Bi-objective Simulation Optimization on Integer Lattices using the Epsilon-Constraint Method in a Retrospective Approximation Framework

Kyle Cooper  
School of Industrial Engineering, Purdue University and Tata Consultancy Services, coopel49@purdue.edu,  
Susan R. Hunter  
School of Industrial Engineering, Purdue University, susanhunter@purdue.edu,  
Kalyani Nagaraj  
School of Industrial Engineering & Management, Oklahoma State University, kalyani.nagaraj@okstate.edu,

We consider multi-objective simulation optimization (MOSO) problems on integer lattices, that is, nonlinear optimization problems in which multiple simultaneous objective functions can only be observed with stochastic error, e.g., as output from a Monte Carlo simulation model. The solution to a MOSO problem is the efficient set, which is the set of all feasible decision points that map to non-dominated points in the objective space. For problems with two objectives, we propose the R-PERLE algorithm, which stands for Retrospective Partitioned Epsilon-constraint with Relaxed Local Enumeration. R-PERLE is designed for simulation efficiency and provably converges to a local efficient set under appropriate regularity conditions. It uses a retrospective approximation (RA) framework and solves each resulting bi-objective sample-path problem only to an error tolerance commensurate with the sampling error. R-PERLE uses the sub-algorithm RLE to certify it has found a sample-path approximate local efficient set. We also propose R-MinRLE, which is a provably-convergent benchmark algorithm for problems with two or more objectives. R-PERLE performs favorably relative to R-MinRLE and the current state of the art, MO-COMPASS, in our numerical experiments. This work points to a family of RA algorithms for MOSO on integer lattices that employ RLE to certify sample-path approximate local efficient sets, and for which we provide the convergence guarantees.

Key words: multi-objective simulation optimization, retrospective approximation, epsilon-constraint

History: Authors’ preprint compiled Saturday 8th June, 2019 at 11:20am.

1. Introduction

Decision-makers increasingly rely on Monte Carlo simulation models to design and optimize complex stochastic systems (Powers et al. 2012). In this context, designing an “optimal” system requires solving an optimization problem in which the objective functions are defined implicitly through the Monte Carlo simulation model, are assumed to be nonlinear, and can only be observed with stochastic error. Such problems are called simulation optimization (SO) problems. Owing to their generality, SO problems arise in a variety of applications
including epidemic modeling (Nsosie et al. 2013), healthcare (Bertsimas et al. 2013), plant breeding (Hunter and McClosky 2016), and transportation (Osorio and Bierlaire 2013). Theory, methods, and algorithms for solving single-objective SO problems have been an active area of research for over thirty years, and mature algorithms exist to solve single-objective SO problems (see, e.g., Fu 2015, Pasupathy and Ghosh 2013, for overviews).

Far fewer resources exist for solving multi-objective simulation optimization (MOSO) problems, despite the fact that many practical applications employ the simultaneous consideration of multiple conflicting objectives in a simulation context (Hunter et al. 2019). We write the MOSO problem in \( d \) simultaneous objectives as

\[
\text{Problem } M_d: \quad \text{minimize}_{x \in X} \{ g(x) = (g_1(x), \ldots, g_d(x)) := (E[G_1(x, \xi)], \ldots, E[G_d(x, \xi))] \},
\]

where \( g: X \rightarrow \mathbb{R}^d \) is an unknown vector-valued function defined implicitly, e.g. through a Monte Carlo simulation oracle; the deterministic constraints, if present, specify a nonempty feasible set \( X \); and \( \xi \) is a random vector. The (global) solution to Problem \( M_d \) is called the efficient set, which is the set of feasible decision points for which no other feasible decision point is at least as good on all objectives and strictly better on at least one objective. The image of the efficient set is called the Pareto set. (We provide more detail on what it means to “minimize” a vector-valued objective function in §2; we refer the reader to Hunter et al. (2019) for a more complete introduction to MOSO.)

Our interest lies in the version of Problem \( M_d \) in which the feasible set is a subset of the integer lattice, \( X \subseteq \mathbb{Z}^q \); henceforth, when we refer to Problem \( M_d \), this property is implied. Usually, the decision variables are natural numbers with physical meaning to the decision-maker, such as the number of people to employ or units of stock to order. SO problems on integer lattices are called integer-ordered SO problems by Pasupathy and Henderson (2006, 2011); at the time of writing, half of the problems in the simopt.org library are integer-ordered single-objective SO problems (Henderson and Pasupathy 2019).

As noted by Hunter et al. (2019), integer-ordered MOSO problems are also common, arising in a variety of applications including aviation (Li et al. 2015b), healthcare (Chen and Wang 2016), environment (Singh and Minsker 2008), logistics and supply chain (Chew et al. 2009), and manufacturing (Andersson et al. 2007). For example, Li et al. (2015b) solve a bi-objective SO problem to manage aircraft spare parts. The objectives are to maximize an expected service level metric and to minimize the expected total cost, including holding
costs. The decision variables include the amount of spare parts inventory to allocate to each repair site. The solution to this problem is the efficient set, i.e., the set of decision points representing the number of spare parts to hold in each location that map to Pareto points in the objective space. Decision-makers then may use the efficient set as input to the decision-making process, perhaps also taking into account factors external to the simulation model.

1.1. Challenges in Solving MOSO Problems on Integer Lattices

When designing algorithms to solve MOSO problems on an integer lattice, several challenges arise. In particular, we consider the following challenges:

C.1 The objective functions are unknown and cannot be observed directly. We only have access to a (possibly computationally intensive) simulation oracle that, at each feasible point \( x \in \mathcal{X} \subseteq \mathbb{Z}^q \), can generate \( n \) simulation replications, or identically distributed copies of the random objective vector \( \mathbf{G}(x, \xi_i) := (G_1(x, \xi_i), \ldots, G_d(x, \xi_i)) \), \( i = 1, \ldots, n \). This information is used to construct the consistent estimator \( \bar{G}_n(x) = \frac{1}{n-1} \sum_{i=1}^{n} \mathbf{G}(x, \xi_i) \) of the unknown objective function values \( \mathbf{g}(x) \). Further, derivative information is not returned automatically by the oracle, implying that algorithms used to solve this MOSO problem must be derivative-free.

C.2 Solving an SO problem often becomes increasingly difficult as the number of objectives increases from one to two, from two to three, and so on.

C.3 MOSO problems are more computationally intensive than their deterministic multi-objective optimization and single-objective SO counterparts. (This statement follows by considering challenges C.1 and C.2 together.)

C.4 The constraints that specify the feasible set \( \mathcal{X} \) may be unknown or hidden (Le Digabel and Wild 2015). Hence we may only be able to query a constraint-satisfaction oracle to determine, without error, whether a point \( x \in \mathbb{Z}^q \) is feasible.

We now discuss these challenges, which are related.

First, challenge C.1 implies that no matter how much simulation effort is expended, there is a positive probability of retrieving an incorrect solution to a MOSO problem. At best, the estimation error drops off slowly according to the canonical rate \( O(1/\sqrt{n}) \) under naïve sampling (Asmussen and Glynn 2007). Because the solution estimates are random variables and may be incorrect, traditional deterministic guarantees on optimality are not available. Thus deterministic derivative-free multi-objective optimization methods (e.g., Ralphs et al.
(2006), Custódio et al. (2011), and Larson et al. (2019, p. 60–62)), which are not explicitly designed to manage random objective function estimators and do not explicitly control sampling error, are not suitable for naïve implementation on Problem $M_d$. See Fu (2002) for more on why it is important to explicitly control stochastic error in SO problems.

Second, while solving a single-objective SO problem is usually considered a computationally intensive task (Fu 2002), challenge C.2 states that solving an SO problem increases in difficulty as the number of objectives increases. The primary increase in difficulty from one to two objectives occurs because MOSO methods identify an entire efficient set, as opposed to a single minimizer. For example, consider one of the primary methods for solving deterministic multi-objective optimization problems, called scalarization. Scalarization reformulates the multiple objectives into a parameterized single-objective problem whose solution is a single efficient point (Miettinen 1999, p. 62). Then, by varying the scalarization parameters, the decision-maker solves many single-objective sub-problems to retrieve different points in the efficient set. When used in the context of MOSO, scalarization implies the need to solve many single-objective SO sub-problems.

Third, MOSO algorithms should explicitly address the computational intensity of solving MOSO problems (challenge C.3). The primary source of computational burden in MOSO algorithms usually lies in the requirement that the algorithms repeatedly conduct expensive stochastic objective function evaluations, especially on the (possibly numerous) points in and around the current estimated efficient set. These function evaluations should be conducted so that the algorithm expends the simulation effort as efficiently as possible while ensuring convergence. Tools that may enhance the efficiency of MOSO algorithms include using common random numbers (CRN) (see Law 2015, p. 588) and exploiting pseudo-gradient information to move toward a local efficient set in the decision space. (See §2 for a definition of a local efficient set; loosely speaking, a local efficient set is the global efficient set on a relevant subset of the feasible space, where the relevant subset is defined by some neighborhood. Methods that locate a local efficient set can be combined later with a method that guides restarts to locate the global efficient set.) Finally, MOSO methods should fully exploit the power of modern parallel computing platforms — either by solving sub-problems in parallel, or obtaining simulation replications in parallel, or both.

Fourth, challenge C.4 acknowledges that even though the constraints are deterministic, they may also be defined implicitly. Thus methods that do not require knowing the feasible set $\mathcal{X}$ in advance, that is, without running the simulation oracle, are useful in practice.
1.2. Existing Methods for Solving MOSO Problems on Integer Lattices

Perhaps owing to the computational complexity of the problem, few algorithms exist to solve MOSO problems on integer lattices that are both provably convergent to a local efficient set and explicitly control sampling error (Hunter et al. 2019). The current state-of-the-art algorithm for identifying a local efficient set as the solution to a MOSO problem on an integer lattice with \( d \geq 2 \) objectives is called MO-COMPASS (Li et al. 2015a); MO-COMPASS is a multi-objective version of COMPASS (Hong and Nelson 2006, Xu et al. 2010).

MO-COMPASS has the following properties. First, under appropriate regularity conditions, MO-COMPASS provably converges to a local efficient set with probability one (challenge C.2). MO-COMPASS explicitly controls sampling error (challenge C.1) by updating the Most Promising Area, which is a subset of the feasible set that the algorithm has deemed likely to contain a local efficient set, and by using a Simulation Allocation Rule to efficiently allocate simulation effort (challenge C.3). Because MO-COMPASS constructs the Most Promising Area from the feasible set in each iteration of the algorithm, the constraints that specify \( X \) must be \textit{a priori} and \textit{known}; that is, the constraints must be provided to the solver explicitly as part of the problem formulation (Le Digabel and Wild 2015). Therefore MO-COMPASS does not address challenge C.4. Finally, we remark that MO-COMPASS does not construct pseudo-gradients per se, or conduct line searches, in response to challenge C.3.

There are also MOSO methods that always provide only an estimator of the global efficient set, and thus do not locate local efficient sets that are not also the global efficient set. These methods include Multi-Objective Probabilistic Branch and Bound (MOPBnB, Huang and Zabinsky 2014) and multi-objective ranking and selection (MORS). MORS methods, which include MOCBA (Lee et al. 2010, Li et al. 2018), multi-objective SCORE (Feldman and Hunter 2018, Applegate et al. 2019), and M-MOBA (Branke and Zhang 2015, Branke et al. 2016), efficiently allocate simulation replications across a “small,” finite, and known feasible set. Since they allow the decision variables to be categorical, MORS methods do not exploit ordering that may exist in the decision space, which makes them unlikely to be competitive methods for problems on large subsets of an integer lattice.

We make two final remarks. First, we know of no existing algorithms for solving MOSO problems on integer lattices that address challenges C.1–C.4 by doing all of the following: explicitly controlling sampling error, provably converging to a local efficient set, moving through the decision space using line searches that exploit pseudo-gradients, and allowing
the constraint set to be hidden. The closest algorithm is by Cooper et al. (2017), who also present an algorithm for identifying a local efficient set as the solution to a MOSO problem on an integer lattice with \( d = 2 \) objectives. However, Cooper et al. (2017) is an early version of the present paper; our work subsumes theirs.

1.3. Overview of Contributions and Solution Approach

We propose a new family of algorithms for solving MOSO problems on integer lattices. Algorithms in our proposed family address challenges C.1–C.4 because they provably converge to a local efficient set under appropriate regularity conditions, explicitly control sampling error, enable the use of CRN and are easily parallelizable, and allow hidden constraints, respectively. We provide a detailed explanation of our approach in the sections that follow.

In brief, our family of algorithms is characterized by its use of a retrospective approximation (RA) framework, together with a sample-path solver that certifies the solution to each sample-path problem is a sample-path approximate local efficient set, to within an error tolerance commensurate with the sampling error.

We propose two pseudo-gradient-based algorithms in the family: R-PERLE and R-MinRLE. Our primary contribution is R-PERLE, which is a tailored algorithm for \( d = 2 \) objectives that uses strategically-conducted line searches that exploit pseudo-gradients. We also propose R-MinRLE, which is a pseudo-gradient-based benchmark algorithm for \( d \geq 2 \) objectives. R-PERLE shows promising numerical performance relative to R-MinRLE and to the current state-of-the-art, MO-COMPASS, on our test problems. Finally, code for our algorithms is available publicly in the PyMOSO software package (Cooper and Hunter 2019).

1.3.1. Retrospective Approximation

First, RA is version of Sample Average Approximation (SAA) that is designed for sampling efficiency (see, e.g., Pasupathy and Ghosh 2013). SAA is an algorithmic framework that replaces the unknown vector-valued functions \( g(\cdot) \) in Problem \( M_d \) with their estimators, resulting in the sample-path problem

Problem \( \overline{M}_{d,n} \):

\[
\text{minimize}_{x \in \mathcal{X}} \left\{ G_n(x) = (G_{1,n}(x), \ldots, G_{d,n}(x)) = \left( \frac{1}{n} \sum_{i=1}^{n} G_1(x, \xi_i), \ldots, \frac{1}{n} \sum_{i=1}^{n} G_d(x, \xi_i) \right) \right\}.
\]

The local and global solutions to Problem \( \overline{M}_{d,n} \) are called sample-path local and global efficient sets, respectively. (These sets are often constructed by locating the sample-path local efficient points. Every point in a sample-path local efficient set is a sample-path local efficient point; see §2.1.1 for definitions.)
Notice that using an algorithm to obtain an estimated solution to Problem \( M_d \) by solving Problem \( \overline{M}_{d,n} \) at a pre-determined sample size, say \( n = 100 \), is not necessarily efficient: the same large sample size is used for all feasible points visited by the algorithm. RA corrects this issue. Instead of solving Problem \( \overline{M}_{d,n} \) for a single, pre-determined, sample size \( n \), an RA framework prescribes solving a sequence of sample-path problems, characterized by the increasing sample size sequence \( \{m_\nu, \nu = 1, 2, \ldots\} \), where \( \nu \) is the RA iteration number. The solution to Problem \( \overline{M}_{d,m_{\nu-1}} \) obtained in RA iteration \( \nu - 1 \) is used as a warm start to solving Problem \( \overline{M}_{d,m_\nu} \) in RA iteration \( \nu \). Thus as the sample size increases, the warm starts are likely to improve, ensuring that large sample sizes are not wasted on severely suboptimal points. Using an RA framework also ensures that simulation replications can be obtained in parallel with CRN (challenge C.3). This algorithmic efficiency arises because within each RA iteration, we use the same sample size at every point visited by the algorithm.

1.3.2. Sample-Path Solver

Within each RA iteration \( \nu \), we require an algorithm to "solve" the sample-path Problem \( \overline{M}_{d,m_\nu} \), which is a deterministic multi-objective optimization problem for fixed values of the random variables \( \xi_i, i = 1, \ldots, n \); we call this algorithm the sample-path solver. We put quotes around "solve" because it would be inefficient to locate a complete sample-path local efficient set as the solution to Problem \( \overline{M}_{d,m_\nu} \) in every RA iteration \( \nu = 1, 2, \ldots \). Instead, we require a sample-path solver that is sensitive to the standard errors of the estimated objective function values at the current sample size (challenge C.1).

When standard errors of the objective vectors are "high," perhaps because the sample size in the current RA iteration is "low," we explicitly control sampling error by obtaining only a sample-path approximate local efficient set to use as a warm start in the next RA iteration. The amount of error allowed in this set is a function of the standard errors of the objective vector estimators at the current sample size. As the standard errors become smaller with larger sample sizes in later RA iterations, the sample-path solver returns an increasingly-complete sample-path approximate local efficient set. (Pasupathy (2010) employs similar concepts in the context of stochastic root finding.)

Assuming we have an appropriate definition of a sample-path approximate local efficient set (formally defined in §3.2), we create sample-path solvers with two key properties: (a) a way to "quickly" identify a subset of points in a sample-path local efficient set using pseudo-gradient information — call this the accelerator; and (b) a way to certify that the set of points obtained is in fact a sample-path approximate local efficient set, and if not, create
such a set from any set of starting points — call this the crawler. We discuss two sample-path solvers with these properties, MinRLE and PERLE. Then, we briefly discuss the SPLINE algorithm (Wang et al. 2013), which enables our accelerators to exploit pseudo-gradients.

The Simple Sample-Path Solver, MinRLE. Our benchmark algorithm, R-MinRLE for \( d \geq 2 \) objectives, results from creating a simple sample-path solver, called MinRLE, that satisfies the two key properties: the Min algorithm is the accelerator, and the RLE (Relaxed Local Enumeration) algorithm is the crawler. First, Min obtains one sample-path local minimizer for each objective using pseudo-gradient-based search. Then, the set of local minimizers is sent to the crawler, RLE, which enumerates the neighborhoods of the points in the set it receives. If RLE cannot certify that the set is a sample-path approximate local efficient set, it crawls through the decision space adding and removing points until it can certify the set. The R-MinRLE algorithm is naïve because using Min to locate minimizers on each objective tends to find the decision vectors that map to “extreme” points of a local Pareto set, resulting in inefficient crawling work for RLE to complete the “center” of the set.

The Tailored Sample-Path Solver, PERLE. To design a more tailored algorithm, we would like the accelerator to help the crawler by performing more pseudo-gradient-based searches in strategic locations. Ideally, the crawler should not crawl at all — it should only certify that the set of points returned by the accelerator is indeed a sample-path approximate local efficient set. To create such an accelerator in \( d = 2 \) objectives, we first obtain one sample-path local minimizer on each objective; the estimated local minima bound our search in the objective space. Then, we use a scalarization technique from the deterministic multi-objective optimization literature called the \( \varepsilon \)-constraint method (see, e.g., Miettinen 1999, p. 85) to search strategically for new sample-path local efficient points within these bounds. The \( \varepsilon \)-constraint method consists of selecting one objective to minimize and posing all others as constraints, where the constraint values are defined by “\( \varepsilon \)’s.”

