Improving Metaheuristic Efficiency for Stochastic Optimization by Sequential Predictive Sampling

Noah Schutte[1][0000−0001−8499−6578], Krzysztof Postek[2][0000−0002−4028−3725], and Neil Yorke-Smith[1][0000−0002−1814−3515]

[1] Delft University of Technology, The Netherlands
{n.j.schutte,n.yorke-smith}@tudelft.nl
[2] Independent Researcher krzysztof.postek@gmail.com

Abstract. Metaheuristics are known to be effective in finding good solutions in combinatorial optimization, but solving stochastic problems is costly due to the need for evaluation of multiple scenarios. We propose a general method to reduce the number of scenario evaluations per solution and thus improve metaheuristic efficiency. We use a sequential sampling procedure exploiting estimates of the solutions' expected objective values. These values are obtained with a predictive model, which is founded on an estimated discrete probability distribution linearly related to all solutions' objective distributions; the probability distribution is continuously refined based on incoming solution evaluation. The proposed method is tested using simulated annealing, but in general applicable to single solution metaheuristics. The method's performance is compared to descriptive sampling and an adaptation of a sequential sampling method assuming noisy evaluations. Experimental results on three problems indicate the proposed method is robust overall, and performs better on average than the baselines on two of the problems.

Keywords: single solution metaheuristics · stochastic optimization · sequential sampling · prediction-based search

1 Introduction

Stochastic optimization problems gain interest because most real-world problems involve uncertainty. Some practical examples of stochastic combinatorial problems are operating room scheduling [38], renewable energy applications [37] and transportation [27]. We define this class of problems as follows.

Definition 1. *Given a finite set of solutions X, random variables ω and a real-valued objective function f, we define a* ***Stochastic Combinatorial Optimization Problem (SCOP)****:* $\min_{x \in X} \mathbb{E}_{\omega}[f(x, \omega)]$.

Noting that the expectation could be replaced by another risk functional like an α-quantile; we will limit ourselves to the expectation for clarity.

SCOPs are significantly more complex than deterministic combinatorial optimization problems. While both aim at finding an optimal solution, in SCOPs

solutions need to be evaluated against uncertainty. Uncertainty does not impact the feasibility of a SCOP's solution, as the set of feasible solutions X we consider is independent of the uncertain parameter ω. However, uncertainty affects the solution's objective value, as it is obtained by evaluating each uncertain parameter realization. Hence SCOPs have additional complexity from solution evaluation.

Several modelling frameworks exist to tackle SCOPs. Exact modelling frameworks, like mathematical or dynamic programming [36,6], are designed to guarantee finding an optimal solution. However, due to the complexity of general SCOPs, these frameworks are often limited to small problems or unrealistic modelling assumptions. Non-exact methods, on the other hand, are particularly successful in finding good solutions in reasonable time. Specifically, *metaheuristics* – general non-exact frameworks that utilize heuristics – have shown to be effective for both deterministic and stochastic problems as they efficiently search the solution space [7,16]. However, metaheuristics for SCOPs are still limited in performance due to facing the challenge of evaluating found solutions against uncertainty [7]. This is in practice often done using simulation, which can be expensive and therefore slows down the search [15].

There is significant research on metaheuristics for specific SCOPs [7,16]. For example, [31] improve the efficiency of adaptive large neighbourhood search for a stochastic surgery scheduling problem. However, there is little work on methods that improve metaheuristic efficiency for SCOPs in general. This paper advances this research area with three specific contributions:

– A generic method that improves the efficiency of metaheuristics when solving SCOPs. The method utilizes prior information in a memory-efficient way and uses a predictive model to estimate the solutions' expected objectives and to estimate the variance of this estimate.
– The notion of a general output scenario distribution. This provides a means to compare solution quality. We provide two approaches to estimate this distribution.
– Application of the method in combination with simulated annealing to demonstrate its effectiveness on three different problems.

2 Background

2.1 Stochastic Optimization

We focus on the discrete stochastic optimization problems, as in Definition 1. Within this framework it is assumed that the distribution of the uncertain parameter ω is known. Given a tractable mathematical formulation of the SCOP, the model can be solved efficiently with a guaranteed optimal solution. However, for a wide range of problems these exact methods become computationally too expensive [7,12].

One reason for this is that uncertainty makes evaluating both objectives and constraints complicated as it potentially requires integration over continuous random variable ω. To circumvent this, discretization is often used to make SCOPs tractable [8]. A common approach for this is *sample average approximation* [18]

in which random samples are being drawn from the continuous probability distribution. These scenarios are assumed as being equally likely and the SCOP then has a deterministic equivalent formulation as an approximation, where each component of each uncertain variable is replaced by all potential scenarios. This still poses a challenge because to compute the average objective value, the objective function $f(x, \omega)$ in Definition 1 still needs to be evaluated for each scenario ω.

This evaluation is challenging because the objective function f need not even be a closed-form function. For example, it can be the optimal value of another optimization problem as in the case of a common class of SCOPs – two-stage problems. There, the goal is to find an optimal *here-and-now* decision before knowing the uncertain parameters' realizations, while the *wait-and-see* decisions are optimized after the uncertain parameters are realized. In other situations f cannot even be mathematically modelled. This is because processes and systems are often modelled by simulation and therefore, in practical SCOPs, evaluating f is often done by a simulation. *Simulation–optimization* thus studies optimization methods under the assumption that $f(x, \omega)$ is a black box. The main challenge there is to limit the number of simulations, as they can be expensive [1]. Because of the above two examples, we assume the following:

Assumption 1 *Objective function f in Def. 1 does not have a closed form, implying that the expected objective $\mathbb{E}_\omega[f(x, \omega)]$ also does not have a closed form.*

Assumption 1 implies that when ω has a continuous distribution, we need to approximate the expected objective using a discrete distribution. In the sequel, Ω is denoted as the set that includes all potential realizations of ω.

2.2 Metaheuristics for Stochastic Optimization

Metaheuristics are designed to find good solutions fast and, within a limited time, they often outperform exact methods. Simulated annealing, tabu search and genetic algorithms are effective metaheuristics trying to find better solutions by modifying earlier-found solutions. The modified solutions are the potential *moves* to make and they are evaluated before further moves are made. This procedure repeats itself until a final solution is returned. Initially designed for solving deterministic problems, metaheuristics have also shown to be effective for SCOPs [7,16]. Stochastics makes the objective evaluation computationally more expensive for a solution x as, based on Assumption 1 and Def. 1, we need to evaluate $f(x, \omega)$ for every scenario $\omega \in \Omega$ to find the exact expectation. The more expensive the evaluation of f and the higher the computational costs of a move (creating a new solution), the slower the metaheuristic search becomes. We show in Example 1 that even for problems with a simple f the required additional evaluations can increase the computational costs significantly. We first properly introduce simulated annealing.

