*Proceedings of the 2017 Winter Simulation Conference*
*W. K. V. Chan, A. D'Ambrogio, G. Zacharewicz, N. Mustafee, G. Wainer, and E. Page, eds.*

# AN EPSILON-CONSTRAINT METHOD FOR INTEGER-ORDERED BI-OBJECTIVE SIMULATION OPTIMIZATION

Kyle Cooper
Susan R. Hunter

School of Industrial Engineering
Purdue University
West Lafayette, IN 47906, USA

Kalyani Nagaraj

School of Industrial Engineering & Management
Oklahoma State University
Stillwater, OK 74078, USA

## ABSTRACT

Consider the context of integer-ordered bi-objective simulation optimization, in which the feasible region is a finite subset of the integer lattice. We propose a retrospective approximation (RA) framework to identify a local Pareto set that involves solving a sequence of sample-path bi-objective optimization problems at increasing sample sizes. We apply the epsilon-constraint method to each sample-path bi-objective optimization problem, thus solving a sequence of constrained single-objective problems in each RA iteration. We solve each constrained single-objective optimization problem using the SPLINE algorithm, thus exploiting gradient-based information. In early RA iterations, when sample sizes are small and standard errors are relatively large, we provide only a rough characterization of the Pareto set by making the number of epsilon-constraint problems a function of the standard error. As the RA algorithm progresses, the granularity of the characterization increases until we solve as many epsilon-constraint problems as there are points in the (finite) image of the local Pareto set. Our algorithm displays promising numerical performance.

## 1 INTRODUCTION

We consider simulation optimization (SO) problems with two simultaneous objectives, that is, nonlinear optimization problems in which *two* objective functions, $g_1$ and $g_2$, may only be observed with error as the output from a Monte Carlo simulation oracle at each decision point $\mathbf{x}$. Bi-objective SO problems are a sub-class of the more general multi-objective simulation optimization (MOSO) problems, which are SO problems having two or more simultaneous objectives. The solution to a MOSO problem is the set of decision points for which no other decision point has objective values that are at least as good on all objectives, and strictly better on at least one objective. We call this set the Pareto set $\mathcal{P}$.

Since usually only one decision can be implemented, MOSO solution methods that return a characterization of the Pareto set $\mathcal{P}$ enable the decision-maker to express solution preferences after the optimization is conducted, instead of before. Such methods are called *a posteriori* methods in the corresponding and relatively well-developed deterministic multi-objective optimization (MOO) literature (see, e.g., Miettinen 1999). Eichfelder (2008) notes that the availability of increased computing power has fueled interest in developing such a posteriori methods.

Decision-makers also are showing increased interest in obtaining multiple Pareto optimal points as the solution to MOSO problems. Several reasons may exist for this increased interest. However, we suspect that as in the MOO context, the primary reason is that it has become increasingly computationally feasible to do so. In addition to the availability of increased computing power, methods to solve single-objective SO problems are increasingly mature and efficient (see, e.g., Chen and Lee 2010, Pasupathy and Ghosh 2013, Fu 2015 for overviews and entry points into the single-objective SO literature). As in the MOO context, these algorithms can be used as single-objective SO "engines" to find multiple Pareto optimal solutions (Marler

and Arora 2004). Example applications of MOSO problems using solution methods that return multiple Pareto points include surgical suite planning (Huschka et al. 2007), aircraft maintenance scheduling (Lee et al. 2007), plant breeding (Hunter and McClosky 2016), and inventory planning (Peirleitner et al. 2016). For further applications and an overview of existing MOSO literature, see Hunter et al. (2017).

Our focus is on finding a *local Pareto set* (defined and discussed in §2) as the solution to a *bi-objective* SO problem having *integer-ordered* decision variables $\mathbf{x}$. Not many solution methods currently exist for these types of problems, however, we believe this class of problems to be important from an applications perspective. Problems with integer-ordered decision variables arise frequently in applications; close to half of the problem entries in the SO problem library, `simopt.org` (Pasupathy and Henderson 2006, 2011), are defined on integer-ordered spaces. Existing papers containing applications of integer-ordered MOSO problems include Huschka et al. (2007), Li et al. (2015), and Peirleitner et al. (2016).

We develop a new method for integer-ordered bi-objective SO that combines several existing methodologies from the SO and MOO literature. First, we use a Retrospective Approximation (RA) framework to solve a sequence of bi-objective sample-path problems with increasing sample sizes (see, e.g., Healy and Schruben 1991, Chen and Schmeiser 2001, and Wang et al. 2013 and references therein). Within an RA iteration, with common random numbers (CRN) at a fixed sample size $m$, each bi-objective sample-path problem is solved using the $\varepsilon$-constraint method. The $\varepsilon$-constraint method is an MOO method that converts a bi-objective problem into a sequence of single-objective constrained problems parameterized by $\varepsilon$ values, e.g., minimize $g_1$ subject to $g_2 \leq \varepsilon$. The solution to each sample-path $\varepsilon$-constraint problem is a sample-path weakly locally Pareto optimal point (defined in §2). For different values of $\varepsilon$, we retrieve a collection of such points. We use the SPLINE algorithm of Wang et al. (2013) to solve the $\varepsilon$-constraint problems, which performs line search and exploits gradient-based information.

Our main contribution lies in our choice of $\varepsilon$ values in each RA iteration. In early RA iterations, when sample sizes are small and standard errors are relatively large, we ensure that the number of $\varepsilon$-constraint problems we solve is appropriately small by making the $\varepsilon$ values a function of the standard error of the estimated objective function values of the estimated locally Pareto optimal points (defined in §2). As the sample sizes increase, these standard errors decrease, and we obtain an increasingly "fine" representation of the image of a local Pareto set (defined in §2). In the limit, we solve only as many $\varepsilon$-constraint problems as we have points in the (finite) image of the local Pareto set. Thus our algorithm is consistent with the philosophy that on early RA iterations, we should not "over-solve" the sample-path problem by locating every point in the sample-path local Pareto set. Instead, we solve the problem only to an appropriate tolerance level before moving on to the next RA iteration, using the previous estimated sample-path local Pareto set as a "warm start" in the next RA iteration.

There are other MOO methods we could have used to solve the deterministic bi-objective sample-path problem on each RA iteration. The reformulation of an MOO problem as a collection of single-objective problems, called *scalarization*, is common in the MOO literature, and many scalarization methods exist (see, e.g., Miettinen 1999, Ehrgott and Wiecek 2005, Eichfelder 2008 in general, and Ralphs et al. 2006, Hamacher et al. 2007 for example bi-objective algorithms for integer-ordered spaces). We choose the $\varepsilon$-constraint method because it has several desirable properties. First, there exists an $\varepsilon$ value that can retrieve each Pareto point, even in "non-convex" portions of the objective function space. Second, unlike the parameters of some scalarization methods, each $\varepsilon$ value has real meaning to the decision-maker as a constraint on one of the objectives. Although our algorithm does not require the decision-maker to input $\varepsilon$ values, it can easily be adapted to accept the decision-maker's choice of one or more particularly interesting or important $\varepsilon$ values. Thus we view the development of efficient MOSO algorithms that use the $\varepsilon$-constraint method as having particular interest.

### 1.1 Related Literature for MOSO with Discrete Decision Variables

As mentioned previously, few methods exist for solving MOSO problems on integer-ordered spaces. Table 1, adapted from Hunter et al. (2017), provides a broad classification of current literature on general solutions methods for MOSO problems that are designed to handle discrete decision variables.

Table 1: Existing MOSO algorithms on discrete feasible sets can be broadly classified by the nature of the decision variables, the search space, and the type of solutions returned to the decision-maker.

| Feasible set | Decision variables | Global / Local | MOSO with $d = 2$ objectives | MOSO with $d \geq 2$ objectives |
|---|---|---|---|---|
| finite | possibly categorical | global | SCORE (Feldman and Hunter 2016), M-MOBA (Branke and Zhang 2015, Branke et al. 2016) | MOCBA (Lee et al. 2010) |
| finite | integer-ordered | local | **This work.** | MO-COMPASS (Li et al. 2015) |
| countable | integer-ordered | local | | |
| bounded | mixed | global | | MOPBnB (Huang and Zabinsky 2014) |

Note: Parts of this table were adapted from Hunter et al. (2017).

