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 ease, 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. \( \xi^k \)) 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:

$$\min f(y, x^k, s^k; \alpha^k) \quad \text{(PH)}$$

s.t. $e(y, x^k, s^k) = 0$ \hspace{1cm} (1)

$g(y, x^k, s^k) \leq 0$ \hspace{1cm} (2)

where $f$ is the cost function with parameter vector $\alpha^k$, 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 $F(y, x^k, s^k)$ 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 $\Omega$ be a set of scenarios $\omega$ from the sample space of $\xi = (\xi^0, \ldots, \xi^{n-1})$. Given a single scenario $\omega$, it is possible to collapse the instantiations of PH for each stage to obtain a flattened (on-line) problem:

$$\min \sum_{k=1}^{n} F(y, x^k_{\omega}, s^k_{\omega}) \quad \text{(PF)}$$

s.t. $e(y, x^k_{\omega}, s^k_{\omega}) = 0$ \hspace{1cm} (3)

$g(y, x^k_{\omega}, s^k_{\omega}) \leq 0$ \hspace{1cm} (4)

$s^{k+1}_{\omega} = T(y, x^k_{\omega}, s^k_{\omega}, \xi^k_{\omega})$ \hspace{1cm} (5)

where $x^k_{\omega}, s^k_{\omega}$ are the on-line decisions/state/realizations for stage $k$ in scenario $\omega$. The only actual adjustment w.r.t. instantiating PH multiple times is that the true cost $F$ is used rather than the heuristic cost function. Since PF assumes the availability of all $\xi^k_{\omega}$ 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:

$$\min f_o(y) + \frac{1}{|\Omega|} \sum_{\omega \in \Omega} \sum_{k=1}^{n} F(y, x^k_{\omega}, s^k_{\omega}) \quad \text{(PO)}$$

s.t. Eq. (3) – (5) \hspace{1cm} (6)

$$s^1_{\omega} = T_o(y, \xi^0_{\omega}) \quad \forall \omega \in \Omega \quad (7)$$

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. $\xi^0_{\omega}$). 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:

$$\min \sum_{k=1}^{n} F(y, x^h, s^h) + \frac{1}{|\Omega|} \sum_{\omega \in \Omega} \sum_{k=h+1}^{n} F(y, x^k_{\omega}, s^k_{\omega}) \quad \text{(PB)}$$

s.t. Eq. (1), (2) \hspace{1cm} (8)

Eq. (3), (4) \hspace{1cm} (9)

Eq. (5) \hspace{1cm} for stage $h$

for $k > h$

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:

\[
- \nabla_{x^k} f = \sum_{i=1}^{\lvert e \rvert} \lambda_{\omega,i}^k \nabla_{x^k} e_i + \sum_{i=1}^{\lvert g \rvert} \mu_{\omega,i}^k \nabla_{x^k} g_i \tag{8}
\]

\[
\mu_{\omega,i}^k g_i = 0 \quad \forall i = 1 .. \lvert g \rvert \tag{9}
\]

\[
\mu_{\omega,i}^k \geq 0 \quad \forall i = 1 .. \lvert g \rvert \tag{10}
\]

where, for sake of readability, \( f(y, x^k, s^k, \alpha_k) \) has been shortened to \( f \), the \( i \)-th component (out of \( |e| \)) of \( e(y, x^k, s^k, \omega) \) to \( e_i \), and the \( i \)-th component (out of \( |g| \)) of \( g(y, x^k, s^k, \omega) \) to \( g_i \).

The \( \lambda_{\omega,i}^k \) and \( \mu_{\omega,i}^k \) variables represent dual multipliers. Eq. (8) corresponds to the gradient cancellation condition, Eq. (9) to complementary slackness, Eq. (10) to dual feasibility (\( \lambda_{\omega,i}^k \) is free), and Eq. (1) to (2) to primal feasibility. Note that here we use the heuristic cost, parameterized in \( \alpha^k \).