Simulated annealing (SA) is an archetypal metaheuristic, outlined high-level in Algorithm 1. SA is a *single solution* metaheuristic keeping a single *current*

Algorithm 1 Simulated Annealing

Input: budget b_0, initial temperature t_0, initial solution x
Output: final solution x

Let $t \leftarrow t_0$, $b \leftarrow b_0$
while $b > 0$ **do**
    Reduce budget b
    Update temperature t
    Get move x' and evaluate x'
    if $\mathbb{P}_{\text{accept}}(x, x', t)$ high enough **then**
        $x \leftarrow x'$
return solution x

solution in memory at all times, as opposed to *population-based* metaheuristics like genetic algorithms. Depending on the quality of a move and the value of a parameter t called the temperature, the new *candidate* solution is accepted or rejected. After this a new move is made in the following iteration. A higher temperature encourages exploration, i.e., the algorithm is more likely to accept solutions, even if they are worse. A lower temperature encourages exploitation, as it only accepts better solutions. Over time, the temperature is decreased within the SA procedure. Given this definition of SA, we now present Example 1.

Example 1. Consider a project scheduling problem in which the goal is to minimize the total makespan of a project. The project consists of set A with n activities that need to be scheduled. The scheduling of activities is constrained by some resources that the activities require. This is known as the Resource Constrained Project Scheduling Problem (RCPSP). A deterministic solution to this problem is most commonly defined as a list of activity start times $(s_{a_1}, \ldots, s_{a_n}) : a_i \in A, s_{a_i} \in \mathbb{R}_+$. If the duration of activities is uncertain however, the actual start times are unknown upfront. Because of this a solution becomes a policy. We define a solution as a *priority list* of activities $(a_1, \ldots, a_n) : a_i \in A$. This list is the order in which activities will start, such that an activity can never start before another activity that is higher on the priority list [11]. Now we use SA to find a good solution. As a move, we swap the place of two activities in the priority list. The evaluation consists of building a realized schedule $f(x, \omega)$ for each scenario $\omega \in \Omega$ and taking the average of the obtained makespans. Swapping two activities is a single operation. Building a realized schedule is n operations, as it consists of scheduling n activities. This is done $|\Omega|$ times. Hence the evaluation is about $|\Omega| * n$ times as expensive.

2.3 Sampling

Recently there has been a resurgence of interest in sampling due to its relevance in sequential decision making problems in general and reinforcement learning (RL) in particular [9]. The *multi-armed bandit problem* is a sub-problem of RL in which the decision maker has to decide between n actions, each with an unknown

but fixed probability distribution of rewards [21]. This problem specifically deals with the trade-off of exploration (new action) and exploitation (best-rewarded action). In our problem setting, the metaheuristic is tasked with solving the sequential decision problem of evaluating at each iteration a new scenario for a current solution, or moving to a new solution: each action maps to a solution and the obtained reward is the evaluated solution objective for some scenario. This is a similar problem setting to RL, but not exactly the same: in our setting the fixed probability distribution of rewards is still unknown, but the fixed probability distribution of uncertain variables that determine the rewards is known. We can exploit this by deciding which samples to draw, i.e., which scenarios to evaluate. In the next section we present two baselines that use this principle. Note that RL in general can be used as a metaheuristic alternative or as a hybrid [32], but typically requires a lot of data and is therefore expected to be inferior in a setting with (relatively) expensive solution evaluations.

3 Baseline Sampling Methods

3.1 Descriptive sampling

Our central question is how to obtain high-quality solutions with the metaheuristic evaluating fewer scenarios per solution. In this section we provide two baselines.

The first idea to decrease the number of evaluations is to create a smaller set of scenarios Ω when using sample average approximation. The drawback is that the set might not be representative anymore for the underlying distribution or data set. A method that creates a more representative set of scenarios compared to random sampling is *descriptive sampling* [29]. When a set of m scenarios is constructed, descriptive sampling creates m equally spaced (by probability density) realizations for each random variable. The scenarios are then constructed by selecting a random permutation for all possible realizations for each random variable, where each realization corresponds to a scenario.

One of the drawbacks of an approach like descriptive sampling is that it creates a representative set based on the distribution of the random variable ω itself. We call this the *input distribution*, which is different from the *output distribution* defined as the distribution of $f(x, \omega)$ given a solution $x \in X$. This distinction is important as we are optimizing over this objective, and we need the approximated output distribution to be representative of the actual output distribution if we want to make good optimization decisions. We show with Example 2 that a representative set for the input distribution is not necessarily a representative set for the output distribution.

Example 2. Consider a RCPSP as described in Example 1, with 3 activities $A = \{a, b, c\}$. Due to limited resources at most 2 activities can be scheduled at the same time. The duration of activity a is fixed at $d_a = 2$, and the duration of b and c are independent and uniformly distributed $d_b, d_c \sim U(0, 4)$. Our goal is to find the priority list for which the expected makespan is minimized. Consider descriptive sampling with two scenarios for realizations of (d_b, d_c). Out

of 4 possible descriptive samples (consisting of durations in $\{1, 3\}$), we assume we get $\omega = (1, 1)$ and $\nu = (3, 3)$. Since d_b and d_c have the same distribution and the project will start with two activities at the same time, effectively there are two different solutions possible: (a, b, c) and (b, c, a). We have the following relationships between the expected makespans of the solutions:

$$\begin{aligned}\mathbb{E}_{\{d_b,d_c\sim U(0,4)\}}[f((b,c,a),(d_b,d_c))] = 3\tfrac{1}{3} \quad &< \quad \mathbb{E}_{\{d_b,d_c\sim U(0,4)\}}[f((a,b,c),(d_b,d_c))] = 3\tfrac{1}{2},\\ \mathbb{E}_{\{(d_b,d_c)\in\{\omega,\nu\}}[f((b,c,a),(d_b,d_c))] = 4 \quad &> \quad \mathbb{E}_{\{(d_b,d_c)\in\{\omega,\nu\}}[f((a,b,c),(d_b,d_c))] = 3\tfrac{1}{2}.\end{aligned}$$

This shows that while the true optimal solution is (b, c, a), descriptive sampling would return (a, b, c) as optimal.