First, we notice a growing body of literature on algorithms that seek global Pareto solutions to MOSO problems on finite sets with decision variables that may be categorical (Feldman and Hunter 2016, Branke and Zhang 2015, Branke et al. 2016). Such problems are referred to as Multi-Objective Ranking and Selection (MORS) problems. MORS algorithms may be applied to a problem instance whose feasible region is a finite subset of an integer lattice. However, since these solution methods are not designed to exploit gradient-based information, they are unlikely competitors for the present context.

The multi-objective probabilistic branch and bound (MOPBnB) algorithm (Huang and Zabinsky 2014), which seeks a global solution to the MOSO problem on a bounded feasible set, handles both continuous and integer decision variables. MOPBnB proceeds by iteratively pruning the feasible region. The likelihood of the retained region being close to the true Pareto set increases with each iteration.

Finally, to the best of our knowledge, MO-COMPASS (Li et al. 2015) is the only provably convergent algorithm created specifically for solving MOSO problems on integer lattices. It employs COMPASS, a local SO solver, to identify candidate local Pareto points and provides probabilistic guarantees on convergence to a local Pareto set. While our algorithm is designed for only two objectives, MO-COMPASS is arguably the most appropriate competitor for our algorithm.

### 1.2 Notational Conventions

We use $\mathbb{Z}^q$ to denote the set of all $q$-dimensional integer-valued vectors, $\mathbb{Z}^q \subset \mathbb{R}^d$. Whenever it is reasonable to do so, capital letters denote random variables $(X)$, script capital letters denote sets $(\mathcal{A})$, vectors appear in bold $(\mathbf{x})$, and random vectors appear in capital bold $(\boldsymbol{X})$. The notation $\mathbf{0}_d$ denotes a $d$-dimensional vector of zeros. For vectors $\mathbf{a} = (a_1, \ldots, a_d) \in \mathbb{R}^d$ and $\mathbf{b} = (b_1, \ldots, b_d) \in \mathbb{R}^d$, we write (a) $\mathbf{a} \leq \mathbf{b}$ if $a_i \leq b_i$ for all $i = 1, \ldots, d$; (b) $\mathbf{a} < \mathbf{b}$ if $a_k < b_k$ for all $k = 1, \ldots, d$; (c) $\mathbf{a} \preceq \mathbf{b}$ if $a_k \leq b_k$ for all $k = 1, \ldots, d$ and $a_{k^*} < b_{k^*}$ for at least one $k^* \in \{1, \ldots, d\}$. We also require the following notions of distance. Let $\mathcal{A} \subset \mathbb{R}^q$ and $\mathcal{B} \subset \mathbb{R}^q$ be two nonempty, bounded sets. Then (a) $d(\mathbf{x}, \mathbf{x}') = ||\mathbf{x} - \mathbf{x}'||$ is the Euclidean distance between two points $\mathbf{x}, \mathbf{x}' \in \mathbb{R}^q$; (b) $d(\mathbf{x}, \mathcal{B}) = \inf_{\mathbf{x}' \in \mathcal{B}} ||\mathbf{x} - \mathbf{x}'||$ is the distance from the point $\mathbf{x} \in \mathbb{R}^q$ to the set $\mathcal{B}$; (c) $\mathbb{D}(\mathcal{A}, \mathcal{B}) = \sup_{\mathbf{x} \in \mathcal{A}} d(\mathbf{x}, \mathcal{B})$ is the distance from set $\mathcal{A}$ to set $\mathcal{B}$; and (d) $d_H(\mathcal{A}, \mathcal{B}) := \max\{\mathbb{D}(\mathcal{A}, \mathcal{B}), \mathbb{D}(\mathcal{B}, \mathcal{A})\}$ is the Hausdorff distance between sets $\mathcal{A}$ and $\mathcal{B}$.

## 2 PROBLEM & SOLUTION CONTEXTS

In this section, we formally define the problems we consider.

### 2.1 The Bi-Objective SO Problem on a Finite Integer Lattice

We consider problems of the form

$$\text{Problem } M\text{:} \quad \underset{\mathbf{x}\in\mathcal{D}}{\text{minimize}} \ \{\mathbf{g}(\mathbf{x}) = (g_1(\mathbf{x}), g_2(\mathbf{x})) := (\mathbb{E}[G_1(\mathbf{x},\xi)], \mathbb{E}[G_2(\mathbf{x},\xi)])\},$$

where the feasible set $\mathcal{D} \subseteq \mathbb{Z}^q$ is a finite integer lattice. For each $\mathbf{x} \in \mathcal{D}$, the corresponding objectives $g_1(\mathbf{x})$ and $g_2(\mathbf{x})$ are expectations of random outcomes $G_1(\mathbf{x},\xi)$ and $G_2(\mathbf{x},\xi)$ (each a function of some random quantity $\xi$) of a Monte Carlo simulation. In the context of two objectives, there may not exist a single point that is the "best" on both objectives. However, some points may dominate others, in the sense that if $\mathbf{x}_1, \mathbf{x}_2 \in \mathcal{D}$, we say $\mathbf{x}_1$ *dominates* $\mathbf{x}_2$ if $\mathbf{g}(\mathbf{x}_1) \preceq \mathbf{g}(\mathbf{x}_2)$; see §1.2 for notation. A point $\mathbf{x}^*$ in the decision space $\mathcal{D}$ is *non-dominated* or *globally Pareto optimal* if there does not exist $\mathbf{x} \in \mathcal{D}$ such that $\mathbf{g}(\mathbf{x}) \preceq \mathbf{g}(\mathbf{x}^*)$. In other words, there does not exist another point that is at least as good in all objectives, and strictly better in at least one objective. The solution to Problem *M* is the collection of all such non-dominated points, which we denote as the *global Pareto set* $\mathcal{P} := \{\mathbf{x} \in \mathcal{D} : \nexists\, \mathbf{x}' \in \mathcal{D} \ni \mathbf{g}(\mathbf{x}') \preceq \mathbf{g}(\mathbf{x})\}$. We denote the image of the set $\mathcal{P}$ as the *global efficient set* $\mathcal{E} := \{\mathbf{g}(\mathbf{x}) : \mathbf{x} \in \mathcal{P}\}$. (Authors use various terms for the set of non-dominated points; see Hunter et al. 2017 for a brief discussion of the conventions.)

Ideally, we would like to obtain the globally Pareto optimal set as the solution to Problem *M*, however, in this paper, we seek only a *local Pareto set*, which we now define. First, recall the definition of a $\mathcal{B}_a$-neighborhood of a point: for $\mathbf{x} \in \mathcal{D} \subseteq \mathbb{Z}^q$ and $a \geq 0$, the $\mathcal{B}_a$-neighborhood of $\mathbf{x}$ is $\mathcal{B}_a(\mathbf{x}) := \{\mathbf{x}' \in \mathbb{Z}^q : ||\mathbf{x} - \mathbf{x}'|| \leq a\}$. Then the corresponding $\mathcal{N}_a$-neighborhood of a set is the union of the $\mathcal{B}_a$-neighborhoods of all the points belonging to the set. That is, for $\mathcal{S} \subset \mathcal{D} \subseteq \mathbb{Z}^q$, the $\mathcal{N}_a$-neighborhood of a set $\mathcal{S}$ is $\mathcal{N}_a(\mathcal{S}) := \cup_{\mathbf{x}\in\mathcal{S}} \mathcal{B}_a(\mathbf{x})$. Since the neighborhood of a set $\mathcal{S}$ contains $\mathcal{S}$, we say that points in $\mathcal{N}_a(\mathcal{S}) \setminus \mathcal{S}$ are *neighborhood-only* points. Henceforth, as is typical in similar settings that deal with local optimality (see, e.g., Li et al. 2015), we only consider $a = 1$.

**Definition 1** A vector $\mathbf{x}^* \in \mathcal{D}$ is $\mathcal{N}_1$-*locally Pareto optimal* (henceforth, locally Pareto optimal) if there does not exist $\mathbf{x} \in \mathcal{D} \cap \mathcal{N}_1(\{\mathbf{x}^*\})$ such that $\mathbf{g}(\mathbf{x}) \preceq \mathbf{g}(\mathbf{x}^*)$.