The \( \varepsilon \)-constraint method has desirable properties for solving problems in two objectives. First, it can retrieve any point in an efficient set, unlike the linear weighted sum method. Second, as we will see, it allows easy control over which parts of the objective space are explored, as a function of the standard errors of the objective vector estimators. For a strategically-chosen objective \( k^* \in \{1, 2\} \), our accelerator solves several sample-path \( \varepsilon \)-constraint problems,

\[
\text{Problem } \bar{S}_{2,m_\nu}(k^*, \varepsilon) : \min_{x \in \bar{X}} G_{k^*,m_\nu}(x) \quad \text{s.t. } \bar{G}_{k^*,m_\nu}(x) \leq \varepsilon \quad \text{for } k^\text{con} \in \{1, 2\}, k^\text{con} \neq k^*,
\]
at \( \varepsilon \) values that are a function of the standard errors of the objective vectors corresponding to already-found sample-path local (weakly) efficient points (see §2.1.1 for a definition); already-found points include the sample-path local minima and warm starts from the previous RA iteration. Our accelerator is called PE, which stands for partitioned \( \varepsilon \)-constraint, because it partitions the objective vector space to search; these searches can be conducted in parallel. Loosely speaking, for each known sample-path local (weakly) efficient point \( X^w_{m_\nu} \) in RA iteration \( \nu \) at sample size \( m_\nu \), the PE algorithm places \( \varepsilon \) values to search for new sample-path local (weakly) efficient points inside the bounds specified by the sample-path local minima and outside the interval specified by \( \bar{G}_{k^{\text{con}},m_\nu}(X^w_{m_\nu}) \pm \hat{s.e.}(\bar{G}_{k^{\text{con}},m_\nu}(X^w_{m_\nu}))(m_\nu^{1/2-\beta_\varepsilon}) \), where \( \hat{s.e.}(\bar{G}_{k^{\text{con}},m_\nu}(X^w_{m_\nu})) \) is the estimated standard error of the constrained objective function estimator at \( X^w_{m_\nu} \), and \( \beta_\varepsilon \in (0, \infty) \) is a parameter. By default, we set \( \beta_\varepsilon = 1/2 \), although convergence is guaranteed for a wide range of values. Justification for our default choice and sensitivity of the algorithm to this parameter appear in the numerical section.

To demonstrate that PE usually provides good starting points to RLE, we consider the convergence properties of the R-PE algorithm, which consists of placing the PE algorithm in an RA framework. We prove that R-PE converges when special structure is present in the objective functions; such structure is not required for the convergence of our main algorithm, R-PERLE. We make two additional remarks here: First, R-PE is useful primarily for analysis. R-PERLE should always be chosen over R-PE in practice. When the required special structure is present, PE and RLE are designed so that RLE is mostly inactive under the default parameter settings. Second, directly employing the \( \varepsilon \)-constraint method becomes more complicated in \( d \geq 3 \) objectives, due in part to difficulties locating the nadir point to bound the Pareto set in the objective space (see, e.g., Miettinen 1999, p. 17). A version of R-PERLE in which the sample-path solver invokes the PE algorithm on any two objectives, followed by the RLE algorithm on all objectives, will provably converge to a local efficient set due to the invocation of RLE. However, it is not clear that such an algorithm applied in \( d \geq 3 \) objectives is a good idea. We leave the development of non-naïve accelerators for \( d \geq 3 \) objectives to future work.

The SPLINE algorithm. Finally, to locate sample-path local minimizers and to solve the sample-path \( \varepsilon \)-constrained problems, we employ an established single-objective, pseudo-gradient-based sample-path solver called SPLINE (Wang et al. 2013). SPLINE conducts
repeated line searches followed by a neighborhood enumeration step to certify that a sample-
path local minimizer has been found. We select the SPLINE algorithm because it is the
primary engine underlying the state-of-the-art single-objective SO algorithm R-SPLINE,
which, like our algorithm, operates within an RA framework. Thus the SPLINE solver is
especially well-suited for our solution context and demonstrates impressive performance on
single-objective SO problems in Wang et al. (2013). Conceivably, other algorithms could be
modified for use in this context. For example, other line search algorithms include Liuzzi et al.
(2018) and the references therein; other derivative-free algorithms are available in Conn et al.
(2009), Audet and Hare (2017). Given that our goal is presenting new MOSO algorithms,
we do not comment further on other possible single-objective sample-path solvers.

1.4. Organization
The sections that follow contain many details required to make our algorithms efficient and
convergent. To help the reader, we provide notation and terminology in §1.5. Then, §2 con-
tains formal definitions of optimality concepts for MOSO, followed by our problem statement.
In §3, we provide details of our solution context, including the definition of a sample-path
approximate local efficient set. Listings of our algorithms appear in §4–§6. Convergence and
efficiency results appear in §8, and numerical results appear in §9. All proofs and some
additional numerical results appear in the Online Appendix.

1.5. Notation and Terminology
With few exceptions, constants are denoted by lower-case letters (a), random variables by
capital letters (X), sets by script capital letters (A), vectors by bold (x), random vectors
by capital bold (X), families of sets by Fraktur (A ∈ A), and operators by blackboard bold
(∈ E[X]). The set of all q-dimensional integer-valued vectors is Zq ⊂ Rq. The set of all d-
dimensional extended real-valued vectors is R d. The d-dimensional vector (0, . . . , 0) is denoted
0 d. The complement of the set A is Ac. The sum of two sets A ⊆ Rd and B ⊆ Rd is the
Minkowski sum, A + B := {a + b : a ∈ A, b ∈ B}. For a sequence of events {An} defined in a
probability space, we say An i.o. if infinitely many of An occur, where An i.o. = lim supn An =
\bigcap_{i=1}^{\infty} \bigcup_{j=i}^{\infty} Aj. Finally, let A ⊆ Rq and B ⊆ Rq be nonempty, bounded sets. Then (a) d(x, x′) = ||x − x′|| is the Euclidean distance between two points x, x′ ∈ Rq; (b) d(x, B) = inf_{x′∈B} ||x − x′|| is the distance from the point x ∈ Rq to the set B; (c) d(A, B) = sup_{x∈A} d(x, B) is the
distance from set A to set B; and (d) dH(A, B) := max\{d(A, B), d(B, A)\} is the Hausdorff
distance between sets A and B.
2. Problem Context: Preliminaries for MOSO on Integer Lattices

In what follows, we define optimality concepts for Problem $M_d$ and provide a formal problem statement. Although R-PERLE is a bi-objective SO algorithm, we retain the generality of $d$ objectives since our benchmark algorithm, R-MinRLE, is defined for $d \geq 2$ objectives.

2.1. Optimality Concepts

Our presentation of optimality concepts follows Hunter et al. (2019), Li et al. (2015a), Wang et al. (2013). To begin, we define a flexible neighborhood structure and notions of dominance.

First, for a decision point $x \in \mathcal{X} \subseteq \mathbb{Z}^d$ and neighborhood size parameter $a \in \mathbb{R}$, $a \geq 1$, define the $N_a$-neighborhood of the point $x$ as $N_a(x) := \{x' \in \mathbb{Z}^d : d(x, x') \leq a\}$. Further, define the $N_a$-neighborhood of a set as the union of the $N_a$-neighborhoods of all the points belonging to the set. That is, for $\mathcal{S} \subset \mathcal{X} \subseteq \mathbb{Z}^d$, the $N_a$-neighborhood of $\mathcal{S}$ is $N_a(\mathcal{S}) := \bigcup_{x \in \mathcal{S}} N_a(x)$. Then for any set $\mathcal{A}$, define $N_a(\mathcal{A}) := N_a(\mathcal{A}) \setminus \mathcal{A}$ as the deleted neighborhood of $\mathcal{A}$. Second, to compare vectors in the objective function space, we define the following.

**Definition 1.** Let $x_1, x_2 \in \mathcal{X}$ and $d \geq 2$. We say the vector $g(x_1)$

1. weakly dominates $g(x_2)$, written as $g(x_1) \leq g(x_2)$, if $g_k(x_1) \leq g_k(x_2)$ for all $k = 1, \ldots, d$;
2. dominates $g(x_2)$, written as $g(x_1) \leq g(x_2)$, if $g(x_1) \leq g(x_2)$ and $g(x_1) \neq g(x_2)$;
3. strictly dominates $g(x_2)$, written as $g(x_1) < g(x_2)$, if $g_k(x_1) < g_k(x_2)$ for all $k = 1, \ldots, d$.

Using these definitions, we define concepts related to optimal points and optimal sets, which are illustrated in Figure 1 for an $N_1$-neighborhood structure.

2.1.1. Minimizers and Efficient Points

Following Wang et al. (2013), for each objective $k \in \{1, \ldots, d\}$, we define local minimizers of the $k$th objective function as follows.

**Definition 2 (Wang et al. 2013).** Given an objective function $g_k : \mathcal{X} \to \mathbb{R}$, a point $x_k^{\text{min}} \in \mathcal{X}$ is an $N_a$-local minimizer of $g_k$ if $g_k(x_k^{\text{min}}) \leq g_k(x)$ for all $x \in N_a(x_k^{\text{min}}) \cap \mathcal{X}$.

We further define local weakly efficient points and local efficient points as follows.

**Definition 3.** A point $x^w \in \mathcal{X}$ is an $N_a$-local weakly efficient point (LWEP) if

1. $\nexists x \in N_a(x^w) \cap \mathcal{X}$ such that $g(x) < g(x^w)$; or equivalently, if
2. $\forall x \in N_a(x^w) \cap \mathcal{X}$, $g(x) \not\leq g(x^w)$; that is, $\exists k \in \{1, \ldots, d\}$ such that $g_k(x^w) \leq g_k(x)$.

**Definition 4.** A point $x^* \in \mathcal{X}$ is an $N_a$-local efficient point (LEP) if

1. $\nexists x \in N_a(x^*) \cap \mathcal{X}$ such that $g(x) \not< g(x^*)$; or, equivalently, if
2. $\forall x \in N_a(x^*) \cap \mathcal{X}$, $g(x) \not< g(x^*)$; that is, one of the following holds: (a) $\exists k \in \{1, \ldots, d\}$ such that $g_k(x^*) < g_k(x)$, or (b) $g(x) = g(x^*)$.  

Notice that every $N_a$-local minimizer of some objective $g_k$, $k \in \{1, \ldots, d\}$, is an $N_a$-LWEP, and if the $N_a$-local minimizer is the unique minimum in its neighborhood, then it is also an $N_a$-LEP. Further, all $N_a$-LEP’s are $N_a$-LWEP’s. We define a global minimizer, a global weakly efficient point, and a global efficient point as an $N_a$-local minimizer, $N_a$-LWEP, and $N_a$-LEP, respectively, in which we set the neighborhood size parameter $a = \infty$.

### 2.1.2. Efficient and Pareto Sets

As in Hunter et al. (2019), we collect the various types of efficient points defined in the previous section into various types of efficient sets, as follows.

**Definition 5.** A set $W_a \subseteq X$, $|W_a| \geq 1$, is an $N_a$-local weakly efficient set if (a) each $x^w \in W_a$ is an $N_a$-LWEP, and (b) no points in $g(W_a)$ strictly dominate other points in $g(W_a)$, and (c) for each $x \in N_{a}^+ (W_a) \cap X$, $\exists x^w \in W_a$ such that $g(x^w) \preceq g(x)$.

**Definition 6.** A set $L_a \subseteq X$, $|L_a| \geq 1$, is an $N_a$-local efficient set if (a) each $x^* \in L_a$ is an $N_a$-LEP, and (b) no points in $g(L_a)$ dominate other points in $g(L_a)$, and (c) for each $x \in N_{a}^+ (L_a) \cap X$, $\exists x^* \in L_a$ such that $g(x^*) \preceq g(x)$.

Notice that every $N_a$-local efficient set is also an $N_a$-local weakly efficient set. Finally, we define the global weakly efficient set, denoted $E^w$, and the global efficient set, denoted $E$, as an $N_a$-local weakly efficient set and $N_a$-local efficient set, respectively, in which the neighborhood size is $a = \infty$. Although our definitions exist primarily in the decision space so far, we also define a $N_a$-local Pareto set as the image of an $N_a$-local efficient set, $g(L_a)$.

We remark here that under our definitions, there may exist $N_a$-LWEP’s that do not belong to an $N_a$-local weakly efficient set. To see an example of such a case, consider Figure 1, and...
notice that $g(x^*_g)$ is not dominated by the image of any points in the $N_1$-neighborhood of $x^*_g$, which are the points $g(x^*_{1,a}), g(x^*_{2,e})$, and $g(x^*_{1,f})$. Therefore, $x^*_g$ is an $N_1$-LWEP. (It is also an $N_1$-LEP.) However, $\{x^*_g\}$ is not an $N_1$-local weakly efficient set because $g(x^*_g)$ does not dominate $g(x^*_{1,a}), g(x^*_{2,e})$, or $g(x^*_{1,f})$. In this example, it is not possible to construct an $N_1$-local weakly efficient set using only the points $x^*_{1,f}, x^*_g, x^*_{2,e}$ because the images of these points do not dominate the images of any other feasible points, and therefore cannot dominate the images of the points in their deleted $N_1$-neighborhood. Thus any $N_1$-local weakly efficient set including $x^*_g$ must also include a member of the global efficient set whose image does not dominate $g(x^*_g)$, such as $x^*_{1,a}, x^*_g, x^*_{2,e}$. But, including any of these points in the candidate $N_1$-local weakly efficient set with $x^*_g$ implies that there exist neighborhood points that violate the definition of an $N_1$-local weakly efficient set. Thus $x^*_g$ does not belong to an $N_1$-local weakly efficient set.

2.2. Problem Statement

Using the optimality concepts defined in the previous section, we consider the following problem statement: Given a neighborhood size $a$ and an oracle capable of producing estimators $\tilde{G}_n(x)$ of $g(x)$ such that $\tilde{G}_n(x) \rightarrow g(x)$ w.p.1 as the sampling effort $n \rightarrow \infty$ for each $x \in X \subseteq \mathbb{Z}^q$, find a local solution to Problem $M_d$, which is an $N_a$-local efficient set.


Recall from §1.3.1 that we employ an RA framework to address our problem statement. First, we briefly revisit the sample-path problem and its solution. Then, we formally define a sample-path approximate local weakly efficient set, which we design for sampling efficiency.

3.1. The Sample-Path Problem and Solution

Given the definitions of optimality concepts in §2.1, we are now able to define optimality concepts as they relate to the sample-path Problem $\overline{M}_{d,n}$. We define sample-path versions of all optimality concepts in §2.1 by replacing the objective function values $g(x)$ and $g_k(x)$ with $\tilde{G}_n(x)$ and $\tilde{G}_{k,n}(x)$, respectively, for all $k \in \{1, \ldots, d\}$. We denote sample-path $N_a$-local minimizers, sample-path $N_a$-LWEP’s, and sample-path $N_a$-LEP’s as $X^*_{k,n}, X^w_n,$ and $X^*_n$, respectively. A local solution to Problem $\overline{M}_{d,n}$ is a sample-path $N_a$-local efficient set. Finally, recall that we solve a sequence of sample-path problems characterized by the increasing sample size sequence $\{m_\nu, \nu = 1, 2, \ldots\}$, where $\nu$ is the RA iteration number. Henceforth for simplicity, within an RA iteration $\nu$ we usually denote the sample size as $n = m_\nu$. 
3.2. A Sample-Path Approximate Local Weakly Efficient Set

As discussed in §1.3.2, we prefer not to locate a complete sample-path \( N_a \)-local efficient set in every RA iteration, since doing so may be inefficient. Instead, we wish to solve the sample-path problem only to within a certain error tolerance that is commensurate with the sampling error. To employ such a concept, we require a relaxed definition of a sample-path \( N_a \)-local efficient set that will enable us to stop our search within an RA iteration early. Thus we define an approximate version of local optimality for Problem \( \overline{M}_{d,n} \), as follows.

**Definition 7.** A set \( A \subseteq X \) is a sample-path approximate \( N_a \)-local weakly efficient set (ALES) for Problem \( \overline{M}_{d,n} \) if no points in \( G_n(A) \) dominate other points in \( G_n(A) \) and, given a vector-valued completeness function \( \delta: X \rightarrow \mathbb{R}^d \) such that \( 0_d \leq \delta(x) \) for all \( x \in X \),

(a) each \( x^w \in A \) is a sample-path \( N_a \)-LWEP, and

(b) for each \( x \in N'_a(A) \cap X \), (i) \( \exists x^w \in A \) such that \( G_n(x^w) \leq G_n(x) \), or (ii) \( \exists x^w \in A \) such that \( (G_n(x) \leq G_n(x^w) \) and \( G_n(x^w) - \delta(x^w) \leq G_n(x) + \delta(x) \), or (iii) \( \forall x^w \in A, G_n(x) \not\leq G_n(x^w), \text{ and } \exists \tilde{x}^w \in A \text{ such that } G_n(\tilde{x}^w) - \delta(\tilde{x}^w) \leq G_n(x) + \delta(x) \) or \( G_n(x) - \delta(x) \leq G_n(x^w) + \delta(\tilde{x}^w) \).

Definition 7 is similar to the definition of a sample-path \( N_a \)-local weakly efficient set, except for Part (b). Definition 7(b) requires that all feasible points in the deleted neighborhood of the ALES are either (i) weakly dominated by a point in the set, or (ii) dominate a point in the set by less than a certain amount, or (iii) do not weakly dominate any points in the set, and would either weakly dominate or be weakly dominated by a point in the set if both were moved by a certain amount. The “certain amount” is specified by the function \( \delta \). We call \( \delta(\cdot) = (\delta_1(\cdot), \ldots, \delta_d(\cdot)) \) the completeness function because it allows the ALES to have neighborhood points that violate the definition of a sample-path \( N_a \)-local weakly efficient set, and bigger values of \( \delta \) result in a “less-complete” ALES. If \( \delta_k(x) = \infty \) for all \( x \in X, k \in \{1, \ldots, d\} \), the ALES is a collection of sample-path \( N_a \)-LWEP’s that may or may not belong to a sample-path \( N_a \)-local weakly efficient set. If \( \delta(x) = 0_d \) for all \( x \in X \), the ALES is a sample-path \( N_a \)-local weakly efficient set.