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

\[
\min f_\omega(y) + \frac{1}{|\Omega|} \sum_{\omega \in \Omega} \sum_{k=1}^n F(y, x^k_\omega, s^k_\omega) \tag{PM}
\]

s.t. Eq. (3) – (5) \( \forall \omega \in \Omega \)

Eq. (6), (7)

Eq. (8) – (10) \( \forall \omega \in \Omega, \forall k = 1 .. n \)

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

We nicknamed this second approach Master Off-line OptimizationN (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 \( G \). 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:

\[
\min \sum_{k=1}^n \sum_{g \in G} c_{g}^k x_{g}^k \tag{P1.1}
\]

s.t. \( \hat{L}^k = \sum_{g \in G} x_{g}^k \) \( \forall k \in G \)

\[
0 \leq \gamma_k + \eta x_{g}^k \leq \Gamma \tag{12}
\]

\[
x_{g}^k \leq \bar{x}_{g}^k \tag{13}
\]

where \( n \) is the number of on-line stages, and \( x_{g}^k \) represents the flow from \( g \) to the VPP (if positive) or in the reverse direction (if negative). All flows must respect the physical bounds \( \bar{x}_{g} \) and \( \bar{x}_{g} \). The flow costs \( c_{g}^k \) correspond to the problem parameters \( c_{g}^k \) in PH. The state variables are the RES energy flow \( x_{g}^1 \), the load to be satisfied \( \hat{L}^k \), and the battery charge \( \gamma_k \). The battery upper limit is \( \Gamma \) and \( \eta \) is the charging efficiency. The off-line decisions do not appear directly in the heuristic model, but they affect instead the state transition function:

\[
\gamma^{k+1} = \gamma^{k} + \eta x_{g}^k \tag{14}
\]

\[
x_{g}^{k+1} = \hat{R}^k + x_{g}^k \tag{15}
\]

\[
\hat{L}^{k+1} = \hat{L}^k + y^k + \xi^k \tag{16}
\]
where $\hat{R}_k$ and $\hat{L}_k$ are the estimated RES production and load, and $\xi_k$ and $\xi_k$ are the corresponding errors (random variables). We assume that the errors follow roughly a Normal distribution $N(0, \sigma^2)$, and that the variance $\sigma^2$ 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:

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

s.t. $\hat{L}_k = \sum_{g \in G} x_{g, \omega}^{k}$ \hspace{1cm} $\forall \omega \in \Omega, \forall k = 1, \ldots n$  \hspace{1cm} (17)

$$\underline{x}_{g} \leq x_{g, \omega}^{k} \leq \overline{x}_{g}$$ \hspace{1cm} $\forall \omega \in \Omega, \forall k = 1, \ldots n$  \hspace{1cm} (18)

$$0 \leq \gamma_{\omega}^{k} \leq \Gamma$$ \hspace{1cm} $\forall k = 1, \ldots n$ \hspace{1cm} (19)

$$\gamma_{\omega}^{k+1} = \gamma_{\omega}^{k} + \eta_{g}^{k} \omega, \quad \forall \omega \in \Omega, \forall k = 1, \ldots n - 1$$ \hspace{1cm} (20)

$$x_{g}^{k+1} = \hat{R}_k + x_{g, \omega}^{k}$$ \hspace{1cm} $\forall \omega \in \Omega, \forall k = 1, \ldots n$ \hspace{1cm} (21)

$$\hat{L}_k = \hat{L}_k + y_k + \xi_k^{k}$$ \hspace{1cm} $\forall \omega \in \Omega, \forall k = 1, \ldots n$ \hspace{1cm} (22)

$$\sum_{k = t}^{n} y_k = 0$$ \hspace{1cm} $\forall t = 1, \ldots n - m$ \hspace{1cm} (23)

$$y_k \leq y_k \leq y^k$$ \hspace{1cm} $\forall k = 1, \ldots n$ \hspace{1cm} (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 $\gamma_{0}^{k}$ 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:

$$\min \sum_{g \in G} c_{g} x_{g}^{h} + \frac{1}{|\Omega|} \sum_{\omega \in \Omega} \sum_{k = h+1}^{n} \sum_{g \in G} c_{g} x_{g, \omega}^{k}$$  \hspace{1cm} (P1.3)

s.t. Eq. (11) – (13)

Eq. (17) – (19) \hspace{1cm} $\forall k > h$

Eq. (20) – (22) \hspace{1cm} $\forall k \geq h$, with $s_{\omega}^{h} = x_{\omega}^{h}$  \hspace{1cm} and $x_{\omega}^{h} = x_{\omega}^{h}$