3.2 Sequential sampling

A second idea aimed at efficient evaluation is to use *sequential sampling* within the metaheuristic. In sequential sampling there is not a given set or number of scenarios sampled, but after every sample a decision is made to continue sampling or not. In the case of *population-based* metaheuristics, multiple new solutions are encountered at the same time. This means that the sequential sampling strategy becomes more elaborate, as there is not just one solution for which the sample decision needs to be made. For example, [5] apply sequential sampling to particle swarm optimization and [14] apply sequential sampling to an evolutionary algorithm.

Similarly, sequential sampling has been applied with SA under noise, where it is assumed that an objective evaluation has added Gaussian noise [2]. In this case, without a predetermined number of evaluations per solution, a solution will only be accepted or rejected when enough evaluations have been done to make this decision. Therefore, at any point there will be three potential decisions: accept the solution, reject the solution or evaluate on more scenarios. [10] used this principle to accept/reject when the solution difference between candidate and current solution is significant. This is inefficient however, as it does not adhere to the fundamental *detailed balance equation* that deterministic SA is based upon. This balance equation ensures that SA reaches equilibrium at each temperature level if given sufficient time, such that it converges in probability to the optimal solution [17]. [2] impose the detailed balance equation conditions at this decision level, under which they maximize the acceptance probability per sample, which is a measure for the efficiency of the algorithm. The authors propose the following acceptance probability:

$$\mathbb{P}_{\text{accept}}(c_n, t) = \min(1, e^{-2(c_n+\sigma^2/(2t))(c_{n-1}+\sigma^2/(2t))/\sigma^2}), \tag{1}$$

where c_n is the cumulative performance difference after n samples:

$$c_n = \textstyle\sum_{\omega\in\Omega_n} f(x, \omega) - f(x', \omega), \tag{2}$$

and Ω_n is the set of n samples. Similarly they derive an optimal rejection rule to apply if the solution is not accepted. However, since this rule is dependent

Algorithm 2 Sequential Difference Sampling

Input: current solution x, candidate solution x'
Output: new current solution x

Let $n \leftarrow 0$, $c_n \leftarrow 0$, $\Omega_x \leftarrow \emptyset$
while *true* **do**
  $n \leftarrow n + 1$
  Sample $\omega \in \Omega \backslash \Omega_x$
  $\Omega_x \leftarrow \Omega_x \cup \omega$
  Evaluate $f(x, \omega)$, $f(x', \omega)$
  $c_n \leftarrow c_{n-1} + f(x, \omega) - f(x', \omega)$
  Determine $\mathbb{P}_{\text{accept}}(c_n, t)$
  if $\mathbb{P}_{\text{accept}}(c_n, t) > u, u \sim U(0, 1)$ **then**
    return solution x'
  else if $c_n < 0$ **then**
    return solution x

on an unknown prior distribution of futures moves, a more simple but effective rejection rule is proposed: Reject if $c_n < 0$, i.e.,

$$\mathbb{P}_{\text{reject}}(c_n, t) = \mathbb{1}_{\{c_n < 0\}}. \tag{3}$$

If the move is neither accepted nor rejected, another sample is taken and this process is repeated as can be seen in Algorithm 2. We fit this procedure to our methodology as described in Section 5.

4 Sequential Predictive Sampling

Our goal is to increase efficiency by reducing the number of scenarios evaluated. We aim to support this goal by utilizing prior information: function evaluations of the earlier-found solutions. In most metaheuristics, solution information is only kept if their corresponding solutions are still considered promising. By contrast, Prudius and Andradóttir [25] introduce an averaging framework in which solutions are evaluated partially during the search: an estimate of the expected objective per solution is stored as the average of all obtained objectives, which makes it possible to improve this estimate by more solution evaluations. Our method, however, does not keep all solution evaluations in memory, as it only deems the more recent solutions evaluations interesting.

The proposed methodology is summarized as follows. Given a new solution, two random scenarios from Ω are evaluated (two is the minimum number that gives us relative information). Based on this evaluation, a predictive model is set up and returns an estimate of the expected objective as well as the variance of this estimate. Based on this, an accept, reject or sample decision can be made in the overarching metaheuristic. The predictive model takes a *general scenario output distribution* as input. This distribution is estimated based on evaluations done on previously found solutions.

Commonly, a solution's expected objective would be determined by evaluating over all scenarios in some set of scenarios Ω. Although we want expected objectives to be representative over all uncertainty (i.e., the whole Ω), evaluating every solution on every scenario is not necessary. As we argue below, this is because not all evaluations are independent.

A fundamental assumption behind any search algorithm is that some solutions are better than others:

Assumption 2 *Some solutions are better than others:* $\exists x, y \in X$ *such that*

$$\mathbb{E}_\omega[f(x,\omega)] < \mathbb{E}_\omega[f(y,\omega)].$$

Similarly, we assume that some scenarios are uniformly better than others, recognizing that uncertainty can have a positive or a negative impact:

Assumption 3 *Some scenarios are better than others: Considering* x *as a discrete uniformly distributed random variable with as support solution space* X*, we assume that for some* $\omega, \nu \in \Omega$ *it holds that*

$$\mathbb{E}_x[f(x,\omega)] < \mathbb{E}_x[f(x,\nu)].$$

Example 3. Due to the structure of a scheduling problem, a realization with long activity durations across all activities is worse than one with short activity durations. Take for examples the two scenarios obtained in Example 2, defined as the two durations of uncertain activities b and c: $\omega = (1,1)$ and $\nu = (3,3)$. Given solution (b,c,a), we get as makespans: $f((b,c,a),\omega) = 3 < f((b,c,a),\nu) = 5$. More generally for this problem, if one scenario ν *dominates* another scenario ω, the makespan can never be better, i.e.,

$$d_i^\nu \geq d_i^\omega \quad \forall i \implies f(x,\nu) \geq f(x,\omega) \quad \forall x \in X.$$

This holds for all problems where the impact of the uncertain parameter is similarly related to the objective function (linearly for some solutions).

Based on Assumption 2 and 3 we go one step further. The fact that both solutions and scenarios can be better or worse, makes it realistic to assume some dependency between evaluations that are based on these solutions and scenarios. When an evaluation $f(x,\omega)$ is perceived as 'good', it is likely that x and ω are both relatively good as well.

4.1 General scenario output distribution estimation

Given our reasoning on dependence of evaluations, we introduce a linear dependence assumption that we consider to hold with a margin of error that is small enough to help us compare evaluations. Even though the assumption will most often not hold exactly, it will help us steer the solution evaluations.