**Definition 2** (Li et al. 2015) A set $\mathcal{P}_1$ is an $\mathcal{N}_1$-*local Pareto set* ($\mathcal{N}_1$-LPS) if for all $\mathbf{x}^* \in \mathcal{P}_1$, there does not exist $\mathbf{x} \in \mathcal{N}_1(\mathcal{P}_1) \cap \mathcal{D}$ such that $\mathbf{g}(\mathbf{x}) \preceq \mathbf{g}(\mathbf{x}^*)$, and for all $\mathbf{x} \in (\mathcal{N}_1(\mathcal{P}_1) \setminus \mathcal{P}_1) \cap \mathcal{D}$, there exists $\mathbf{x}^* \in \mathcal{P}_1$ such that $\mathbf{g}(\mathbf{x}^*) \preceq \mathbf{g}(\mathbf{x})$.

That is, a set $\mathcal{P}_1$ is an $\mathcal{N}_1$-LPS if all points in the $\mathcal{N}_1$-neighborhood are not dominated by any other points in the $\mathcal{N}_1$-neighborhood, and all $\mathcal{N}_1$-neighborhood-only points are dominated by at least one point in the set $\mathcal{P}_1$. As in MO-COMPASS (Li et al. 2015), we seek an $\mathcal{N}_1$-LPS as the solution to Problem *M*.

### 2.2 The Sample-Path Problem

Since we are in an SO context, we cannot observe the objective functions in Problem *M* directly. Instead, we can only observe estimates of the objective functions from, for example, a Monte Carlo simulation oracle. Given a simulation budget $m$, we assume we can construct an estimator $\bar{\boldsymbol{G}}_m(\mathbf{x}) := (\bar{G}_{1,m}(\mathbf{x}), \bar{G}_{2,m}(\mathbf{x})) = (\frac{1}{m}\sum_{i=1}^m G_1(\mathbf{x},\xi_i), \frac{1}{m}\sum_{i=1}^m G_2(\mathbf{x},\xi_i))$ for all $\mathbf{x} \in \mathcal{D}$, where $(G_1(\mathbf{x},\xi_1), G_2(\mathbf{x},\xi_1)), \ldots, (G_1(\mathbf{x},\xi_m), G_2(\mathbf{x},\xi_m))$ are $m$ independent and identically distributed copies of the random objective vector $(G_1(\mathbf{x},\xi), G_2(\mathbf{x},\xi))$ generated by the oracle. Additionally, for $k = 1, 2$, we have $\bar{G}_{k,m}(\mathbf{x}) \to g_k(\mathbf{x})$ w.p.1 as $m \to \infty$. Using these estimators, the we define the (deterministic) *sample-path problem* as

$$\text{Problem } \hat{M}_m\text{:} \quad \underset{\mathbf{x}\in\mathcal{D}}{\text{minimize}} \ (\bar{G}_{1,m}(\mathbf{x}), \bar{G}_{2,m}(\mathbf{x})).$$

The solution to Problem $\hat{M}_m$ is the estimated global Pareto set $\hat{\mathcal{P}} := \{\mathbf{x} \in \mathcal{D} : \nexists\, \mathbf{x}' \in \mathcal{D} \ni \bar{\boldsymbol{G}}_m(\mathbf{x}') \preceq \bar{\boldsymbol{G}}_m(\mathbf{x})\}$, and its estimated image is $\hat{\mathcal{E}} := \{\bar{\boldsymbol{G}}_m(\boldsymbol{X}) : \boldsymbol{X} \in \hat{\mathcal{P}}\}$. Thus $\hat{\mathcal{P}}$ and $\hat{\mathcal{E}}$ estimate $\mathcal{P}$ and $\mathcal{E}$.

Instead of solving the sample-path problem to global optimality, after fixing a sample size $m$, we seek only a sample-path $\mathcal{N}_1$-*estimated approximate local Pareto set*, defined as follows.

**Definition 3** A set $\hat{\mathcal{P}}_1 \subseteq \mathcal{D}$ is called an $\mathcal{N}_1$-*estimated approximate local Pareto set* ($\mathcal{N}_1$-EALPS) if

1. for all $\mathbf{x}^* \in \hat{\mathcal{P}}_1$, there does not exist $\mathbf{x} \in \hat{\mathcal{P}}_1$ such that $\bar{\boldsymbol{G}}_m(\mathbf{x}) \preceq \bar{\boldsymbol{G}}_m(\mathbf{x}^*)$,
2. for all $\mathbf{x}^* \in \hat{\mathcal{P}}_1$, there does not exist $\mathbf{x} \in (\mathcal{N}_1(\hat{\mathcal{P}}_1) \setminus \hat{\mathcal{P}}_1) \cap \mathcal{D}$ such that $\bar{\boldsymbol{G}}_m(\mathbf{x}) \preceq (\bar{\boldsymbol{G}}_m(\mathbf{x}^*) - \boldsymbol{\delta}(\mathbf{x}^*))$,
3. for all $\mathbf{x} \in (\mathcal{N}_1(\hat{\mathcal{P}}_1) \setminus \hat{\mathcal{P}}_1) \cap \mathcal{D}$, there exists $\mathbf{x}^* \in \hat{\mathcal{P}}_1$ such that $(\bar{\boldsymbol{G}}_m(\mathbf{x}^*) - \boldsymbol{\delta}(\mathbf{x}^*)) \preceq \bar{\boldsymbol{G}}_m(\mathbf{x})$,

where $\mathbf{0}_2 \leq \boldsymbol{\delta}(\mathbf{x}^*) \in \mathbb{R}^2$ for all $\mathbf{x}^* \in \hat{\mathcal{P}}_1$.

Definition 3 defines a relaxed version of an estimated LPS. Condition 1 ensures no point in the $\mathcal{N}_1$-EALPS dominates any other point in the $\mathcal{N}_1$-EALPS. Condition 2 allows points $\mathbf{x}^*$ in the $\mathcal{N}_1$-EALPS to be dominated by neighborhood-only points, as long as the neighborhood-only points do not dominate $\mathbf{x}^*$ by more than some amount $\boldsymbol{\delta}(\mathbf{x}^*)$. Condition 3 allows neighborhood-only points not to be dominated by anyone in the $\mathcal{N}_1$-EALPS, as long as there exists a point $\mathbf{x}^*$ in the $\mathcal{N}_1$-EALPS such that $\bar{\boldsymbol{G}}_m(\mathbf{x}^*) - \boldsymbol{\delta}(\mathbf{x}^*)$ dominates it. Notice that when $\boldsymbol{\delta}(\mathbf{x}^*) = \mathbf{0}_2$ for all $\mathbf{x}^* \in \hat{\mathcal{P}}_1$, we retrieve an estimated version of an $\mathcal{N}_1$-LPS. When solving the sample-path Problem $\hat{M}_m$, we seek an $\mathcal{N}_1$-EALPS, in which the relaxation is determined by the parameter $\boldsymbol{\delta}(\cdot)$. We suggest setting $\boldsymbol{\delta}(\cdot)$ as a function of the estimated standard errors in §3.1.

### 2.3 The $\varepsilon$-Constraint Sample-Path Problem

Since the sample-path Problem $\hat{M}_m$ is deterministic, we can find an $\mathcal{N}_1$-EALPS using deterministic methods. We solve the sample-path Problem $\hat{M}_m$ by solving a sequence of $\varepsilon$-constraint problems. Given parameters $m$, $\varepsilon$, and a chosen objective $k^*$, we define the sample-path $\varepsilon$-constraint problem as

$$\text{Problem } \hat{M}_m(\varepsilon, k^*): \quad \underset{\mathbf{x}\in\mathcal{D}}{\text{minimize}}\ \bar{G}_{k^*,m}(\mathbf{x}) \quad \text{s.t.} \quad \bar{G}_{k^{\text{con}},m}(\mathbf{x}) \leq \varepsilon \text{ for } k^{\text{con}} \neq k^*.$$

With minor modifications to the results in Miettinen (1999), p. 85–86), it follows that a local solution to Problem $\hat{M}_m(k^*, \varepsilon)$ is a sample-path locally weakly Pareto optimal point, where in general, we say that a vector $\mathbf{x}^* \in \mathcal{D}$ is $\mathcal{N}_1$-*locally weakly Pareto optimal* (henceforth, locally weakly Pareto optimal) if there does not exist $\mathbf{x} \in \mathcal{D} \cap \mathcal{N}_1(\{\mathbf{x}^*\})$ such that $\mathbf{g}(\mathbf{x}) < \mathbf{g}(\mathbf{x}^*)$. As in Wang et al. (2013), we say that a point $\mathbf{x}^* \in \mathcal{D}$ is an $\mathcal{N}_1$-local minimum (henceforth, a local minimum) of objective $k^*$ if $g_{k^*}(\mathbf{x}) \geq g_{k^*}(\mathbf{x}^*)$ for all $\mathbf{x} \in \mathcal{N}_1(\{\mathbf{x}^*\}) \cap \mathcal{D}$. By varying the parameter $\varepsilon$, we retrieve various constrained local minima on the chosen objective $k^*$ that belong to an $\mathcal{N}_1$-EALPS.

