

# Stochastic Optimization Algorithm based on Deterministic Approximations

Mohan Krishnamoorthy<sup>1</sup> <sup>a</sup>, Alexander Brodsky<sup>2</sup> <sup>b</sup> and Daniel A. Menascé<sup>2</sup> <sup>c</sup>

<sup>1</sup>Mathematics and Computer Science Division, Argonne National Laboratory, Lemont, IL 60439, U.S.A.

<sup>2</sup>Department of Computer Science, George Mason University, Fairfax, VA 22030, U.S.A.

**Keywords:** Decision Support, Decision Guidance, Deterministic Approximations, Stochastic Simulation Optimization, Heuristic Algorithm.


**Abstract:** We consider steady-state production processes that have feasibility constraints and metrics of cost and throughput that are stochastic functions of process controls. We propose an efficient stochastic optimization algorithm for the problem of finding process controls that minimize the expectation of cost while satisfying deterministic feasibility constraints and stochastic steady state demand for the output product with a given high probability. The proposed algorithm is based on (1) a series of deterministic approximations to produce a candidate set of near-optimal control settings for the production process, and (2) stochastic simulations on the candidate set using optimal simulation budget allocation methods. We demonstrate the proposed algorithm on a use case of a real-world heat-sink production process that involves contract suppliers and manufacturers as well as unit manufacturing processes of shearing, milling, drilling, and machining, and conduct an experimental study that shows that the proposed algorithm significantly outperforms four popular simulation-based stochastic optimization algorithms.


## 1 INTRODUCTION


This paper considers steady-state processes that produce a discrete product and have feasibility constraints and metrics of cost and throughput that are stochastic functions of process controls. We are concerned with the development of a one-stage stochastic optimization algorithm for the problem of finding process controls that minimize the expectation of cost while satisfying deterministic feasibility constraints and stochastic steady state demand for the output product with a given high probability. These problems are prevalent in manufacturing processes, such as machining, assembly lines, and supply chain management. The problem tries to find process controls such as speed of the machines that not only satisfy the feasibility constraints of being within a constant capacity, but also satisfy the production demand. In the stochastic case, we want to find these process controls at the minimum expected cost and for demand that is satisfied with a high probability of 95%.

Stochastic optimization have typically been performed using simulation-based optimization techniques (see (Amaran et al., 2016) for a review of such techniques). Tools like SIMULINK (Dabney and Harman, 2001) and Modelica (Provan and Venturini, 2012) allow users to run stochastic simulations on models of complex systems in mechanical, hydraulic, thermal, control, and electrical power. Tools like OMOptim (Thieriot et al., 2011), Efficient Traceable Model-Based Dynamic Optimization (EDOp) (OpenModelica, 2009), and jMetal (Durillo and Nebro, 2011) use simulation models to heuristically guide a trial and error search for the optimal answer. The above approaches are limited because they use simulation as a black box instead of using the problem's underlying mathematical structure.

From the work on Mathematical Programming (MP), we know that, for deterministic problems, utilizing the mathematical structure can lead to significantly better results in terms of optimality of results and computational complexity compared to simulation-based approaches (e.g., (Amaran et al., 2016) and (Klemmt et al., 2009)). For this reason, a number of approaches have been developed to bridge the gap between stochastic simulation and MP. For

<sup>a</sup>  <https://orcid.org/0000-0002-0828-4066>

<sup>b</sup>  <https://orcid.org/0000-0002-0312-2105>

<sup>c</sup>  <https://orcid.org/0000-0002-4085-6212>

instance, (Thompson and Davis, 1990) propose an integrated approach that combines simulation with MP where the MP problem is constructed from the original stochastic problem with uncertainties being resolved to their mean values by using a sample of black-box simulations. This strategy of extracting an MP from the original problem is also used by (Paraskevopoulos et al., 1991) to solve the optimal capacity planning problem by incorporating the original objective function augmented with a penalty on the sensitivity of the objective function to various types of uncertainty. The authors of (Xu et al., 2016) propose an ordinal transformation framework, consisting of a two-stage optimization framework that first extracts a low fidelity model using simulation or a queuing network model using assumptions that simplify the original problem and then uses this model to reduce the search space over which high fidelity simulations are run to find the optimal solution to the original problem. However, extraction of the mathematical structure through sampling using a black-box simulation is computationally expensive, especially for real-world production processes composed of complex process networks.

