Journal of Decision Systems

ISSN: 1246-0125 (Print) 2116-7052 (Online) Journal homepage: http://www.tandfonline.com/loi/tjds20

Stochastic decision optimisation based on deterministic approximations of processes described as closed-form arithmetic simulation

Mohan Krishnamoorthy, Alexander Brodsky & Daniel A. Menascé

To cite this article: Mohan Krishnamoorthy, Alexander Brodsky & Daniel A. Menascé (2018): Stochastic decision optimisation based on deterministic approximations of processes described as closed-form arithmetic simulation, Journal of Decision Systems, DOI: 10.1080/12460125.2018.1468174

To link to this article: https://doi.org/10.1080/12460125.2018.1468174

Published online: 07 May 2018.

JOURNAL OF DECISION SYSTEMS, 2018
https://doi.org/10.1080/12460125.2018.1468174

Taylor & Francis
Taylor & Francis Group

Stochastic decision optimisation based on deterministic approximations of processes described as closed-form arithmetic simulation

Mohan Krishnamoorthy, Alexander Brodsky and Daniel A. Menascé

Department of Computer Science, George Mason University, Fairfax, VA, USA

ABSTRACT

We propose an efficient one-stage stochastic optimisation algorithm for the problem of finding process controls that minimise the expectation of cost while satisfying multiple deterministic and stochastic feasibility constraints with a given high probability. The proposed algorithm is based on a series of deterministic approximations to produce a candidate solution set and on a refinement step using stochastic simulations with optimal simulation budget allocation. We conduct an experimental study for a real-world manufacturing service network, which shows that the proposed algorithm significantly outperforms four popular simulation-based stochastic optimisation algorithms.

ARTICLE HISTORY
Received 9 January 2018
Accepted 11 April 2018

KEYWORDS
Decision support; decision guidance; deterministic approximations; stochastic simulation optimisation; heuristic algorithm

1. Introduction

In this paper, we consider processes with feasibility constraints and metrics of interest, including process cost, that are stochastic functions of process controls. We focus on the development of a one-stage stochastic optimisation algorithm for the problem of finding process controls that minimise the expectation of cost while satisfying multiple deterministic and stochastic feasibility constraints with a given high probability. These problems are prevalent in many domains including manufacturing, supply chain management, energy and power.

One-stage stochastic optimisation have been extensively studied, e.g. see overviews Figueira and Almada-Lobo (2014) and Juan et al. (2015), which outline key techniques. They typically are based on simulation-based optimisation, e.g. see Amaran, Sahinidis, Sharda, and Bury (2016) and Nguyen, Reiter, and Rigo (2014).

However, the general limitation of simulation-based approaches is that simulation is used as a black box, and the underlying mathematical structure is not utilised. In Mathematical Programming (MP) applied to deterministic optimisation problems, utilising the mathematical structure of the problem typically gives significantly better results in terms of optimality and computational time compared to simulation-based approaches (e.g. see Amaran et al., 2016). For this reason, a number of approaches have been developed to bridge the gap between stochastic simulation and MP, by extracting mathematical structure

CONTACT Mohan Krishnamoorthy mkrishn4@gmu.edu

through simulation. For example, in Thompson and Davis (1990), random variables in a simulation are resolved to their means; in Paraskevopoulos, Karakitsos, and Rustem (1991), to solve the optimal capacity planning problem, the original objective function is augmented with a penalty on the sensitivity of the objective function to various types of uncertainty. However, extraction of the mathematical structure through sampling using a black-box simulation is computationally expensive, especially for real-world processes composed of complex process networks.

Instead of extracting mathematical structure using simulation, in Krishnamoorthy, Brodsky, and Menascé (2015), we used symbolic analysis of simulation model code as part of a stochastic optimisation algorithm for temporal production processes. In Krishnamoorthy, Brodsky, and Menascé (2017), we generalised the approach from Krishnamoorthy et al. (2015) to general stochastic optimisation problems where both the objective and the demand constraint are stochastic and may involve non-linear arithmetic. However, the proposed algorithm was limited to a single, as opposed to multiple, stochastic demand constraints. Lifting this limitation is the focus of this paper.