We set the completeness function using the estimated standard errors of the objective function values \( \hat{G}_{k,n}(x) \) for all \( k \in \{1, \ldots, d\}, x \in X \). Thus we require the following definitions. For all \( x \in X, k \in \{1, \ldots, d\} \), define the variance \( \sigma_k^2(x) := \mathbb{V}(G_k(x, \xi)) < \infty \). (The assumption of finite variances is made formal in §8.) Then, let the estimated standard deviation of the \( k \)th objective value at \( x \in X \) be \( \hat{\sigma}_{k,n}(x) := \sqrt{\frac{n-1}{n} \sum_{i=1}^n (G_k(x, \xi_i) - \hat{G}_{k,n}(x))^2} \), and let the
standard error of the $k$th estimated objective value be $\hat{s.e.}(\hat{G}_{k,n}(x)) := \hat{\sigma}_{k,n}(x)/n^{1/2}$. Further, for each objective $k \in \{1, \ldots, d\}$ and for all $x \in \mathcal{X}$, $\beta \in (0, \infty]$, define the function $\hat{f}_k(x, \beta) := \hat{s.e.}(\hat{G}_{k,n}(x))(n^{1/2-\beta})$ if $\beta \in (0, \infty)$ and $\hat{f}_k(x, \infty) := 0$. Let $\hat{f}(x, \beta) := (\hat{f}_1(x, \beta), \ldots, \hat{f}_d(x, \beta))$.

For the remainder of the paper, we consider the ALES completeness function specified by $\hat{f}$. Since the value of this function is random for each $x \in \mathcal{X}$, henceforth, we denote the completeness function as $\hat{\delta}(x) := \hat{f}(x, \beta_\delta)$ for all $x \in \mathcal{X}$, where $\beta_\delta \in (0, \infty]$ is the completeness parameter. Smaller values of $\beta_\delta$ correspond to larger values of $\hat{\delta}(x)$, thus specifying a less-complete, or “smaller,” ALES. The value $\beta_\delta = \infty$ implies that the ALES is a sample-path $\mathcal{N}_a$-local weakly efficient set. Since the completeness function is a function of the standard error, under the regularity conditions in §8, the value $\hat{\delta}(x) \to 0$ w.p.1 for each $x \in \mathcal{X}$ as the sampling effort increases.

4. The Main Algorithm: R-PERLE for Two Objectives

We now provide an overview of our main RA algorithm, R-PERLE, which is listed in Algorithm 1. R-PERLE employs an RA framework that solves the sequence of sample-path Problems $\overline{M}_{2,m_\nu}$ at increasing sample sizes $\{m_\nu, \nu = 1, 2, \ldots\}$, where $\nu$ is the RA iteration number. Within an RA iteration, the PERLE algorithm is the sample-path solver. We implicitly define the PERLE algorithm as calling PE followed by RLE within one RA iteration $\nu$ (Algorithm 1 Steps 3 and 4). The solution to Problem $\overline{M}_{2,m_\nu}$ found by PERLE on the $\nu$th RA iteration, denoted $\hat{A}_\nu$ in Algorithm 1, is guaranteed by RLE to be an ALES. For efficiency, R-PERLE uses the sample-path solution from the preceding RA iteration, $\hat{A}_{\nu-1}$ which is an ALES for Problem $\overline{M}_{2,m_{\nu-1}}$, as an initial set of points for finding an ALES that solves Problem $\overline{M}_{2,m_\nu}$. Given the amount of detail inherent in the algorithms PE and RLE, we address these algorithms separately in §5 and §6, respectively.

R-PERLE requires a few input parameters in addition to a sequence of sample sizes $\{m_\nu, \nu = 1, 2, \ldots\}$ and an initial feasible point $x_0 \in \mathcal{X}$, which is declared as a global variable.

**Algorithm 1: The R-PERLE Algorithm for $d = 2$**

```
Input: initial point $x_0 \in \mathcal{X}$; sequence of sample sizes to expend at each visited point, $\{m_\nu\}$; sequence of limits on oracle calls during search, $\{b_\nu\}$; $\varepsilon$-placement and ALES parameters, $\beta = (\beta_\varepsilon, \beta_\delta)$

1 Initialize: $\hat{A}_0 = \{x_0\}$ and set $x_0$ as a global variable
2 for $\nu = 1, 2, \ldots$ with CRN do
3   $\hat{A}_\nu = $ PE($\hat{A}_{\nu-1}, m_\nu, b_\nu, \beta_\varepsilon$) /partition and solve $\varepsilon$-constraint problems
4   $\hat{A}_\nu = $ RLE($\hat{A}_\nu, m_\nu, b_\nu, \beta_\delta$) /guarantee the returned set is an ALES
```
in Algorithm 1 Step 1. First, R-PERLE requires a sequence of limits on oracle calls during search, \( \{b_\nu, \nu = 1, 2, \ldots\} \), that prevents chase-offs in the case of “bad” sample-path realizations of Problem \( \overline{M}_{2,m_\nu} \). We set this sequence so that for large enough RA iteration numbers \( \nu \), search “time outs” due to binding \( b_\nu \) do not occur w.p.1. R-PERLE requires parameters \( \beta = (\beta_\varepsilon, \beta_\delta) \), which control the completeness of the ALES and are discussed in the sections that follow. We suppress the choice of neighborhood size, which is \( a = 1 \) by default. The convergence properties of R-PERLE under different parameter values are discussed in §8, and we specify the default settings used in our numerical examples in §9.1. Under the default settings, the initial feasible point \( x_0 \in \mathcal{X} \) is the only required user-specified input.

Finally, for algorithmic efficiency, everywhere the oracle is called at a point \( \mathbf{x} \) with sample size \( n = m_\nu \) within an RA iteration, we assume the triple \( (\mathbf{x}, \hat{G}_n(\mathbf{x}), \hat{s.e.}(\hat{G}_n(\mathbf{x})) \) is stored and made available to all relevant subroutines within an RA iteration. Thus everywhere a candidate ALES is passed between functions, we assume the estimated objective function values of the neighborhood points are made available to all relevant subroutines, especially to RLE. This practice enhances efficiency by removing the need to re-sample neighborhood points when checking whether a candidate ALES is truly an ALES. All stored points visited and simulation replications obtained is cleared between RA iterations. For readability, we usually suppress the passing of this information between algorithms.

5. The PE Algorithm for Two Objectives

The PE algorithm is the first algorithm in the PERLE solver. In RA iteration \( \nu \), the PE algorithm chooses an objective \( k^* \) to minimize and uses the \( \varepsilon \)-constraint method to solve Problem \( \overline{S}_{2,n}(k^*, \varepsilon) \) at carefully-chosen \( \varepsilon \) values; recall that \( n = m_\nu \) is the current sample size in RA iteration \( \nu \). On an integer lattice, the sample-path \( \mathcal{N}_a \)-local minimizer for Problem \( \overline{S}_{2,n}(k^*, \varepsilon) \) is guaranteed to be an \( \mathcal{N}_a \)-LWEP for the original sample-path Problem \( \overline{M}_{2,n} \) (see, e.g., Miettinen 1999, for the continuous context). Except in pathological cases, varying the \( \varepsilon \) values and solving each resulting sample-path \( \varepsilon \)-constraint problem results in locating multiple sample-path \( \mathcal{N}_a \)-LWEP’s for Problem \( \overline{M}_{2,n} \). Thus PE yields a collection of sample-path \( \mathcal{N}_a \)-LWEP’s that includes a local minimizer on each objective and the sample-path \( \mathcal{N}_a \)-LWEP’s that result from solving Problem \( \overline{S}_{2,n}(k^*, \varepsilon) \) for each chosen \( \varepsilon \).

5.1. PE Algorithm Listing

We now discuss PE (Algorithm 2) in detail. First, to ensure that all sample-path \( \varepsilon \)-constraint problems have a non-empty feasible set, in Step 1, PE obtains updated sample-path \( \mathcal{N}_a \)-local
Algorithm 2: $\mathcal{A}_{\text{new}} = \text{PE}(\mathcal{A}_{\text{old}}, n, b, \beta_k)$

**Input:** estimated efficient set from the last RA iteration, $\mathcal{A}_{\text{old}} \subseteq \mathcal{X}$; sample size, $n$; limit on oracle calls during search, $b$; epsilon placement parameter, $\beta_k$

**Output:** $\mathcal{A}_{\text{new}} \subseteq \mathcal{X}$, a collection of sample-path $N_a$-LWEP’s

1. $\mathcal{A}_{\text{old}}^0 = \text{Min}(\mathcal{A}_{\text{old}} \cup \{x_0\}, n, b)$  
   \text{//SEARCH: update sample-path $N_a$-local minimizers}

2. $[\sim, \mathcal{A}_c, \sim] = \text{RemoveNonLWEP}(\mathcal{A}_{\text{old}}^0)$  
   \text{//get the set of sample-path $N_a$-LWEP’s in $\mathcal{A}_{\text{old}}^0$}

3. if $\mathcal{A}_{\text{old}} = \emptyset$ then $\mathcal{A}_{\text{old}}^\varepsilon \leftarrow \mathcal{A}_{\text{old}}^0$  
   \text{//Min update timed out, no other sample-path $N_a$-LWEP’s exist}

4. Initialize: $\varepsilon^0 \leftarrow |\mathcal{A}_{\text{old}}^\varepsilon|$  
   \text{//set $\varepsilon$’s using sample-path $N_a$-LWEP’s}

5. for $k^* = 1, 2$ do  
   \text{//determine objective to minimize, $k^*$}

6. Initialize: $k^* \leftarrow k$ for $k \in \{1, 2\}$ such that $k \neq k^*$

7. Sort $G_n(\mathcal{A}_{\text{old}})$ on $k^*$-con to get $(X_{i_1}^{w_1}, \ldots, X_{i_0}^{w_0})$ where $\bar{G}_{k,\text{con},n}(X_{i_1}^{w_1}) \leq \ldots \leq \bar{G}_{k,\text{con},n}(X_{i_0}^{w_0})$

8. Set constraint lower bound $L_k \leftarrow \bar{G}_{k,\text{con},n}(X_{i_1}^{w_1}) + \bar{f}_{k,\text{con}}(X_{i_1}^{w_1})$

9. for $i = 2, \ldots, c^0$ do

10. \begin{align*}
      \varepsilon_{i,k}^0(i) &\leftarrow \bar{G}_{k,\text{con},n}(X_{i}^{w}) - \bar{f}_{k,\text{con}}(X_{i}^{w}, \beta_k) \\
      \varepsilon_{i,k}^0(i) &\leftarrow \bar{G}_{k,\text{con},n}(X_{i}^{w}) + \bar{f}_{k,\text{con}}(X_{i}^{w}, \beta_k)
    \end{align*}

11. $C_{\text{new}} \leftarrow \{\varepsilon_{i,k}^0(i) : i \in \{2, \ldots, c^0\}, L_k \prec \varepsilon_{i,k}^0(i), \varepsilon_{i,k}^0(i) \notin \{\varepsilon_{i,k}^0(i'), \varepsilon_{i',k}^0(i')\} \text{ for all } i' \in \{2, \ldots, c^0\}\}$

12. $C_{\text{new}} \leftarrow \{C_{\text{new}}\}$

13. $K^* \leftarrow \arg \min\{C_{\text{new}} : k \in \{1, 2\}\}$  
   \text{//choose least $\varepsilon$-constraints; break ties randomly}

14. $C \leftarrow C_{K^*}$ and $K_{\text{con}} \leftarrow k$ for $k \in \{1, 2\}$ such that $k \neq K^*$

15. if $C > 0$ then

16. Sort $C_{K^*}$ in ascending order to get the ordered list $(\varepsilon_{1}, \ldots, \varepsilon_{C})$

17. for $j = 1, 2, \ldots, C$ do  
   \text{//partition space}

18. $\varepsilon_j \leftarrow \max(L_k, \max(\varepsilon_{i,k}^0(i) : i \in \{2, \ldots, c^0\}))$  
   \text{//get traceback lower bound}

19. Initialize: $\varepsilon_{\text{new}} \leftarrow \varepsilon_j$, $\mathcal{A}_{\text{new}} \leftarrow \emptyset$, $T \leftarrow \emptyset$

20. while $\varepsilon_j < \varepsilon_{\text{new}}$ do  
   \text{//find new sample-path $N_a$-LWEP’s}

21. $X_n \leftarrow \arg \min\{G_{k,\text{con},n}(X) : X \in T \cup \mathcal{A}_{\text{new}}, G_{k,\text{con},n}(X) \leq \varepsilon_{\text{new}}\}$

22. $X_n \subseteq T$, $N(X_n) = \text{SPLINE}(K^*, X_n, n, b, \varepsilon_{\text{new}})$  
   \text{//SEARCH: solve $\varepsilon$-constrained}

23. $\mathcal{A}_{\text{new}} \leftarrow \mathcal{A}_{\text{new}} \cup \{X_n\}$ and $T \leftarrow T \cup T'$

24. $\mathcal{A}_{\text{LWEP}} = \mathcal{A}_{\text{new}} \cup \mathcal{A}_{\text{LWEP}} \cup \{x_0\}$  
   \text{//collect new sample-path $N_a$-LWEP’s}

25. $\mathcal{A}_{\text{new}} = \text{Remove Dominated}(G_n(\mathcal{A}_{\text{LWEP}} \cup \mathcal{A}_{\text{LWEP}} \cup \{x_0\}))$  
   \text{//remove dominated, do no harm}

minimizers on each objective, stored in the set $\mathcal{A}_{\text{old}}^0$. (The Min algorithm called in Step 1 is listed in Algorithm 3 and discussed in §5.2.) The sample-path $N_a$-local minimizers at current sample size $n$ ensure that for the constrained objective $k_{\text{con}}$, no $\varepsilon$ values are placed outside of the interval $[\bar{G}_{k,\text{con},n}(X_{\text{min}}), \bar{G}_{k,\text{con},n}(X_{k^*}, n)]$ for $k_{\text{con}} \neq k^*$. Thus for every $\varepsilon$-constraint problem posed, there exists a sample-path feasible point in the set $\mathcal{A}_{\text{old}}^0$.

To select an objective $k^* \in \{1, 2\}$ to minimize, in Steps 2 through 14, PE places constraints a function of the standard error away from the images of known sample-path $N_a$-LWEP’s on both objectives, and ultimately selects the objective that results in solving the least number of $\varepsilon$-constraint problems. This strategy ensures that we do not constrain an objective with relatively small standard errors, thus wasting simulation effort attempting to order many points on a high standard error objective. To determine the set of $\varepsilon$-constraint values for
Figure 2 The figure shows part of a “traceback” from PE Steps 20–24, assuming objective 1 is constrained and that in PE Step 4, $\hat{A}^w_n$ has exactly two sample-path local minimizers: one on each objective, represented by outlined solid black dots. The traceback places the first $\varepsilon$-constraint at $\varepsilon_{new,1}$, thus retrieving the outlined black star point. Then, it places $\varepsilon_{new,2}$ and retrieves the black star point, and so on, until a value of $\varepsilon_{new} < \varepsilon^*_1$.

each objective, first, in Step 2, PE creates an initial set of known sample-path $N_a$-LWEP’s, $\hat{A}^w_n$, from the points in $\hat{A}^0_n$ using the RemoveNonLWEP function (discussed in §6.2). If there are no such points in $\hat{A}^0_n$ in Step 3, the search budget value $b$ must have been binding in the Min algorithm. Then PE sets $\varepsilon$ values based on $\hat{A}^0_n$. Since $b$ is non-binding in the limit, for large enough RA iteration number $\nu$, all $\varepsilon$ values will be set based on known sample-path $N_a$-LWEP’s. In Steps 5 through 12, PE sorts the initial points and partitions the objective space by placing $\varepsilon$ values in each part of the objective space where the standard error intervals of the initial points do not overlap. The standard error intervals in Steps 5 through 12 are defined using the function $\hat{f}$ from §3.2. For all known sample-path $N_a$-LWEP’s, denoted $X^w_n$, the PE algorithm does not place any new $\varepsilon$ values in the interval $\tilde{G}_{k,con,n}(X^w_n) \pm \hat{f}_{k,con}(X^w_n, \beta_\varepsilon)$, where $\beta_\varepsilon \in (0, \infty)$ is a parameter. The $\beta_\varepsilon$ parameter controls how large the standard error intervals are; a smaller $\beta_\varepsilon$ value results in wider intervals. Once both sets of $\varepsilon$-constraint values have been determined, Steps 13 and 14 select the objective with the least $\varepsilon$’s to minimize, where ties are broken randomly.

If the objective chosen to minimize results in one or more $\varepsilon$-constraint problems, Steps 15 through 25 solve these problems. First, PE partitions the objective space based on the $\varepsilon$ values. Then, each $\varepsilon$-constraint problem is solved using the SPLINE algorithm (see §5.3). Within a partition, once a new sample-path $N_a$-LWEP is found, PE performs what we call a traceback in Step 24 by attempting to place a new $\varepsilon$ value that is both within the current partition and that sets as infeasible the newly-found sample-path $N_a$-LWEP, as shown in Figure 2. If the new $\varepsilon$ value is outside the partition, the search ends; otherwise, the new $\varepsilon$-constraint problem is solved, and this process repeats until no more $\varepsilon$ values can be placed.
in the current partition. The new sample-path $\mathcal{N}_a$-LWEP’s found across all partitions are collected into a set of sample-path $\mathcal{N}_a$-LWEP’s for Problem $\overline{M}_{2,n}$ and returned by PE in Step 26. We remark here that as the RA iteration number $\nu \rightarrow \infty$ in R-PERLE, under appropriate regularity conditions, the PE algorithm solves as many $\varepsilon$-constraint problems as there are points in the $\mathcal{N}_a$-local efficient set to which the algorithm converges (see §8).

5.2. The Min Algorithm for Many Objectives

The Min algorithm for $d \geq 2$ objectives, called in PE Step 1 and listed in Algorithm 3, is a relatively simple algorithm that takes in any set of feasible points, brings up the sample sizes, updates the sample-path $\mathcal{N}_a$-local minimizers on each objective, and removes any points whose images are sample-path dominated. The resulting set of sample-path non-dominated points and sample-path $\mathcal{N}_a$-local minimizers are returned as the set $\hat{A}^*_n$.

5.3. The SPLINE Algorithm for One Objective

The SPLINE algorithm (Wang et al. 2013) is the engine underlying all of our single-objective searches, that is, solving the $\varepsilon$-constraint problems in PE Step 22 and finding sample-path $\mathcal{N}_a$-local minimizers in Min Step 3. SPLINE (Algorithm 4) finds a sample-path $\mathcal{N}_a$-local minimizer of objective $k$ on a feasible set $\tilde{X} \subseteq \mathcal{X}$ using sample size $n$. Our version of SPLINE contains minor modifications that allow us to input an objective to minimize and additional bound constraints on the search space, specified by $\mathcal{G}$, that define $\tilde{X}$. Our ability to pass additional bound constraints to SPLINE enables us to restrict the search space when solving $\varepsilon$-constrained problems in PE. We also output the search trajectory for later use.

The SPLINE algorithm consists of two primary steps: SPLI and NE, which are called iteratively until a sample-path $\mathcal{N}_a$-local minimizer is found, or the search times out. The SPLI algorithm conducts a pseudogradient-based line search with piecewise linear interpolation. The NE algorithm performs neighborhood enumeration to either move to a better
Algorithm 4: \([X^*, T, \mathcal{N}(X^*)] = SPLINE(k, X_0, n, b, G)\)

**Input:** objective \(k\); initial point \(X_0 \in \mathcal{X} \subseteq \mathcal{X}\); sample size \(n\); limit on search oracle calls, \(b\); optional bound constraint values \(G\) that define \(\mathcal{X} \subseteq \mathcal{X}\).