In (Krishnamoorthy et al., 2015), instead of extracting the mathematical structure using black-box simulation, the mathematical structure is extracted from a white-box simulation code analysis as part of a heuristic algorithm to solve a stochastic optimization problem of finding controls for temporal production processes with inventories as to minimize the total cost while satisfying the stochastic demand with a predefined probability. Similar to the previous approaches, the mathematical structure is used for approximating a candidate set of solutions by solving a series of deterministic MP problems that approximate the stochastic simulation. However, the class of problems considered in (Krishnamoorthy et al., 2015) is limited to processes described using piece-wise linear arithmetic. In (Krishnamoorthy et al., 2018), the approach from (Krishnamoorthy et al., 2015) is generalized to solve stochastic optimization problems that may involve non-linear arithmetic with stochastic objective and multiple demand constraints, like those in temporal production processes. However, many production processes, especially in manufacturing, have models that are in steady-state and have only a single demand constraint that can be solved more easily. To close this gap, this paper specializes the approach from (Krishnamoorthy et al., 2018) for a single demand constraint to solve the stochastic optimization problems for steady-state production processes described using non-linear arithmetic.

More specifically, the contributions of this paper

are three-fold: (A) a specialized heuristic algorithm called Stochastic Optimization Algorithm Based on Deterministic Approximations (SODA) to solve the problem of finding production process controls that minimize the expectation of cost while satisfying the deterministic process feasibility constraints and stochastic steady state demand for the output product with a given high probability. The proposed algorithm is based on (1) a series of deterministic approximations to produce a candidate set of near-optimal control settings for the production process, and (2) stochastic simulations on the candidate set using optimal simulation budget allocation methods (e.g., (Chen et al., 2000), (Chen and Lee, 2011), (Lee et al., 2012)). (B) a demonstration of the proposed algorithm on a use case of a real-world heat-sink production process that involves 10 processes including contract suppliers and manufacturers as well as unit manufacturing processes of shearing, milling, drilling, and machining with models from the literature that use non-linear physics-based equations. (C) an initial experimental study using the heat-sink production process to compare the proposed algorithm with four popular simulation-based stochastic optimization algorithms viz., Nondominated Sorting Genetic Algorithm 2 (NSGA2) (Deb et al., 2002), Indicator Based Evolutionary Algorithm (IBEA) (Zitzler and Künzli, 2004), Strength Pareto Evolutionary Algorithm 2 (SPEA2) (Zitzler et al., 2001), and Speed-constrained Multi-objective Particle swarm optimization (SMPSO) (Nebro et al., 2009). The experimental study demonstrates that SODA significantly outperforms the other algorithms in terms of optimality of results and computation time. In particular, running over a 12-process problem using a 8-core server with 16GB RAM, in 40 minutes, SODA achieves a production cost lower than that of competing algorithms by 61%; in 16 hours SODA achieves 29% better cost; and, in 3 days it achieves 7% better cost.

The rest of this paper is organized as follows. Section 2 formally describes the stochastic optimization problem over steady-state production processes. SODA, including deterministic approximations, is presented in section 3. Experimental results are presented in section 4. Finally, section 5 concludes with some future research directions.

## 2 OPTIMIZATION OF PRODUCTION PROCESSES WITH SCFA

We now specialize the stochastic optimization problem from (Krishnamoorthy et al., 2018) for steady-state processes with a single feasibility constraint over a stochastic metric. The stochastic optimization 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  $y_1 = \text{expr}_1, \dots, y_n = \text{expr}_n$  where  $\text{expr}_i, 1 \leq i \leq n$  is either

- (a) An arithmetic or boolean expression in terms of a subset of the elements of  $\vec{X}$  and/or  $y_1, \dots, y_{i-1}$ . We say that  $y_i$  is arithmetic or boolean if the  $\text{expr}_i$  is arithmetic or boolean correspondingly.
- (b) An expression invoking  $PD(\vec{P})$ , a function that draws from a probability distribution using parameters  $\vec{P}$  that are a subset of the elements of  $\vec{X}$  and/or  $y_1, \dots, y_{i-1}$ .