More specifically, the contributions of this paper are two-fold: First, we propose a heuristic algorithm called general Stochastic Optimisation Algorithm based on Deterministic Approximations (g-SODA). The proposed algorithm is based on (1) a series of deterministic approximations to produce a candidate set of near-optimal control settings for the process, and (2) stochastic simulations on the candidate set using optimal simulation budget allocation methods (e.g. see Chen & Lee, 2011; Lee, Pujowidianto, Li, Chen, & Yap, 2012). Second, we conduct an initial experimental study over a real world use case of a heat-sink service network to compare the proposed algorithm with four popular simulation-based stochastic optimisation algorithms viz., Nondominated Sorting Genetic Algorithm 2 (NSGA2) (Deb, Pratap, Agarwal, & Meyarivan, 2002), Indicator Based Evolutionary Algorithm (IBEA) (Zitzler & Künzli, 2004), Strength Pareto Evolutionary Algorithm 2 (SPEA2) (Zitzler, Laumanns, & Thiele, 2001) and Speed-constrained Multi-objective Particle swarm optimisation (SMPSO) (Nebro et al., 2009). The experimental study demonstrates that g-SODA significantly outperforms the other algorithms in terms of optimality of results and computation time. In particular, running over a 12-process problem with 22 decision variables and 21 stochastic constraints using a 8-core server with 16GB RAM, g-SODA achieves a production cost lower than that of competing algorithms by 42% in one day and 25% better cost in three days.

The rest of this paper is organised as follows. Section 2 formally describes the stochastic optimisation problem over processes with multiple stochastic feasibility constraints. The algorithm for g-SODA, including deterministic approximations, is presented in Section 3. Section 4 describes the experimental study to evaluate g-SODA's performance. Finally, Section 5 concludes with some future research directions.

2. Stochastic optimisation over processes with multiple stochastic feasibility constraints and closed-form non-linear arithmetic

We now borrow the stochastic optimisation problem from Krishnamoorthy et al. (2017) and extend it for processes that have feasibility constraints over multiple stochastic metrics and are described using non-linear arithmetic. The stochastic optimisation problem for such processes assumes a stochastic closed-form arithmetic (SCFA) simulation of the following

form. A SCFA simulation on input variable $\vec{X}$ is a sequence $(y_1 = expr_1), \ldots, (y_n = expr_n)$ where $expr_i, 1 \leq i \leq n$ is either

(a) An arithmetic or Boolean expression in terms of a subset of the elements of $\vec{X}$ and/or $y_1, \ldots, y_{i-1}$. We say that y_i is arithmetic or Boolean if $expr_i$ is arithmetic or Boolean correspondingly.
(b) An expression invoking $PD(\vec{P})$, which is a function that draws from a probability distribution (e.g. Gaussian, exponential and uniform) using parameters $\vec{P}$ that are a subset of the elements of $\vec{X}$ and/or $y_1, \ldots, y_{i-1}$.

We say that $y_i, 1 \leq i \leq n$ is stochastic if, recursively,

(a) $expr_i$ invokes $PD(\vec{P})$, or
(b) $expr_i$ uses at least one stochastic variable $y_j, 1 \leq j < i$

If y_i is not stochastic, we say that it is deterministic. Also, we say that a SCFA simulation $\mathbb{S}$ computes a variable v if $v = y_i$, for some $1 \leq i \leq n$.