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:

$$-c_{g}^{h} = \lambda_{g}^{h} + \mu_{g, \omega}^{h} - \nu_{g, \omega}^{k}$$ \hspace{1cm} $\forall g \in G$ \hspace{1cm} (25)

$$\mu_{g, \omega}^{h}(x_{g, \omega}^{h} + \Pi_g) = 0$$ \hspace{1cm} $\forall g \in G$ \hspace{1cm} (27)

$$\nu_{g, \omega}^{h}(x_{g, \omega}^{h} - x_{g, \omega}^{k}) = 0$$ \hspace{1cm} $\forall g \in G$ \hspace{1cm} (26)

$$\mu_{g, \omega}^{h}(x_{g, \omega}^{h} + \Pi_g) = 0$$ \hspace{1cm} $\forall g \in G$ \hspace{1cm} (28)

$$\nu_{g, \omega}^{h}(x_{g, \omega}^{h} - x_{g, \omega}^{k}) = 0$$ \hspace{1cm} $\forall g \in G$ \hspace{1cm} (29)

$$\lambda_{g}^{h}, \mu_{g, \omega}^{h}, \nu_{g, \omega}^{k} \geq 0$$ \hspace{1cm} $\forall g \in G$ \hspace{1cm} (30)

$$\mu_{g, \omega}^{h}, \nu_{g, \omega}^{k} \geq 0$$ \hspace{1cm} $\forall g \in G$ \hspace{1cm} (31)

where $\mu_{g, \omega}^{h}$ and $\nu_{g, \omega}^{k}$ are the multipliers associated to the physical flow bounds, while $\mu_{g, \omega}^{h}$ and $\nu_{g, \omega}^{k}$ are associated to the battery capacity bounds. The multiplier $\lambda_{g}^{h}$ 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:

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

s.t. Eq. (17) – (24) \hspace{1cm} Eq. (25) – (31) \hspace{1cm} $\forall \omega \in \Omega, \forall k = 1, \ldots n$

where the decision variables are $y_k^{h}, x_{g, \omega}^{k}, \mu_{g, \omega}^{h}, \nu_{g, \omega}^{k}, \mu_{g, \omega}^{h}, \nu_{g, \omega}^{k}$.

To those, we add the cost $c_{g}^{h}$ 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:

$$\min \sum_{j \in V_h} c_{h, j} x_{h, j}$$  \hspace{1cm} (P2.1)

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

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

where $x_{h, j} = 1$ 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 $c_{h, j}$ 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:

\[ V_{h^*} = V_h \setminus \{ h^* \} \]  
\[ c_{h^*, j} = c_{h, j} \]

where \( h^* \) is the index of the next node selected by the heuristic and \( c_{h, j} \) is the travel time from \( h^* \) to \( j \) (a random variable). Note 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 \( \omega \) 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:

\[
\begin{align*}
\min & \frac{1}{|Q|} \sum_{\omega \in Q} \sum_{k \in K} \sum_{i, j \in V} \xi_{i,j} x^\omega_{k,i,j} \\
\text{s.t.} & \sum_{j \in V} x^\omega_{k,i,j} = y_{k,i} \quad \forall k \in K, \forall i \in V \\
& \sum_{i \in V} x^\omega_{k,i,j} = y_{k,j} \quad \forall k \in K, \forall j \in V \\
& y_{k,0} = 1 \quad \forall k \in K \\
& t^\omega_{k,j} = t^\omega_{k,i} + M(M + 1) x^\omega_{k,i,j} \quad \forall k \in K, \forall i, j \in V, V^+ \\
& t^\omega_{k,0} = 0 \quad \forall k \in K \\
& \sum_{i \in V} q_i y_{k,i} \leq C_k \quad \forall k \in K \\
& \sum_{k \in K} y_{k,i} = 1 \quad \forall i \in V^+ \\
\end{align*}
\]

where all constraints where an \( \omega \) apex appears should be posted \( \forall \omega \in Q \). All \( x \) and \( y \) variables are binary, and \( y_{k,i} = 1 \) 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 \( t^\omega_{k,i} \) 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:

\[
\begin{align*}
\min & \sum_{j \in V_h} c_{h, j} x_{k,i,j} + \frac{1}{|Q|} \sum_{\omega \in Q} \sum_{i \in V_h} \sum_{j \in V_h} \xi_{i,j} x^\omega_{k,i,j} \\
\text{s.t.} & \text{Eq. (32)} \\
& \text{Eq. (36)} \text{ restricted to } V_h \setminus \{0\} \\
& \text{Eq. (37) – (39)} \text{ restricted to } V_h \\
& t^\omega_{k,h} = 0
\end{align*}
\]