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

- (a)  $\text{expr}_i$  invokes  $PD(\vec{P})$ , or
- (b)  $\text{expr}_i$  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  $1 \leq i \leq n$ .

This paper considers the stochastic optimization problem of finding process controls that minimize the cost expectation while satisfying deterministic process constraints and steady state demand for the output product with a given probability. More formally, the stochastic optimization problem is of the form:

$$\begin{aligned} & \text{minimize}_{\vec{x} \in \vec{D}} E(\text{cost}(\vec{X})) \\ & \text{subject to} \quad C(\vec{X}) \wedge \\ & \quad P(\text{thru}(\vec{X}) \geq \theta) \geq \alpha \end{aligned} \quad (1)$$

where  $\vec{D} = D_1 \times \dots \times D_n$  is the domain for decision variables  $\vec{X}$ ,  $\vec{X}$  is a vector of decision variables over  $\vec{D}$ ,  $\text{cost}(\vec{X})$  is a random variable defined in terms of  $\vec{X}$ ,  $\text{thru}(\vec{X})$  is a random variable defined in terms of  $\vec{X}$ ,  $C(\vec{X})$  is a deterministic constraint in  $\vec{X}$  i.e., a function  $C: \vec{D} \rightarrow \{\text{true}, \text{false}\}$ ,  $\theta \in \mathbb{R}$  is a throughput threshold,  $\alpha \in [0, 1]$  is a probability threshold, and  $P(\text{thru}(\vec{X}) \geq \theta)$  is the probability that  $\text{thru}(\vec{X})$  is greater than or equal to  $\theta$

Note in this problem that upon increasing  $\theta$  to some  $\theta'$ , the size of the space of alternatives that satisfy the stochastic demand constraint in (1) increases

and hence it can be said that the best solution, i.e., the minimum expected cost is monotonically improving in  $\theta'$ . We assume that the random variables,  $\text{cost}(\vec{X})$  and  $\text{thru}(\vec{X})$  as well as the deterministic constraint  $C(\vec{X})$  are expressed by an SCFA simulation  $\mathbb{S}$  that computes the stochastic arithmetic variable  $(\text{cost}, \text{thru}) \in \mathbb{R}^2$  as well as the deterministic boolean variable  $C \in \{\text{true}, \text{false}\}$ . Many complex real-world processes can be formulated as SCFA simulations as described in (Krishnamoorthy et al., 2017).

## 3 STOCHASTIC OPTIMIZATION ALGORITHM BASED ON DETERMINISTIC APPROXIMATIONS

This section presents the Stochastic Optimization Algorithm Based on Deterministic Approximations (SODA). The problem of optimizing stochastic production processes can be solved using simulation-based optimization approaches by initializing the control settings and performing simulations to check whether the throughput satisfies the demand with sufficient probability. But that 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 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 SODA is shown in Fig. 1. To generate a small set of candidate control settings, SODA performs deterministic approximations of the original stochastic problem by defining 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. The deterministic approximations  $\text{cost}_0(\vec{X})$  and  $\text{thru}_0(\vec{X})$  of  $\text{cost}(\vec{X})$  and  $\text{thru}(\vec{X})$ , respectively, can be expressed using  $\mathbb{S}_0$ . To optimize this reduced problem, a deterministic optimization problem (see (2)) approximates the stochastic optimization problem of (1) is used as a heuristic.

$$\begin{aligned} & \text{minimize}_{\vec{x} \in \vec{D}} \text{cost}_0(\vec{X}) \\ & \text{subject to} \quad C(\vec{X}) \wedge \\ & \quad \text{thru}_0(\vec{X}) \geq \theta' \end{aligned} \quad (2)$$

where  $\theta' \geq \theta$  is a conservative approximation of  $\theta$ .

