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, kalyanin@gmail.com

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 Thursday 20$^{\text{th}}$ June, 2019 at 6:40pm.

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 (Nsoesie 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

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

where $\mathbf{g}\colon \mathcal{X} \to \mathbb{R}^d$ 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 $\mathcal{X}$; and $\boldsymbol{\xi}$ 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, $\mathcal{X} \subseteq \mathbb{Z}^q$; 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, approximately 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 $\mathbf{x} \in \mathcal{X} \subseteq \mathbb{Z}^q$, can generate n *simulation replications*, or identically distributed copies of the random objective vector $(G_1(\mathbf{x}, \boldsymbol{\xi}_i), \ldots, G_d(\mathbf{x}, \boldsymbol{\xi}_i))$, $i = 1, \ldots, n$. This information is used to construct the consistent estimator $\bar{\mathbf{G}}_n(\mathbf{x}) = (\bar{G}_{1,n}(\mathbf{x}), \ldots, \bar{G}_{d,n}(\mathbf{x})) = \left(\frac{1}{n}\sum_{i=1}^{n} G_1(\mathbf{x}, \boldsymbol{\xi}_i), \ldots, \frac{1}{n}\sum_{i=1}^{n} G_d(\mathbf{x}, \boldsymbol{\xi}_i)\right)$ of the unknown objective function values $\mathbf{g}(\mathbf{x})$. 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 $\mathbf{x} \in \mathbb{Z}^q$ is feasible.

We now discuss these challenges, which are related.

First, consider challenge C.1: Why can't we directly apply a "naïve implementation" of an existing deterministic derivative-free multi-objective optimization algorithm to solve Problem M_d? By "naïve implementation," we mean that the algorithm is unchanged except that at each point $\mathbf{x} \in \mathcal{X}$ it visits, the estimated objective vector $\bar{\mathbf{G}}_n(\mathbf{x})$ is used in place of the true value $\mathbf{g}(\mathbf{x})$. Such an implementation implicitly uses an algorithmic framework called Sample Average Approximation (SAA) (see, e.g., Shapiro et al. 2009, Kim et al. 2015) in

which we replace the unknown vector-valued function $\mathbf{g}(\cdot)$ in Problem M_d with its estimator $\bar{\mathbf{G}}_n(\cdot)$, resulting in the *sample-path problem*

$$\text{Problem } \overline{M}_{d,n}\text{: minimize}_{\mathbf{x}\in\mathcal{X}} \left\{\bar{\mathbf{G}}_n(\mathbf{x}) = \left(\tfrac{1}{n}\sum_{i=1}^{n} G_1(\mathbf{x},\boldsymbol{\xi}_i), \ldots, \tfrac{1}{n}\sum_{i=1}^{n} G_d(\mathbf{x},\boldsymbol{\xi}_i)\right)\right\}.$$

In this context, several complications and questions arise. To begin, we must consider the possibility that for a particular n, the solution to Problem $\overline{M}_{d,n}$ may not exist — even when the solution to Problem M_d does exist. Such a scenario is possible because the objective function estimators are random variables. Then, we must answer the question, how large should the sample size n be? For example, we may wish to determine a value of n that ensures the solution to Problem $\overline{M}_{d,n}$ is within a certain error tolerance of the solution to Problem M_d with high probability, under certain regularity conditions. Notice that because the solution estimators are random variables, traditional deterministic guarantees on optimality are not available; we can make only probabilistic guarantees. Determining the value of n is not a question of optimization but of stochastic error control. We refer the reader to Fu (2002) for a discussion of the importance of explicit stochastic error control in the context of single-objective SO; the same arguments apply to multi-objective SO. Finally, assuming we obtain a suitable value of n, this method of solving Problem M_d is known to be inefficient both in theory and in practice (Pasupathy 2010, Royset and Szechtman 2013) — the same sample size n is used at all decision points, regardless of how close to optimal they are or how large the estimated standard error of each objective vector estimator is. Thus in general, deterministic derivative-free multi-objective optimization methods (e.g., Ralphs et al. 2006, Custódio et al. 2011, Larson et al. 2019, p. 60–62) cannot directly be applied to solve Problem M_d.