We first define two uniform discrete distributions that we will use to relate evaluations: *solution output distribution* E and *general scenario output distribution* D with as sets of atoms:

$$E_X := \{e_x : x \in X, e_x := \mathbb{E}_\omega[f(x,\omega)]\},$$
$$D_\Omega := \{d_\omega : \omega \in \Omega, d_\omega := \mathbb{E}_x[f(x,\omega)]\}.$$

Assumption 4 *There exist linear relationships between solutions given a scenario, i.e., for any $x, y \in X$:*

$$f(x,\omega)/f(y,\omega) = e_x/e_y \quad \forall \omega \in \Omega,$$

where the linear relation is decomposed as a fraction e_x/e_y. Further, there exist linear relationships between scenarios given a solution, i.e., for any $\omega, \nu \in \Omega$:

$$f(x,\omega)/f(x,\nu) = d_\omega/d_\nu \quad \forall x \in X,$$

where the linear relation is decomposed as a fraction d_ω/d_ν.

Lemma 1. *Given Assumption 4, we have $\forall x, y \in X$, $\forall \omega, \nu \in \Omega$:*

$$f(x,\omega)/f(y,\nu) = e_x d_\omega/(e_y d_\nu) \;\Rightarrow\; \forall c \in \mathbb{R}\colon f(x,\omega) = c\, e_x d_\omega \wedge f(y,\nu) = c\, e_y d_\nu.$$

We note that constant c in Lemma 1 is irrelevant in comparing evaluations and we therefore omit it from here on. Given Assumption 4, Lemma 1 ensures we can have a rank-1 decomposition of matrix $\boldsymbol{F}$, $\boldsymbol{F} = \boldsymbol{e}\boldsymbol{d}^T$, where $\boldsymbol{e}$ and $\boldsymbol{d}$ denote the vectorized estimates of sets E_X and D_Ω. In practice, Lemma 1 only holds when the objective function is linear in the uncertain parameters, which is unlikely when f is considered to not have a closed-form (Assumption 1). When Assumption 4 holds, the stochastic problem effectively becomes a deterministic problem where the solution quality can be compared exactly when all solutions are evaluated on the same, single scenario $\omega \in \Omega$. Therefore, Assumption 4 serves as a strong baseline. It ensures that when uncertainty is (close to) linear in the evaluation function, the method performs similar to assuming the problem is deterministic. Now that we have defined general scenario output distribution D, we present a method to estimate it. We will use linearity throughout such that the estimator is perfect if the rank-1 decomposition $\boldsymbol{F} = \boldsymbol{e}\boldsymbol{d}^T$ exists.

Bayesian inference is popular in estimating distributions [13]. However, the general scenario output distribution can have any kind of shape and therefore we refrain from assuming a prior distribution. A prior distribution can be especially troublesome since observing only a few evaluations per solution will update the posterior only slightly for each new solution. Furthermore, solution evaluations are not random draws from the same distribution. Because of this we use discrete linear updates. Given an output scenario distribution D and Assumption 4, we can update this distribution assuming the evaluations from the last obtained solution are most representative of the general scenario output distribution. Denoting x as the last evaluated solution, we use:

$$d_\omega \leftarrow r f(x,\omega) \frac{\sum_{\omega \in \Omega_x} d_\omega}{\sum_{\omega \in \Omega_x} f(x,\omega)} + (1-r) d_\omega, \quad \forall \omega \in \Omega_x, \tag{4}$$

where Ω_x is the set of scenarios on which x is evaluated and r is the rate by which the current values are replaced, which makes the updates more smooth. When using SA, this update can be done after each iteration. Note that if the rank-1 decomposition is equal to the actual evaluation matrix, i.e., $\boldsymbol{F} = \boldsymbol{e}\boldsymbol{d}^T$, these updates ensure the general scenario output distribution has a linear relationship with all solution specific scenario output distributions. This leads to a perfect predictor given a linear predictive model, hence we will use this model class.

4.2 Predictive model

Now we have defined an approach to obtain a general scenario output distribution D with Equation 4, we use this distribution to compare obtained solutions during the metaheuristic search. Since we are developing a sequential sampling method, this comparison will include specifying a accept, reject or sample decision as introduced in Section 3.2. When evaluating a solution x, we set up the following predictive model:

$$f(x,\omega) = \beta_x d_\omega + \varepsilon_{x,\omega}, \tag{5}$$

where $\varepsilon_{x,\omega}$ is an error term and we estimate β_x by the ordinary least squares estimator that minimizes squared errors of the predictive model:

$$\hat{\beta}_x := \operatorname*{argmin}_{\beta_x} \sum_{\omega\in\Omega_x} (f(x,\omega) - \beta_x d_\omega)^2 = \frac{\sum_{\omega\in\Omega_x} d_\omega f(x,\omega)}{\sum_{\omega\in\Omega_x} d_\omega^2}.$$

Given the predictive model from Equation 5 with $\beta_x = \hat{\beta}_x$, we get an estimate of the expected objective of solution x:

$$\hat{\mu}_x := \hat{\mathbb{E}}_\omega[f(x,\omega)] = \frac{1}{|\Omega|}\Big(\sum_{\omega\in\Omega_x^c} \hat{\beta}_x d_\omega + \sum_{\omega\in\Omega_x} f(x,\omega)\Big),$$

where $\Omega_x^c = \Omega \setminus \Omega_x$. An estimate of the objective value by itself is useful, but additionally we also get a measure of the uncertainty of the prediction by estimating the variances as:

$$\widehat{\operatorname{Var}}(\mu_x) = \frac{1}{|\Omega|^2}\Big(\Big(\sum_{\omega\in\Omega_x^c} d_\omega\Big)^2 \operatorname{Var}(\hat{\beta}_x) + |\Omega_x^c| \operatorname{Var}(\varepsilon_x)\Big).$$

$$\widehat{\operatorname{Var}}(\varepsilon_x) = \frac{1}{|\Omega_x| - 1} \sum_{\omega\in\Omega_x} (f(x,\omega) - \hat{\beta}_x d_\omega)^2.$$

$$\operatorname{Var}(\hat{\beta}_x) = \widehat{\operatorname{Var}}(\varepsilon_x) / \sum_{\omega\in\Omega_x} d_\omega^2.$$

Given this setup, we know the expected objective of a solution x has a Student's t-distribution with $|\Omega_x| - 1$ degrees of freedom, shifted by mean $\hat{\mu}_x$ and scaled by estimated variance $\widehat{\operatorname{Var}}(\mu_x)$, assuming error term $\varepsilon_{x,\omega}$ is normally distributed. Hence, we effectively have an estimated variance of: $\hat{\sigma}_x^2 := \widehat{\operatorname{Var}}(\mu_x)\frac{|\Omega_x|-1}{|\Omega_x|-3}$.