### 2.4 Questions Raised by Using the $\varepsilon$-Constrant Method in an RA Framework

We now consider specific questions raised by solving Problem $M$ using the $\varepsilon$-constraint method within an RA framework. Briefly, our proposed solution method proceeds as follows. Given an increasing sequence of sample sizes $\{m_\nu\}$ and a sequence of objectives to minimize $\{k^*_\nu\}$, at each RA iteration $\nu = 1, 2, \ldots$ we solve Problem $\hat{M}_{m_\nu}(\varepsilon_{\nu,i}, k^*_\nu)$ for a finite set of $\varepsilon_{\nu,i}$ values, $\varepsilon_{\nu,i} \in \Xi_\nu$. For each $\varepsilon$-constraint problem we solve, we obtain one weakly locally Pareto optimal point. We ensure that together, these points form an $\mathcal{N}_1$-EALPS. Then we ask, how should the $\varepsilon_{\nu,i}$ values be chosen to ensure the $\mathcal{N}_1$-EALPS converges to an $\mathcal{N}_1$-LPS at the fastest possible rate?

Due to space constraints, we do not discuss algorithm convergence or the rate of convergence in this paper. However, we make the following remark on convergence requirements.

**Remark 1** A "separation" assumption similar to that in MO-COMPASS is required: there exists a $\kappa > 0$ such that for all $\mathbf{x}, \mathbf{x}' \in \mathcal{D}$ and all $k \in \{1, 2\}$, $\mathbf{x} \neq \mathbf{x}'$ implies $|g_k(\mathbf{x}) - g_k(\mathbf{x}')| \geq \kappa$. This assumption implies that there are no weakly locally Pareto optimal points that are not also locally Pareto optimal points. Further, algorithm parameters from §3, such as $\{m_\nu\}, \{b_\nu\}, \gamma$, and $\beta$ must be chosen appropriately to ensure convergence. We are likely to inherit optimal algorithm parameters from R-SPLINE (Wang et al. 2013) and cgR-SPLINE (Nagaraj and Pasupathy 2016).

## 3 A RETROSPECTIVE APPROXIMATION EPSILON-CONSTRAINT ALGORITHM

We begin with a broad overview of the RA algorithm to solve the bi-objective SO problem on integer-ordered spaces using the $\varepsilon$-constraint method; then, we discuss the sub-routines of our proposed algorithm in detail.

The RA algorithm appears in Algorithm 1, and consists of repeatedly solving bi-objective sample-path Problem $\hat{M}_{m_\nu}$ using a schedule of increasing sample sizes $\{m_\nu\}$. For each sample size $m_\nu$ in the sequence, we solve sample-path Problem $\hat{M}_{m_\nu}$ using the algorithm EPSILONSWEEP. EPSILONSWEEP solves a sequence of $\varepsilon$-constrained sample-path problems and returns an $\mathcal{N}_1$-EALPS. The information carried forward from the $(\nu-1)$nth RA iteration to the $\nu$th RA iteration is the sample-path $\mathcal{N}_1$-EALPS returned by EPSILONSWEEP, which is used as the warm start solution for solving the sample-path problem in the $\nu$th iteration.

**Algorithm 1** An RA algorithm to solve Problem $M$ via the $\varepsilon$-constraint method

**Require:** initial solutions for each objective $\mathbf{x}_1^0$ and $\mathbf{x}_2^0$; a sequence containing the objective to minimize $\{k_\nu^*\}$; a schedule of sample sizes $\{m_\nu\}$ to expend on each visited solution at iteration $\nu$; a schedule of the maximum number of oracle calls allowed in SPLINE, $\{b_\nu\}$.

1: Initialize $\hat{\mathcal{P}}_0 = \{\mathbf{x}_1^0, \mathbf{x}_2^0\}$.
2: **for** $\nu = 1, 2, \ldots$ with CRN **do**
3: $\quad\hat{\mathcal{P}}_\nu$ = EPSILONSWEEP$(\hat{\mathcal{P}}_{\nu-1}, k_\nu^*, m_\nu, b_\nu)$
4: **end for**

In the following sections, we discuss the details of the EPSILONSWEEP algorithm, including the algorithms it calls, TRACEBACK and LPSCRAWL. We highlight three key features of our (deterministic) EPSILONSWEEP algorithm:

1. The number of $\varepsilon$-constraint problems we solve in each RA iteration is a function of the standard errors of the estimated objective function values of the estimated Pareto points. On average, smaller sample sizes solve fewer $\varepsilon$-constraint problems. In the limit, we solve as many $\varepsilon$-constraint problems as there are points in the image of the LPS.
2. In each RA iteration, initial $\varepsilon$ values are set to *disqualify* the local Pareto points found in the previous RA iteration. The function TRACEBACK uses the SPLINE algorithm to search for new sample-path local Pareto points in between the initial $\varepsilon$ values. This function searches for new sample-path local Pareto points by "tracing back" through the trajectory of a SPLINE call, adding $\varepsilon$ values to disqualify each new sample-path local Pareto point it finds, and re-starting the search from feasible points in the trajectory.
3. The function LPSCRAWL returns a sample-path $\mathcal{N}_1$-EALPS, which ensures we do not "over-solve" the sample-path problem at low sample sizes. The amount of relaxation $\boldsymbol{\delta}(\cdot)$ used in LPSCRAWL is also a function of the standard errors. There exist pathological Problems $M$ such that without LPSCRAWL, our RA algorithm may not converge. In "nice" problems (like our test problem in §4 that has a single $\mathcal{N}_1$-LPS), with properly chosen parameters, the algorithm should converge even when LPSCRAWL is mostly inactive.

For exposition in all algorithms that follow, everywhere the oracle is queried with sample size $m$ at decision vector $\mathbf{x}$, we assume the triple $(\mathbf{x}, \bar{\boldsymbol{G}}_m(\mathbf{x}), \widehat{\text{s.e.}}(\bar{\boldsymbol{G}}_m(\mathbf{x})))$ is stored for future reference. Thus for legibility, we omit the explicit passing of the additional information $(\bar{\boldsymbol{G}}_m(\mathbf{x}), \widehat{\text{s.e.}}(\bar{\boldsymbol{G}}_m(\mathbf{x})))$ between functions.

### 3.1 The EPSILONSWEEP Algorithm

Algorithm 2, EPSILONSWEEP, solves the sample-path bi-objective problem by solving a sequence of $\varepsilon$-constraint problems. The broad idea of the algorithm is to initialize the $\varepsilon$ values so as to disqualify the Pareto points carried forward from the last RA iteration, and then employ a search procedure to find new sample-path locally Pareto optimal points that are at least some function of the standard deviation away from the known sample-path locally Pareto optimal points.