**Output:** local solution \(X^*\) on \(\mathcal{X} \subseteq \mathcal{X}\); sample-path search set, \(T\); neighborhood points, \(\mathcal{N}(X^*)\).

1. Initialize: search oracle calls spent so far \(b^* \leftarrow 0\), \(X_{NE} \leftarrow X_0\), and \(T \leftarrow \{X_0\}\).
2. repeat
   3. \([b', X_{SPLI}, G_n(X_{SPLI})] = SPLINE(k, X_{NE}, n, b, G)\)  \(\text{/search: line search with interpolation}\)
   4. if \(G_{k,n}(X_{SPLI}) > G_{k,n}(X_{NE})\) then \(X_{SPLI} \leftarrow X_{NE}\) \(\text{/SPLI cannot cause harm}\)
   5. \([b'', X_{NE}, G_n(X_{NE}), \mathcal{N}(X_{NE})] = NE(k, X_{SPLI}, n, G)\) \(\text{/neighborhood enumeration}\)
   6. \(T \leftarrow T \cup \{X_{SPLI}, X_{NE}\}\) \(\text{/update trajectory}\)
   7. \(b' \leftarrow b^* + b''\) \(\text{/update oracle calls expended}\)
   8. until \(G_{k,n}(X_{NE}) = G_{k,n}(X_{SPLI})\) or \(b^* > b\) \(\text{/find a local solution or time out}\)

neighborhood point, or to certify a local minimum has been found. We refer the reader to Wang et al. (2013) for detailed listings and further explanations of SPLI and NE.

6. The RLE Algorithm for Many Objectives

The RLE algorithm is the second algorithm in the PERLE solver. The collection of sample-path \(N_a\)-LWEP’s found by the PE algorithm is sent to RLE to certify that this collection of points is indeed an ALES, or to create an ALES using this collection of points as an initial set. Without RLE to certify an ALES, an algorithm like PE that relies only on collecting sample-path \(N_a\)-LWEP’s may “get stuck” by returning points that do not belong to the same, or to any, sample-path \(N_a\)-local weakly efficient set. For example, in Figure 1, the PE algorithm may return \(S = \{x_{1min}, x^*, x_{2max}\}\), which is a set of \(N_a\)-LWEP’s containing a minimum on each objective but that is not an \(N_a\)-local weakly efficient set. The RLE algorithm is designed to crawl out of the sample-path version of this scenario when the completeness function is small enough. In what follows, we first define a key concept used in RLE called the sample-path non-conforming neighborhood. Then, we discuss the RLE algorithm in detail.

6.1. The Sample-Path Non-Conforming Neighborhood

Suppose we have a set of points \(S \subseteq \mathcal{X}\) such that none of the estimated images of points in \(S\) dominate the estimated images of other points in \(S\). The non-conforming neighborhood of \(S\) is the set of points in the deleted neighborhood of \(S\) that prevent it from being an ALES.

**Definition 8.** Let \(S \subseteq \mathcal{X}\) be a collection of feasible points such that no points in \(G_n(S)\) dominate other points in \(\bar{G}_n(S)\). Then given a completeness function \(\delta: \mathcal{X} \to \mathbb{R}^d\) such that \(0 \leq \delta(x)\) for all \(x \in \mathcal{X}\), define the sample-path non-conforming neighborhood (NCN) of \(S\), \(N^{nc}_a(S)\), as all feasible points in the deleted neighborhood of \(S\), \(x \in N^{nc}_a(S) \cap \mathcal{X}\), such that

(a) \(\exists \bar{x} \in S\) such that \(x \in N_a(\bar{x})\) and \(\bar{G}_n(x) < G_n(\bar{x})\), or
Figure 3  Let \( S = \{x\} \) with two objectives, and let \( x \in \mathcal{N}(S) \cap \mathcal{X} \) be in the deleted neighborhood of \( S \). Def. 8(a) adds \( x \) to the NCN if \( x \in \mathcal{N}(\bar{x}) \) and \( \bar{G}(x) \) is in the light gray area. If \( x \) does not satisfy Def. 8(a), Def. 8(b) adds \( x \) to the NCN if its \( \delta \) box, defined by corners \( \bar{G}(x) \pm \delta(x) \), is contained in the dark gray area.

First, Definition 8(a) adds \( x \in \mathcal{N}(\bar{x}) \subseteq \mathcal{X} \) to the NCN if it prevents a point in \( S \) from being a sample-path \( \mathcal{N}(\bar{x}) \)-LWEP. Definition 8(b) also adds a feasible deleted-neighborhood point to the NCN if it violates the conditions of Definition 7(b), that is, the point (i) is not weakly dominated by any points in \( S \), and (ii) does not dominate any points in \( S \) by less than a certain amount, and (iii) weakly dominates a point in \( S \), or would not either weakly dominate or be weakly dominated by a point in \( S \) if both were moved by a certain amount.

Given a single point \( S = \{x\} \subseteq \mathcal{X} \), Figure 3 shows regions of the objective space that correspond to a feasible point in the deleted neighborhood of \( S \), \( x \in \mathcal{N}(\bar{x}) \cap \mathcal{X} \), being declared a member of the NCN. Definition 8(a) implies \( x \) is in the NCN if it is an \( \mathcal{N}(\bar{x}) \)-neighbor of \( \bar{x} \) and \( \bar{G}(x) \) is in the light gray region of Figure 3. If \( x \) does not meet the requirements of Definition 8(a), Definition 8(b) implies that \( x \) is in the NCN if its entire "\( \delta \) box," defined by the corners \( \bar{G}(x) \pm \delta(x) \), is completely contained in the dark gray shaded region of Figure 3. Thus there is no overlap between the \( \delta \) boxes of \( x \) and \( \bar{x} \) on any objective. Henceforth, we use \( \delta(x) = \bar{f}(x, \beta) \) for all \( x \in \mathcal{X} \) (see §3.2) as the NCN completeness function.

6.2. RLE Algorithm Listing
We now discuss RLE (Algorithm 5) in detail. To guarantee an ALES, in Step 1, RLE first removes any points in \( S \) whose estimated objective vectors are dominated by the estimated objective vectors of other points in \( S \). Then, in Step 2, RLE calculates the NCN of the remaining points using the function GetNCN. If the NCN is empty in Step 2, then RLE certifies an ALES and the algorithm terminates; otherwise, RLE enters a search phase.
Algorithm 5: $\mathcal{A}_{\text{ALES}} = \text{RLE}(\mathcal{S}, n, b, \beta_\delta)$

**Input:** set of points $\mathcal{S} \subset \mathcal{X}$; limit on search oracle calls, $b$; ALES completeness parameter $\beta_\delta$

**Output:** $\mathcal{A}_{\text{ALES}}$, which is an ALES for the sample-path problem with sample size $n$

1. $\mathcal{S} = \text{RemoveDominated}(\mathcal{G}_n(\mathcal{S} \cup \{x_0\}))$
2. $\mathcal{N}^{\text{nc}} = \text{GetNCN}(\mathcal{S}, \beta_\delta)$ /get non-conforming neighborhood
3. Initialize: outer search oracle calls spent so far $b^* \leftarrow 0$
4. while $b^* \leq b$ and $\mathcal{N}^{\text{nc}} \neq \emptyset$ do /SEARCH: traverse sample-path $\mathcal{N}_a$-LWEP chains
5. $[b', \mathcal{N}^{\text{nc}}, \mathcal{N}^*_2] = \text{RemoveNonLWEP}(\mathcal{N}^{\text{nc}})$
6. $\mathcal{S} \leftarrow \mathcal{S} \cup \mathcal{N}^*$ /add neighborhood sample-path $\mathcal{N}_a$-LWEP’s to $\mathcal{S}$
7. if $\mathcal{N}^w = \emptyset$ then /the neighbors are dominated by their neighbors: $\mathcal{N}^*_2 \neq \emptyset$
   
   Initialize: $\mathcal{X}^\text{new} \leftarrow \mathcal{N}^*_2$, $\mathcal{X}^w \leftarrow \emptyset$, and inner search oracle calls spent so far $b^{**} \leftarrow 0$
8. while $b^{**} \leq b$ and $\mathcal{X}^w = \emptyset$ do /SEARCH: traverse dominating chains
9. $[b'', \mathcal{X}^w, \mathcal{N}^*_2] = \text{RemoveNonLWEP}(\mathcal{X}^\text{new})$
10. $\mathcal{X}^\text{new} \leftarrow \mathcal{N}^*_2$ and $b^{**} \leftarrow b^{**} + b''$
11. $\mathcal{S} \leftarrow \mathcal{S} \cup \mathcal{X}^w$ /keep progress if search times out
12. if $\mathcal{X}^w = \emptyset$ then $\mathcal{S} \leftarrow \mathcal{S} \cup \mathcal{X}^\text{new}$
13. $\mathcal{S} = \text{RemoveDominated}(\mathcal{G}_n(\mathcal{S} \cup \{x_0\}))$
14. $[b'', \mathcal{N}^{\text{nc}}] = \text{GetNCN}(\mathcal{S}, \beta_\delta)$ /get non-conforming neighborhood
15. $b^* \leftarrow b^* + b' + b''$

17. return $\mathcal{A}_{\text{ALES}} \leftarrow \mathcal{S}$

The “outer” RLE search phase, which begins in Step 4, checks to see if any members of the NCN are also sample-path $\mathcal{N}_a$-LWEP’s using RemoveNonLWEP. The RemoveNonLWEP function takes an input set of feasible points and outputs three quantities: (a) the number of simulation replications expended, (b) the set of sample-path $\mathcal{N}_a$-LWEP’s in the input set, and (c) a set of points in the neighborhood of the input set whose images sample-path dominate the images of the members of the input set. Thus when RemoveNonLWEP is passed a non-empty NCN, it enumerates the neighborhood of the NCN; these points are neighbors of neighbors of the original set. If the NCN contains sample-path $\mathcal{N}_a$-LWEP’s in Step 5, RLE adds them to $\mathcal{S}$ in Step 6. If no members of the NCN are sample-path $\mathcal{N}_a$-LWEP’s in Step 7, there must exist neighbors of the NCN that dominate points in the NCN, denoted as $\mathcal{N}^*_2$ in Steps 5 and 8. If this is the case, RLE enters an “inner” search phase.

The “inner” RLE search phase, which begins in Step 9, allows the algorithm to find new sample-path $\mathcal{N}_a$-LWEP’s by traversing points whose estimated images are sample-path dominated. Once a sample-path $\mathcal{N}_a$-LWEP is found or the inner search times out, the new points are added to $\mathcal{S}$ in Steps 12 and 13. After removing points whose images are sample-path dominated by the images of other points in $\mathcal{S}$, in Step 15, RLE checks the new set $\mathcal{S}$ to see if it is an ALES. If not, this process repeats until a complete ALES is found, or a total outer
search budget has been exhausted. Since the search budget sequence is non-binding in the limit, for large enough RA iteration number $\nu$, RLE guarantees an ALES.

7. Other Algorithms: R-PE and R-MinRLE

To assess and understand the performance of R-PERLE, we find it helpful to define and analyze two other RA algorithms: R-PE and R-MinRLE, which we discuss in this section. We discuss R-PE for two objectives first, followed by R-MinRLE for many objectives.

First, we define the R-PE algorithm for two objectives as identical to R-PERLE (Algorithm 1), except without the call to RLE in Step 4. To show that the PE algorithm delivers “good points” to the RLE algorithm, in §8, we prove convergence of the R-PE algorithm under a set of fairly restrictive assumptions; these assumptions are violated in the example in Figure 1. R-PE is also useful for numerically analyzing the $\beta_\epsilon$ parameter in §9.3.

The R-MinRLE algorithm, listed in Algorithm 6, is arguably the most general algorithm we propose, since it is defined for $d \geq 2$ objectives. R-MinRLE is like R-PERLE except that instead of obtaining a set of sample-path $\mathcal{N}_a$-LWEP’s from PE on each RA iteration, R-MinRLE uses the Min algorithm (Algorithm 3) to update the sample-path $\mathcal{N}_a$-local minimizer on each objective before invoking RLE. We define the sample-path solver MinRLE as calling Min followed by RLE within an RA iteration; the MinRLE algorithm locates an ALES for sample-path Problem $\overline{\mathcal{M}}_{d,m_\nu}$ for $d \geq 2$ and each $\nu = 1, 2, \ldots$.

Loosely speaking, for two objectives, notice that R-MinRLE is likely to exhibit “outside-in” convergence behavior. That is, because R-MinRLE only guarantees locating the sample-path $\mathcal{N}_a$-local minimizers on each RA iteration, if the completeness parameter $\beta_\delta$ is “small” so that RLE crawls less, MinRLE locates the sample-path $\mathcal{N}_a$-local minimizers and perhaps a few points nearby to complete an ALES. All externalities being equal, this ALES is less likely to contain points that map to the “center” of a sample-path $\mathcal{N}_a$-local Pareto set than a

---

**Algorithm 6:** The R-MinRLE Algorithm for $d \geq 2$

<table>
<thead>
<tr>
<th>Input: initial point $x_0 \in \mathcal{X}$; sequence of sample sizes to expend at each visited point, ${m_\nu}$; sequence of limits on oracle calls during search, ${b_\nu}$; ALES completeness parameter, $\beta_\delta$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initialize: $\mathcal{A}_0 = {x_0}$ and set $x_0$ as a global variable</td>
</tr>
<tr>
<td>for $\nu = 1, 2, \ldots$ with CRN do</td>
</tr>
<tr>
<td>$\mathcal{A}<em>{\min} = \text{Min}(\mathcal{A}</em>{\nu-1}, m_\nu, b_\nu)$</td>
</tr>
<tr>
<td>$\mathcal{A}<em>{\nu} = \text{RLE}(\mathcal{A}</em>{\min}, m_\nu, b_\nu, \beta_\delta)$</td>
</tr>
<tr>
<td>/update the sample-path $\mathcal{N}_a$-local minimizers</td>
</tr>
<tr>
<td>/guarantee the returned set is an ALES</td>
</tr>
</tbody>
</table>
corresponding ALES located by PERLE. Therefore in our numerical experiments, comparisons with R-MinRLE demonstrate the usefulness of using the PE algorithm as a precursor to RLE within an RA iteration, as opposed to the naïve Min algorithm.

8. Asymptotic Behavior

We now study the asymptotic behavior of our RA algorithms and show that, under appropriate regularity conditions, R-PERLE, R-PE, and R-MinRLE converge to an $N_a$-local efficient set w.p.1. In what follows, first, we discuss the assumptions required for our results. Then, we prove the convergence of RA algorithms that rely on RLE. The proof is general in the sense that, under our regularity conditions, any RA algorithm with a sample-path solver consisting of an accelerator that returns a candidate ALES in finite time and invokes RLE last using the completeness function $\hat{\delta} = \hat{f}(x, \beta_\delta)$ with $\beta_\delta \in (0, \infty]$ will converge. We also prove the convergence of R-PE under fairly restrictive assumptions. Finally, in §8.4, we provide sampling efficiency results. Throughout this section, we assume the search budget sequence $\{b_\nu, \nu = 1, 2, \ldots\}$ is non-binding w.p.1 for all $\nu$ large enough, under our regularity conditions.

8.1. Preliminaries and Assumptions

We require several regularity conditions on both the true, unknown objective functions and the sample-path objective functions. We require Assumptions 1–3 in all of our results.

Assumption 1. (Wang et al. 2013, p. 12) For each $x \in \mathcal{X}$ and $k \in \{1, \ldots, d\}$, there exists $\alpha_k > 0$, dependent on $x$, such that $S_k(x, \alpha_k) := \{x' \in \mathcal{X}: g_k(x') \leq g_k(x) + \alpha_k\}$ is finite.

Assumption 2. (Wang et al. 2013, p. 12) For each $k \in \{1, \ldots, d\}$, we assume the following. Let $S_k(x, \alpha_k)$ be as in Assumption 1, and define $\hat{S}_{k, \nu}(x) := \{x' \in \mathcal{X}: \hat{G}_{k, m_\nu}(x') \leq \hat{G}_{k, m_\nu}(x)\}$ for all $\nu = 1, 2, \ldots$. Given $x \in \mathcal{X}$, there exists a sequence $\{p_\nu\}_\nu^\infty$ such that $\mathbb{P}\{\hat{x} \in \hat{S}_{k, \nu}(x)\} \leq p_\nu$ and $\sum_{\nu=1}^\infty p_\nu < \infty$ for all $x \in \mathcal{X} \setminus S_k(x, \alpha_k)$.

Assumption 3. All variances are finite, that is, $\max_{k \in \{1, \ldots, d\}} \sigma^2_k(x) < \infty$ for all $x \in \mathcal{X}$.

First, Assumption 1 implies that at each feasible point $x \in \mathcal{X}$ and for every objective $k$, there exists a constant such that the level set created by adding the constant to $g_k(x)$ is finite. Since this property holds for every objective $k$, then under Assumption 1, the union of the level sets over the objectives $k$ is also finite. That is, for each $x \in \mathcal{X}$, define $\alpha := (\alpha_1, \ldots, \alpha_d)$, where $\alpha$ also depends on $x$. Define the set $S(x, \alpha) := \bigcup_{k=1}^d S_k(x, \alpha_k)$ as the set of all feasible
points that map to objective values that are not strictly dominated by the point \( g(x) + \alpha \); notice that \( g(S^c(x, \alpha)) = \{ g(x) + \alpha \} + \{ y \in \mathbb{R}^d : 0_d < y \} \) is the set of points that \( g(x) + \alpha \) strictly dominates for all \( x \in \mathcal{X} \). Then under Assumption 1, \( S(x, \alpha) \) is finite for all \( x \in \mathcal{X} \).

Assumption 1 implies the nonempty global weakly efficient set exists and all \( \mathcal{N}_a \)-local weakly efficient sets are finite. We present this result in Lemma 1 without proof; we first define the following notation. Given a neighborhood size \( a \in (0, \infty] \), let \( \mathcal{W}_a \) and \( \mathcal{L}_a \) be the collection of all possible \( \mathcal{N}_a \)-local weakly efficient sets and \( \mathcal{N}_a \)-local efficient sets for Problem \( M_d \), respectively, where \( \mathcal{L}_a \subseteq \mathcal{W}_a \). Since the global weakly efficient set is also an \( \mathcal{N}_a \)-local weakly efficient set for \( a \geq 1 \), notice that \( |\mathcal{W}_a| \geq 1 \) if the global weakly efficient set exists.

**Lemma 1.** Under Assumption 1, given \( a \geq 1 \), the following hold:

(a) The global weakly efficient set, \( \mathcal{E}^w \subseteq \mathcal{X} \), exists and is nonempty.