The use of a linear model ensures that if the evaluations' relationships are linear, the predictive model can predict perfectly: $\hat{\sigma}_x^2 = 0$.

4.3 Full *SeqPre* procedure

We summarize the full procedure in Algorithm 3. The comments therein show the exact implementation of the procedure for the experiments. We note that in the case of SA, the procedure replaces algorithmic steps 5–7 of Algorithm 1. All the input variables are updated and kept in memory during the search. Note that in Step 3 we always select x' if it does not have any evaluations yet. Note also that the while loop (steps 2–12) is guaranteed to terminate since scenario set Ω is finite and therefore when all scenarios are evaluated $\hat{\sigma}_y^2 = 0$ and $\mathbb{P}_{\text{accept}}(x,x',t), \mathbb{P}_{\text{reject}}(x,x',t) \in \{0,1\}$.

Algorithm 3 Sequential Predictive Sampling

Input: current and candidate solution x, x', estimation parameters $\hat{\mu}_x, \hat{\sigma}_x^2$, scenario set Ω, evaluated scenario set Ω_x, general scenario output distribution D, temperature t
Output: new current solution x

Update D ▷ using Equation 4
while *true* **do**
  Select $y \in \{x, x'\}$ ▷ with $\mathbb{P}(y = x) = |\Omega_x|/(|\Omega_x| + |\Omega_{x'}|)$
  Sample $\omega \in \Omega \backslash \Omega_y$ ▷ at random
  $\Omega_y \leftarrow \Omega_y \cup \omega$
  Evaluate $f(y, \omega)$
  Update $\hat{\mu}_y$ and $\hat{\sigma}_y^2$ ▷ using predictive model in Equation 5
  Determine $\mathbb{P}_{\text{accept}}(x, x', t)$, $\mathbb{P}_{\text{reject}}(x, x', t)$ ▷ using Equation 1, 3
  if $\mathbb{P}_{\text{accept}}(x, x', t) > u, u \sim U(0, 1)$ **then**
    return solution x'
  else if $\mathbb{P}_{\text{reject}}(x, x', t) > u, u \sim U(0, 1)$ **then**
    return solution x

5 Experiments

We examine the performance of our proposed method on three problems. Experiments were implemented in Python and performed on a single 2.0GHz CPU running Ubuntu 20.4 with 32GB RAM. Source code is available on GitHub [30].

5.1 General experimental configuration

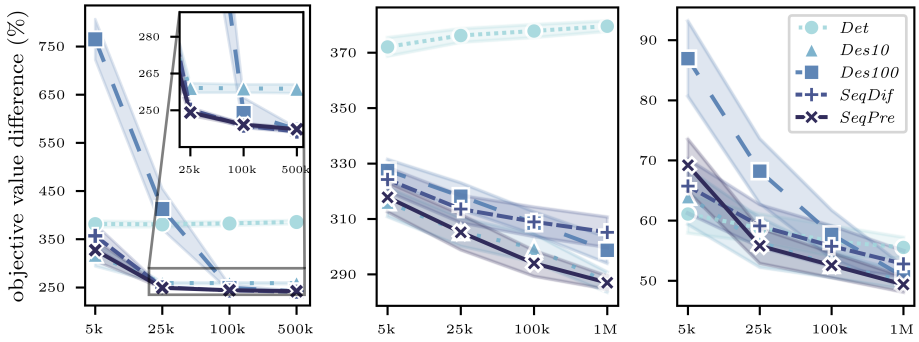
Each method uses the same underlying SA procedure as outlined in Algorithm 1. To compare solution objectives we give each search algorithm a fixed budget b_0 of evaluations. This is common practice for search algorithms as it shows efficiency independent of the underlying machine specifications [20]. We also denote runtime in the results, however we want to emphasize that the experimental problems have relatively cheap evaluations (non-extensive simulations) which makes the overhead greater than it would be for more practical problems. As annealing schedule, geometric annealing is used with a fixed final temperature for fair comparison between methods:

$$t \leftarrow t_0(t_\text{f}/t_0)^{(1-b/b_0)}.$$

The final temperature is set at a small value $t_\text{f} = 0.01$. The initial temperature is:

$$t_0 = -h \left\lceil \min_{x \in X} f(x, \mathbb{E}_{\omega \in \Omega}[\omega]) \right\rceil / \ln(p),$$

where $\lceil \cdot \rceil$ rounds up to the closest power of 10 and $h = 0.1$, $p = 0.5$ are such that initially solutions that are 10% worse are accepted with a probability of 0.5.



(a) SSFORU, instance size 150 (b) SCVRPTW, mean (c) SRCPSP, mean

Fig. 1: Objectives values and 95% bounds in % difference to the optimal deterministic solution obj. value based on the test set and relative to budgets b_0.

5.2 Methods compared

We use three benchmarks that evaluate a fixed number of scenarios at each iteration. One of them evaluates the single scenario where each uncertain parameter attains its average. This is searching for a deterministic optimal solution, so we call this method *Det*. The other two variants use descriptive sampling to obtain a set of scenarios. Scenario sets of sizes 10 and 100 are used and the methods are denoted as *Des10* and *Des100*. Size 10 has shown to be effective for some problems in literature [28,4], while size 100 is assumed to be large enough to be representative for the input distribution.

Two methods are proposed to compare to these benchmarks. The first method is sequential *difference* sampling (*SeqDif*) based on [2] as shown in Algorithm 2. A random difference is sampled between the two solutions that are compared. However, we do not assume Gaussian noise with known variance, so we estimate the variance by updating it throughout the search:

$$\hat{\sigma}_i^2 \leftarrow (\hat{\sigma}_{i-1}\sqrt{i-1} + c^i_{n_i})^2/i,$$

where i is the iteration and $c^i_{n_i}$ is the cumulative difference as defined in Equation 2 when an accept or reject decision is made after n_i samples for iteration i.

The second method is our proposed sequential *predictive* sampling approach (*SeqPre*), as shown in Algorithm 3. Hyper-parameters as defined in Section 4 are set at $k = 50$ and $r = 0.9$ (chosen based on empirical results). Both sequential approaches use as decision rule the steps 9-12 in Algorithm 2, and a descriptive set of 100 scenarios as Ω. The final obtained solutions are evaluated on a test set of 1000 randomly drawn scenarios with a unique seed.