**Algorithm 2** $\hat{\mathcal{P}}_{\text{new}}$ = EPSILONSWEEP$(\hat{\mathcal{P}}_{\text{old}}, k^*, m, b)$

**Require:** initial estimated Pareto set $\hat{\mathcal{P}}_{\text{old}}$; sample size $m$; limit on oracle calls within SPLINE $b$.
**Ensure:** updated estimated Pareto set $\hat{\mathcal{P}}_{\text{new}}$.
1: Initialize $k^* = k_v^*$ and $k^{\text{con}} = k$ for $k \in \{1,2\}, k \neq k^*$.
2: **for** all $\boldsymbol{X}^* \in \hat{\mathcal{P}}_{\text{old}}$ **do** obtain $\bar{\boldsymbol{G}}_m(\boldsymbol{X}^*)$ and $\widehat{\text{s.e.}}(\bar{\boldsymbol{G}}_m(\boldsymbol{X}^*))$. {Bring up estimators to new sample size $m$.}
3: Set $[\boldsymbol{X}_1^{\min}, \boldsymbol{X}_2^{\min}]$ = GETMIN$(\hat{\mathcal{P}}_{\text{old}}, m, b)$. {Search for new minima.}
4: **if** $\boldsymbol{X}_1^{\min} = \boldsymbol{X}_2^{\min}$ **then**
5: **return** $\hat{\mathcal{P}}_{\text{new}} = \{\boldsymbol{X}_1^{\min}\}$.
6: **else**
7: Set $\hat{\mathcal{P}}_\varepsilon = \{\boldsymbol{X}_1^{\min}, \boldsymbol{X}_2^{\min}\} \cup \hat{\mathcal{P}}_{\text{old}}$. {Initialize the set that determines the $\varepsilon$ values.}
8: Set constraints $\hat{\Xi} = \{\bar{G}_{k^{\text{con}},m}(\boldsymbol{X}_{k^{\text{con}}}^{\min})\} \cup \{\bar{G}_{k^{\text{con}},m}(\boldsymbol{X}^*) - f(\widehat{\text{s.e.}}(\bar{G}_{k^{\text{con}},m}(\boldsymbol{X}^*))) : \boldsymbol{X}^* \in \hat{\mathcal{P}}_\varepsilon\}$.
9: Update $\hat{\Xi} = \hat{\Xi} \setminus \{\varepsilon \in \hat{\Xi} : \varepsilon < \bar{G}_{k^{\text{con}},m}(\boldsymbol{X}_{k^{\text{con}}}^{\min})\}$. {Remove infeasible constraints.}
10: Sort elements of $\hat{\Xi}$ in ascending order, and denote the $i$th element as $\varepsilon_{[i]}$, $i = 0, 1, \ldots, |\hat{\Xi}|$.
11: **for** $i = 1, 2, \ldots, |\hat{\Xi}|$ **do**
12: Find $\boldsymbol{X}_i^0 = \text{argmin}_{\boldsymbol{X}^* \in \hat{\mathcal{P}}_\varepsilon} \{\bar{G}_{k^*,m}(\boldsymbol{X}^*) : \bar{G}_{k^{\text{con}},m}(\boldsymbol{X}^*) \leq \varepsilon_{[i-1]}\}$.
13: $\hat{\mathcal{P}}_i$ = TRACEBACK$(\boldsymbol{X}_i^0, \varepsilon_{[i-1]}, \varepsilon_{[i]}, k^*, k^{\text{con}}, m, b)$.
14: **end for**
15: Update $\hat{\mathcal{P}}_{\text{new}}$ =GETPARETOS$(\hat{\mathcal{P}}_\varepsilon \cup (\cup_{i=1}^{|\hat{\Xi}|} \hat{\mathcal{P}}_i))$.
16: Set $\hat{\mathcal{P}}_{\text{new}}$ =LPSCRAWL$(\hat{\mathcal{P}}_{\text{new}})$.
17: **return** $\hat{\mathcal{P}}_{\text{new}}$.
18: **end if**

To be more specific, the algorithm works as follows. After initializing parameters, in Step 2, we obtain estimators at the current sample size $m = m_v$ for all points in the previous $\mathcal{N}_1$-EALPS, $\hat{\mathcal{P}}_{v-1}$. This step helps ensure good $\varepsilon$ choices in the sample-path problem. (Henceforth, when discussing algorithms that operate within an RA iteration, we drop the subscript $v$). Then, in Step 3, we obtain estimators for the sample-path local minima on each objective using the SPLINE algorithm (Algorithm 5) inside the function GETMIN (Algorithm 3). While we use the SPLINE algorithm to find the local minimum, any convergent algorithm can be used in this step. If the minima from GETMIN are equal in EPSILONSWEEP Step 4, then the algorithm returns this value as the new estimated LPS in Step 5.

If the minima do not equal each other, the $\varepsilon$ values are determined in Step 8 by (a) setting one $\varepsilon$ value equal to the estimated minimum objective function value on the constrained objective, which serves as the smallest $\varepsilon$ value, and (b) setting all other epsilon values equal to the estimated objective function values of the Pareto points, minus a function $f$, of their estimated standard error on the constrained objective. As in cgR-SPLINE (Nagaraj and Pasupathy 2016), we define $f$ as

$$f(\widehat{\text{s.e.}}(\bar{G}_{k^{\text{con}},m}(\mathbf{x}))) := \widehat{\text{s.e.}}(\bar{G}_{k^{\text{con}},m}(\mathbf{x}))(m^{\frac{1}{2}-\gamma}) = \hat{\sigma}_{k^{\text{con}}}(\mathbf{x})/m^\gamma,$$

**Algorithm 3** $[\boldsymbol{X}_1^{\min}, \boldsymbol{X}_2^{\min}]$ = GETMIN$(\mathcal{S}, m, b)$

**Require:** a set of points $\mathcal{S} \subseteq \mathcal{D}$.
**Ensure:** Minima on each objective, $\boldsymbol{X}_1^{\min}$ and $\boldsymbol{X}_2^{\min}$.
1: Calculate $\boldsymbol{X}_{1,\text{old}}^{\min} = \text{argmin}_{\boldsymbol{X}^* \in \mathcal{S}} \{\bar{G}_{1,m}(\boldsymbol{X}^*)\}$ and $\boldsymbol{X}_{2,\text{old}}^{\min} = \text{argmin}_{\boldsymbol{X}^* \in \mathcal{S}} \{\bar{G}_{2,m}(\boldsymbol{X}^*)\}$.
2: **for** $k = 1, 2$ **do** {Solve two unconstrained single-objective problems.}
3: $[\boldsymbol{X}_k^{\min}, \sim] = \text{SPLINE}(\boldsymbol{X}_k^0, \mathcal{D}, m, b)$.
4: **end for**
5: **if** $\boldsymbol{X}_2^{\min} \preceq \boldsymbol{X}_1^{\min}$ **then** $[\boldsymbol{X}_1^{\min}, \sim] = \text{SPLINE}(\boldsymbol{X}_2^{\min}, \mathcal{D}, m, b)$. {Find a better local min if one dominates the other.}
6: **if** $\boldsymbol{X}_1^{\min} \preceq \boldsymbol{X}_2^{\min}$ **then** $[\boldsymbol{X}_2^{\min}, \sim] = \text{SPLINE}(\boldsymbol{X}_1^{\min}, \mathcal{D}, m, b)$.

where $\gamma \in (0, \frac{1}{2})$ and $\hat{\sigma}_{k^{\text{con}}}(\mathbf{x})$ is the estimated standard deviation on the constrained objective after obtaining $m$ samples at $\mathbf{x} \in \mathcal{D}$. Thus in each $\varepsilon$-constraint problem, the current estimated Pareto points are rendered *infeasible* by their respective $\varepsilon$ values. Step 9 removes all sample-path infeasible $\varepsilon$-constraint problems.

After a sorting step, Steps 11–14 divide the search space based on the $\varepsilon$ values and use the SPLINE algorithm within the TRACEBACK function to find new sample-path local Pareto points. These steps are designed to be executed in parallel; Step 12 ensures every SPLINE algorithm call within the TRACEBACK algorithm has a feasible starting solution. In Step 15, the results of each TRACEBACK search are collected into a new set of estimated local Pareto points, and any sample-path dominated points are eliminated from the set using the function GETPARETOS.

Finally, we call LPSCRAWL (Algorithm 6) to ensure that we have a sample-path $\mathcal{N}_1$-EALPS, which is returned in the $v$th RA iteration. This function is analogous to the "neighborhood enumeration" step in the SPLINE Algorithm 5, which checks that a point is truly a sample-path $\mathcal{N}_1$-local minimum on the feasible space. Define the relaxation of the $\mathcal{N}_1$-EALPS as $\boldsymbol{\delta}(\mathbf{x}) := (\hat{\sigma}_1(\mathbf{x})/m^{\beta}, \hat{\sigma}_2(\mathbf{x})/m^{\beta})$, where $\hat{\sigma}_k(\mathbf{x})$ is the estimated standard deviation on the $k$th objective, $k \in \{1,2\}$, and $\beta \in (0,\infty)$.