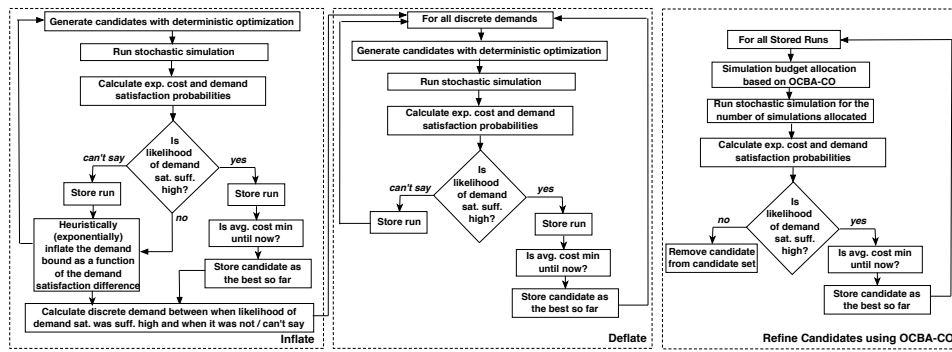


Figure 1: Overview of SODA.

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

Intuitively, the *inflate* phase tries to increase the throughput to satisfy the demand with the desired probability. When the current candidate control settings do not generate the throughput that satisfies the original user-defined demand with a desired probability, the demand parameter itself is exponentially inflated such that this inflation does not render future iterations infeasible. This is ensured by precomputing the feasible throughput range, which is the interval between the requested demand and the throughput bound where the production process is at capacity. This bound is computed by solving the optimization problem in (3). Then, the demand parameter is inflated within the throughput range. This inflation of the demand parameter yields higher controls for the machines and thus increases the overall throughput. However, this may result in the throughput overshooting the demand in the stochastic setting, which degrades the objective cost.

$$\begin{aligned}
 & \underset{\vec{X} \in \vec{D}}{\text{maximize}} && thru(\vec{X}) \\
 & \text{subject to} && C(\vec{X})
 \end{aligned} \tag{3}$$

To overcome this, in the *deflate* phase, SODA scales back the demand by splitting it into a number of points, separated by a small epsilon, in the interval between the inflated demand where throughput overshoot the original demand and the previous demand at which the last inflation occurred. Deterministic approximations are run for this lower demand to check whether the throughput still satisfies the demand with desired probability while yielding a better objective cost. In this way, the *inflate* and *deflate* phases find a better demand threshold for which it can get the right

balance between optimum cost and demand 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 Fig. 1). In this phase, SODA refines the candidates in the candidate set generated using deterministic approximation by running Monte Carlo simulations on them. To maximize the likelihood of selecting the best candidate, i.e., candidate with the least cost and sufficient probability of demand satisfaction, this phase uses the Optimal Computing Budget Allocation method for Constraint Optimization (OCBA-CO) (Lee et al., 2012) for the ranking and selection problem with stochastic constraints. OCBA-CO allocates budget among the candidates so that the greatest fraction of the budget is allocated to the most critical candidates in the candidate set. This apportions the computational budget in a way that minimizes the effort of simulating candidates that are not critical or those that have low variability. This phase is performed iteratively for some delta budget that is allocated among the candidates using OCBA-CO and, in each iteration, the additional simulations on the candidates can yield a better objective cost or a higher confidence of demand satisfaction. In each subsequent iteration, OCBA-CO uses the updated estimates to find new promising candidates and allocates a greater fraction of the delta budget to them to check if these candidates can be further refined.

In this way, SODA uses the model knowledge in *inflate*, *deflate*, and *refineCandidates* phases to provide optimal control settings of the non-linear processes that a process operator can use on the manufacturing floor in a stochastic environment.

## 4 EXPERIMENTAL RESULTS

This section discusses experimental results that evaluate SODA by comparing the quality of objective cost and rate of convergence obtained from SODA with other metaheuristic simulation-based optimization algorithms. We used the SCFA simulation of the Heat-Sink Production Process (HSPP) described in (Krishnamoorthy et al., 2017) in the experiments. The SCFA simulation of HSPP is written in JSONiq and SODA is written in Java. The deterministic approximations for SODA are performed using a system that automatically converts the SCFA simulation of HSPP into a deterministic optimization problem, which is a deterministic abstraction of the original SCFA simulation.