To clarify, consider a simple SCFA simulation example, $\mathbb{S}_{simple}$ that consists of the following sequence of expressions:

```
1: stochSpeed := speed + 𝒩(0,σ)
2: stochTime := f(stochSpeed)
3: throughput := 1/stochTime
4: cost := throughput × pricePerUnit
5: C := lb ≤ speed ≤ ub
```

In this example, the SCFA simulation computes the stochastic arithmetic variables of *cost* and *throughput* as well as the deterministic Boolean variable *C*. The variable *speed* is deterministic (e.g. machine speed), and it should be bounded within some lower bound (*lb*) and upper bound (*ub*). The Boolean variable *C* describes whether *speed* is bounded. Also, the effects of *speed* is stochastic (*stochSpeed*) due to normally distributed random noise $\mathcal{N}(0, \sigma)$. For the sake of simplicity, assume that the stochastic time to produce one item (*stochTime*) is obtained from a function described in terms of *stochSpeed*. Then, *throughput* is computed as the inverse of *stochTime* and *cost* is computed as the product of *throughput* and a fixed parameter of price to produce one unit of item (*pricePerUnit*).

This paper considers the stochastic optimisation problem of finding process controls that minimise the cost expectation while satisfying multiple deterministic and stochastic feasibility constraints with a given probability. More formally, the stochastic optimisation problem is of the form:

$$\begin{aligned} \underset{\vec{X} \in \vec{D}}{\text{minimise}} \quad & E(cost(\vec{X})) \\ \text{subject to} \quad & C(\vec{X}) \wedge \\ & \forall_{i \in \{1,\ldots,k\}} P(m_i(\vec{X}) \geq \theta_i) \geq \alpha_i, \end{aligned} \tag{1}$$

where $\vec{D} = D_1 \times \cdots \times D_n$ is the domain for decision variables $\vec{X}$
$\vec{X}$ is a vector of decision variables that range over $\vec{D}$
$cost(\vec{X})$ is a random variable defined in terms of $\vec{X}$
$C(\vec{X})$ is a deterministic constraint in $\vec{X}$ i.e. a function $C : \vec{D} \rightarrow \{true, false\}$

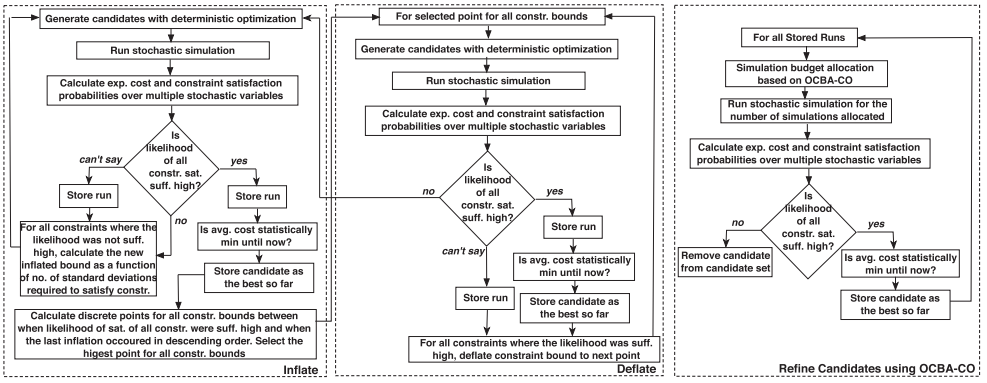

Figure 1. Overview of g-SODA.

$m_1(\vec{X}), \ldots, m_k(\vec{X})$ are random variables defined in terms of $\vec{X}$
$\theta_1, \ldots, \theta_k \in \mathbb{R}^k$ are the constraint thresholds
$\alpha_1, \ldots, \alpha_k \in [0,1]^k$ are the probability thresholds, and
$P(m_i(\vec{X}) \geq \theta_i)$ is the probability that $m_i(\vec{X}) \geq \theta_i$, and $i \in \{1, \ldots, k\}$

We assume that the random variables, $cost(\vec{X})$, $m_1(\vec{X}), \ldots, m_k(\vec{X})$ as well as the deterministic constraint $C(\vec{X})$ are expressed by an SCFA simulation $\mathbb{S}$ that computes the corresponding stochastic arithmetic variables *cost*, $m_1, \ldots, m_k$ as well as the deterministic Boolean variable $C \in \{true, false\}$.