### 3.2 The TRACEBACK Algorithm

In this section, we discuss the details of the TRACEBACK algorithm, which is the main search mechanism inside EPSILONSWEEP. The TRACEBACK algorithm is designed to work with the SPLINE Algorithm 5. Since the first step of TRACEBACK is a call to SPLINE, we discuss SPLINE first.

SPLINE is a single-objective optimization algorithm that iterates between two functions, SPLI and NE. The SPLI function conducts a line search with piecewise-linear interpolation. The direction of the line search is determined by pseudo-gradients, which are constructed from the piecewise-linear interpolation. The NE function performs neighborhood enumeration to check if the current solution is a local minimum. For a complete listing of SPLI and a discussion of NE, we refer the reader to Wang et al. (2013). Given

**Algorithm 4** $\hat{\mathcal{P}}_{\text{sub}} = \text{TRACEBACK}(\boldsymbol{X}_0, \varepsilon_\ell, \varepsilon_u, k^*, k^{\text{con}}, m, b)$

**Require:** initial solution $\boldsymbol{X}_0$, objective to minimize $k^*$, objective to constraint $k^{\text{con}}$, constraints $\varepsilon_\ell, \varepsilon_u$, sample size $m$, limit on oracle calls within SPLINE $b$.
**Ensure:** a subset of the sample-path Pareto set $\hat{\mathcal{P}}_{\text{sub}}$.
1: $[\boldsymbol{X}^*, \mathcal{T}] = \text{SPLINE}(\boldsymbol{X}_0, \mathcal{D}_{k^*}(\varepsilon_u), m, b)$.
2: Set $\hat{\mathcal{P}}_{\text{sub}} = \{\boldsymbol{X}^*\}$.
3: Set $\varepsilon = \bar{G}_{k^{\text{con}},m}(\boldsymbol{X}^*) - f(\widehat{\text{s.e.}}(\bar{G}_{k^{\text{con}},m}(\boldsymbol{X}^*)))$.
4: **while** $\varepsilon > \varepsilon_\ell$ **do** {Trace back the SPLINE trajectory to find more Pareto points.}
5: Update $\mathcal{T} = \{\boldsymbol{X} \in \mathcal{T} : \bar{G}_{k^{\text{con}},m}(\boldsymbol{X}) < \varepsilon\}$.
6: Update $\boldsymbol{X}_0 = \text{argmin}_{\boldsymbol{X} \in \mathcal{T}} \bar{G}_{k^*,m}(\boldsymbol{X})$.
7: $[\boldsymbol{X}^*, \mathcal{T}'] = \text{SPLINE}(\boldsymbol{X}_0, \mathcal{D}_{k^*}(\varepsilon), m, b)$.
8: Update $\hat{\mathcal{P}}_{\text{sub}} = \hat{\mathcal{P}}_{\text{sub}} \cup \{\boldsymbol{X}^*\}$ and $\mathcal{T} = \mathcal{T} \cup \mathcal{T}'$. {Update Pareto set and trajectory set.}
9: Update $\varepsilon = \bar{G}_{k^{\text{con}},m}(\boldsymbol{X}^*) - f(\widehat{\text{s.e.}}(\bar{G}_{k^{\text{con}},m}(\boldsymbol{X}^*)))$.
10: **end while**

**Algorithm 5** $[\boldsymbol{X}^*, \mathcal{T}] = \text{SPLINE}(\boldsymbol{X}_0, \mathcal{D}, m, b)$

**Require:** initial solution $\boldsymbol{X}_0 \in \mathcal{D}$; sample size $m$; limit on oracle calls $b$.
**Ensure:** local solution $\boldsymbol{X}^*$ on $\mathcal{D}$; sample-path search set $\mathcal{T}$.
1: Initialize $n = 0$, $\boldsymbol{X}_{\text{NE}} = \boldsymbol{X}_0$, and $\mathcal{T} = \{\boldsymbol{X}_0\}$.
2: **repeat**
3: $[n', \boldsymbol{X}_{\text{SPLI}}] = \text{SPLI}(\boldsymbol{X}_{\text{NE}}, m, b)$ {Line search with piecewise-linear interpolation.}
4: $[n'', \boldsymbol{X}_{\text{NE}}] = \text{NE}(\boldsymbol{X}_{\text{SPLI}}, m)$ {Neighborhood enumeration to check for local optimality on $\mathcal{D}$.}
5: Set $n = n + n' + n''$ and $\mathcal{T} = \mathcal{T} \cup \{\boldsymbol{X}_{\text{SPLI}}, \boldsymbol{X}_{\text{NE}}\}$. {Update oracle calls expended and search trajectory.}
6: **until** $\boldsymbol{X}_{\text{NE}} = \boldsymbol{X}_{\text{SPLI}}$ or $n > b$ {Line search ends on a local solution.}

a feasible starting solution $\mathbf{x}_0 \in \mathcal{D}$, SPLINE is guaranteed to return a local minimum on $\mathcal{D}$. Note that SPLINE treats points outside the feasible set $\mathcal{D}$ as hard constraints, i.e., the Monte Carlo simulation is not defined on $\mathcal{D}^c$. There is no "recovery" mechanism if SPLINE begins at an infeasible point $\mathbf{x} \in \mathcal{D}^c$.

We provide two minor modifications to the SPLINE listing in Wang et al. (2013). First, we modify SPLINE so that it accepts a feasible domain $\mathcal{D}$ as a parameter. Since we wish to run SPLINE on only a subset of the feasible space, we use the bound constraint $\varepsilon$ to disqualify a portion of the feasible space, and define $\mathcal{D}_{k^*}(\varepsilon) := \{\mathbf{x} \in \mathcal{D} : g_{k^{\text{con}}}(\mathbf{x}) < \varepsilon \text{ for all } k^{\text{con}} \in \{1,2\}, k^{\text{con}} \neq k^*\}$. SPLINE is still guaranteed to return a local minimum on such a space (and hence an estimated weakly locally Pareto optimal point), as long as we ensure a feasible starting solution $\mathbf{x}_0 \in \mathcal{D}_{k^*}(\varepsilon)$. By EPSILONSWEEP Step 12, the starting solution $\boldsymbol{X}_0$ in TRACEBACK is guaranteed to be sample-path feasible for all $\varepsilon$-constraint problems solved within TRACEBACK. Second, we also modify the SPLINE algorithm to ensure that it saves and returns its *trajectory*, $\mathcal{T}$, which is the set of all points returned by SPLI or NE. This modification enables the TRACEBACK algorithm to "trace back" through the initial SPLINE trajectory created in Step 1.

After obtaining an initial local sample-path optimal point $\boldsymbol{X}^*$ on the feasible space constrained by the largest $\varepsilon$ value, $\varepsilon_u$, in Step 1, TRACEBACK updates the estimated subset of the local Pareto set in Step 2, and sets a new $\varepsilon$ constraint that disqualifies the point $\boldsymbol{X}^*$ in Step 3. If this new $\varepsilon$ value is still within the search interval $[\varepsilon_\ell, \varepsilon_u]$, then the SPLINE search trajectory is truncated to only points that are feasible for the new $\varepsilon$ value, and a new start solution is found as the best feasible point in the trajectory in Steps 5 and 6, respectively. The SPLINE algorithm is re-started from the new start solution, and yields a new estimated local Pareto point in Step 7. This point is added to the estimated subset of the local Pareto set, the trajectory is updated by appending the points from the new search, a new epsilon value is set that disqualifies the latest estimated local Pareto point (Steps 8–9), and the process is repeated. In this sense, the algorithm "traces back" through the space $[\varepsilon_\ell, \varepsilon_u]$ for new Pareto points and returns a set of estimated local Pareto points on $[\varepsilon_\ell, \varepsilon_u]$.