5.3 Description of example problems

Surgery Scheduling in Flexible Operating Rooms under Uncertainty (SSFORU) has as goal to plan as many elective surgeries as possible in a given set of operating

rooms, while minimizing idle and overtime for the staff and waiting time for the patients. At any time during the day an emergency surgery can come in that needs to be performed as soon as any operating room is available. This problem has been studied in literature in several modifications [26] and was proposed in this form by [34], with instance sizes of $n \in \{70, 100, 150, 200\}$.

We use the most generally used costs configuration ($c_{\text{over}} = 26$, $c_{\text{idle}} = \frac{2}{3}c_{\text{over}}$, $c_{\text{waiting}} = \frac{1}{5}c_{\text{over}}$, $c_{\text{not-scheduling}} = 30 \times \frac{3}{4}c_{\text{over}}$) in literature [24,23,33]. Emergency inter-arrival times are modelled as exponentially distributed with a rate of 4 per day and surgery duration times are modelled as log-normal distributed fitted by type. In the realization of a schedule, emergency surgeries are assigned to an operating room based on the lowest expected duration of the remaining planned elective surgeries. In the SA procedure moves are defined as removing or adding elective surgeries and shifting the times of already planned surgeries.

Stochastic Resource Constrained Project Scheduling Problem (SRCPSP) is a well studied problem, both the deterministic variant and different stochastic adaptations. The goal is to minimize the total duration of a project that consists of activities. Some activities can only start when certain others are finished (precedence relationships), while they also have resource requirements and there are limited resources available. We define a solution as a priority list of activities as in Example 1, while the evaluation of a schedule consists of simulating the realized schedule and obtaining the makespan. We test on instances 1.1–1.5 from the PSPLIB [19] with 30 activities. The duration of the activities is uncertain and modelled as exponentially distributed with as mean the given duration in the instance. In the SA procedure we define a move as a swap between two consecutive activities in the priority list. These are done randomly, limited to swaps that adhere to the precedence relationships.

Stochastic Capacitated Vehicle Routing Problem with Time Windows (SCVRPTW) has a given set of the same vehicles stationed at a depot, with the goal to serve a given set of customers. The vehicles have some capacity, while each customer has demand and a time window in which delivery is allowed. A solution is a route for reach vehicle. We test on instances R101–R105 introduced by [35].

Both demand and travel times are uncertain and modelled as exponentially distributed with as mean the given demand and the Euclidean distance respectively. Similarly to [22], costs are determined by distance travelled. The evaluation of a solution and scenario consists of a simulation in which vehicles travel to the customers. When a vehicle does not have enough product or arrives after the end of the time window, a cost penalty is incurred equal to a single trip back and forth to the depot. When a vehicle arrives before the start of a time window it waits. In the SA procedure a move is defined as for each route: 1) randomly adding or removing a customer, or doing nothing; 2) randomly swapping two consecutive customers.

5.4 Experimental results

Tables 1 and 2 show the experimental results. Baseline *Det* performs poorly, due to over-fitting on the single scenario it uses. We see similar behaviour in lower

magnitudes for *Des10*. Figure 1a illustrates that this over-fitting causes solutions to not improve for larger budgets. On the other hand, *Des100* does improve with larger budgets, but performs poorly on smaller budgets.

In general the aim of the sequential sampling methods is to reach similar objectives as *Des100* in less time. We see that for SSFORU, *SeqDif* and *SeqPre* perform similarly and better than the benchmarks. For SRCPSP and SCVRPTW however, *SeqPre* performs significantly better than *SeqDif*. This could be caused by the fact that differences of individual scenarios are not good enough estimates for the actual differences, or because the estimated variance is not accurate enough. Another difference between problem results is that *Des10* performs close to the best methods for SRCPSP and SCVRPTW. Specifically for SRCPSP we can state that this is caused by the fact that uncertainty does not change solutions that much, as the obtained solution is still competitive despite over-fitting. This confirms the findings in [3].

6 Conclusions and Future Work

The central idea of this paper is that metaheuristic efficiency can be improved by reducing the number evaluations per solution. We propose a general methodology based on a sequential sampling procedure and a predictive model that utilizes earlier obtained information memory-efficiently. This is done by modelling linear relationships between both solutions and scenarios. We applied the approach to SA on three diverse problems and found consistently strong performance.

Future work includes applying this method on metaheuristics in addition to SA: For single solution metaheuristics the method is readily applicable, while for population-based metaheuristcs the procedure of accepting/rejecting solutions needs to be adjusted. A second avenue is exploring potential further improvements: modelling more complex relationships between evaluations; and including solution characteristics in the predictive model, as currently it only uses solution evaluations. Finally, more sophisticated machine learning methods could be applied to estimate a general scenario distribution or predict objectives directly from all obtained information. It can require more computing power to learn a strong predictive model than to perform the search, but this could still be useful when offline computation is cheaper.

Acknowledgements. We thank the reviewers for their suggestions. This work was partially supported by Epistemic AI and by TAILOR, both funded by EU Horizon 2020 under grants 964505 and 952215 respectively.