(b) All \( \mathcal{N}_a \)-local weakly efficient sets are finite; that is, \( 1 \leq |\mathcal{W}_a| < \infty \) for all \( \mathcal{W}_a \in \mathcal{W}_a \).

Now, let us turn our attention to Assumption 2, which is a condition defined by Wang et al. (2013) ensuring that the probability of incorrectly estimating a level set decays sufficiently fast. Wang et al. (2013) provide a detailed discussion of the conditions under which Assumption 2 holds. For completeness, we include the conditions below as Lemma 2; recall that the variance \( \sigma_k^2(x) \) is \( \mathbb{V}(G_k(x, \xi)) \) for each \( x \in \mathcal{X} \) and objective \( k \in \{1, \ldots, d\} \). Essentially, Lemma 2 implies that Assumption 2 holds under a large-deviations regime or under the conditions of the Central Limit Theorem, whenever the sample size sequence increases at a sufficiently fast rate. For compactness, we refer the reader to Wang et al. (2013) for additional explanation of Assumption 2 and Lemma 2.

**Lemma 2.** (see Wang et al. 2013, p. 13–14) Assumption 2 holds if one of the following two sets of conditions holds:

**C1.** (a) for all \( k \in \{1, \ldots, d\} \), the sequence of random variables \( \{ \bar{G}_{k,m_\nu}(x) - g_k(x) \} \) is governed by a large-deviation principle with rate function \( I_{k,x}(s) \) (Dembo and Zeitouni 1998); (b) each \( I_{k,x}(s) \) is such that for any \( \epsilon > 0 \), \( \inf_{x \in \mathcal{X}, k \in \{1, \ldots, d\}} \min(I_{k,x}(\epsilon), I_{k,x}(-\epsilon)) = \eta > 0 \); and (c) the sequence of sample sizes \( \{m_\nu\} \) increases faster than logarithmically, that is, \( \limsup_{\nu \rightarrow \infty} (m_\nu)^{-1} (\log \nu)^{1+\Delta_1} = 0 \) for some \( \Delta_1 > 0 \).

**C2.** (a) for all \( k \in \{1, \ldots, d\} \), a central limit theorem holds on the sequence of random variables \( \{ \bar{G}_{k,m_\nu}(x) \} \) for each \( x \in \mathcal{X} \), that is, \( \sqrt{m_\nu}(\sigma_k(x))^{-1}(\bar{G}_{k,m_\nu}(x) - g_k(x)) \Rightarrow Z \), where \( \sigma_k(x) > 0 \) satisfies \( \sup_{x \in \mathcal{X}} \sigma_k^2(x) < \infty \), and (b) as \( \nu \rightarrow \infty \), \( \sup_y |F_{k,x,m_\nu}(y) - \Phi(y)| = \)
\[ O(1/\sqrt{m_n}) \text{ for all } \mathbf{x} \in \mathcal{X}, \text{ where } F_{k,\mathbf{x},m_n}(\cdot) \text{ denotes the cumulative distribution function of} \]
\[ \sqrt{m_n}(\sigma_k(\mathbf{x}))^{-1}(\mathcal{G}_{k,m_n}(\mathbf{x}) - g_k(\mathbf{x})), \text{ and (c) the sequences of sample sizes } \{m_n\} \text{ satisfies} \]
\[ \limsup_{n \to \infty} (m_n)^{-1/2} = 0 \text{ for some } \Delta^2 > 0. \]

The primary implication of Assumptions 1 and 2 is the convergence of the estimated level sets into the true level sets, as described in the following Lemma 3. Before we present the lemma, recall that \( S(\mathbf{x}, \alpha) = \bigcup_{k=1}^{d} S_k(\mathbf{x}, \alpha_k) \) is finite, and define \( \hat{S}_\nu(\mathbf{x}) := \bigcup_{k=1}^{d} \hat{S}_{k,\nu}(\mathbf{x}) \) as the set of all decision points estimated as being at least as good as \( \mathbf{x} \) on at least one objective.

A proof sketch for Lemma 3 appears in the Online Appendix.

**Lemma 3.** Under Assumptions 1 and 2,

(a) (see Wang et al. 2013, p. 15) for each \( k \in \{1, \ldots, d\} \) and any \( \mathbf{x} \in \mathcal{X} \), the sets \( \hat{S}_{k,\nu}(\mathbf{x}) \) converge almost surely into the set \( S_k(\mathbf{x}, \alpha_k) \), that is, \( \mathbb{P}\{\hat{S}_{k,\nu}(\mathbf{x}) \not\subseteq S_k(\mathbf{x}, \alpha_k) \text{ i.o.}\} = 0 \);

(b) the sets \( \hat{S}_\nu(\mathbf{x}) \) converge almost surely into the sets \( S(\mathbf{x}, \alpha) \) for any \( \mathbf{x} \in \mathcal{X} \), that is, \( \mathbb{P}\{\hat{S}_\nu(\mathbf{x}) \not\subseteq S(\mathbf{x}, \alpha) \text{ i.o.}\} = 0 \).

Finally, we need our last required assumption, Assumption 3, because our \( \varepsilon \)-placement and ALES completeness parameters rely on the estimated standard errors of the objective function values. Notice that Assumption 3 is implied under the conditions of Lemma 2.

In addition to Assumptions 1–3 discussed above, some of our results require additional structure on the true, unknown objective functions in Problem \( M_d \). We present these assumptions as Assumptions 4–6, and then we discuss their implications.

**Assumption 4.** For all \( \mathbf{x}, \mathbf{x}' \in \mathcal{X} \), if \( g(\mathbf{x}) = g(\mathbf{x}') \), then \( \mathbf{x} = \mathbf{x}' \).

**Assumption 5.** There exists \( \kappa > 0 \) such that \( \min_{k \in \{1, \ldots, d\}} \inf \{|g_k(\mathbf{x}) - g_k(\mathbf{x}')| : \mathbf{x}, \mathbf{x}' \in \mathcal{X}, \mathbf{x} \neq \mathbf{x}'\} > \kappa \).

**Assumption 6.** Given \( a \in [1, \infty) \), all \( N_a \)-LWEP’s are global efficient points and there exists exactly one \( N_a \)-local efficient set that solves Problem \( M_d \) which is also the global efficient set, \( \mathcal{E} \).

Assumption 4 ensures that two or more decision points in the feasible space do not map to the same point in the objective space. Under this assumption, the following Lemma 4 holds regarding the existence of an \( N_a \)-local efficient set within each \( N_a \)-local weakly efficient set. We present the result without proof; intuitively, it follows because Assumption 4 prevents the points in the \( N_a \)-local weakly efficient set from having identical objective vector values. Thus the set must contain \( N_a \)-LEP’s, from which the \( N_a \)-local efficient set can be constructed.
Lemma 4. Under Assumptions 1 and 4, given \( a \in (0, \infty) \), all \( N_a \)-local weakly efficient sets contain an \( N_a \)-local efficient set; that is, for each \( W_a \in \mathcal{W}_a \), \( \exists L_a^* \in \mathcal{L}_a \) such that \( W_a \supseteq L_a^* \).

Finally, we remark on Assumptions 5 and 6. Assumption 5, which subsumes Assumption 4, ensures that each feasible point is distinguishable on each objective. Under this assumption, every \( N_a \)-LWEP is an \( N_a \)-LEP, and every \( N_a \)-local weakly efficient set is an \( N_a \)-local efficient set. Assumption 6 is required for the convergence of R-PE, and stipulates that every \( N_a \)-LWEP is a global efficient point, and there exists exactly one \( N_a \)-local efficient set, which is also the global efficient set. Under this assumption, the R-PE algorithm cannot “get stuck” by returning parts of different \( N_a \)-local efficient sets.

8.2. Convergence of R-PERLE and R-MinRLE

We now consider the convergence of the algorithms R-PERLE and R-MinRLE under the regularity conditions discussed in the previous section. These algorithms invoke RLE to guarantee that the set of points returned at the end of each RA iteration is an ALES. Theorem 1 and its proof are presented for \( d \geq 2 \) since R-MinRLE converges for two or more objectives; the proof appears in the Online Appendix. Given appropriate parameter values, the proof of convergence of Theorem 1 holds for any RA algorithm with a sample-path solver that ensures the accelerator returns a set in finite time and invokes RLE last.

Theorem 1. Let Assumptions 1–3 hold. For any neighborhood size \( a \in [1, \infty) \), initial point \( x_0 \in \mathcal{X} \), \( \varepsilon \)-placement rule \( \beta_\varepsilon \in (0, \infty) \), and completeness parameter \( \beta_\delta \in (0, \infty] \), R-PERLE \((d = 2)\) and R-MinRLE \((d \geq 2)\) generate a sequence of estimated solutions \( \{\hat{A}_\nu\} \) such that

(a) \( \{\hat{A}_\nu\} \) converges into an \( N_a \)-local weakly efficient set almost surely, that is, \( \exists W_a \in \mathcal{W}_a \) such that \( \mathbb{P}\{\hat{A}_\nu \not\subseteq W_a \text{ i.o.}\} = 0 \);

(b) under Assumption 4, \( \{\hat{A}_\nu\} \) contains an \( N_a \)-local efficient set infinitely often almost surely, that is, \( \exists L_a \in \mathcal{L}_a \) such that \( \mathbb{P}\{L_a \not\subseteq \hat{A}_\nu \text{ i.o.}\} = 0 \);

(c) under Assumption 5, \( \{\hat{A}_\nu\} \) converges to an \( N_a \)-local efficient set almost surely, that is, \( \exists L_a \in \mathcal{L}_a \) such that \( \mathbb{P}\{\hat{A}_\nu \neq L_a \text{ i.o.}\} = 0 \).

(d) under Assumptions 5 and 6, \( \{\hat{A}_\nu\} \) converges to the global efficient set almost surely, that is, \( \mathbb{P}\{\hat{A}_\nu \not\subseteq \mathcal{E} \text{ i.o.}\} = 0 \).

Theorem 1 presents a series of convergence results that require increasingly stringent assumptions on Problem \( M_d \). At a minimum, under our required Assumptions 1–3, R-PERLE and R-MinRLE converge into an \( N_a \)-local weakly efficient set almost surely.
8.3. Convergence of R-PE

We now consider the convergence of R-PE, which does not rely on RLE to certify that each RA iteration returns an ALES. To show the convergence of R-PE in Theorem 2, first, we notice that for each objective $k \in \{1, 2\}$, the sequence of sample-path $N_a$-local minimizers produced by PE in Step 1 across RA iterations, defined as $\{ \mathcal{M}_\nu, \nu = 1, 2, \ldots \}$ where $\mathcal{M}_\nu = \{ \mathbf{X}_{min}^{1, m_\nu}, \mathbf{X}_{min}^{2, m_\nu} \}$ for all $\nu = 1, 2, \ldots$, converges into the set of all true $N_a$-local minimizers of objective $g_k$ over the feasible set $\mathcal{X}$, $\mathcal{M}_a^* \subseteq \mathcal{X}$, almost surely as $\nu \to \infty$. Since this result, presented in Lemma 5, follows almost directly from Wang et al. (2013) under Assumptions 1 and 2, we do not provide a proof. The proof of convergence of R-PE, which requires our most restrictive assumptions on the underlying Problem $M_2$, appears in the Online Appendix.

**Lemma 5.** Under Assumptions 1 and 2, for $d = 2$, any neighborhood size $a \in [1, \infty)$, initial point $x_0 \in \mathcal{X}$, and $\varepsilon$-placement rule, across RA iterations, PE Step 1 generates a sequence of sample-path $N_a$-local minimizers $\{ \mathcal{M}_\nu \}$ that converges into $\mathcal{M}_a^*$ almost surely, that is, $\mathbb{P}\{ \mathcal{M}_\nu \not\subseteq \mathcal{M}_a^* \text{ i.o.} \} = 0$.

**Theorem 2.** Under Assumptions 1–6, for $d = 2$, any neighborhood size $a \in [1, \infty)$, initial point $x_0 \in \mathcal{X}$, and $\varepsilon$-placement rule specified by $\beta_\varepsilon \in (0, \infty)$, R-PE generates a sequence of estimated solutions $\{ \hat{\mathcal{A}}_\nu \}$ that converges almost surely to the global efficient set $\mathcal{E}$, in the sense that $\mathbb{P}\{ \hat{\mathcal{A}}_\nu \neq \mathcal{E} \text{ i.o.} \} = 0$.

These results imply that R-PE provides “good” starting points to RLE. If $\beta_\varepsilon = \beta_\delta$ and Assumptions 1–6 hold, RLE should be mostly inactive in R-PERLE.

8.4. Sampling Efficiency

Finally, we provide a result on the sampling efficiency of our algorithms. This result provides insight into how to set the algorithm parameter values in §9.1 to achieve exponential convergence. In Theorem 3, let $\mathcal{X}^w$ denote the set of all $N_1$-LWEP’s for Problem $M_d$, and let $\mathcal{X}_\nu^w$ denote the set of all sample-path $N_1$-LWEP’s on the $\nu$th RA iteration. Further, let $\hat{\mathcal{A}}_\nu$ denote the solution returned on the $\nu$th RA iteration of R-PERLE ($d = 2$), R-PE ($d = 2$), or R-MinRLE ($d \geq 2$) for any $x_0 \in \mathcal{X}$, $\varepsilon$-placement rule $\beta_\varepsilon \in (0, \infty)$, and completeness parameter $\beta_\delta \in (0, \infty]$. A proof of Theorem 3 appears in the Online Appendix.

**Theorem 3.** Let the neighborhood size $a = 1$ and suppose the feasible set $\mathcal{X} \subset \mathbb{Z}^q$ is finite with $\max_{k \in \{1, \ldots, d\}} \sup_{x \in \mathcal{X}} \sigma_k^2(x) < \infty$. For all objectives $k \in \{1, \ldots, d\}$, let the sequence of
random variables \( \{ \bar{G}_{k,m,v}(x) - g_k(x) \} \) be governed by a large-deviation principle with rate function \( I_{k,x}(s) \), as stipulated in Lemma 2. Then the following hold:

(a) \( \mathbb{P}\{ \bar{X}_w \not\subseteq X^w \} = O(e^{-\gamma m_v}) \) for some \( \gamma > 0 \).

(b) If the sequence of sample sizes increases to infinity at least linearly in R-PERLE, R-PE, and R-MinRLE, that is, if \( \limsup_{v \to \infty} m_v^{-1} v < \infty \), then

(i) \( \mathbb{P}\{ \hat{A}_v \not\subseteq X^w \} = O(e^{-\gamma m_v}) \) for some \( \gamma > 0 \),

(ii) under Assumption 5 and 6, \( \mathbb{P}\{ \hat{A}_v \neq \mathcal{E} \} = O(e^{-\gamma m_v}) \) for some \( \gamma > 0 \).

9. Numerical Experiments

We conduct numerical experiments to compare our main algorithm, R-PERLE, to our benchmark algorithm, R-MinRLE, and to the current state-of-the-art, MO-COMPASS. First, in §9.1, we discuss algorithm implementation and parameters. Then, in §9.2, we compare the performance of these algorithms on three test problems with known solutions. In §9.3, we explore the performance of R-PERLE on the same three test problems across a variety of \( \beta = (\beta_\epsilon, \beta_\delta) \) values. To demonstrate our algorithms on a real-world SO problem, we also create a new bi-objective bus scheduling problem based on the single-objective version in Wang et al. (2013). We run R-PERLE on the bi-objective 9-bus problem, for which the \( \mathcal{N}_1 \)-local weakly efficient sets are unknown. A description of the bi-objective bus scheduling problem and all corresponding numerical results appear in the Online Appendix.

Finally, while the overhead required to run a MOSO algorithm is often considered negligible relative to the computational time required to obtain one simulation replication, there may be scenarios in which the relative computational overhead of each algorithm becomes a consideration. The Online Appendix contains information on the computational time required to run our algorithms and MO-COMPASS on our test problems.

9.1. R-PERLE and R-MinRLE Implementation and Parameters

While our definitions and algorithms allow a flexible neighborhood size \( a \), Wang et al. (2013) note the tension between the faster convergence enabled by \( a = 1 \) and the certification of the local solution as optimal in a larger neighborhood. By default, we set \( a = 1 \).

To improve algorithmic efficiency, recall from §4 that we store \( (x, \bar{G}_n(x), s, \bar{e}, (\bar{G}_n(x))) \) at all feasible points visited within an RA iteration; we clear this list between RA iterations. We remark here that we could also modify the algorithms GetNCN and the version of RemoveNonLWEP implemented in RLE to return only a subset of the non-conforming
points encountered in the neighborhood and the first sample-path \( N_\varepsilon \)-LWEP encountered, respectively, rather than all such points. (Note that we require a full RemoveNonLWEP for use in PE Step 2.) Further, we could update the initial point, \( x_0 \), to a sample-path \( N_\varepsilon \)-LWEP a finite number of times before fixing it at a particular value for the remainder of the algorithm. These modifications would not affect the convergence properties of the algorithm and may improve algorithmic efficiency, especially in higher dimensions. In the current numerical experiments, we implement the algorithms as written in the pseudocode.

In the numerics that follow, we set the monotone-increasing sample size sequence as \( m_\nu = \lceil 2 \times 1.1^\nu \rceil \) for all \( \nu \geq 1 \). This sequence satisfies the requirements of Lemma 2 and Theorem 3 in §8. To ensure every search terminates in finite time, but that for large enough \( \nu \), the sample size limit \( b_\nu \) will not be reached, we set the sequence \( b_\nu = \lceil 8 \times 1.2^\nu \rceil \) for all \( \nu \geq 1 \). Each search we conduct inside PE, Min, and RLE gets a fresh limiting sample size.

We control the placement of the \( \varepsilon \) values in PE and the completeness of the ALES returned by RLE using the parameters \( \beta_\varepsilon \) and \( \beta_\delta \). There is considerable flexibility in setting these parameters; by default, we use \( \beta_\varepsilon = \beta_\delta = 1/2 \) unless otherwise specified, as in §9.3.

9.2. Algorithm Performance on Test Problems with Known Solutions

We compare the performances of R-PERLE, R-MinRLE, and MO-COMPASS on three increasingly-complicated test problems. These test problems were chosen for their known features and diverse properties. In each independent run of an algorithm, we use an initial point \( x_0 \) that is generated uniformly from the feasible set \( \mathcal{X} \), which is finite in our test problems. Within an algorithm run, we use CRN across points visited. We configure MO-COMPASS, including its Simulation Allocation Rule, as close as possible to Li et al. (2015a, p. 10). In the following plots, the algorithm performance at each value of the total simulation budget \( t \) is dependent on its previous performance.