For the comparison algorithms, we used the jMetal package (Durillo and Nebro, 2011). The algorithms chosen for comparison include Nondominated Sorting Genetic Algorithm 2 (NSGA2) (Deb et al., 2002), Indicator Based Evolutionary Algorithm (IBEA) (Zitzler and Künzli, 2004), Strength Pareto Evolutionary Algorithm 2 (SPEA2) (Zitzler et al., 2001), and Speed-constrained Multi-objective Particle swarm optimization (SMPSO) (Nebro et al., 2009). These are popular multi-objective simulation-based optimization algorithms and they were chosen because they performed better than other single and multi-objective simulation-based optimization 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 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 SODA ran for. All these algorithms use the SCFA of HSPP to perform the cost, throughput and feasibility constraint computations. The computation of the demand satisfaction information from expected throughput is also similar to that of SODA. The cost and demand satisfaction information is then used by the jMetal algorithms to further increase or decrease the control settings using heuristics. The source code for SCFA simulation of HSPP, SODA algorithm and the comparison using the jMetal package is available at (Krishnamoorthy et al., 2020)

Initially, SODA was run with two different settings. In the first setting, SODA was run for 16 hours and the number of candidates collected from the *In-*

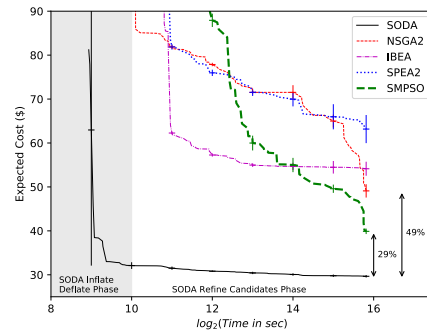


Figure 2: Estimated average cost for the elapsed time of 16 hours.

*flateDeflate* phase was 2,000. In the second setting, SODA was run for approximately 72 hours (3 days) and the number of candidates collected was 2,000. The *inflateDeflate* was run for a certain amount of time and to accumulate the required candidates within this time, the bound on the deterministic constraints over each production process in HSPP was increased so that the throughput range was wide. Additionally, all production processes were stochastic due to 5% of standard normal noise added to the controls. Finally, the demand from the HSPP was set to be 4. The comparison algorithms were also run with the same (relevant) parameters.

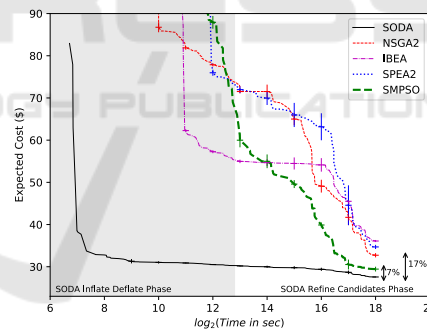


Figure 3: Estimated average cost for the elapsed time of 3 days.

The data collected from the experiments include the estimated average costs achieved at different elapsed time points for SODA in two settings and all the comparison algorithms. Figure 2 shows the estimated average costs achieved after 16 hours whereas Fig. 3 shows the estimated average costs achieved after about three days. Each experiment was run multiple times and 95% confidence bars are included at certain elapsed time points in both figures.

It can be seen that both settings of SODA perform better than the comparison algorithms initially. This is because SODA uses deterministic approximation to reduce the search space of potential candidates quickly whereas the competing algorithms start at a

random point in the search space and need a number of additional iterations (and time) to find candidates closer to those found by SODA.

As time progresses, SODA in both settings is able to achieve much better expected cost in the *inflateDeflate* phase than the other algorithms. Hence, SODA converges quicker toward the points close to the (near) optimal solution found at the end of the algorithm. This is because SODA uses heuristics in the *inflate* and *deflate* phases that reduce the search space quickly, which allows SODA to quickly converge toward a more promising candidate.

Also, the solutions found by SODA at the end of the experiment are much better than those found by the competing algorithms. After 16 hours, the expected cost found by SODA was 29% better than the nearest comparison algorithm (SMPSO) and 49% better than the second best comparison algorithm (NSGA2) (see Fig. 2). After three days though, the advantage of SODA over the nearest comparison algorithm (SMPSO) reduces to 7% and that to the second best algorithm (NSGA2) reduces to 17% (see Fig. 3). This is not surprising since SMPSO and NSGA2 use strong meta-heuristics and given enough time such algorithms will eventually converge toward the (near) optimal solution found by SODA. Although, it should be noted from Fig. 3 that SODA reaches close to the optimal solution found at the end much more quickly than its competition. Also, since the x-axis in Fig. 3 is in log scale, it should be noted that out of the total experiment time of about 259,200 sec, the other algorithms stop improving at around 180,911 sec whereas SODA continues to improve for a longer time (until about 210,562 sec) due to a good collection of candidates from the *InflateDeflate* phase and performing simulation refinements using an optimal budget allocation scheme of OCBA-CO in *RefineCandidates* phase. We also ran the Tukey-Kramer procedure for pairwise comparisons of correlated means on all six algorithm types. By doing so, we confirmed that SODA was indeed better than the other algorithms when the experiment ended.