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 re-formulates 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 (possibly expensive) stochastic objective vector evaluations, particularly on the (possibly numerous) points in and around the current estimated efficient set. These objective vector 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 local structure and 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 later be embedded within 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. We remark that as high-performance computing environments become increasingly hierarchical (Gropp and Snir 2013), hierarchical algorithms, such as global solvers that coordinate the efforts of multiple asynchronous local solvers, each of which solves sub-problems in parallel, may provide scalability.

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 $\mathcal{X}$ must be *a priori* and *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.

Finally, 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 Problem M_d 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 SAA that is designed for sampling efficiency (Pasupathy and Ghosh 2013). Recall that SAA is an algorithmic framework that replaces the unknown vector-valued function $\mathbf{g}(\cdot)$ in Problem M_d with its estimator $\bar{\mathbf{G}}_n(\cdot)$, resulting in the sample-path problem, Problem $\overline{M}_{d,n}$. 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 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.)

As discussed in §1.1, 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 ν 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 ν. 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 ν, 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 $\boldsymbol{\xi}_i,\ i = 1, \ldots, n$; 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; Pasupathy (2010) employs similar concepts in the context of stochastic root finding. 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. (In case the sample-path approximate local efficient set does not exist at the current sample size m_ν, a bound b_ν limits the total number of simulation replications obtained within RA iteration ν and ensures the solver stops in finite time. Regularity conditions in §8 ensure that as ν increases, a sample-path local efficient set exists and b_ν is non-binding.)

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 *ε-constraint method* (see, e.g., Miettinen 1999, p. 85) to search strategically for new sample-path local efficient points within these bounds. The ε-constraint method consists of selecting one objective to minimize and posing all others as constraints, where the constraint values are defined by "ε's."

The ε-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 ε-constraint problems,

$$\text{Problem } \bar{S}_{2,m_\nu}(k^*, \varepsilon)\text{: } \underset{\mathbf{x}\in\mathcal{X}}{\text{minimize}}\ \bar{G}_{k^*,m_\nu}(\mathbf{x}) \ \text{ s.t. } \ \bar{G}_{k^{\text{con}},m_\nu}(\mathbf{x}) \leq \varepsilon \text{ for } k^{\text{con}} \in \{1, 2\}, k^{\text{con}} \neq k^*,$$