9.2.1. Test Problem A

Our first test problem is a modified version of a problem that appears in Kim and Ryu (2011). We define Problem \( T_A \) as

\[
\text{Problem } T_A: \quad \text{minimize}_{x \in \mathcal{X}} \left\{ g_1(x) = \mathbb{E}[(x_1/10 - 2\xi_1)^2 + (x_2/10 - \xi_2)^2] \right\} \\
g_2(x) = \mathbb{E}[x_1^2/100 + (x_2/10 - 2\xi_3)^2]
\]

where \( \mathcal{X} = \tilde{\mathcal{X}}_{A1} \times \tilde{\mathcal{X}}_{A2} \) and \( \tilde{\mathcal{X}}_{A1} = \tilde{\mathcal{X}}_{A2} = \{0, 1, 2, \ldots, 50\}, |\mathcal{X}| = 2601, \) and \( \xi_i \) are independent chi-squared random variables with one degree of freedom so that \( \mathbb{E}[\xi_i] = 1 \) and \( \mathbb{V} \mathbb{E}(\xi_i) = 2 \).
for all $i \in \{1, 2, 3\}$. Thus the random objective values returned by the simulation oracle are independent for each $x \in X$. Problem $T_A$ has one $\mathcal{N}_1$-local efficient set which equals the global efficient set, but it also has $\mathcal{N}_1$-LEP’s that do not belong to this set. Problem $T_A$ satisfies only Assumptions 1–3, although Assumption 4 holds for points that are members of the global efficient set. A picture of Problem $T_A$ appears in Figure 4.

We measure the solution quality returned by each algorithm using sample quantiles of the coverage error. The coverage error is defined by Hunter et al. (2019) as the Hausdorff distance between the image of the set returned by the algorithm and the image of the true efficient set as a function of $t$, $d_H(g(\hat{A}(t)), g(\mathcal{E}))$, where $\hat{A}(t)$ denotes the set returned by an algorithm after expending a total of $t$ simulation replications. Figure 5 shows the sample quantiles of the coverage error for 1,000 independent runs each of R-PERLE, R-MinRLE, and MO-COMPASS on Problem $T_A$.

Figure 5 shows that R-PERLE and R-MinRLE out-perform MO-COMPASS on Problem $T_A$. The performances of R-PERLE and R-MinRLE are similar, with R-PERLE performing slightly better for lower simulation budgets $t$.

9.2.2. Test Problem B Our second test problem is a modified version of a test problem that appears in Ryu and Kim (2014). We define Problem $T_B$ as

$$
\text{Problem } T_B: \quad \text{minimize}_{x \in \mathcal{X}} \left\{ \begin{array}{l}
g_1(x) = \mathbb{E}[\xi_1 h_1(x)] \\
g_2(x) = \mathbb{E}[\xi_1 \xi_2 f(x_2) h_1(x_1), f(x_2)]
\end{array} \right.
$$
where $X = X_{B1} \times X_{B2}$ and $X_{B1} = X_{B2} = \{0, 1, \ldots, 100\}, |X| = 10,201$, $h_1(x_1) = 4x_1/100$, and $h_2(h_1, f)$ and $f(x_2)$ are defined as

$$h_2(h_1, f) = \begin{cases} 
1 - (h_1/f)^\alpha & \text{if } h_1 \leq f, \\
0 & \text{otherwise}; 
\end{cases}$$

$$f(x_2) = \begin{cases} 
4 - 3 \exp\left\{-\frac{(x_2-20)^2}{2}\right\} & \text{if } 0 \leq x_2 \leq 40, \\
4 - 2 \exp\left\{-\frac{(x_2-70)^2}{20}\right\} & \text{if } 40 < x_2 \leq 100; 
\end{cases}$$

and $\alpha = 0.25 + 3.75(f(x_2) - 1)$. As in the previous test problem, $\xi_1$ and $\xi_2$ are independent chi-squared random variables with one degree of freedom. Unlike in Problem $T_A$, Problem $T_B$ has dependence between the random objective function values returned by the simulation oracle. Problem $T_B$ has two $N_1$-local weakly efficient sets, one of which is the global weakly efficient set. Since Problem $T_B$ has global weakly efficient points that are not also global efficient points, it satisfies only Assumptions 1–4. A picture of Problem $T_B$ appears in Figure 6.

By Theorem 1 Parts (a) and (b), our algorithms converge into an $N_1$-local weakly efficient set almost surely, and contain an $N_1$-local efficient set almost surely. Nevertheless, for this problem, we use the local coverage error as our solution quality metric (Hunter et al. 2019). In our context, the local coverage error is the Hausdorff distance from the set $g(\hat{A}(t))$ to the nearest $N_1$-local Pareto set as a function of the total simulation work done, $\min_{L_1 \in \mathcal{L}_1} d_H(g(\hat{A}(t)), g(L_1))$. This metric penalizes all algorithms for returning the points that are global weakly efficient set members but not global efficient set members, which may not be distinguishable with finite sample size. Figure 7 shows the sample quantiles of the local coverage error for 1,000 independent runs each of R-PERLE, R-MinRLE, and MO-COMPASS on Problem $T_B$.

Figure 7 shows that R-PERLE out-performs both R-MinRLE and MO-COMPASS on Problem $T_B$. R-MinRLE eventually out-performs MO-COMPASS, but initially suffers from
Cooper, Hunter, and Nagaraj: Bi-objective SO on Integer Lattices using the \(\varepsilon\)-Constraint Method in an RA Framework

high variance in its performance. We believe this behavior occurs because R-MinRLE crawls from the “outside in,” and the sample path \(\mathcal{N}_1\)-local minimizers on each objective may be members of the global weakly efficient set and not the global efficient set. Also, R-MinRLE may not retrieve the “middle” of the \(\mathcal{N}_1\)-local Pareto set until the sample sizes become large enough that the completeness function values are small enough for RLE to crawl there. Thus R-PERLE’s ability to retrieve the middle of the \(\mathcal{N}_1\)-local Pareto set is likely a crucial aspect of its speedy convergence in Problem \(T_B\).

9.2.3. Test Problem C

Our third test problem, Problem \(T_C\), is also a modified version of a test problem that appears in Ryu and Kim (2014). We define Problem \(T_C\) as

\[
\text{Problem } T_C: \quad \text{minimize}_{x \in \mathcal{X}} \begin{cases} 
  g_1(x) = \mathbb{E} \left[ \sum_{i=1}^{2} -10 \xi_i \exp \left\{ -0.2 \sqrt{x_i^2 + x_{i+1}^2} \right\} \right] \\
  g_2(x) = \mathbb{E} \left[ \sum_{i=1}^{3} \xi_i \left( |x_i|^{0.8} + 5 \sin^3(x_i) \right) \right]
\end{cases}
\]

where \(\mathcal{X} = \tilde{X}_{C1} \times \tilde{X}_{C2} \times \tilde{X}_{C3}\), \(\tilde{X}_{Ci} = \{-5, -4.5, -4.0, -3.5, \ldots, 5\}\) for all \(i \in \{1, 2, 3\}\), \(|\mathcal{X}| = 9,261\), and \(\xi_1, \xi_2,\) and \(\xi_3\) are independent chi-squared random variables with one degree of freedom so that \(\mathbb{E} [\xi_i] = 1\) and \(\mathbb{V} (\xi_i) = 2\) for all \(i \in \{1, 2, 3\}\). We map Problem \(T_C\) to an integer lattice so that the \(\mathcal{N}_1\)-neighborhood corresponds to points within distance 0.5 in the original feasible space. Like Problem \(T_B\), Problem \(T_C\) has dependence between the random objective function values returned by the simulation oracle. Problem \(T_C\) has multiple feasible points that map to the same objective vector value. Therefore Problem \(T_C\) only satisfies Assumptions 1–3. Problem \(T_C\) appears in Figure 8.

Figure 8 Problem \(T_C\): Black circles and gray stars represent points in the global weakly efficient set and the \(\mathcal{N}_1\)-local weakly efficient set members, respectively (left) and their images (right).

Figure 9 Problem \(T_C\): Sample quantiles (.25, .5, .75) of the local weakly coverage error across 1,000 independent runs per algorithm.
By Theorem 1 Part (a), our algorithm returns a solution that converges into an $\mathcal{N}_1$-local weakly efficient set w.p.1., with no guarantees on completeness. Nevertheless, we use the local weakly coverage error as our solution quality metric, which we define as $\min_{W_1 \in W_1} d_H(g(\hat{A}(t)), g(W_1))$. Since all $\mathcal{N}_1$-local efficient sets are also $\mathcal{N}_1$-local weakly efficient sets, this metric is less stringent than local coverage error. Algorithm performances based on the local weakly coverage error, calculated across a collection of 516 unique $\mathcal{N}_1$-local weakly efficient sets, appear in Figure 9. Our method for locating the $\mathcal{N}_1$-local weakly efficient sets for Problem $T_C$ appears in the Online Appendix.

Figure 9 shows that both R-PERLE and R-MinRLE out-perform MO-COMPASS on Problem $T_C$. In many of the $\mathcal{N}_1$-local weakly efficient sets, the $\mathcal{N}_1$-local weakly efficient set members are not neighbors. Thus the $\mathcal{N}_1$-local weakly efficient set members may be far away from each other in the feasible space, and often are isolated, as seen in Figure 8. We believe the relative efficiency of R-PERLE and R-MinRLE occurs because RLE crawls to find a sample-path $\mathcal{N}_1$-LWEP that completes the sample-path $\mathcal{N}_1$-local weakly efficient set, even if the required sample-path $\mathcal{N}_1$-LWEP is far away in the feasible space. Since MO-COMPASS operates by updating a region of the feasible space called the Most Promising Area, we suspect that the isolated, scattered nature of the $\mathcal{N}_1$-local weakly efficient set members reduces the likelihood that all set members are contained within the Most Promising Area.

9.3. R-PERLE Performance Across a Range of $\beta$ Values

We explore R-PERLE’s performance on our test problems across a variety of $\beta = (\beta_\varepsilon, \beta_\delta)$ values. Recall that for PE, smaller $\beta_\varepsilon$ values result in solving fewer sample-path $\varepsilon$-constraint problems, and larger $\beta_\varepsilon$ values correspond to solving more sample-path $\varepsilon$-constraint problems. For the RLE algorithm, smaller $\beta_\delta$ implies less crawling and a less-complete ALES, and larger $\beta_\delta$ corresponds to more crawling and a more-complete ALES. Across 1,000 independent runs of R-PE or R-PERLE on Problems $T_A$, $T_B$, and $T_C$, Figures 10, 11, and 12 show the sample quantiles of the respective coverage errors at the total simulation budget of $t = 0.4 \times 10^6$ (corresponding to the first $t$-axis tick mark in Figures 5, 7, and 9) across a variety of parameter settings. Each independent run uses CRN across the $\beta$ values.

At total simulation budget $t = 0.4 \times 10^6$ on Problem $T_A$, there seems to be a “sweet spot” for setting $\beta_\varepsilon$ in the interval $(0.2, 0.4)$, as seen in the left and center panels of Figure 10. Relative to our sampling error, $\beta_\varepsilon < 0.2$ causes the algorithm to find too few sample-path $\mathcal{N}_1$-LWEP’s, while $\beta_\varepsilon > 0.4$ cause the algorithm to find too many. Given that $\beta_\varepsilon = 0.5$, the
R-PERLE performance in the right panel of Figure 10 is fairly robust to different values of $\beta_\delta$. Notice that with $\beta_\varepsilon = 0.5$, for small values of $\beta_\delta$, R-PE and R-PERLE return similar sets. On Problem $T_B$, however, solving more $\varepsilon$-constraint problems and crawling more in RLE seems to improve algorithm performance. We suspect that here, correlation between the objectives and using CRN implies each sample-path problem is similar to the true problem. Thus the ordering of the points in the sample-path problem is similar to the ordering of the points in the true problem with high probability, except among the global weakly efficient points. Thus finding more $\mathcal{N}_1$-LWEP’s in PE and crawling more in RLE is usually better.

Problem $T_C$ is a difficult problem for which R-PE is not guaranteed to converge. In the left panel of Figure 12, like in the left panel of Figure 10, R-PE exhibits $u$-shaped behavior as a function of $\beta_\varepsilon$. However, the center and right panels of Figure 12 tell an interesting story for R-PERLE. It seems that in Problem $T_C$, it is best to solve few $\varepsilon$-constraint problems (smaller $\beta_\varepsilon$) and let RLE do the work of finding the disconnected $\mathcal{N}_1$-local weakly efficient set members, with the sweet spot for $\beta_\delta$ shown in the right panel of Figure 12. Interestingly, that more effort should be expended in RLE and less effort in PE explains the good performance of R-MinRLE in Figure 9. Finally, we remark that across all the problems, without prior knowledge of the problem structure, our default $\beta_\varepsilon$ and $\beta_\delta$ values seem reasonable.

---

**Figure 10** Problem $T_A$: Sample quantiles (0.25, 0.50, 0.75) of the coverage error at $t = 0.4 \times 10^6$ across 1,000 independent runs of R-PE (left), R-PERLE, $\beta_\delta = 0.5$ (center), R-PERLE, $\beta_\varepsilon = 0.5$ (right).

**Figure 11** Problem $T_B$: Sample quantiles (0.25, 0.50, 0.75) of the local coverage error at $t = 0.4 \times 10^6$ across 1,000 independent runs of R-PE (left), R-PERLE, $\beta_\delta = 0.5$ (center), R-PERLE, $\beta_\varepsilon = 0.5$ (right).
10. Concluding Remarks

We propose R-PERLE, a new, provably-convergent algorithm for bi-objective SO on integer lattices. We also propose R-MinRLE as a benchmark algorithm for MOSO on integer lattices with two or more objectives. R-PERLE out-performs both R-MinRLE and the current state-of-the-art algorithm, MO-COMPASS, on our test problems. This work points to a family of RA algorithms for MOSO on integer lattices that employ an accelerator plus RLE for sample-path certification of an ALES, where the convergence guarantees are provided by Theorem 1. Both R-PERLE and R-MinRLE, as well as infrastructure for creating new accelerators, are publicly available in the PyMOSO software package (Cooper and Hunter 2019).

Acknowledgments

The authors thank Eric Applegate for suggesting improvements to our algorithm implementations, Raghu Pasupathy for discussions about the bi-objective bus scheduling problem, and the associate editor and anonymous referees for comments that improved the paper. S. R. Hunter and K. Nagaraj were supported by the National Science Foundation grant CMMI-1554144.

References


Online Appendices for
Bi-objective Simulation Optimization on Integer Lattices
using the Epsilon-Constraint Method
in a Retrospective Approximation Framework

Kyle Cooper
School of Industrial Engineering, Purdue University and Tata Consultancy Services, coope149@purdue.edu

Susan R. Hunter
School of Industrial Engineering, Purdue University, susanhunter@purdue.edu

Kalyani Nagaraj
School of Industrial Engineering & Management, Oklahoma State University, kalyani.nagaraj@okstate.edu

A. Proof Sketch of Lemma 3

Proof Sketch. The proof of Lemma 3 Part (a) is provided in Wang et al. (2013, p. 15) and follows from the first Borel-Cantelli lemma (Billingsley 1995, p. 59). By Lemma 3 Part (a), for each objective \( k \in \{1, \ldots, d\} \), there exists \( \tilde{\nu}_k \), dependent on \( \alpha, x \), and the random realization, such that for all \( \nu \geq \tilde{\nu}_k \), \( \hat{S}_{k,\nu}(x) \subseteq S_k(x, \alpha_k) \) w.p.1. Let \( \tilde{\nu} := \max_k \{\tilde{\nu}_k\} \), so that for all \( \nu \geq \tilde{\nu} \), \( \hat{S}_{\nu}(x) \subseteq S(x, \alpha) \) w.p.1. \( \square \)

B. Proof of Theorem 1

Proof of Theorem 1 Part (a). For every \( \nu \), R-PERLE and R-MinRLE return a set \( \hat{A}_\nu \) in finite time. Thus both algorithms produce an infinite sequence of solutions \( \{\hat{A}_\nu\} \). Further, notice that R-PERLE and R-MinRLE never return a set \( \hat{A}_\nu \) containing a point whose estimated objective vector is dominated by \( G_{\nu}(x_0) \) (see Algorithm 5, RLE Steps 1 and 14). Now consider the union of the level sets corresponding to the starting point \( x_0, S(x_0, \alpha) \). By Lemma 3, there exists \( \tilde{\nu} \) such that for all \( \nu \geq \tilde{\nu} \), \( \hat{A}_\nu \subseteq \hat{S}_\nu(x_0) \subseteq S(x_0, \alpha) \) w.p.1. Since \( S(x_0, \alpha) \) is finite, then any sequence of estimated efficient points \( \{X^{*}_\nu : X^{*}_\nu \in \hat{A}_\nu \text{ for all } \nu = 1, 2, \ldots\} \) is bounded w.p.1. Using an argument similar to that in Wang et al. (2013, Theorem 5.4, p. 15), we now prove that \( \{\hat{A}_\nu\} \) converges into an \( N_a \)-local weakly efficient set w.p.1.

Since \( S(x_0, \alpha) \) is finite and \( a \in (0, \infty) \), then \( N_a(S(x_0, \alpha)) \cap \mathcal{X} \) is also finite. Thus for all \( k \in \{1, \ldots, d\} \), \( G_{k,\nu}(\cdot) \) uniformly converges to \( g_k(\cdot) \) w.p.1 as \( \nu \to \infty \) on the set \( N_a(S(x_0, \alpha)) \cap \mathcal{X} \). Let the set \( \mathcal{D}_k := \{(x, x') : x, x' \in N_a(S(x_0, \alpha)) \cap \mathcal{X}, g_k(x') \neq g_k(x)\} \) be the set of all pairs...
of feasible points in the level set neighborhood that have different true objective function values on objective $k$, and let $\kappa_1 = \min_{k \in \{1, \ldots, d\}} \inf_{\mathcal{D}_k} |g_k(x) - g_k(x')| > 0$ be the smallest difference in objective values across these pairs; Assumption 1 implies $\kappa_1 > 0$. Then w.p.1, there exists $\nu'$ (dependent on neighborhood size $a$, initial point $x_0$, the constants $\kappa_1$ and $\alpha$, and the random realization) such that for all $\nu \geq \nu'$, $\max_{k \in \{1, \ldots, d\}} |\bar{G}_{k,m}(x) - g_k(x)| < \kappa_1/4$ for all $x \in \mathcal{N}_a(S(x_0, \alpha)) \cap \mathcal{X}$. Since $\beta_\delta \in (0, \infty]$, the ALES completeness function $\hat{\delta}(\cdot) = \hat{f}_k(\cdot, \beta) = \delta_{k,m}(\cdot)/m^\beta$ uniformly converges to zero w.p.1 on the finite set $\mathcal{N}_a(S(x_0, \alpha)) \cap \mathcal{X}$ as $\nu \to \infty$ for all $k \in \{1, \ldots, d\}$. Then w.p.1, there exists $\nu''$ (dependent on the same quantities as $\nu'$ and dependent on $\beta_\delta$) such that for all $\nu \geq \nu''$, $\max_{k \in \{1, \ldots, d\}} \hat{\delta}(x) < \kappa_1/4$ for all $x \in \mathcal{N}_a(S(x_0, \alpha)) \cap \mathcal{X}$.