For example, in $\mathbb{S}_{simple}$, *speed* is the decision variable and the stochastic metrics are *stochTime*, *throughput* and *cost*. Then the stochastic optimisation problem for $\mathbb{S}_{simple}$ is to minimise the objective of *E(cost)* subject to the satisfaction of the deterministic constraint *C* and *throughput* is higher than some bound (θ) with a given probability (α).

3. General optimisation algorithm over stochastic closed-form arithmetic simulations

This section presents the general Stochastic Optimisation Algorithm based on Deterministic Approximations (g-SODA) over the SCFA simulations as described in Section 2. This stochastic optimisation problem can be solved using simulation-based optimisation approaches by initialising the control settings and performing simulations to check whether the stochastic constraints are satisfied with sufficient probability. But such an approach is inefficient because the stochastic space of this problem is very large and hence this approach will typically converge very slowly to the optimum solution. So, the key idea of g-SODA is that instead of working with a large number of choices in the stochastic space, we use deterministic approximations to generate a small set of candidate control settings and then validate these control settings in the stochastic space using simulations.

An overview of g-SODA is shown in Figure 1. To generate a small set of candidate control settings, g-SODA performs deterministic approximations of the original stochastic problem. To achieve this, g-SODA defines a deterministic computation $\mathbb{S}_0$ from the SCFA simulation $\mathbb{S}$ described in Section 2 by replacing every expression that uses a probability distribution $PD(\vec{P})$ with the expectation of that distribution. Thus, the deterministic approximations $cost_0(\vec{X}), m_{0,1}(\vec{X}), \ldots, m_{0,k}(\vec{X})$ of $cost(\vec{X}), m_1(\vec{X}), \ldots, m_k(\vec{X})$, respectively, can be expressed

using $\mathbb{S}_0$. To optimise this reduced problem, a deterministic optimisation problem that approximates the stochastic optimisation problem shown in Equation (1) is used as a heuristic. This deterministic optimisation problem can be described as follows:

$$\begin{aligned} \underset{\vec{X}\in\vec{D}}{\text{minimise}} \quad & cost_0(\vec{X}) \\ \text{subject to} \quad & C(\vec{X}) \wedge \\ & m_{0,1}(\vec{X}) \geq \theta_1{}' \wedge \cdots \wedge m_{0,k}(\vec{X}) \geq \theta_k{}', \end{aligned} \tag{2}$$

where $\theta_1{}' \geq \theta_1, \ldots, \theta_k{}' \geq \theta_k$ are conservative approximations of $\theta_1, \ldots, \theta_k$.

This deterministic approximation is performed iteratively such that the control settings found in the current iteration are more likely to satisfy the stochastic constraints with the desired probability than in the previous iterations. This is possible because of the *inflate* phase (left box of Figure 1) and the *deflate* phase (middle box of Figure 1) of g-SODA.

The *inflate* phase is intuitively trying to increase the bounds to satisfy the stochastic constraints with the desired probability. When the current candidate control settings do not generate any metric that satisfies its respective user-defined bound with a desired probability, this bound parameter is artificially inflated as a function of the number of standard deviations required so that the metric satisfies the bound with desired probability. This inflation very quickly yields metrics that satisfy its respective user-defined bound. However, this may result in the metric overshooting the bound, which degrades the objective cost. For example, in $\mathbb{S}_{simple}$, if *throughput* does not satisfy θ with probability (α), then the inflate phase performs a series of deterministic approximations using θ', which is an inflated value of θ in an effort to find *speed* that produces *throughput* that satisfies θ with probability α. However, this may result in a *throughput* that is much greater than θ, which degrades *cost*.