### 3.3 The LPSCRAWL Algorithm

Finally, we discuss the LPSCRAWL algorithm, which is listed as Algorithm 6. Assuming appropriate parameter choices, LPSCRAWL is designed to ensure that our algorithm returns an LPS in the limit, but does not waste samples on expensive naïve searching. We define the set of neighborhood points that do not conform to the definition of an $\mathcal{N}_1$-EALPS as follows.

**Definition 4** Let $\hat{\mathcal{P}} \subseteq \mathcal{D}$ be a collection of sample-path local Pareto points. Then define the *non-conforming neighborhood* $\mathcal{X}_1(\hat{\mathcal{P}})$ as the set of points $\mathbf{x} \in (\mathcal{N}_1(\hat{\mathcal{P}}) \setminus \hat{\mathcal{P}}) \cap \mathcal{D}$ such that

1. there exists $\mathbf{x}^* \in \hat{\mathcal{P}}$ such that $\bar{\boldsymbol{G}}(\mathbf{x}) \preceq \bar{\boldsymbol{G}}(\mathbf{x}^*) - \boldsymbol{\delta}(\mathbf{x}^*)$, or
2. there does not exist $\mathbf{x}^* \in \hat{\mathcal{P}}$ such that $(\bar{\boldsymbol{G}}(\mathbf{x}^*) - \boldsymbol{\delta}(\mathbf{x}^*)) \preceq \bar{\boldsymbol{G}}(\mathbf{x})$.

That is, Definition 4 is the set of all points in the neighborhood of $\hat{\mathcal{P}}$ that do not satisfy Conditions 2 or 3 of Definition 3. The function LPSCRAWL first checks if the candidate estimated Pareto set is an $\mathcal{N}_1$-EALPS. If so, it returns the candidate set. Otherwise, it performs a recursive call after adding all non-conforming

**Algorithm 6** $\hat{\mathcal{P}}_{\text{local}} = \text{LPSCRAWL}(\hat{\mathcal{P}})$

**Require:** candidate local Pareto set $\hat{\mathcal{P}}$.
**Ensure:** $\hat{\mathcal{P}}_{\text{local}}$ is an approximate Local Pareto Set.
1: $\mathcal{X}_1(\hat{\mathcal{P}}) =$EALPSCHECK$(\hat{\mathcal{P}})$. {Check if $\hat{\mathcal{P}}$ is an $\mathcal{N}_1$-EALPS.}
2: **if** $\mathcal{X}_1(\hat{\mathcal{P}}) = \emptyset$ **then**
3: **return** $\hat{\mathcal{P}}_{\text{local}} = \hat{\mathcal{P}}$.
4: **else**
5: Update $\hat{\mathcal{P}} =$GETPARETOS$(\hat{\mathcal{P}} \cup \mathcal{X}_1(\hat{\mathcal{P}}))$.
6: **return** $\hat{\mathcal{P}}_{\text{local}} =$ LPSCRAWL$(\hat{\mathcal{P}})$.
7: **end if**

neighborhood points $\mathcal{X}_1(\hat{\mathcal{P}})$ to the candidate Pareto set and removing the dominated points. Notice that when the parameters $\boldsymbol{\delta}(\mathbf{x}^*)$ are "large" for each $\mathbf{x}^* \in \hat{\mathcal{P}}$, LPSCRAWL terminates quickly.

## 4 NUMERICAL EXPERIMENTS

In this section, we discuss the performance of proposed Algorithm 1 and MO-COMPASS on a modified version of a test problem given by Kim and Ryu (2011). (Kim and Ryu (2011) considers MOSO problems on continuous spaces, and thus does not appear in Table 1.) Our modifications discretize and scale the domain of the test problem to coincide with the integer lattice, and reduce the variance of the objective functions. The following is the modified version of the problem from Kim and Ryu (2011):

$$\text{Problem } M_{\text{KR}}\text{:} \qquad \text{minimize} \begin{cases} g_1(\mathbf{x}) = \mathbb{E}\left[(x_1/10 - 2\xi_1)^2 + (x_2/10 - \xi_2)^2\right] \\ g_2(\mathbf{x}) = \mathbb{E}\left[x_1^2/100 + (x_2/10 - 2\xi_3)^2\right] \end{cases}$$
$$\text{s.t.} \quad \mathbf{x} \in \mathcal{D} = \mathcal{X}_1 \times \mathcal{X}_2,$$

where $\mathcal{X}_1 = \mathcal{X}_2 = \{0, 1, 2, \ldots, 50\}$ and $\xi_1, \xi_2, \xi_3$ are independent chi-square random variables, each with one degree of freedom. The size of the feasible and Pareto sets are $|\mathcal{D}| = 2601$ and $|\mathcal{P}| = 49$, respectively. Notice that there is no correlation between the objectives.

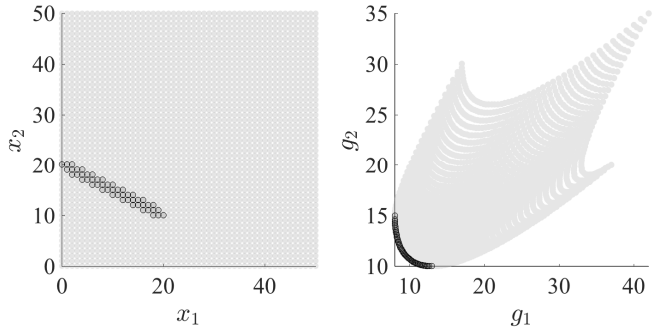


Figure 1: The black circles represent the Pareto set $\mathcal{P}$ (left panel) and the efficient set $\mathcal{E}$ (right panel) in Problem $M_{\text{KR}}$.

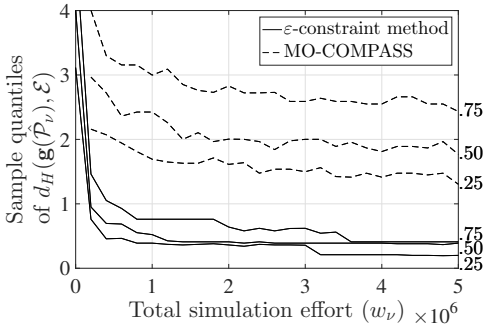


Figure 2: Sample quantiles of the Hausdorff distance on Problem $M_{\text{KR}}$.

To compare the two algorithms, we ran one hundred independent sample paths of each of Algorithm 1 and MO-COMPASS on Problem $M_{\text{KR}}$. We use CRN across algorithms within each independent run. Algorithm 1 takes five parameters: we set the schedule of sample sizes $\{m_\nu\}$ and the schedule of limits on oracle calls in SPLINE $\{b_\nu\}$ exactly as recommended in Wang et al. (2013), $\{m_\nu\} = \{2 * 1.1^\nu\}_{\nu=1}^{\infty}$ and $\{b_\nu\} = \{8 * 1.1^\nu\}_{\nu=1}^{\infty}$; we set $\gamma = 0.4$, $\beta = 0.5$, and $\{k_\nu^*\} = \{1, 2, 1, 2, 1, 2, \ldots\}$. In MO-COMPASS, we configure the algorithm as close as possible to the recommended settings in Li et al. (2015).

For both algorithms, Figure 2 shows the sample quantiles of $d_H(g(\hat{\mathcal{P}}_\nu), \mathcal{E})$, which is the Hausdorff distance between the set containing the true objective function values of the estimated Pareto set and the true efficient set of Problem $M_{\text{KR}}$, as a function of the total simulation effort $w_\nu := \sum_{i=1}^{\nu} m_i$. There is autocorrelation across total simulation effort values $w_\nu$. As seen in Figure 2, Algorithm 1 exhibits faster convergence to the true Pareto set than MO-COMPASS.

## CONCLUDING REMARKS

We present an $\varepsilon$-constraint algorithm in an RA framework for integer-ordered bi-objective SO. Though we have not proven that our algorithm converges to an $\mathcal{N}_1$-LPS, it appears to converge faster than its competitor on our test problem.

## ACKNOWLEDGMENTS

This work was supported by the National Science Foundation through grant CMMI-1554144.