Henceforth, let $\nu \geq \max\{\nu', \nu''\}$. Combining the above results, for all $x \in \mathcal{N}_a(S(x_0, \alpha)) \cap \mathcal{X}$, $\max_{k \in \{1, \ldots, d\}} |\bar{G}_{k,m}(x) - \hat{G}_{k,m}(x) - g_k(x)| < \kappa_1/2$ w.p.1. Thus for all $k \in \{1, \ldots, d\}$ and for all $x, x' \in \mathcal{N}_a(S(x_0, \alpha)) \cap \mathcal{X}$, the following hold:

R1. if $g_k(x) < g_k(x')$, then $\bar{G}_{k,m}(x) + \hat{\delta}(x) < \bar{G}_{k,m}(x') - \hat{\delta}(x')$ w.p.1;

R2. if $\bar{G}_{k,m}(x) + \hat{\delta}(x) \leq \bar{G}_{k,m}(x') - \hat{\delta}(x')$, then $g_k(x) \leq g_k(x')$ w.p.1.

Further, for all $x, x' \in \mathcal{N}_a(S(x_0, \alpha)) \cap \mathcal{X}$, if $G_{m',m}(x) \notin G_{m',m}(x')$, then $\exists k \in \{1, \ldots, d\}$ such that $\bar{G}_{k,m}(x') < \bar{G}_{k,m}(x)$, implying that $g_k(x') \leq g_k(x)$ w.p.1. This result, along with results R1 and R2 above, imply that for all $x, x' \in \mathcal{N}_a(S(x_0, \alpha)) \cap \mathcal{X}$,

R3. if $G_{m',m}(x) \notin G_{m',m}(x')$, then $g(x) \leq g(x')$ w.p.1;

R4. if $G_{m',m}(x) + \hat{\delta}(x) \leq G_{m',m}(x') - \hat{\delta}(x)$, then $g(x) \leq g(x')$ w.p.1.

Now let $\nu \geq \max\{\tilde{\nu}, \nu', \nu''\}$. Then by Lemma 3, the set of decision points $A_\nu$ returned by each algorithm lie in $S(x_0, \alpha)$ w.p.1. We now consider all parts of the definition of an ALES (Definition 7). First, Algorithm 5 ensures no points in $G_n(A_\nu)$ dominate other points in $G_n(A_\nu)$ (RLE Steps 1 and 14). Thus result R3 above implies that no points in $g(A_\nu)$ strictly dominate other points in $g(A_\nu)$. Second, Algorithm 5 ensures each point in $X_\nu \in A_\nu$ is a sample-path $\mathcal{N}_a$-LWEP (e.g., RLE Steps 2, 5, 10, and 15). Thus result R3 above implies that all points in $A_\nu$ are $\mathcal{N}_a$-LWEP’s. Third, Algorithm 5 ensures that the NCN of $A_\nu$ is empty (RLE Steps 2 and 15). Then applying results R3 and R4 above, for all $x \in \mathcal{N}_a(A_\nu) \cap \mathcal{X}$, (i) $\exists X_\nu \in A$ such that $g(X_\nu) \leq g(X)$ w.p.1, or (ii) $\exists X_\nu \in A$ such that $(g(X) \leq g(X_\nu)$ and $G_{m',m}(X_\nu) - \hat{\delta}(X_\nu) \leq G_{m',m}(X) + \hat{\delta}(X))$ w.p.1, which happens with probability zero unless $g(X) = g(X_\nu)$, or (iii) employing the complements of the previous two conditions, $\forall X_\nu \in A$, $g(X_\nu) \notin g(X) \notin g(X_\nu)$, and $\exists X_\nu \in A$ such that $G_{m',m}(X_\nu) - \hat{\delta}(X_\nu) \leq G_{m',m}(X) + \hat{\delta}(X)$ or
Theorem 1 Part (b). Let \( \nu \geq \max\{\tilde{\nu}, \nu', \nu''\} \). By the proof of Theorem 1 Part (a), \( \hat{A}_\nu \) is an \( N_a \)-local weakly efficient set w.p.1. Under Assumption 4, by Lemma 4, there exists \( L_a \in L_a \) such that \( \hat{A}_\nu \supseteq L_a \) w.p.1, and the result holds.

Proof of Theorem 1 Part (c). Assumption 5 implies that all \( N_a \)-local weakly efficient sets are \( N_a \)-local efficient sets. Thus \( W_a = L_a \), and the result follows from Theorem 1 Parts (a) and (b).

Proof of Theorem 1 Part (d). Assumptions 5 and 6 imply that the \( N_a \)-local efficient set in Theorem 1 Part (c) is the global efficient set, and the result follows.

C. Proof of Theorem 2

Proof. For every \( \nu \), R-PE returns a set \( \hat{A}_\nu \) in finite time, thus producing an infinite sequence of solutions \( \{\hat{A}_\nu\} \). Further, R-PE never returns a set \( \hat{A}_\nu \) containing a point whose estimated objective vector is dominated by \( G_{m_\nu}(x_0) \) (see Algorithm 2, PE Step 26). Recall that for all \( \nu \geq \tilde{\nu} \), \( \hat{A}_\nu \subseteq \hat{S}_\nu(x_0) \subseteq S(x_0, \alpha) \) w.p.1, and since \( S(x_0, \alpha) \) is finite, then any sequence of estimated efficient points \( \{X_\nu^*: X_\nu^* \in \hat{A}_\nu \text{ for all } \nu = 1, 2, \ldots\} \) is bounded w.p.1.

As in the proof of Theorem 1 Part (a), for all \( k \in \{1, 2\} \), \( \hat{G}_{k,m_\nu}(\cdot) \) uniformly converges to \( g_k(\cdot) \) w.p.1 as \( \nu \to \infty \) on the finite set \( N_a(S(x_0, \alpha)) \cap \mathcal{X} \). Since there are only two objectives and \( \beta_\varepsilon \in (0, \infty) \), the maximum \( \varepsilon \)-placement distance \( \max_{k,\varepsilon} \widehat{f}_{k,\varepsilon}(\cdot, \beta_\varepsilon) = \)
max_{k \in \{1, 2\}} \sigma_{k, m} / m^{\beta_\epsilon}\) also uniformly converges to zero w.p.1 as \(\nu \to \infty\) on \(N_a(S(x_0, \alpha)) \cap \mathcal{X}\). Let \(\kappa > 0\) be as in Assumption 5. Then w.p.1, there exists \(\nu'_{\text{PE}}\) (dependent on \(a, x_0, \kappa, \alpha, \) and the random realization) such that for all \(\nu \geq \nu'_{\text{PE}}\) and all \(x \in N_a(S(x_0, \alpha)) \cap \mathcal{X}\), we have \(\max_{k \in \{1, 2\}} |\tilde{G}_{k, m} (x) - g_k(x)| < \kappa/4\) w.p.1. Also w.p.1, there exists \(\nu''_{\text{PE}}\) (dependent on the same quantities as \(\nu'_{\text{PE}}\) and dependent on \(\beta_\epsilon\)) such that for all \(\nu \geq \nu''_{\text{PE}}\), \(\max_{k \in \{1, 2\}} |\hat{f}_{k, \epsilon} (x, \beta_\epsilon) - \tilde{f}_{k, \epsilon} (x')| - g_{k, \epsilon}(x) < \kappa/2\) w.p.1. Henceforth, let \(\nu \geq \max\{\nu'_{\text{PE}}, \nu''_{\text{PE}}\}\). Then for all \(x, x' \in N(S(x_0, \alpha)) \cap \mathcal{X}\), the following hold w.p.1:

1. \(\forall k \in \{1, 2\}, g_k(x) < g_k(x')\) if and only if \(\tilde{G}_{k, m} (x) + \hat{f}_{k} (x, \beta_\epsilon) < \tilde{G}_{k, m} (x') + \hat{f}_{k} (x', \beta_\epsilon)\);
2. if \(G_{m} (x) \not\subseteq G_{m} (x')\), then \(g(x) \not\subseteq g(x')\), that is, \(\exists k \in \{1, 2\}\) such that \(g_k(x') < g_k(x)\).

Under Assumption 1, for any \(x_0 \in \mathcal{X}\), \(\mathcal{E} \subseteq S(x_0, \alpha)\). Therefore results R5, R6 and Assumptions 5–6 imply that all points in \(\mathcal{E}\) are both sample-path \(N_a\)-LWEP’s and sample-path global efficient points w.p.1. Further, all points in \(N_a(\mathcal{E})\) are not sample-path \(N_a\)-LWEP’s w.p.1. Let \(c_\epsilon := |\mathcal{E}| \geq 1\), and for any objective \(k \in \{1, 2\}\), sort the elements of \(\mathcal{E}\) on objective \(k\) so that \(g_k(x^*_{k(i)}) < \ldots < g_k(x^*_{k(c_\epsilon)})\), where \(x^*_{k(i)}\) denotes the \(i\)th ordered element of \(\mathcal{E}\) on objective \(k\), \(i = 1, \ldots, c_\epsilon\). If \(c_\epsilon \geq 2\), then result R5 implies that w.p.1 for all \(i = 1, \ldots, c_\epsilon - 1\),

\[
\tilde{G}_{k, m} (x^*_{k(i)}) + \hat{f}_{k, \epsilon} (x^*_{k(i)}, \beta_\epsilon) < \tilde{G}_{k, m} (x^*_{k(i+1)}) - \hat{f}_{k, \epsilon} (x^*_{k(i+1)}).
\]

By Lemma 5, w.p.1 there exists \(\nu''\) (dependent on the same quantities as \(\nu'_{\text{PE}}\)) such that for all \(\nu \geq \nu''\), the updated sample-path \(N_a\)-local minimizers returned as part of PE Step 1, which we call \(M_{\nu}\), are such that \(M_{\nu} \subseteq M^*_a\). Under Assumptions 5–6, the set \(M^*_a = \{x^1_{\text{min}}, x^2_{\text{min}}\}\) contains the unique global minimizers for each objective \(k \in \{1, 2\}\).

Henceforth, let \(\nu > \max\{\bar{\nu}, \nu'_{\text{PE}}, \nu''_{\text{PE}}, \nu''\}\), and let \(\{k^*_\nu, \nu = 1, 2, \ldots\}\) be any sequence of objectives minimized, where \(k^*_\nu \neq k^*_\nu\) for each \(\nu\). Then by Lemma 3, the set of decision points \(\hat{A}_\nu\) returned by R-PE lie in \(S(x_0, \alpha)\) w.p.1, as does the set of points used to set the \(\epsilon\) values in Algorithm 2, PE Step 2, which is a set of sample-path \(N_a\)-LWEP’s we call \(\hat{A}_\nu^w\). Since all points in \(\hat{A}_\nu^w\) are sample-path \(N_a\)-LWEP’s, then results R5, R6 and Assumptions 5–6 ensure that \(\hat{A}_\nu^w \subseteq \mathcal{E}\) w.p.1; further, \(\mathcal{M}_\nu \subseteq \hat{A}_\nu^w\), where \(\mathcal{M}_\nu = \{x^1_{\text{min}}, x^2_{\text{min}}\}\) w.p.1. If \(c_\epsilon \in \{1, 2\}\), the proof is complete, since \(\hat{A}_\nu^w\) is returned as \(\hat{A}_\nu\) in Algorithm 2, PE Step 26, and no other points have entered the set w.p.1. Now suppose \(c_\epsilon \geq 3\). All points in \(\hat{A}_\nu^w \cup \mathcal{E}\) can be ordered on \(k^\text{con}\) as in line (1). Points in \(\mathcal{E} \setminus \hat{A}_\nu^w\) are retrieved by Algorithm 2, PE Steps 15–25, and
carried forward to $\hat{A}_{\nu+1}$; no other points enter the set w.p.1. Then it follows that for all $\nu^* > \nu + 1$, $\hat{A}_{\nu^*} = \mathcal{E}$ w.p.1, and the result holds. □

D. Proof of Theorem 3

Proof of Theorem 3 Part (a). Let $\mathcal{D} \subseteq \mathcal{X}$ be any subset of the feasible set. Since $\mathcal{X}$ is finite, $\mathcal{D}$ is finite. Let $\mathcal{B}_{k,\nu}^*(\mathcal{D})$ denote the set of sample-path global minimizers of objective $g_k$, $k \in \{1, \ldots, d\}$ on the set $\mathcal{D}$, and let $\mathcal{B}_{k}^*(\mathcal{D})$ denote the corresponding set of true global minimizers on $\mathcal{D}$. Then under our assumptions, by Wang et al. (2013, p. 16), for all $k \in \{1, \ldots, d\}$ and all $\mathcal{D} \subseteq \mathcal{X}$, there exists $\eta > 0$ such that for large enough $\nu$,

$$
\mathbb{P}\{\mathcal{B}_{k,\nu}^*(\mathcal{D}) \not\subseteq \mathcal{B}_{k}^*(\mathcal{D})\} \leq |\mathcal{D}| e^{-m_\nu \eta}.
$$

(2)

Recall that $\mathcal{X} \subseteq \mathbb{Z}^d$ and let $x \in \mathcal{X}$ be a feasible point. Letting $e_i$ denote a $q$-dimensional vector of zeros with one in the $i$th place, divide $\mathcal{N}_1(x)$ into $2q$ sub-neighborhoods that include $x$ and exactly one other neighborhood point in each direction, $\mathcal{N}_{1,i}(x) := \{x, x + e_i\}$ and $\mathcal{N}_{1,-i}(x) := \{x, x - e_i\}$ for all $i \in \{1, \ldots, q\}$.

For every non-$\mathcal{N}_1$-LWEP $x \in \mathcal{X} \setminus \mathcal{X}^w$, there must exist $x' \in \mathcal{N}_1(x) \cap \mathcal{X}$ such that $g(x')$ strictly dominates $g(x)$. Then for every $x \in \mathcal{X} \setminus \mathcal{X}^w$, there exists $j \in \{-q, \ldots, -1, 1, \ldots, q\}$ and $x' \in \mathcal{N}_1(x) \cap \mathcal{X}$ such that $\mathcal{N}_{1,j}(x) = \{x, x'\}$ and $g(x')$ strictly dominates $g(x)$. Thus $x$ is not a global minimizer on $\mathcal{N}_{1,j}(x)$ on any objective. If $x \in \mathcal{X} \setminus \mathcal{X}^w$ is nonetheless estimated as an $\mathcal{N}_1$-LWEP, that is, $x \in \mathcal{X}^w_\nu$ on its $\mathcal{N}_1$-neighborhood, there must exist an objective $k \in \{1, \ldots, d\}$ such that $\mathcal{B}_{k,\nu}^*(\mathcal{N}_{1,j}(x)) \not\subseteq \mathcal{B}_{k}^*(\mathcal{N}_{1,j}(x))$. Then for large enough $\nu$,

$$
\mathbb{P}\{\mathcal{X}^w_\nu \not\subseteq \mathcal{X}^w\} \leq \sum_{x \in \mathcal{X} \setminus \mathcal{X}^w} \mathbb{P}\{x \in \mathcal{X}^w_\nu\} \\
\leq \sum_{x \in \mathcal{X} \setminus \mathcal{X}^w} \sum_{j \in \{-q, \ldots, -1, 1, \ldots, q\}} \sum_{k \in \{1, \ldots, d\}} \mathbb{P}\{\mathcal{B}_{k,\nu}^*(\mathcal{N}_{1,j}(x)) \not\subseteq \mathcal{B}_{k}^*(\mathcal{N}_{1,j}(x))\} \\
\leq \sum_{x \in \mathcal{X} \setminus \mathcal{X}^w} \sum_{j \in \{-q, \ldots, -1, 1, \ldots, q\}} \sum_{k \in \{1, \ldots, d\}} 2 e^{-m_\nu \eta} \leq |\mathcal{X}| 4qde^{-m_\nu \eta},
$$

where $\eta > 0$ denotes the relevant constant from line (2).

Proof of Theorem 3 Part (b). We begin by noticing that item (i) follows from Theorem 3 Part (a), along with the assumption that sample sizes increase at least linearly and the fact that our algorithms guarantee $\hat{A}_\nu$ contains only sample-path $\mathcal{N}_1$-LWEP’s.

To prove item (ii), notice that under our assumptions, all $\mathcal{N}_1$-LWEP’s are global efficient points. Therefore by item (i), $\mathbb{P}\{\hat{A}_\nu \not\subseteq \mathcal{E}\} = O(e^{-\gamma m_\nu})$ for some $\gamma > 0$. We now consider
\[ P\{E \not\subseteq \hat{A}_\nu\} \leq \sum_{x \in E} P\{x \not\in \hat{A}_\nu\}. \]

Notice that if \( x \in E \) is not in \( \hat{A}_\nu \), then it must have been incorrectly estimated as dominated by at least one point in its neighborhood. Then by a proof similar to that of Theorem 3 Part (a), it follows that \( P\{E \not\subseteq \hat{A}_\nu\} = O(e^{-\gamma m_\nu}) \) for some \( \gamma > 0 \), which implies the result. \( \square \)

E. Finding \( N_1 \)-local weakly efficient sets in Problem \( T_C \)

To calculate the collection of all possible \( N_1 \)-local weakly efficient sets in Problem \( T_C \), we first locate all \( N_1 \)-LWEP’s; we find 512. Then, starting from each \( N_1 \)-LWEP, we run a program similar to RLE with no relaxation to find the smallest complete \( N_1 \)-local weakly efficient set that contains the \( N_1 \)-LWEP. We refer to these \( N_1 \)-local weakly efficient sets as level-1 \( N_1 \)-local weakly efficient sets; after removing duplicate sets, we find 39. Then, we take all possible unions of two level-1 \( N_1 \)-local weakly efficient sets, remove any dominated points, and check if this set is a new, unique \( N_1 \)-local weakly efficient set. We refer to all new, unique \( N_1 \)-local weakly efficient sets that are found by taking the union of two level-1 \( N_1 \)-local weakly efficient sets as level-2 \( N_1 \)-local weakly efficient sets. We repeat this process for level-3 and so on, up to level-8. We found one level-7 \( N_1 \)-local weakly efficient set and no level-8 \( N_1 \)-local weakly efficient sets. The total number of unique \( N_1 \)-local weakly efficient sets found in this manner, up to level-8, was 516. All together, these 516 \( N_1 \)-local weakly efficient sets contain just 73 points; we call these points \( N_1 \)-local weakly efficient set members in Figure 8. Recall that there are 512 \( N_1 \)-LWEP’s, so not all \( N_1 \)-LWEP’s are members of an \( N_1 \)-local weakly efficient set.