We believe that SODA is sensitive to three categories of parameters of the production process viz. (1) the throughput range as determined by the interval between the requested demand and the throughput bound where the production process is at capacity, (2) the level of noise added to the controls of each production process, and (3) the coefficient used to inflate the demand parameter, where the higher value corresponds to inflating more aggressively. To evaluate SODA against the comparison algorithms for these categories, we ran the experiment for different settings of each of these parameters. For the through-

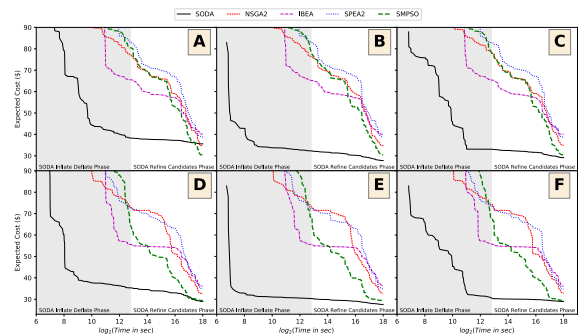


Figure 4: Estimated average cost for the elapsed time of 3 days for a wide throughput range where the noise level and inflation jump is A: high (10%) and very aggressive (1.3x); B: high (10%) and aggressive (1x); C: high (10%) and less aggressive (0.8x); D: low (5%) and very aggressive (1.3x); E: low (5%) and aggressive (1x); F: low (5%) and less aggressive (0.8x).

put range, we chose a wide range where the capacity of each production process of HSPP was high and a narrow range where the capacity was lower. For the level of noise, we chose 5% of standard normal noise as before as well as a higher noise of 10% of standard normal noise. Finally, the inflation jump was (a) more aggressive when a larger fraction of multiplicative exponential factor was added to the current demand, e.g. 1.3x, (b) aggressive when a smaller fraction of this factor was added to the current demand, e.g. 1x, and (c) less aggressive when an even smaller fraction of this factor was added to the current demand, e.g. 0.8x. To understand the effects of these parameters comprehensively, SODA and the comparison algorithms were run for 72 hours (3 days). The time for the *inflateDeflate* phase and the number of candidates collected during this phase was maintained the same as before. Also, the demand of HSPP was maintained at 4 so that we could perform a direct comparison of the estimated average costs over the elapsed time across these experiments.

Figures 4 and 5 show the estimated average costs achieved after three days when the throughput range was wide and when the throughput range was narrow, respectively. To increase or decrease the throughput range, the capacity of the production process was increased or decreased such that the throughput bound obtained by solving the optimization model in equation 3 would change accordingly. In Figs. 4 and 5 the first row shows results for the case when the noise level is 10% of the standard normal noise and the second row shows results for the case where the noise level is 5% of that noise. The first, second and third column of both these figures show results for the case when the level of inflation jump was more aggressive, aggressive and less aggressive, respectively. Since

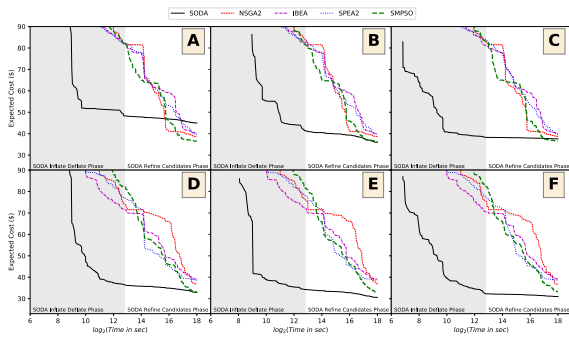


Figure 5: Estimated average cost for the elapsed time of 3 days for a narrow throughput range where the noise level and inflation jump is A: high (10%) and very aggressive (1.3x); B: high (10%) and aggressive (1x); C: high (10%) and less aggressive (0.8x); D: low (5%) and very aggressive (1.3x); E: low (5%) and aggressive (1x); F: low (5%) and less aggressive (0.8x).

the inflation jump occurs in the *inflateDeflate* phase of SODA, the results from the comparison algorithms remained the same in any particular row of Figs. 4 and 5.