at ε 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 ε-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 $\mathbf{X}^{\text{w}}_{m_\nu}$ in RA iteration ν at sample size m_ν, the PE algorithm places ε 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 $\bar{G}_{k^{\text{con}},m_\nu}(\mathbf{X}^{\text{w}}_{m_\nu}) \pm \widehat{\text{s.e.}}(\bar{G}_{k^{\text{con}},m_\nu}(\mathbf{X}^{\text{w}}_{m_\nu}))(m_\nu^{1/2-\beta_\varepsilon})$, where $\widehat{\text{s.e.}}(\bar{G}_{k^{\text{con}},m_\nu}(\mathbf{X}^{\text{w}}_{m_\nu}))$ is the estimated standard error of the constrained objective function estimator at $\mathbf{X}^{\text{w}}_{m_\nu}$, 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 ε-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 ε-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 contains 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 ($\mathcal{A}$), vectors by bold ($\mathbf{x}$), random vectors by capital bold ($\mathbf{X}$), families of sets by Fraktur ($\mathcal{A} \in \mathfrak{A}$), and operators by blackboard bold ($\mathbb{E}[X]$). The set of all q-dimensional integer-valued vectors is $\mathbb{Z}^q \subset \mathbb{R}^q$. The set of all d-dimensional extended real-valued vectors is $\overline{\mathbb{R}}^d$. The d-dimensional vector $(0, \ldots, 0)$ is denoted $\mathbf{0}_d$. The complement of the set $\mathcal{A}$ is $\mathcal{A}^c$. The sum of two sets $\mathcal{A} \subseteq \mathbb{R}^d$ and $\mathcal{B} \subseteq \mathbb{R}^d$ is the Minkowski sum, $\mathcal{A} + \mathcal{B} := \{\mathbf{a} + \mathbf{b} \colon \mathbf{a} \in \mathcal{A}, \mathbf{b} \in \mathcal{B}\}$. For a sequence of events $\{\mathcal{A}_n\}$ defined in a probability space, we say $\mathcal{A}_n$ *i.o.* if infinitely many of $\mathcal{A}_n$ occur, where $\mathcal{A}_n$ *i.o.* $= \limsup_n \mathcal{A}_n = \cap_{i=1}^{\infty} \cup_{j=i}^{\infty} \mathcal{A}_j$. Finally, let $\mathcal{A} \subset \mathbb{R}^q$ and $\mathcal{B} \subset \mathbb{R}^q$ be nonempty, bounded sets. Then (a) $d(\mathbf{x}, \mathbf{x}') = ||\mathbf{x} - \mathbf{x}'||$ is the Euclidean distance between two points $\mathbf{x}, \mathbf{x}' \in \mathbb{R}^q$; (b) $d(\mathbf{x}, \mathcal{B}) = \inf_{\mathbf{x}' \in \mathcal{B}} ||\mathbf{x} - \mathbf{x}'||$ is the distance from the point $\mathbf{x} \in \mathbb{R}^q$ to the set $\mathcal{B}$; (c) $d(\mathcal{A}, \mathcal{B}) = \sup_{\mathbf{x} \in \mathcal{A}} d(\mathbf{x}, \mathcal{B})$ is the distance from set $\mathcal{A}$ to set $\mathcal{B}$; and (d) $d_H(\mathcal{A}, \mathcal{B}) := \max\{d(\mathcal{A}, \mathcal{B}), d(\mathcal{B}, \mathcal{A})\}$ is the Hausdorff distance between sets $\mathcal{A}$ and $\mathcal{B}$.

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 $\mathbf{x} \in \mathcal{X} \subseteq \mathbb{Z}^q$ and neighborhood size parameter $a \in \overline{\mathbb{R}}$, $a \geq 1$, define the $\mathcal{N}_a$-neighborhood of the point $\mathbf{x}$ as $\mathcal{N}_a(\mathbf{x}) := \{\mathbf{x}' \in \mathbb{Z}^q : d(\mathbf{x}, \mathbf{x}') \leq a\}$. Further, define the

$\mathcal{N}_a$-neighborhood of a set as the union of the $\mathcal{N}_a$-neighborhoods of all the points belonging to the set. That is, for $\mathcal{S} \subset \mathcal{X} \subseteq \mathbb{Z}^q$, the $\mathcal{N}_a$-neighborhood of $\mathcal{S}$ is $\mathcal{N}_a(\mathcal{S}) := \cup_{\mathbf{x}\in\mathcal{S}} \mathcal{N}_a(\mathbf{x})$. Then for any set $\mathcal{A}$, define $\mathcal{N}'_a(\mathcal{A}) := \mathcal{N}_a(\mathcal{A}) \setminus \mathcal{A}$ as the *deleted neighborhood* of $\mathcal{A}$. Second, to compare vectors in the objective function space, we define the following.

DEFINITION 1. Let $\mathbf{x}_1, \mathbf{x}_2 \in \mathcal{X}$ and $d \geq 2$. We say the vector $\mathbf{g}(\mathbf{x}_1)$

1. *weakly dominates* $\mathbf{g}(\mathbf{x}_2)$, written as $\mathbf{g}(\mathbf{x}_1) \leqq \mathbf{g}(\mathbf{x}_2)$, if $g_k(\mathbf{x}_1) \leq g_k(\mathbf{x}_2)$ for all $k = 1, \ldots, d$;
2. *dominates* $\mathbf{g}(\mathbf{x}_2)$, written as $\mathbf{g}(\mathbf{x}_1) \leq \mathbf{g}(\mathbf{x}_2)$, if $\mathbf{g}(\mathbf{x}_1) \leqq \mathbf{g}(\mathbf{x}_2)$ and $\mathbf{g}(\mathbf{x}_1) \neq \mathbf{g}(\mathbf{x}_2)$;
3. *strictly dominates* $\mathbf{g}(\mathbf{x}_2)$, written as $\mathbf{g}(\mathbf{x}_1) < \mathbf{g}(\mathbf{x}_2)$, if $g_k(\mathbf{x}_1) < g_k(\mathbf{x}_2)$ 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 $\mathcal{N}_1$-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 kth objective function as follows.

