edu/bthoa/TSPTWBenchmarkDataSets.htm

References

[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.

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

[Bertsimas and Simchi-Levi, 1996] Dimitris Bertsimas and David Simchi-Levi. A new generation of vehicle routing research: robust algorithms, addressing uncertainty. *Operations Research*, 44(2):286–304, 1996.

[Birge and Louveaux, 1997] John Birge and Francois Louveaux. Introduction to stochastic programming. series in operations research and financial engineering, 1997.

[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.

[Clavier *et al.*, 2015] Juan Clavier, Franois Bouffard, Dmitry Rimorov, and Gza Jos. Generation dispatch techniques for remote communities with flexible demand. *IEEE Transactions on Sustainable Energy*, 6(3):720–728, 2015.

[De Filippo *et al.*, 2017] Allegra De Filippo, Michele Lombardi, Michela Milano, and Alberto Borghetti. Robust optimization for virtual power plants. In *Conference of the Italian Association for Artificial Intelligence*, pages 17–30. Springer, 2017.

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

[Gamou *et al.*, 2002] Satoshi Gamou, Ryohei Yokoyama, and Koichi Ito. Optimal unit sizing of cogeneration systems in consideration of uncertain energy demands as continuous random variables. *Energy Conversion and Management*, 43(9):1349 – 1361, 2002.

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

[Kall and Wallace, 1994] Peter Kall and Stein Wallace. *Stochastic programming*. Springer, 1994.

[Lee *et al.*, 2012] Chungmok Lee, Kyungsik Lee, and Sungsoo Park. Robust vehicle routing problem with deadlines and travel time/demand uncertainty. *Journal of the Operational Research Society*, 63(9):1294–1306, 2012.

[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.

[Miller *et al.*, 1960] Clair Miller, Albert Tucker, and Richard Zemlin. Integer programming formulation of traveling salesman problems. *J. ACM*, 7(4):326–329, October 1960.

[Morales *et al.*, 2013] Juans Morales, Antonio Conejo, Henrik Madsen, Pierre Pinson, and Marco Zugno. *Integrating renewables in electricity markets: operational problems*, volume 205. Springer Science & Business Media, 2013.

[Palma-Behnke *et al.*, 2011] Rodrigo Palma-Behnke, Carlos Benavides, E. Aranda, Jacqueline Llanos, and Doris Saez. Energy management system for a renewable based microgrid with a demand side management mechanism. In *2011 IEEE Symposium on Computational Intelligence Applications In Smart Grid (CIASG)*, pages 1–8, April 2011.

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

[Shapiro and Philpott, 2007] Alexander Shapiro and Andy Philpott. A tutorial on stochastic programming. *Manuscript. Available at www2. isye. gatech. edu/ashapiro/publications. html*, 17, 2007.

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

[Taş *et al.*, 2013] Duygu Taş, Nico Dellaert, Tom Van Woensel, and Ton De Kok. Vehicle routing problem with stochastic travel times including soft time windows and service costs. *Computers & Operations Research*, 40(1):214–224, 2013.

[Toth and Vigo, 2002] Paolo Toth and Daniele Vigo. *The vehicle routing problem*. SIAM, 2002.

[Winston, 2004] Wayne Leslie Winston. *Operations research: applications and algorithms*, volume 3. 2004.