n	Method	$b_0 = 5$k			$b_0 = 25$k			$b_0 = 100$k			$b_0 = 500$k		
		Test	Train	s	Test	Train	s	Test	Train	s	Test	Train	s
70	*Det*	298.1±4.2	32	9	344.9±5.5	29	47	327.2±6.2	29	189	322.1±6.0	29	972
	Des10	**188.9**±2.3	120	8	179.0±2.0	103	40	170.7±1.7	94	151	165.8±1.1	89	748
	Des100	394.7±15.7	388	6	204.5±9.0	194	44	163.1±1.2	152	190	158.2±0.6	144	943
	SeqDif	199.3±3.4	190	10	**165.6**±1.3	155	59	*__159.4__±0.6	146	233	**156.4**±0.9	142	1172
	SeqPre	190.9±2.4	181	11	169.3±0.8	158	61	165.8±0.9	152	243	158.5±0.6	143	1309
100	*Det*	809.4±7.0	60	11	819.0±6.9	48	58	837.6±7.9	46	227	836.4±6.4	46	1156
	Des10	**594.5**±13.5	455	12	547.6±3.6	384	56	540.8±3.7	365	241	539.0±3.3	355	1202
	Des100	1194.8±41.8	1177	8	669.6±35.5	647	60	514.6±1.6	483	238	505.4±1.4	466	1290
	SeqDif	655.3±18.6	629	14	526.9±2.1	501	78	**507.7**±1.5	475	354	*__501.7__±1.1	462	1747
	SeqPre	**594.5**±5.3	571	14	*__521.1__±1.7	494	83	510.9±1.8	478	349	505.5±1.4	469	1807
150	*Det*	381.5±3.5	32	17	380.9±3.0	24	66	382.9±2.5	20	367	385.8±2.1	18	1796
	Des10	**317.9**±12.3	257	17	259.1±1.0	188	71	258.9±0.9	178	288	258.7±0.9	175	1323
	Des100	764.7±21.5	764	13	412.7±21.6	407	76	248.9±3.2	238	291	242.2±0.3	228	1412
	SeqDif	357.0±12.5	353	19	250.2±0.5	242	77	**243.7**±0.4	232	327	*__241.3__±0.3	227	1400
	SeqPre	327.4±11.2	323	17	**249.1**±0.7	239	93	244.1±0.3	233	341	242.2±0.4	229	1753
200	*Det*	**210.7**±0.7	26	13	209.2±0.5	20	60	209.3±0.4	20	228	208.6±0.3	18	1156
	Des10	219.8±10.8	188	22	140.6±0.7	106	69	138.9±0.7	102	252	140.0±0.8	100	1030
	Des100	631.7±16.7	624	31	335.1±17.3	329	144	138.7±2.4	135	384	**130.7**±0.1	126	1485
	SeqDif	274.2±18.0	269	21	135.9±0.4	132	87	**131.8**±0.2	128	321	**130.7**±0.2	125	1354
	SeqPre	257.2±10.6	252	32	**134.7**±0.3	132	90	132.1±0.2	128	306	131.3±0.1	126	1672

Table 1: Results SSFORU per instance size n for different evaluation budgets b_0: average test (± SE) and train objectives in % difference to the optimal deterministic objective based on 20 random seeds, and average time s in seconds. * shows significant difference between the 2 best methods (paired t-test, $\alpha = 0.05$).

	Method	$b_0 = 5$k			$b_0 = 25$k			$b_0 = 100$k			$b_0 = 1$M		
		Test	Train	s	Test	Train	s	Test	Train	s	Test	Train	s
SRCPSP	*Det*	**61.1**±1.6	9	2	58.8±1.2	5	8	56.6±1.3	3	31	55.5±0.9	1	317
	Des10	64.0±2.4	52	2	56.4±2.1	43	8	**52.6**±1.1	39	35	50.1±0.7	35	351
	Des100	86.9±3.2	85	2	68.2±2.8	66	8	57.7±2.0	55	33	50.7±0.9	48	348
	SeqDif	65.7±2.3	64	2	59.1±1.8	57	9	55.7±1.7	54	39	52.8±1.1	51	397
	SeqPre	69.2±2.2	66	3	**55.8**±1.5	53	17	**52.6**±1.1	50	67	**49.4**±0.7	47	677
SCVRPTW	*Det*	372.1±1.7	175	6	376.2±1.3	163	32	377.8±1.2	155	127	379.6±1.0	145	1278
	Des10	**316.1**±2.7	313	1	305.3±2.6	302	6	299.5±2.6	297	22	287.2±1.9	285	217
	Des100	327.5±2.0	327	1	318.2±2.5	318	4	308.5±2.1	308	15	298.7±2.4	298	129
	SeqDif	324.2±2.2	324	2	313.6±2.5	313	10	309.0±2.7	308	49	305.2±2.8	305	551
	SeqPre	317.8±2.7	317	4	**305.2**±3.3	305	19	*__294.0__±2.4	294	81	**287.0**±1.2	287	844

Table 2: Results SRCPSP & SCVRPTW for different evaluation budgets b_0: test (± SE) and train objectives in % difference to the optimal deterministic objective averaged over 10 random seeds and 5 instances, and average time s in seconds. * shows significant difference between the 2 best methods (paired t-test, $\alpha = 0.05$).

References

1. Amaran, S., Sahinidis, N.V., Sharda, B., Bury, S.J.: Simulation optimization: a review of algorithms and applications. Annals of Operations Research **240**, 351–380, (2016). https://doi.org/10.1007/s10479-015-2019-x
2. Ball, R.C., Branke, J., Meisel, S.: Optimal sampling for simulated annealing under noise. INFORMS Journal on Computing **30**, 200–215, (2018). https://doi.org/10.1287/ijoc.2017.0774
3. Ballestín, F.: When it is worthwhile to work with the stochastic RCPSP? Journal of Scheduling **10**, 153–166, (2007). https://doi.org/10.1007/s10951-007-0012-1
4. Ballestín, F., Leus, R.: Resource-constrained project scheduling for timely project completion with stochastic activity durations. Production and Operations Management **18**, 459–474, (2009). https://doi.org/10.3401/poms.1080.01023
5. Bartz-Beielstein, T., Blum, D., Branke, J.: Particle swarm optimization and sequential sampling in noisy environments. In: Metaheuristics: Progress in Complex Systems Optimization. pp. 261–273. Springer, (2007). https://doi.org/10.1007/978-0-387-71921-4_14
6. Bellman, R.: Dynamic programming. Science **153**(3731), 34–37, (1966). https://doi.org/10.1126/science.153.3731.34
7. Bianchi, L., Dorigo, M., Gambardella, L.M., Gutjahr, W.J.: A survey on metaheuristics for stochastic combinatorial optimization. Natural Computing **8**, 239–287, (2009). https://doi.org/10.1007/s11047-008-9098-4
8. Birge, J.R., Louveaux, F.: Introduction to Stochastic Programming. Springer, (2011). https://doi.org/10.1007/978-1-4614-0237-4
9. Bouneffouf, D., Rish, I., Aggarwal, C.: Survey on applications of multi-armed and contextual bandits. In: 2020 IEEE Congress on Evolutionary Computation (CEC). pp. 1–8. (2020). https://doi.org/10.1109/CEC48606.2020.9185782
10. Bulgak, A.A., Sanders, J.L.: Integrating a modified simulated annealing algorithm with the simulation of a manufacturing system to optimize buffer sizes in automatic assembly systems. In: 1988 Winter Simulation Conference Proceedings. pp. 684–690. (1988). https://doi.org/10.1109/WSC.1988.716241
11. Chen, Z., Demeulemeester, E., Bai, S., Guo, Y.: Efficient priority rules for the stochastic resource-constrained project scheduling problem. European Journal of Operational Research **270**, 957–967, (2018). https://doi.org/10.1016/j.ejor.2018.04.025
12. Dumouchelle, J., Julien, E., Kurtz, J., Khalil, E.B.: Neur2ro: Neural two-stage robust optimization. arXiv preprint (2023). https://doi.org/10.48550/ARXIV.2310.04345
13. Gelman, A., Carlin, J.B., Stern, H.S., Rubin, D.B.: Bayesian Data Analysis. Chapman and Hall/CRC, (1995). https://doi.org/10.1201/9780429258411
14. Groves, M., Branke, J.: Sequential sampling for noisy optimisation with cma-es. In: Proceedings of the 2018 Genetic and Evolutionary Computation Conference. pp. 1023–1030. Association for Computing Machinery, Inc, (2018). https://doi.org/10.1145/3205455.3205559
15. Juan, A.A., Faulin, J., Grasman, S.E., Rabe, M., Figueira, G.: A review of simheuristics: Extending metaheuristics to deal with stochastic combinatorial optimization problems. Operations Research Perspectives **2**, 62–72, (2015). https://doi.org/10.1016/j.orp.2015.03.001
16. Juan, A.A., Keenan, P., Martí, R., McGarraghy, S., Panadero, J., Carroll, P., Oliva, D.: A review of the role of heuristics in stochastic optimisation: from metaheuristics to learnheuristics. Annals of Operations Research (2021). https://doi.org/10.1007/s10479-021-04142-9