DEFINITION 2 (WANG ET AL. 2013). Given an objective function $g_k : \mathcal{X} \to \mathbb{R}$, a point $\mathbf{x}_k^{\min} \in \mathcal{X}$ is an $\mathcal{N}_a$-*local minimizer* of g_k if $g_k(\mathbf{x}_k^{\min}) \leq g_k(\mathbf{x})$ for all $\mathbf{x} \in \mathcal{N}_a(\mathbf{x}_k^{\min}) \cap \mathcal{X}$.

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

DEFINITION 3. A point $\mathbf{x}^{\mathrm{w}} \in \mathcal{X}$ is an $\mathcal{N}_a$-*local weakly efficient point* (LWEP) if

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

DEFINITION 4. A point $\mathbf{x}^* \in \mathcal{X}$ is an $\mathcal{N}_a$-*local efficient point* (LEP) if

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

Notice that every $\mathcal{N}_a$-local minimizer of some objective g_k, $k \in \{1, \ldots, d\}$, is an $\mathcal{N}_a$-LWEP, and if the $\mathcal{N}_a$-local minimizer is the unique minimum in its neighborhood, then it is also an $\mathcal{N}_a$-LEP. Further, all $\mathcal{N}_a$-LEP's are $\mathcal{N}_a$-LWEP's. We define a *global minimizer*, a *global weakly efficient point*, and a *global efficient point* as an $\mathcal{N}_a$-local minimizer, $\mathcal{N}_a$-LWEP, and $\mathcal{N}_a$-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 $\mathcal{W}_a \subseteq \mathcal{X}, |\mathcal{W}_a| \geq 1$, is an $\mathcal{N}_a$-*local weakly efficient set* if (a) each $\mathbf{x}^{\mathrm{w}} \in \mathcal{W}_a$ is an $\mathcal{N}_a$-LWEP, and (b) no points in $\mathbf{g}(\mathcal{W}_a)$ strictly dominate other points in $\mathbf{g}(\mathcal{W}_a)$, and (c) for each $\mathbf{x} \in \mathcal{N}'_a(\mathcal{W}_a) \cap \mathcal{X}$, $\exists \mathbf{x}^{\mathrm{w}} \in \mathcal{W}_a$ such that $\mathbf{g}(\mathbf{x}^{\mathrm{w}}) \leqq \mathbf{g}(\mathbf{x})$.