Table 1 shows the percentage difference of the estimated average costs between SODA and the most competitive comparison algorithm, SMPSO, at the end of the experiment. When the throughput range is wide and the inflation jump is more aggressive (see Figs. 4A and 4D), SODA decreases the cost more rapidly than when the inflation jump is less aggressive (see Figs. 4C and 4F). This is because SODA is able to quickly find candidates that satisfy the demand with sufficient probability because of the more aggressive inflation jump. The initial candidates found by the more aggressive case (not shown here) have high expected cost due to the throughput range being wide. Additionally, SODA is affected by greater amount of noise since greater inflation is required to satisfy the demand with sufficient probability. This yields worse candidates collected during the *inflateDeflate* phase when the inflation jump is less aggressive and the noise is high (see Fig. 4C). It seems that when the throughput range is wide, the most effective strategy for inflation jump is for it to be neither less nor more aggressive because the cost decreases rapidly in the *inflateDeflate* phase and the candidates collected in this phase give the best gain in the expected cost over the comparison algorithms at the end of the experiment (see Figs. 4B and 4E).

When the throughput range is narrow and noise is high, we can see that the more aggressive case shown in Fig. 5A performs very poorly. A possible explanation is that the inflated demand keeps hitting the throughput boundary due to more aggressive inflation. This either yields higher expected cost or for the interval explored by deflate not to be wide

enough for the expected cost to improve. This phenomenon improves when the level of aggression is reduced as shown in Fig. 5B and the candidates collected in the *inflateDeflate* phase are the best for this scenario when the inflation level is less aggressive as shown in Fig. 5C. When the noise level is reduced, the candidates collected in the *inflateDeflate* phase with more aggressive inflation jump (see Fig. 5D) are better as compared to when the noise level is higher (see Fig. 5A). This happens because more candidates satisfy the demand with sufficient probability within the narrow throughput range when the noise level is low. Hence, when the throughput range is narrow, it seems that the most effective strategy for inflation level is for it to be between normal and less aggressive since SODA achieves the best gain over the comparison algorithms at this inflation level. This is because SODA explores the narrow throughput range better for feasible candidates and finds better optimal candidates among them at this inflation level.

Table 1: Percentage difference of the estimated average cost between SODA and the most competitive comparison algorithm SMPSO found at the end of the experiment where positive percentages mean SODA did better and negative percentages mean SMPSO did better.

Inflation jump level	Wide throughput range and 10% noise	Wide throughput range and 5% noise	Narrow throughput range and 10% noise	Narrow throughput range and 5% noise
Very aggressive (1.3x)	-16	2	-23	1
Aggressive (1x)	9	7	2	8
Less aggressive (0.8x)	5	2	-2	7

## 5 CONCLUSIONS

This paper presented an efficient stochastic optimization algorithm called SODA for the problem of finding process controls of a steady-state production processes that minimize the expectation of cost while satisfying the deterministic feasibility constraints and stochastic steady state demand for the output product with a given high probability. SODA is based on performing (1) a series of deterministic approximations to produce a candidate set of near-optimal control settings for the production process, and (2) stochastic simulations on the candidate set using optimal simulation budget allocation methods.

This paper also demonstrated SODA on a use case of a real-world heat-sink production process that involves contract suppliers and manufacturers as well as unit manufacturing processes of shearing, milling, drilling, and machining. Finally, an experimental study showed that SODA significantly outperforms four popular simulation-based stochastic optimization algorithms. In particular, the study showed that SODA performs better than the best competing algo-

rithm by 29% after 16 hours and by 7% after three days. The study also showed that SODA can solve the optimization problem over a complex network of production processes and SODA is able to scale well for such a network. However, the efficiency of SODA is limited by the total budget allocated to OCBA-CO, which is an input parameter to the algorithm.