17. Kirkpatrick, S., Gelatt, C.D., Vecchi, M.P.: Optimization by simulated annealing. Science **220**, 671–680, (1983). https://doi.org/10.1126/science.220.4598.671
18. Kleywegt, A.J., Shapiro, A., Homem-de-mello, T.: The sample average approximation method for stochastic discrete optimization. Society for Industrial and Applied Mathematics **12**, 479–502, (2001). https://doi.org/10.1137/S1052623499363220
19. Kolisch, Sprecher: Psplib a project scheduling problem library. European Journal of Operational Research **96**, 205–216, (1996). https://doi.org/10.1016/S0377-2217(96)00170-1
20. Kolisch, R., Hartmann, S.: Heuristic algorithms for solving the resource-constrained project scheduling problem: Classification and computational analysis. In: Project Scheduling. International Series in Operations Research & Management Science. vol. 14, pp. 147–178. Springer, (1999). https://doi.org/10.1007/978-1-4615-5533-9_7
21. Lattimore, T., Szepesvári, C.: Bandit algorithms. Cambridge University Press, (2020). https://doi.org/10.1017/9781108571401
22. Lei, H., Laporte, G., Guo, B.: The capacitated vehicle routing problem with stochastic demands and time windows. Computers and Operations Research **38**, 1775–1783, (2011). https://doi.org/10.1016/j.cor.2011.02.007
23. Liu, N., Truong, V.A., Wang, X., Anderson, B.R.: Integrated scheduling and capacity planning with considerations for patients' length-of-stays. Production and Operations Management **28**, 1735–1756, (2019). https://doi.org/10.1111/poms.13012
24. Min, D., Yih, Y.: Scheduling elective surgery under uncertainty and downstream capacity constraints. European Journal of Operational Research **206**, 642–652, (2010). https://doi.org/10.1016/j.ejor.2010.03.014
25. Prudius, A.A., Andradóttir, S.: Averaging frameworks for simulation optimization with applications to simulated annealing. Naval Research Logistics **59**, 411–429, (2012). https://doi.org/10.1002/nav.21496
26. Rahimi, I., Gandomi, A.H.: A comprehensive review and analysis of operating room and surgery scheduling. Archives of Computational Methods in Engineering **28**, 1667–1688, (2021). https://doi.org/10.1007/s11831-020-09432-2
27. Ritzinger, U., Puchinger, J., Hartl, R.F.: A survey on dynamic and stochastic vehicle routing problems. International Journal of Production Research **54**, 215–231, (2016). https://doi.org/10.1080/00207543.2015.1043403
28. Rostami, S., Creemers, S., Leus, R.: New strategies for stochastic resource-constrained project scheduling. Journal of Scheduling **21**, 349–365, (2018). https://doi.org/10.1007/s10951-016-0505-x
29. Saliby, E.: Descriptive sampling: A better approach to Monte Carlo simulation. Source: The Journal of the Operational Research Society **41**, 1133–1142, (1990). https://doi.org/10.2307/2583110
30. Schutte, N.: Codebase experiments sequential predictive sampling, github.com/NoahJSchutte/sequential-predictive-sampling. Last accessed 29 Nov 2022
31. Schutte, N., van den Houten, K., Eigbe, E.: Dynamic scenario reduction for simulation based optimization under uncertainty, (2022), https://drive.google.com/file/d/1kxzgO8ZhW2bjXo1vwVskK4_5LNRbp5vj/view?usp=sharing. Last accessed 29 Nov 2022
32. Seyyedabbasi, A.: A reinforcement learning-based metaheuristic algorithm for solving global optimization problems. Advances in Engineering Software **178**, 103411, (2023). https://doi.org/10.1016/j.advengsoft.2023.103411
33. Shehadeh, K.S.: Data-driven distributionally robust surgery planning in flexible operating rooms over a wasserstein ambiguity. Computers and Operations Research **146**, (2022). https://doi.org/10.1016/j.cor.2022.105927

34. Shehadeh, K.S., Zuluaga, L.F.: 14th AIMMS-MOPTA optimization modeling competition: Surgery scheduling in flexible operating rooms under uncertainty, (2022), https://iccopt2022.lehigh.edu/competition-and-prizes/aimms-mopta-competition/. Last accessed 29 Nov 2022
35. Solomon, M.M.: Algorithms for the vehicle routing and scheduling problems with time window constraints. Operations Research **35**, 254–265, (1987). https://doi.org/10.1287/opre.35.2.254
36. Vajda, S.: Mathematical programming. Courier Corporation, (2009)
37. Zakaria, A., Ismail, F.B., Lipu, M.S., Hannan, M.A.: Uncertainty models for stochastic optimization in renewable energy applications. Renewable Energy **145**, 1543–1571, (2020). https://doi.org/10.1016/j.renene.2019.07.081
38. Zhu, S., Fan, W., Yang, S., Pei, J., Pardalos, P.M.: Operating room planning and surgical case scheduling: a review of literature. Journal of Combinatorial Optimization **37**, 757–805, (2019). https://doi.org/10.1007/s10878-018-0322-6