## REFERENCES

Branke, J., and W. Zhang. 2015. "A new myopic sequential sampling algorithm for multi-objective problems". In *Proceedings of the 2015 Winter Simulation Conference*, edited by L. Yilmaz, W. K. V. Chan, I. Moon, T. M. K. Roeder, C. Macal, and M. D. Rossetti, 3589–3598. Piscataway, NJ: IEEE: Institute of Electrical and Electronics Engineers, Inc.

Branke, J., W. Zhang, and Y. Tao. 2016. "Multiobjective ranking and selection based on hypervolume". In *Proceedings of the 2016 Winter Simulation Conference*, edited by T. M. K. Roeder, P. I. Frazier, R. Szechtman, E. Zhou, T. Huschka, and S. E. Chick, 859–870. Piscataway, NJ: IEEE: Institute of Electrical and Electronics Engineers, Inc.

Chen, C.-H., and L. H. Lee. 2010. *Stochastic Simulation Optimization: An Optimal Computing Budget Allocation*. Hackensack, NJ: World Scientific.

Chen, H., and B. W. Schmeiser. 2001. "Stochastic root finding via retrospective approximation". *IIE Transactions* 33:259–275.

Ehrgott, M., and M. M. Wiecek. 2005. "Multiobjective Programming". In *Multiple Criteria Decision Analysis: State of the Art Surveys*, edited by J. Figueira, S. Greco, and M. Ehrgott, International Series in Operations Research & Management Science, Chapter 17. Springer New York.

Eichfelder, G. 2008. *Adaptive Scalarization Methods in Multiobjective Optimization*. Berlin Heidelberg: Springer.

Feldman, G., and S. R. Hunter. 2016. "SCORE allocations for bi-objective ranking and selection". *Optimization Online*.

Fu, M. (Ed.) 2015. *Handbook of Simulation Optimization*, Volume 216 of *International Series in Operations Research & Management Science*. New York: Springer-Verlag.

Hamacher, H. W., C. R. Pedersen, and S. Ruzika. 2007. "Finding representative systems for discrete bicriterion optimization problems". *Operations Research Letters* 35 (3): 336–344.

Healy, K., and L. W. Schruben. 1991. "Retrospective simulation response optimization". In *Proceedings of the 1991 Winter Simulation Conference*, edited by B. L. Nelson, W. D. Kelton, and G. M. Clark, 901–906.

Huang, H., and Z. B. Zabinsky. 2014. "Multiple objective probabilistic branch and bound for Pareto optimal approximation". In *Proceedings of the 2014 Winter Simulation Conference*, edited by A. Tolk, S. Y. Diallo, I. O. Ryzhov, L. Yilmaz, S. Buckley, and J. A. Miller, 3916–3927. Piscataway, NJ: IEEE: Institute of Electrical and Electronics Engineers, Inc.

Hunter, S. R., E. A. Applegate, V. Arora, B. Chong, K. Cooper, O. Rincón-Guevara, and C. Vivas-Valencia. 2017. "An introduction to multi-objective simulation optimization". *Optimization Online*.

Hunter, S. R., and B. McClosky. 2016. "Maximizing quantitative traits in the mating design problem via simulation-based Pareto estimation". *IIE Transactions* 48 (6): 565–578.

Huschka, T. R., B. T. Denton, S. Gul, and J. W. Fowler. 2007. "Bi-criteria evaluation of an outpatient procedure center via simulation". In *Proceedings of the 2007 Winter Simulation Conference*, edited by S. G. Henderson, B. Biller, M.-H. Hsieh, J. Shortle, J. D. Tew, and R. R. Barton, 1510–1518. Piscataway, NJ: IEEE: Institute of Electrical and Electronics Engineers, Inc.

Kim, S., and J. Ryu. 2011. "The sample average approximation method for multi-objective stochastic optimization". In *Proceedings of the 2011 Winter Simulation Conference*, edited by S. Jain, R. R. Creasey, J. Himmelspach, K. P. White, and M. Fu, 4026–4037. Piscataway, NJ: IEEE: Institute of Electrical and Electronics Engineers, Inc.

Lee, L. H., E. P. Chew, S. Teng, and D. Goldsman. 2010. "Finding the non-dominated Pareto set for multi-objective simulation models". *IIE Transactions* 42:656–674.

Lee, L. H., C. U. Lee, and Y. P. Tan. 2007. "A multi-objective genetic algorithm for robust flight scheduling using simulation". *European Journal of Operational Research* 177 (3): 1948–1968.

Li, H., L. H. Lee, E. P. Chew, and P. Lendermann. 2015. "MO-COMPASS: A fast convergent search algorithm for multi-objective discrete optimization via simulation". *IIE Transactions* 47 (11): 1153–1169.

Li, H., Y. Zhu, G. Pedrielli, N. A. Pujowidianto, and Y. Chen. 2015. "The object-oriented discrete event simulation modeling: a case study on aircraft spare part management". In *Proceedings of the 2015 Winter Simulation Conference*, edited by L. Yilmaz, W. K. V. Chan, T. M. K. Roeder, C. Macal, and M. Rosetti, 3514–3525. Piscataway, NJ: IEEE: Institute of Electrical and Electronics Engineers, Inc.

Marler, R. T., and J. S. Arora. 2004. "Survey of multi-objective optimization methods for engineering". *Structural and Multidisciplinary Optimization* 26:369–395.

Miettinen, K. 1999. *Nonlinear Multiobjective Optimization*. Boston: Kluwer Academic Publishers.

K. Nagaraj and R. Pasupathy 2016. "Stochastically constrained simulation optimization on integer-ordered spaces: The cgR-SPLINE algorithm". http://www.optimization-online.org/DB_HTML/2015/10/5139.html.

Pasupathy, R., and S. Ghosh. 2013. "Simulation optimization: a concise overview and implementation guide". In *TutORials in Operations Research*, edited by H. Topaloglu, Chapter 7, 122–150. Catonsville, MD: INFORMS.

Pasupathy, R., and S. G. Henderson. 2006. "A testbed of simulation-optimization problems". In *Proceedings of the 2006 Winter Simulation Conference*, edited by L. F. Perrone, F. P. Wieland, J. Liu, B. G. Lawson, D. M. Nicol, and R. M. Fujimoto, 255–263. Piscataway, NJ: IEEE: Institute of Electrical and Electronics Engineers, Inc.

Pasupathy, R., and S. G. Henderson. 2011. "SimOpt: A library of simulation optimization problems". In *Proceedings of the 2011 Winter Simulation Conference*, edited by S. Jain, R. R. Creasey, J. Himmelspach, K. P. White, and M. Fu. Piscataway, NJ: IEEE: Institute of Electrical and Electronics Engineers, Inc.

Peirleitner, A. J., K. Altendorfer, and T. Felberbauer. 2016. "A simulation approach for multi-stage supply chain optimization to analyze real world transportation effects". In *Proceedings of the 2016 Winter Simulation Conference*, edited by T. M. K. Roeder, P. I. Frazier, R. Szechtman, E. Zhou, T. Huschka, and S. E. Chick, 2272–2283. Piscataway, NJ: IEEE: Institute of Electrical and Electronics Engineers, Inc.

Ralphs, T. K., M. J. Saltzman, and M. M. Wiecek. 2006. "An improved algorithm for solving biobjective integer programs". *Annals of Operations Research* 147 (1): 43–70.

Wang, H., R. Pasupathy, and B. W. Schmeiser. 2013. "Integer-ordered simulation optimization using R-SPLINE: Retrospective Search using Piecewise-Linear Interpolation and Neighborhood Enumeration". *ACM Transactions on Modeling and Computer Simulation* 23 (3): 17:1–17:24.

## AUTHOR BIOGRAPHIES

**KYLE COOPER** is a Ph.D. student at Purdue University and researcher for Tata Consultancy Services. His email address is coope149@purdue.edu.

**SUSAN R. HUNTER** is an assistant professor in the School of Industrial Engineering at Purdue University. Her research interests include Monte Carlo methods and simulation optimization, especially in the presence of multiple performance measures. Her e-mail address is susanhunter@purdue.edu, and her website is http://web.ics.purdue.edu/~hunter63/.

**KALYANI NAGARAJ** is an assistant professor in the School of Industrial Engineering & Management at Oklahoma State University. Her research interests include Monte Carlo methodology, with a focus on simulation optimization and rare-event estimation. Her email address is kalyani.nagaraj@okstate.edu.