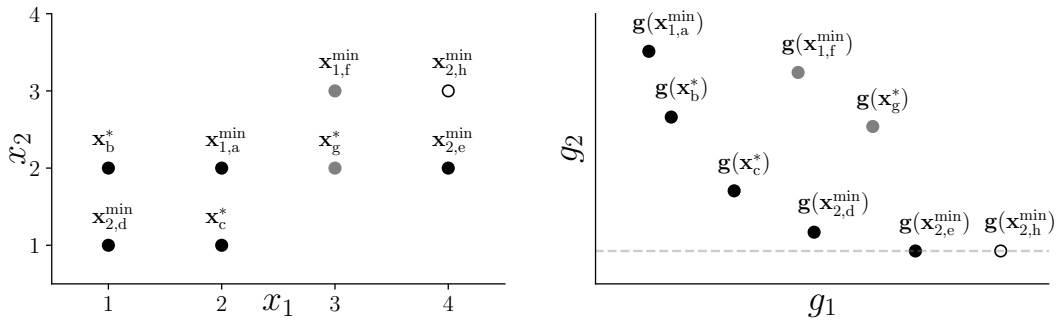


Figure 1 **The figure shows a feasible set on the left, and its image appears on the right. On the left, solid black dots represent global efficient points, open black dots represent global weakly efficient points, and solid gray dots represent $\mathcal{N}_1$-LEP's. $\mathcal{N}_1$-local minimizers are denoted by the superscript** $\min$. **The set $\mathcal{L}_1 = \{\mathbf{x}^{\min}_{1,\mathrm{a}}, \mathbf{x}^*_{\mathrm{b}}, \mathbf{x}^*_{\mathrm{c}}, \mathbf{x}^{\min}_{2,\mathrm{d}}\}$ is an $\mathcal{N}_1$-local efficient set. The set $\mathcal{E} = \{\mathbf{x}^{\min}_{1,\mathrm{a}}, \mathbf{x}^*_{\mathrm{b}}, \mathbf{x}^*_{\mathrm{c}}, \mathbf{x}^{\min}_{2,\mathrm{d}}, \mathbf{x}^{\min}_{2,\mathrm{e}}\}$ is the global efficient set. The set $\mathcal{E}^{\mathrm{w}} = \mathcal{E} \cup \{\mathbf{x}^{\min}_{2,\mathrm{h}}\}$ is the global weakly efficient set. The point $\mathbf{x}^*_{\mathrm{g}}$ is an $\mathcal{N}_1$-LEP that does not belong to an $\mathcal{N}_1$-local efficient set or an $\mathcal{N}_1$-local weakly efficient set.**

DEFINITION 6. A set $\mathcal{L}_a \subseteq \mathcal{X}, |\mathcal{L}_a| \geq 1$, is an *$\mathcal{N}_a$-local efficient set* if (a) each $\mathbf{x}^* \in \mathcal{L}_a$ is an $\mathcal{N}_a$-LEP, and (b) no points in $\mathbf{g}(\mathcal{L}_a)$ dominate other points in $\mathbf{g}(\mathcal{L}_a)$, and (c) for each $\mathbf{x} \in \mathcal{N}'_a(\mathcal{L}_a) \cap \mathcal{X}$, $\exists \mathbf{x}^* \in \mathcal{L}_a$ such that $\mathbf{g}(\mathbf{x}^*) \leq \mathbf{g}(\mathbf{x})$.

Notice that every $\mathcal{N}_a$-local efficient set is also an $\mathcal{N}_a$-local weakly efficient set. Finally, we define the *global weakly efficient set*, denoted $\mathcal{E}^{\mathrm{w}}$, and the *global efficient set*, denoted $\mathcal{E}$, as an $\mathcal{N}_a$-local weakly efficient set and $\mathcal{N}_a$-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 $\mathcal{N}_a$-local Pareto set as the image of an $\mathcal{N}_a$-local efficient set, $\mathbf{g}(\mathcal{L}_a)$.

We remark here that under our definitions, there may exist $\mathcal{N}_a$-LWEP's that do not belong to an $\mathcal{N}_a$-local weakly efficient set. To see an example of such a case, consider Figure 1, and notice that $\mathbf{g}(\mathbf{x}^*_{\mathrm{g}})$ is not dominated by the image of any points in the $\mathcal{N}_1$-neighborhood of $\mathbf{x}^*_{\mathrm{g}}$, which are the points $\mathbf{g}(\mathbf{x}^{\min}_{1,\mathrm{a}}), \mathbf{g}(\mathbf{x}^{\min}_{2,\mathrm{e}})$, and $\mathbf{g}(\mathbf{x}^{\min}_{1,\mathrm{f}})$. Therefore, $\mathbf{x}^*_{\mathrm{g}}$ is an $\mathcal{N}_1$-LWEP. (It is also an $\mathcal{N}_1$-LEP.) However, $\{\mathbf{x}^*_{\mathrm{g}}\}$ is not an $\mathcal{N}_1$-local weakly efficient set because $\mathbf{g}(\mathbf{x}^*_{\mathrm{g}})$ does not dominate $\mathbf{g}(\mathbf{x}^{\min}_{1,\mathrm{a}}), \mathbf{g}(\mathbf{x}^{\min}_{2,\mathrm{e}})$, or $\mathbf{g}(\mathbf{x}^{\min}_{1,\mathrm{f}})$. In this example, it is not possible to construct an $\mathcal{N}_1$-local weakly efficient set using only the points $\mathbf{x}^{\min}_{1,\mathrm{f}}, \mathbf{x}^*_{\mathrm{g}}$, and $\mathbf{x}^{\min}_{2,\mathrm{h}}$ 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 $\mathcal{N}_1$-neighborhood. Thus any $\mathcal{N}_1$-local weakly efficient set including $\mathbf{x}^*_{\mathrm{g}}$ must also include a member of the global efficient set whose image does not dominate $\mathbf{g}(\mathbf{x}^*_{\mathrm{g}})$, such as $\mathbf{x}^{\min}_{1,\mathrm{a}}, \mathbf{x}^*_{\mathrm{b}}$, or $\mathbf{x}^{\min}_{2,\mathrm{e}}$. But, including any of these points in the candidate $\mathcal{N}_1$-local weakly efficient set with $\mathbf{x}^*_{\mathrm{g}}$ implies that there exist neighborhood