To overcome this, in the *deflate* phase, g-SODA scales the inflated bounds back by calculating the discrete points for all bounds between when the constraint was satisfied with sufficient probability and when the last inflation occurred. Deterministic approximations are run for this lower bound to check whether the updated metric still satisfies the user-defined bound with desired probability while yielding a better objective cost. For example, in $\mathbb{S}_{simple}$, θ' is scaled back iteratively so long as the deterministic approximations using the new θ' yields a *speed* that produces *throughput* that satisfies θ with probability α. In this way, the *inflate* and *deflate* phases find a more optimum bound threshold for all stochastic constraints for which it can get the right balance between optimum cost and stochastic constraint satisfaction with desired probability.

After the iterative *inflate* and *deflate* procedure, more simulations may be needed to check if a promising candidate that currently is not a top candidate could be the optimal solution or to choose an optimal solution from multiple candidates. To resolve this, these candidates are further refined in the *refineCandidates* phase (right box in Figure 1) where a number of simulations are allocated to each candidate as obtained from an optimal simulation budget allocation method. These simulations are run on these candidates to check if this produces a new optimal solution. For example, in $\mathbb{S}_{simple}$, more simulations are run on all the candidates collected from inflate and deflate phases i.e. *speed* that yielded *throughput* satisfying θ with probability α, to check whether they can be refined further to produce a new optimal solution.

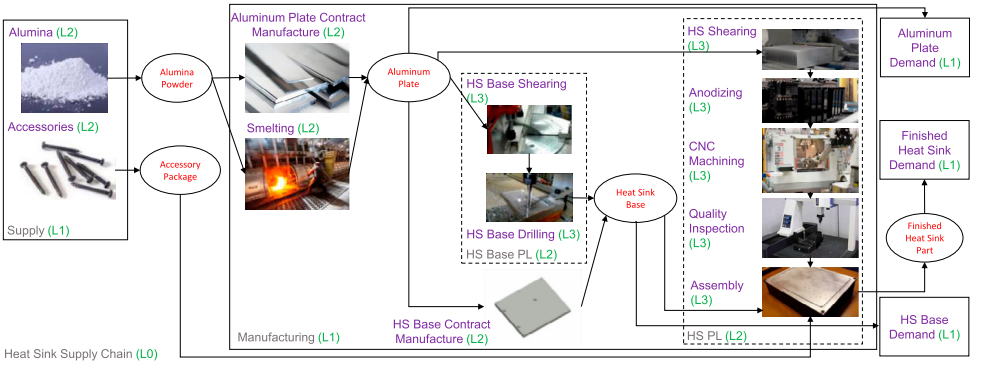

Figure 2. Graphical representation of the Heat-Sink Production Process (HSPP).

4. Experimental study

This section describes the experimental study that was conducted to evaluate g-SODA. In this study, a real-world use-case of the Heat Sink Production Process (HSPP) is considered. The graphical representation for the HSPP is shown in Figure 2. Within HSPP, each activity and physical good is represented by an image and a labelled circle, respectively. Vendors can provide raw material, e.g. *Alumina Powder*. As an example of a contract manufacturer, the *HS Base Contract Manufacture* provides the service of machining a part for the product's final assembly. Within service networks, it is likely that there a multiple paths to successfully provide the same physical good. For instance, the *Heat Sink Base* can either be provided by a contract manufacturer or it can be provided by the OEM's own production line, shown in the dotted box in the middle of the diagram. This production line includes two unit manufacturing processes, namely shearing and drilling. The Heat Sink's service network culminates in a relatively complex production line that includes five activities, namely shearing, anodising, CNC machining, quality inspection and final assembly. Finally, the three demand processes shown on the right in Figure 2 are virtual processes that dictate the minimum number of *Aluminium Plate*, *Finished Heat Sink Part*, and *Heat Sink Base* to be produced to satisfy the customer demand.