Future research directions include: (a) dynamically executing the inflate deflate phase and candidate refinement phase of SODA to improve the exploration of the search space; and (b) comparing SODA with an existing Stochastic Optimization Algorithm Based on Deterministic Approximations.

## ACKNOWLEDGEMENTS

The authors are partly supported by the National Institute of Standards and Technology Cooperative Agreement 70NANB12H277.

## REFERENCES

- Amaran, S., Sahinidis, N. V., Sharda, B., and Bury, S. J. (2016). Simulation optimization: a review of algorithms and applications. *Annals of Operations Research*, 240(1):351–380.
- Chen, C. H. and Lee, L. H. (2011). *Stochastic Simulation Optimization: An Optimal Computing Budget Allocation*. World Scientific Publishing Company, Hackensack, NJ, USA.
- Chen, C.-H., Lin, J., Yücesan, E., and Chick, S. E. (2000). Simulation budget allocation for further enhancing the efficiency of ordinal optimization. *Discrete Event Dynamic Systems*, 10(3):251–270.
- Dabney, J. B. and Harman, T. L. (2001). *Mastering SIMULINK 4*. Prentice Hall, Upper Saddle River, NJ, USA, 1st edition.
- Deb, K., Pratap, A., Agarwal, S., and Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197.
- Durillo, J. J. and Nebro, A. J. (2011). jMetal: A Java framework for multi-objective optimization. *Advances in Engineering Software*, 42(10):760–771.
- Klemmt, A., Horn, S., Weigert, G., and Wolter, K.-J. (2009). Simulation-based optimization vs. mathematical programming: A hybrid approach for optimizing scheduling problems. *Robotics and Computer-Integrated Manufacturing*, 25(6):917–925.
- Krishnamoorthy, M., Brodsky, A., and Menascé, D. A. (2018). Stochastic decision optimisation based on deterministic approximations of processes described as closed-form arithmetic simulation. *Journal of Decision Systems*, 27(sup1):227–235.
- Krishnamoorthy, M., Brodsky, A., and 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*, pages 30–46.
- Krishnamoorthy, M., Brodsky, A., and Menascé, D. A. (2017). Stochastic optimization for steady state production processes based on deterministic approximations. Technical Report GMU-CS-TR-2017-3, 4400 University Drive MSN 4A5, Fairfax, VA 22030-4444 USA.
- Krishnamoorthy, M., Brodsky, A., and Menascé, D. A. (2020). *Source code for SODA algorithm*.
- Lee, L. H., Pujowidianto, N. A., Li, L. W., Chen, C. H., and 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., and 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)*, pages 66–73. IEEE Press.
- OpenModelica (2009). *Efficient Traceable Model-Based Dynamic Optimization - EDop*, <https://openmodelica.org/research/omoptim/edop>. OpenModelica.
- Paraskevopoulos, D., Karakitsos, E., and Rustem, B. (1991). Robust Capacity Planning under Uncertainty. *Management Science*, 37(7):787–800.
- Provan, G. and Venturini, A. (2012). Stochastic simulation and inference using modelica. In *Proceedings of the 9th International Modelica Conference*, pages 829–837.
- Thieriota, H., Namera, M., Torabzadeh-Tarib, M., Fritzon, P., Singh, R., and Kocherry, J. J. (2011). Towards design optimization with openmodelica emphasizing parameter optimization with genetic algorithms. In *Modelica Conference*. Modelica Association.
- Thompson, S. D. and 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.
- Xu, J., Zhang, S., Huang, E., Chen, C., Lee, L. H., and Celik, N. (2016). MO<sup>2</sup>TOS: Multi-fidelity optimization with ordinal transformation and optimal sampling. *Asia-Pacific Journal of Operational Research*, 33(03):1650017.
- Zitzler, E. and Künzli, S. (2004). Indicator-based selection in multiobjective search. In *Proceedings of the 8th International Conference on Parallel Problem Solving from Nature, 2004*, pages 832–842. Springer.
- Zitzler, E., Laumanns, M., and Thiele, L. (2001). SPEA2: Improving the strength pareto evolutionary algorithm. Technical Report 103, Computer Engineering and Networks Laboratory (TIK), Swiss Federal Institute of Technology (ETH), Zurich, Switzerland.