F. A Bi-objective Integer Bus Scheduling Problem

We create a bi-objective version of the integer bus-scheduling problem described by Wang et al. (2013), as follows. Suppose passengers arrive to a bus station according to a Poisson process with arrival rate \( \lambda = 10 \) people per time unit. During a day that is \( \tau = 100 \) time units long, the decision-maker would like to schedule \( b \in \{1, 2, \ldots, q\} \) infinite-capacity buses so that the expected cost of operating the buses, \( g_1(x) \), and the passengers’ expected total waiting time, \( g_2(x) \), are minimized. The decision variable \( x = (x_1, x_2, \ldots, x_q) \) is an integer bus schedule, where we assume there is a no-cost bus at time 0 and a pre-scheduled bus at time \( \tau \). Scheduling one of the \( q \) total buses at time 0 or \( \tau \), or at the same time as any other bus, corresponds to not using that bus. The feasible set is \( X = \{0, 1, \ldots, 100\}\). We note here that this problem has a many-to-one mapping, and thus violates Assumptions 4 and 5.
We define the objective functions implicitly through a Monte Carlo simulation model. To specify the simulation model, without loss of generality, label the buses so that $x_0 := 0 \leq x_1 \leq x_2 \leq \ldots \leq x_q \leq x_{q+1} := \tau$, and let $N_i(x_\ell) - N_i(x_{\ell-1})$ denote the number of passenger arrivals between bus $\ell - 1$ and bus $\ell$ on the $i$th day, $\ell = 1, 2, \ldots, q + 1$. On the $i$th day and given an integer bus schedule $\mathbf{x}$, the simulation model returns (a) the cost $G_1(\mathbf{x}, \xi_i) = \sum_{\ell=1}^{q+1} c_0 \mathbb{I}\{x_\ell - x_{\ell-1} > 0\} + (N_i(x_\ell) - N_i(x_{\ell-1}))^\gamma$, where $c_0$ is a constant operating cost per bus and $\gamma$ is a constant, and (b) the observed total waiting time $G_2(\mathbf{x}, \xi_i) = \sum_{j=1}^{N_i(\tau)} W_{ij}$, where $W_{ij}$ is the wait time of the $j$th passenger for $j = 1, 2, \ldots, N_i(\tau)$ passengers on the $i$th day.

We implement the 9-bus scheduling problem with $c_0 = 100$ and $\gamma = 1/2$. For any feasible $\mathbf{x}$, we approximate the true expected cost as

$$g_1(\mathbf{x}) = \mathbb{E}[G_1(\mathbf{x}, \xi_i)] = \sum_{\ell=1}^{q+1} c_0 \mathbb{I}\{x_\ell - x_{\ell-1} > 0\} + \mathbb{E}\left[\sqrt{N_i(x_\ell) - N_i(x_{\ell-1})}\right]$$

$$\approx \sum_{\ell=1}^{q+1} c_0 \mathbb{I}\{x_\ell - x_{\ell-1} > 0\} + \sqrt{\lambda(x_\ell - x_{\ell-1})},$$

where the approximation to the expected value of the square root of a Poisson random variable is better for larger values of $(x_\ell - x_{\ell-1})$. Since $\lambda(x_\ell - x_{\ell-1}) \geq 10$ whenever $x_\ell \neq x_{\ell-1}$, the approximation error is relatively small for the values we consider. We calculate the true expected wait time as

$$g_2(\mathbf{x}) = \mathbb{E}[G_2(\mathbf{x}, \xi_i)] = \mathbb{E}\left[\sum_{j=1}^{N_i(\tau)} W_{ij}\right] = \left(\lambda / 2\right) \sum_{\ell=1}^{q+1} (x_\ell - x_{\ell-1})^2.$$

The points that minimize the $g_1$ objective correspond to scheduling none of the $q$ buses, $x_1^{\text{min}} = (0, 0, \ldots, 0)$ or $x_2^{\text{min}} = (100, 100, \ldots, 100)$. The global solution on the $g_2$ objective is known to be $x_2^{\text{min}} = (10, 20, 30, 40, 50, 60, 70, 80, 90)$. Thus under our approximation of the true expected cost, the ideal point is $(g_1(\mathbf{x}_1^{\text{min}}), g_2(\mathbf{x}_2^{\text{min}})) = (131.6, 5000)$, and the nadir point is $(g_1(\mathbf{x}_2^{\text{min}}), g_2(\mathbf{x}_1^{\text{min}})) = (1100, 50000)$.

Since the feasible set is large ($|\mathcal{X}| = 1.0937 \times 10^{18}$) and there is a many-to-one mapping, it is too computationally intensive to locate all possible $\mathcal{N}_1$-local weakly efficient sets, as we did in Test Problem C. Therefore, the collection of $\mathcal{N}_1$-local weakly efficient sets remains unknown. However, to gain intuition on the structure of this problem, it is conceptually helpful to constrain the problem into ten sub-problems that correspond to the exact number of buses scheduled at non-null times: zero, one, two, and so on, up to nine.

For example, suppose we constrain ourselves to scheduling exactly one bus at a non-null time. The point $(0, 0, 0, 0, 0, 0, 0, 0, 1)$ puts the non-null bus as close as possible to zero, which
is a local minimizer for the expected cost. The point \((0,0,0,0,0,0,0,0,50)\) places the bus in the middle of the time interval and is a local minimizer for the expected total wait time. These minimizers are not unique. Then, we enumerate the points \((0,0,0,0,0,0,0,0,i)\) for \(i = 1, 2, \ldots, 50\), and plot the points on the left-hand side Figure 13. Likewise, for the problem of scheduling exactly two buses at non-null times, the point \((0,0,0,0,0,0,0,1,2)\) clusters the non-null buses as close as possible to zero and is a local minimizer for the expected cost. The point \((0,0,0,0,0,0,33,66)\) evenly spaces the two buses and is a local minimizer for the expected total wait time. These minimizers are not unique. Then, we enumerate the points \((0,0,0,0,0,0,i,j)\) for \(i = 1, 2, \ldots, 33\) and \(j = i + 1, \ldots, 66\), remove the duplicate and dominated points, and plot their values in the right-hand side of Figure 13; we calculate that there are 212 points in this graph. We did not enumerate the corresponding values for scheduling three or more non-null buses.

While the plots in Figure 13 and the constrained sub-problems are helpful for understanding the structure of the problem, the points \((0,0,0,0,0,0,0,1)\) and \((0,0,0,0,0,0,1,2)\) are not actually an \(N_1\)-local minimizers on the cost objective in the unconstrained 9-bus scheduling problem — their neighbors, \((0,0,0,0,0,0,0,0)\) and \((0,0,0,0,0,0,0,1,1)\), respectively, have a lower expected cost value. Since the scheduling of an additional non-null bus incurs an immediate cost of \(c_0 = 100\) monetary units, we expect that the global Pareto set will consist of nine such clusters of Pareto points, plus one global minimum point on the expected cost objective corresponding to scheduling zero buses at non-null times.
We perform 24 independent runs of R-PERLE on this difficult problem using a total simulation effort of $t = 1 \times 10^6$ simulation oracle calls. We configure R-PERLE as described in §9.1, and we use CRN across points visited. For consistency with the R-SPLINE implementation of the bus scheduling problem in Wang et al. (2013), we start all 24 runs from the initial feasible point $x_0 = (1, 1, \ldots, 1)$, which Raghu Pasupathy confirmed through personal communication as the starting points for the numerical runs in Wang et al. (2013). Since this problem violates Assumptions 4 and 5, we remark here that at best, R-PERLE converges into an $N_1$-local weakly efficient set w.p.1 as the simulation budget increases to infinity.

We are not able to run MO-COMPASS on this problem because our current implementation of the algorithm cannot handle a problem with a feasible set of this size. We know of no publicly available version of MO-COMPASS code that could run on this problem, therefore, we omit a comparison with MO-COMPASS.

The following 24 plots report our results, where each plot is the result of one independent R-PERLE run. Since the $N_1$-local weakly efficient sets are unknown, we report only the true objective function values of the ALES returned at the end of each R-PERLE run, $g(\hat{A}(t))$. The plots also display the number of RA iterations completed, the cardinality of the returned ALES, and the ideal and nadir points for reference. As expected, the performance of R-PERLE varies across sample paths. However, R-PERLE seems to find points across a variety of the Pareto “clusters” corresponding to scheduling different numbers of non-null buses.
G. Computational Time

As noted in the main body of the paper, usually, the time required to obtain one simulation replication is so much larger than the overhead required by a MOSO algorithm that the overhead is considered negligible. However, in the event that obtaining one simulation replication is very fast, the computational overhead of the algorithm may become a relevant characteristic for practitioners to consider when selecting an algorithm. In this section, we report statistics on the time it takes to run one sample path of each algorithm on each of the Test Problems A, B, and C. We also report statistics on the time it takes to run the bus-scheduling problem.

To obtain our results, we take a fixed-budget approach. That is, we fix the total simulation budget for all algorithms on all test problems to be \( t \) simulation replications. Given the total simulation budget \( t \), we complete 100 independent runs of each algorithm on each test problem, and record statistics regarding the computational time required and solution quality achieved for all problems with known solutions.

For Test Problems A, B, and C, which have known solutions, we use \( t = 5 \times 10^6 \). We select this simulation budget because it is the total simulation budget we used in §9.2 to numerically demonstrate the convergence of our algorithms, before we cropped the figures down to reflect a total simulation budget of \( 4 \times 10^6 \). Further, this budget will provide an upper bound for smaller total simulation budget values on the same problems and using the same or similar algorithm implementations. For the bi-objective 9-bus scheduling problem, we use \( t = 1 \times 10^6 \), which is consistent with the results in §F.

To measure computational time, let the random variable \( T \) denote the computational time required for one independent run of one algorithm on one test problem. Notice that \( T \) depends on the particular path taken by the algorithm in response to the random variables in the current run, as well as the time required to obtain one simulation replication. In our Test
Problems A, B, and C, obtaining one simulation replication is fast — as fast as generating several chi-squared random variates and performing the algebra required to evaluate the objective functions. Obtaining one simulation replication in the bi-objective 9-bus scheduling problem requires running a Monte Carlo simulation that, for a given bus schedule, simulates the arrival of passengers and collects the observed cost and wait times.

To measure solution quality, we use the metrics for Test Problems A, B, and C from the main body of the paper. For brevity, we define the random variables representing quality as

\[ Q_A = d_H(\mathbf{g}(\hat{A}(t)), \mathbf{g}(\mathcal{E})), \quad Q_B = \min_{\mathcal{L}_1 \in \mathcal{L}_1} d_H(\mathbf{g}(\hat{A}(t)), \mathbf{g}(\mathcal{L}_1)), \quad Q_C = \min_{\mathcal{W}_1 \in \mathcal{W}_1} d_H(\mathbf{g}(\hat{A}(t)), \mathbf{g}(\mathcal{W}_1)), \]

for Test Problems A, B, and C, respectively. We do not report solution quality for the 9-bus scheduling problem because the collection of \( \mathcal{N}_1 \)-local weakly efficient sets is unknown.

Given an algorithm and a test problem, we calculate the following statistics regarding computational time and solution quality across the 100 independent runs, where each run collects up to \( t \) simulation replications. To measure computational time, let \( \bar{T} = (100)^{-1} \sum_{i=1}^{100} T_i \) be the average time required to perform an independent run up to \( t \) simulation replications. Let \( \hat{s.d.}(T) = \sqrt{(99)^{-1} \sum_{i=1}^{100} (T_i - \bar{T})^2} \) denote the estimated standard deviation of \( T \), and let \( \hat{s.e.}(\bar{T}) = \hat{s.d.}(T)/\sqrt{100} \) be the estimated standard error of \( \bar{T} \). Likewise, for each quality metric \( Q^J, J \in \{A, B, C\} \), the average solution quality achieved by an independent run up to \( t \) simulation replications is \( \bar{Q}^J := (100)^{-1} \sum_{i=1}^{100} Q_i^J \). Let the estimated standard deviation of \( Q^J \) be \( \hat{s.d.}(Q^J) = \sqrt{(99)^{-1} \sum_{i=1}^{100} (Q_i^J - \bar{Q}^J)^2} \), and let the estimated standard error of \( \bar{Q}^J \) be \( \hat{s.e.}(\bar{Q}^J) = \hat{s.d.}(Q^J)/\sqrt{100} \).

All algorithms are coded in Python. The runs for R-PERLE and R-MinRLE were completed using the PyMOSO software package (Cooper and Hunter 2019). Our code for MO-COMPASS is compatible with PyMOSO and uses its infrastructure for random number stream management. To obtain 100 runs of each algorithm on each test problem, we use 5 nodes of a high-performance computing cluster, where each node has two Haswell CPU’s at 2.60GHz, 20 cores, and 128GB of memory per node. We obtain the independent runs in an embarrassingly parallel fashion, with one run per core. We complete all runs of a single algorithm on a single test problem before proceeding to the next algorithm and test problem combination. We report the results of our numerical experiments on Test Problems A, B, and C in Tables 1, 2, and 3. We report the results for the bus scheduling problem in Table 4.
Table 1  Test Problem A: The table reports computational time and solution quality statistics calculated across 100 independent runs of each algorithm using a total simulation budget of $t = 5 \times 10^6$ simulation replications.

<table>
<thead>
<tr>
<th>Performance Metric</th>
<th>Statistic</th>
<th>R-PERLE</th>
<th>R-MinRLE</th>
<th>MO-COMPASS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time in Minutes</td>
<td>$\bar{T}$</td>
<td>38.46</td>
<td>39.20</td>
<td>37.43</td>
</tr>
<tr>
<td>s.d.($T$)</td>
<td></td>
<td>1.20</td>
<td>0.84</td>
<td>0.68</td>
</tr>
<tr>
<td>s.e.($\bar{T}$)</td>
<td></td>
<td>0.12</td>
<td>0.08</td>
<td>0.07</td>
</tr>
<tr>
<td>Solution Quality</td>
<td>$\bar{Q}^A$</td>
<td>0.209</td>
<td>0.202</td>
<td>1.958</td>
</tr>
<tr>
<td>s.d.($Q^A$)</td>
<td></td>
<td>0.144</td>
<td>0.113</td>
<td>1.627</td>
</tr>
<tr>
<td>s.e.($\bar{Q}^A$)</td>
<td></td>
<td>0.014</td>
<td>0.011</td>
<td>0.163</td>
</tr>
</tbody>
</table>

Table 2  Test Problem B: The table reports computational time and solution quality statistics calculated across 100 independent runs of each algorithm using a total simulation budget of $t = 5 \times 10^6$ simulation replications.

<table>
<thead>
<tr>
<th>Performance Metric</th>
<th>Statistic</th>
<th>R-PERLE</th>
<th>R-MinRLE</th>
<th>MO-COMPASS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time in Minutes</td>
<td>$\bar{T}$</td>
<td>33.13</td>
<td>34.03</td>
<td>34.72</td>
</tr>
<tr>
<td>s.d.($T$)</td>
<td></td>
<td>0.83</td>
<td>0.84</td>
<td>0.98</td>
</tr>
<tr>
<td>s.e.($\bar{T}$)</td>
<td></td>
<td>0.08</td>
<td>0.08</td>
<td>0.10</td>
</tr>
<tr>
<td>Solution Quality</td>
<td>$\bar{Q}^B$</td>
<td>0.000</td>
<td>0.000</td>
<td>0.653</td>
</tr>
<tr>
<td>s.d.($Q^B$)</td>
<td></td>
<td>0.000</td>
<td>0.000</td>
<td>0.680</td>
</tr>
<tr>
<td>s.e.($\bar{Q}^B$)</td>
<td></td>
<td>0.000</td>
<td>0.000</td>
<td>0.068</td>
</tr>
</tbody>
</table>

Table 3  Test Problem C: The table reports computational time and solution quality statistics calculated across 100 independent runs of each algorithm using a total simulation budget of $t = 5 \times 10^6$ simulation replications.

<table>
<thead>
<tr>
<th>Performance Metric</th>
<th>Statistic</th>
<th>R-PERLE</th>
<th>R-MinRLE</th>
<th>MO-COMPASS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time in Minutes</td>
<td>$\bar{T}$</td>
<td>39.44</td>
<td>39.69</td>
<td>46.32</td>
</tr>
<tr>
<td>s.d.($T$)</td>
<td></td>
<td>1.04</td>
<td>0.81</td>
<td>3.31</td>
</tr>
<tr>
<td>s.e.($\bar{T}$)</td>
<td></td>
<td>0.10</td>
<td>0.08</td>
<td>0.33</td>
</tr>
<tr>
<td>Solution Quality</td>
<td>$\bar{Q}^C$</td>
<td>0.227</td>
<td>0.264</td>
<td>4.476</td>
</tr>
<tr>
<td>s.d.($Q^C$)</td>
<td></td>
<td>0.222</td>
<td>0.229</td>
<td>3.524</td>
</tr>
<tr>
<td>s.e.($\bar{Q}^C$)</td>
<td></td>
<td>0.022</td>
<td>0.023</td>
<td>0.352</td>
</tr>
</tbody>
</table>
Table 4  Bi-objective 9-Bus Scheduling Problem: The table reports computational time and solution quality statistics calculated across 100 independent runs of each algorithm using a total simulation budget of $t = 1 \times 10^6$ simulation replications.

<table>
<thead>
<tr>
<th>Performance Metric Statistic</th>
<th>R-PERLE</th>
<th>R-MinRLE</th>
<th>MO-COMPASS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time in Hours</td>
<td>$\overline{T}$</td>
<td>16.43</td>
<td>15.25</td>
</tr>
<tr>
<td>$\text{s.d.}(T)$</td>
<td>0.97</td>
<td>0.35</td>
<td>$-$</td>
</tr>
<tr>
<td>$\text{s.e.}(\overline{T})$</td>
<td>0.10</td>
<td>0.03</td>
<td>$-$</td>
</tr>
</tbody>
</table>

$^a$ Results are not available due to large computational time or memory limitations.

Tables 1, 2, and 3 show that R-PERLE, R-MinRLE, and MO-COMPASS require similar computational time for problems with low-dimensional feasible spaces. The feasible spaces for Test Problems $A$ and $B$ are both subsets of $\mathbb{Z}^2$. While all algorithms are slower on Test Problem $C$, which has a feasible space that is a subset of $\mathbb{Z}^3$, MO-COMPASS is slower than R-PERLE and R-MinRLE and has a larger estimated standard deviation of the runtime. We believe that MO-COMPASS is relatively slower because the speed of the calculations required to update the Most Promising Area may be more sensitive to the dimensionality of the feasible space than the calculations required by R-PERLE and R-MinRLE.

Table 4 shows that R-PERLE and R-MinRLE require similar computational time for problems with large, higher-dimensional feasible spaces, with R-MinRLE being slightly faster. While R-PERLE and R-MinRLE require an average computational time of just over 30 minutes for a total budget of 5 million simulation replications on Test Problems $A$, $B$, and $C$, the bi-objective 9-bus scheduling problem takes much longer: Both R-PERLE and R-MinRLE require over 15 hours, on average, for a total budget of 1 million simulation replications.

References