To conduct the experimental study, first the SCFA simulation of HSPP is written in JSONiq. An overview of the SCFA simulation for a production process in given in Krishnamoorthy et al. (2017). The code for g-SODA is written in Java. For the comparison algorithms, we used the jMetal package, which is an object-oriented Java-based framework for optimisation with metaheuristics (Durillo & Nebro, 2011). The algorithms chosen for comparison include Nondominated Sorting Genetic Algorithm 2 (NSGA2) (Deb et al., 2002), Indicator Based Evolutionary Algorithm (IBEA) (Zitzler & Künzli, 2004), Strength Pareto Evolutionary Algorithm 2 (SPEA2) (Zitzler et al., 2001) and Speed-constrained Multi-objective Particle swarm optimisation (SMPSO) (Nebro et al., 2009). These are popular multi-objective simulation-based optimisation algorithms, and they were chosen because we found that these algorithms performed better over one-dimensional SODA than the other single- and multi-objective simulation-based optimisation algorithms available in jMetal. The swarm/population size for the selected algorithms was set to 100. Also, we ran these algorithms with the following operators (wherever applicable): (a) polynomial mutation operator with probability of $(noOfDecisionVariables)^{-1}$ and distribution index of 20; (b) simulated binary crossover

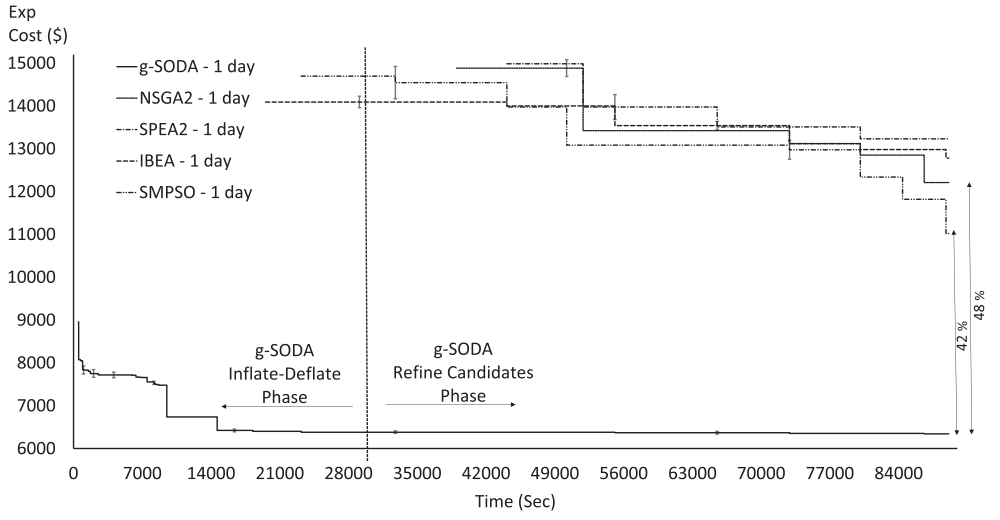

Figure 3. Estimated average cost for the elapsed time (max runtime = one day) (x-axis in normal scale).

operator with probability of 0.9 and distribution index of 20 and (c) binary tournament selection operator. For each algorithm, the maximum number of evaluations was set to 500,000 and the maximum time to complete these evaluations was set to the time that g-SODA ran for.

We ran g-SODA in two different settings: one day with 5000 iterations of *InflateDeflate* and 72 hours (three days) with 10,000 iterations of *InflateDeflate*. There were 22 decision variables and 21 stochastic constraints in the stochastic optimisation problem over HSPP. The probability bound and the confidence level for all stochastic constraints were set to 0.95 and 99%, respectively. Finally, the demand for aluminium plate, finished Heat Sink and Heat Sink Base were set to 8, 4 and 6 completed products, respectively. The comparison algorithms were also run with the same (relevant) parameters.