points that violate the definition of an $\mathcal{N}_1$-local weakly efficient set. Thus $\mathbf{x}_{\mathrm{g}}^*$ does not belong to an $\mathcal{N}_1$-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 $\bar{\mathbf{G}}_n(\mathbf{x})$ of $\mathbf{g}(\mathbf{x})$ such that $\bar{\mathbf{G}}_n(\mathbf{x}) \to \mathbf{g}(\mathbf{x})$ w.p.1 as the sampling effort $n \to \infty$ for each $\mathbf{x} \in \mathcal{X} \subseteq \mathbb{Z}^q$, find a local solution to Problem M_d, which is an $\mathcal{N}_a$-local efficient set.

3. Solution Context: Preliminaries for the RA Framework

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 $\mathbf{g}(\mathbf{x})$ and $g_k(\mathbf{x})$ with $\bar{\mathbf{G}}_n(\mathbf{x})$ and $\bar{G}_{k,n}(\mathbf{x})$, respectively, for all $k \in \{1, \ldots, d\}$. We denote sample-path $\mathcal{N}_a$-local minimizers, sample-path $\mathcal{N}_a$-LWEP's, and sample-path $\mathcal{N}_a$-LEP's as $\mathbf{X}_{k,n}^{\min}$, $\mathbf{X}_n^{\mathrm{w}}$, and $\mathbf{X}_n^*$, respectively. A local solution to Problem $\overline{M}_{d,n}$ is a sample-path $\mathcal{N}_a$-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 ν is the RA iteration number. Henceforth for simplicity, within an RA iteration ν 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 $\mathcal{N}_a$-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 $\mathcal{N}_a$-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 $\mathcal{A} \subseteq \mathcal{X}$ is a *sample-path approximate $\mathcal{N}_a$-local weakly efficient set* (ALES) for Problem $\overline{M}_{d,n}$ if no points in $\bar{\mathbf{G}}_n(\mathcal{A})$ dominate other points in $\bar{\mathbf{G}}_n(\mathcal{A})$ and, given a vector-valued *completeness function* $\boldsymbol{\delta} \colon \mathcal{X} \to \overline{\mathbb{R}}^d$ such that $\mathbf{0}_d \leqq \boldsymbol{\delta}(\mathbf{x})$ for all $\mathbf{x} \in \mathcal{X}$,

(a) each $\mathbf{x}^{\mathrm{w}} \in \mathcal{A}$ is a sample-path $\mathcal{N}_a$-LWEP, and