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 \( \omega \) we obtain:

\[
\begin{align*}
(c_{h,j} + \lambda^\omega_{k,h,j} x^\omega_{k,h,j} = 0 \quad \forall j \in V_h \\
(c_{h,j} + \lambda^\omega_{k,h,j} \geq 0 \quad \forall j \in V_h \\
\end{align*}
\]

where \( \lambda^\omega_{k,h,j} \) 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 \( r^\omega_{k,j} \) such that \( r^\omega_{k,j} = 1 \) 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:

\[
\begin{align*}
\min & \frac{1}{|Q|} \sum_{\omega \in Q} \sum_{k \in K} \sum_{i,j \in V} \xi_{i,j} x^\omega_{k,i,j} \\
\text{s.t.} & \text{Eq. (36) – (42)} \\
& (c_{ij} + \lambda^\omega_{k,i,j} x^\omega_{k,i,j}(1 - r^\omega_{k,j}) = 0 \quad \forall k \in K, \forall i, j \in V \\
& (c_{ij} + \lambda^\omega_{k,i,j} \geq 0 \quad \forall k \in K, \forall i, j \in V \\
& r^\omega_{k,i} = y_{k,i} \quad \forall i \in V \\
& r^\omega_{k,j} = r^\omega_{k,h,i} x^\omega_{k,h,h,j} \quad \forall i \in V, \forall h \in V, \forall j \in V \\
& \xi_{i,j} \leq c_{ij} \leq \tilde{c}_{ij} \quad \forall i, j \in V
\end{align*}
\]

The decision variables are \( y_{k,i}, x^\omega_{k,i,j}, \lambda^\omega_{k,i,j}, r^\omega_{k,j} \), plus the “virtual travel times” \( c_{ij} \), i.e., the parameters for the on-line heuristic. The constraints on the \( r^\omega_{k,j} \) 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 EXPEC-TATION, 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 realization 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
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 [Escapita and Ochao, 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, 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.

<table>
<thead>
<tr>
<th>Instance</th>
<th>Oracle (k€)</th>
<th>Baseline (k€)</th>
<th>BOON (k€)</th>
<th>MOON (k€)</th>
</tr>
</thead>
<tbody>
<tr>
<td>I1</td>
<td>331.362</td>
<td>404.622</td>
<td>342.061</td>
<td>344.604</td>
</tr>
<tr>
<td>I2</td>
<td>247.213</td>
<td>311.145</td>
<td>265.326</td>
<td>263.808</td>
</tr>
<tr>
<td>I3</td>
<td>393.818</td>
<td>462.577</td>
<td>404.322</td>
<td>408.721</td>
</tr>
<tr>
<td>I4</td>
<td>798.388</td>
<td>923.243</td>
<td>819.249</td>
<td>811.119</td>
</tr>
<tr>
<td>I5</td>
<td>565.607</td>
<td>684.197</td>
<td>580.174</td>
<td>573.934</td>
</tr>
<tr>
<td>I6</td>
<td>856.955</td>
<td>984.904</td>
<td>874.585</td>
<td>868.764</td>
</tr>
</tbody>
</table>

**Table 1: Cost values for the different VPP models**

<table>
<thead>
<tr>
<th>Instance</th>
<th>Off-line part (sec)</th>
<th>On-line part (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Baseline</td>
<td>MOON</td>
</tr>
<tr>
<td>I1</td>
<td>1.699</td>
<td>6.255</td>
</tr>
<tr>
<td>I2</td>
<td>2.477</td>
<td>17.445</td>
</tr>
<tr>
<td>I3</td>
<td>2.532</td>
<td>25.938</td>
</tr>
<tr>
<td>I4</td>
<td>186.798</td>
<td>338.998</td>
</tr>
<tr>
<td>I5</td>
<td>243.330</td>
<td>357.543</td>
</tr>
<tr>
<td>I6</td>
<td>361.537</td>
<td>490.856</td>
</tr>
</tbody>
</table>

**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