The data collected from the experiments include the estimated average costs achieved at different elapsed time points for g-SODA in the two settings and all the comparison algorithms. Figure 3 shows the estimated average costs achieved after one day, whereas Figure 4 shows the estimated average costs achieved after about three days. Each experiment was run multiple times and 95% confidence bars are included around the mean at certain elapsed time point in both figures.

It can be observed that both settings of g-SODA perform better than the comparison algorithms initially. In fact, g-SODA finds the first feasible solution with expected cost of about $9000 in 610 sec (10 min), whereas the first feasible solution with expected cost of $14,000 is only found by the competing algorithms after about 19,500 sec (five hours). As time progresses, g-SODA in both settings continues to find feasible solution with reduced expected cost in the *InflateDeflate* phase itself. In particular, the first three significant drops in objective cost from $9000 to $6700 comes after about 921 sec (15 min), 9509 sec (2.6 hours) and 14638 sec (four hours). During these times, the competing algorithms have not yet found a feasible solution. At the end of the experiment, the feasible solutions found by g-SODA have much better expected cost than those found by the competing algorithms. After one day, the expected cost of the solutions found by g-SODA was 42% better than the

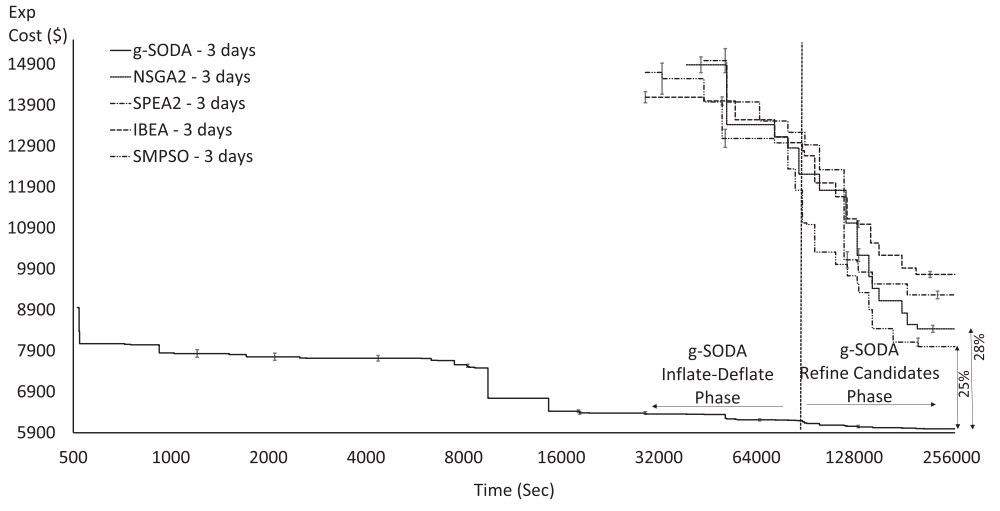

Figure 4. Estimated average cost for the elapsed time (max runtime = three days) (x-axis in log-scale).

nearest comparison algorithm (SMPSO) (see Figure 3). After three days, the advantage for g-SODA over the nearest comparison algorithm (SMPSO) reduces to 25% and that to the second best algorithm (NSGA2) is 28% (see Figure 4).

5. Conclusion and future work

We proposed an efficient one-stage stochastic optimisation algorithm called g-SODA for the problem of finding process controls that minimise the expectation of cost while satisfying multiple deterministic and stochastic feasibility constraints with a given high probability, and conducted an experimental study to demonstrate that g-SODA considerably outperforms the best simulation-based algorithms.

Future research direction include (1) comparing g-SODA with additional algorithms, including Bayesian optimisation, and (2) studying how evolutionary and other local search algorithms can be leveraged to improve the selection and refinement phase of g-SODA.

Disclosure statement

No potential conflict of interest was reported by the authors.