(b) for each $\mathbf{x} \in \mathcal{N}_a'(\mathcal{A}) \cap \mathcal{X}$, (i) $\exists \mathbf{x}^{\mathrm{w}} \in \mathcal{A}$ such that $\bar{\mathbf{G}}_n(\mathbf{x}^{\mathrm{w}}) \leqq \bar{\mathbf{G}}_n(\mathbf{x})$, or (ii) $\exists \mathbf{x}^{\mathrm{w}} \in \mathcal{A}$ such that ($\bar{\mathbf{G}}_n(\mathbf{x}) \leq \bar{\mathbf{G}}_n(\mathbf{x}^{\mathrm{w}})$ and $\bar{\mathbf{G}}_n(\mathbf{x}^{\mathrm{w}}) - \boldsymbol{\delta}(\mathbf{x}^{\mathrm{w}}) \leqq \bar{\mathbf{G}}_n(\mathbf{x}) + \boldsymbol{\delta}(\mathbf{x})$), or (iii) $\forall \mathbf{x}^{\mathrm{w}} \in \mathcal{A}$, $\bar{\mathbf{G}}_n(\mathbf{x}) \nleqq \bar{\mathbf{G}}_n(\mathbf{x}^{\mathrm{w}})$, and $\exists \tilde{\mathbf{x}}^{\mathrm{w}} \in \mathcal{A}$ such that $\bar{\mathbf{G}}_n(\tilde{\mathbf{x}}^{\mathrm{w}}) - \boldsymbol{\delta}(\tilde{\mathbf{x}}^{\mathrm{w}}) \leqq \bar{\mathbf{G}}_n(\mathbf{x}) + \boldsymbol{\delta}(\mathbf{x})$ or $\bar{\mathbf{G}}_n(\mathbf{x}) - \boldsymbol{\delta}(\mathbf{x}) \leqq \bar{\mathbf{G}}_n(\tilde{\mathbf{x}}^{\mathrm{w}}) + \boldsymbol{\delta}(\tilde{\mathbf{x}}^{\mathrm{w}})$.

Definition 7 is similar to the definition of a sample-path $\mathcal{N}_a$-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 $\boldsymbol{\delta}$. We call $\boldsymbol{\delta}(\cdot) = (\delta_1(\cdot), \ldots, \delta_d(\cdot))$ the completeness function because it allows the ALES to have neighborhood points that violate the definition of a sample-path $\mathcal{N}_a$-local weakly efficient set, and bigger values of $\boldsymbol{\delta}$ result in a "less-complete" ALES. If $\delta_k(\mathbf{x}) = \infty$ for all $\mathbf{x} \in \mathcal{X}, k \in \{1, \ldots, d\}$, the ALES is a collection of sample-path $\mathcal{N}_a$-LWEP's that may or may not belong to a sample-path $\mathcal{N}_a$-local weakly efficient set. If $\boldsymbol{\delta}(\mathbf{x}) = \mathbf{0}_d$ for all $\mathbf{x} \in \mathcal{X}$, the ALES is a sample-path $\mathcal{N}_a$-local weakly efficient set.

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

For the remainder of the paper, we consider the ALES completeness function specified by $\hat{\mathbf{f}}$. Since the value of this function is random for each $\mathbf{x} \in \mathcal{X}$, henceforth, we denote the completeness function as $\hat{\boldsymbol{\delta}}(\mathbf{x}) := \hat{\mathbf{f}}(\mathbf{x}, \beta_\delta)$ for all $\mathbf{x} \in \mathcal{X}$, where $\beta_\delta \in (0, \infty]$ is the *completeness parameter*. Smaller values of β_δ correspond to larger values of $\hat{\boldsymbol{\delta}}(\mathbf{x})$, 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 $\hat{\boldsymbol{\delta}}(\mathbf{x}) \to 0$ w.p.1 for each $\mathbf{x} \in \mathcal{X}$ 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 ν 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 ν (Algorithm 1 Steps 3 and 4). The solution to Problem $\overline{M}_{2,m_\nu}$ found by PERLE on the νth RA iteration, denoted $\hat{\mathcal{A}}_\nu$ in Algorithm 1, is guaranteed by RLE to be an ALES under mild regularity conditions. For efficiency, R-PERLE uses the sample-path solution from the preceding RA iteration, $\hat{\mathcal{A}}_{\nu-1}$ 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 $\mathbf{x}_0 \in \mathcal{X}$, which is declared as a global variable 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 $\overline{M}_{2,m_\nu}$, e.g., when the solution does not exist. We set this sequence so that for large enough RA iteration numbers ν, under the assumptions in §8, search "time outs" due to binding b_ν do not occur w.p.1. R-PERLE requires parameters $\boldsymbol{\beta} = (\beta_\varepsilon, \beta_\delta)$, 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 $\mathbf{x}_0 \in \mathcal{X}$ 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 $(\mathbf{x}, \bar{\mathbf{G}}_n(\mathbf{x}), \widehat{\text{s.e.}}(\bar{\mathbf{G}}_n(\mathbf{x})))$ is stored