Funding

This work was partially supported by National Institute of Standards and Technology [grant number 70NANB12H277].

ORCID

Mohan Krishnamoorthy http://orcid.org/0000-0002-0828-4066
Alexander Brodsky http://orcid.org/0000-0002-0312-2105
Daniel A. Menascé http://orcid.org/0000-0002-4085-6212

References

Amaran, S., Sahinidis, N. V., Sharda, B., & Bury, S. J. (2016). Simulation optimization: a review of algorithms and applications. *Annals of Operations Research, 240*(1), 351–380.

Chen, C. H., & Lee, L. H. (2011). *Stochastic Simulation Optimization: An Optimal Computing Budget Allocation*. Hackensack, NJ: World Scientific Publishing Company.

Deb, K., Pratap, A., Agarwal, S., & Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation, 6*(2), 182–197.

Durillo, J. J., & Nebro, A. J. (2011). jMetal: A Java framework for multi-objective optimization. *Advances in Engineering Software, 42*(10), 760–771.

Figueira, G., & Almada-Lobo, B. (2014). Hybrid simulation-optimization methods: A taxonomy and discussion. *Simulation Modelling Practice and Theory, 46*, 118–134.

Juan, A. A., Faulin, J., Grasman, S. E., Rabe, M., & Figueira, G. (2015). A review of simheuristics: Extending metaheuristics to deal with stochastic combinatorial optimization problems. *Operations Research Perspectives, 2*, 62–72.

Krishnamoorthy, M., Brodsky, A., & Menascé, D. (2015). Optimizing stochastic temporal manufacturing processes with inventories: An efficient heuristic algorithm based on deterministic approximations. In *Proceedings of the 14th INFORMS Computing Society Conference* (pp. 30–46). Richmond, VA, USA.

Krishnamoorthy, M., Brodsky, A., & Menascé, D. A. (2017). *Stochastic optimization for steady state production processes based on deterministic approximations* (Technical Report GMU-CS-TR-2017-3). Department of Computer Science, George Mason University, 4400 University Drive MSN 4A5, Fairfax, VA 22030–4444 USA.

Lee, L. H., Pujowidianto, N. A., Li, L. W., Chen, C. H., & Yap, C. M. (2012). Approximate simulation budget allocation for selecting the best design in the presence of stochastic constraints. *IEEE Transactions on Automatic Control, 57*(11), 2940–2945.

Nebro, A., Durillo, J., García-Nieto, J., Coello Coello, C., Luna, F., & Alba, E. (2009). Smpso: A new pso-based metaheuristic for multi-objective optimization. In *2009 IEEE Symposium on Computational Intelligence in Multicriteria Decision-Making (MCDM 2009)* (pp. 66-73). Nashville, TN, USA: IEEE Press.

Nguyen, A.-T., Reiter, S., & Rigo, P. (2014). A review on simulation-based optimization methods applied to building performance analysis. *Applied Energy, 113*, 1043–1058.

Paraskevopoulos, D., Karakitsos, E., & Rustem, B. (1991). Robust Capacity Planning under Uncertainty. *Management Science, 37*(7), 787–800.

Thompson, S. D., & Davis, W. J. (1990). An integrated approach for modeling uncertainty in aggregate production planning. *IEEE Transactions on Systems, Man, and Cybernetics, 20*(5), 1000–1012.

Zitzler, E., & Künzli, S. (2004). Indicator-based selection in multiobjective search. In *Proceedings of the 8th International Conference on Parallel Problem Solving from Nature, 2004* (pp. 832–842). Berlin, Heidelberg: Springer.

Zitzler, E., Laumanns, M., & Thiele, L. (2001). *SPEA2: Improving the strength pareto evolutionary algorithm* (Technical Report 103). Zurich, Switzerland: Computer Engineering and Networks Laboratory (TIK), Swiss Federal Institute of Technology (ETH).