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

Input: initial point $\mathbf{x}_0 \in \mathcal{X}$; sequence of sample sizes to expend at each visited point, $\{m_\nu\}$; sequence of limits on oracle calls during search, $\{b_\nu\}$; ε-placement and ALES parameters, $\boldsymbol{\beta} = (\beta_\varepsilon, \beta_\delta)$

1 Initialize: $\hat{\mathcal{A}}_0 = \{\mathbf{x}_0\}$ and set $\mathbf{x}_0$ as a global variable
2 **for** $\nu = 1, 2, \ldots$ *with CRN* **do**
3 $\quad \hat{\mathcal{A}}_\nu = \text{PE}(\hat{\mathcal{A}}_{\nu-1}, m_\nu, b_\nu, \beta_\varepsilon)$ `/partition and solve ε-constraint problems`
4 $\quad \hat{\mathcal{A}}_\nu = \text{RLE}(\hat{\mathcal{A}}_\nu, m_\nu, b_\nu, \beta_\delta)$ `/guarantee the returned set is an ALES`

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 ν, the PE algorithm chooses an objective k^* to minimize and uses the ε-constraint method to solve Problem $\bar{S}_{2,n}(k^*, \varepsilon)$ at carefully-chosen ε values; recall that $n = m_\nu$ is the current sample size in RA iteration ν. On an integer lattice, the sample-path $\mathcal{N}_a$-local minimizer for Problem $\bar{S}_{2,n}(k^*, \varepsilon)$ 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 ε values and solving each resulting sample-path ε-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 $\bar{S}_{2,n}(k^*, \varepsilon)$ for each chosen ε.

5.1. PE Algorithm Listing

We now discuss PE (Algorithm 2) in detail. First, to ensure that all sample-path ε-constraint problems have a non-empty feasible set, in Step 1, PE obtains updated sample-path $\mathcal{N}_a$-local minimizers on each objective, stored in the set $\hat{\mathcal{A}}_n^0$. (The Min algorithm called in Step 1 is listed in Algorithm 3 and discussed in §5.2.) The sample-path $\mathcal{N}_a$-local minimizers at current sample size n ensure that for the constrained objective k^{con}, no ε values are placed outside of the interval $(\bar{G}_{k^{\text{con}},n}(\mathbf{X}^{\min}_{k^{\text{con}},n}), \bar{G}_{k^{\text{con}},n}(\mathbf{X}^{\min}_{k^*,n})]$ for $k^{\text{con}} \neq k^*$. Thus for every ε-constraint problem posed, there exists a sample-path feasible point in the set $\hat{\mathcal{A}}_n^0$.

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 $\mathcal{N}_a$-LWEP's on both objectives, and ultimately selects the objective that results in solving the *least* number of ε-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

Algorithm 2: $\hat{\mathcal{A}}_{\text{new}} = \text{PE}(\hat{\mathcal{A}}_{\text{old}}, n, b, \beta_\varepsilon)$

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

Output: $\hat{\mathcal{A}}_{\text{new}} \subseteq \mathcal{X}$, a collection of sample-path $\mathcal{N}_a$-LWEP's

1 $\hat{\mathcal{A}}_n^0 = \text{Min}(\hat{\mathcal{A}}_{\text{old}} \cup \{\mathbf{x}_0\}, n, b)$ `/SEARCH: update sample-path` $\mathcal{N}_a$ `-local minimizers`

2 $[\sim, \hat{\mathcal{A}}_n^{\text{w}}, \sim] = \text{RemoveNonLWEP}(\hat{\mathcal{A}}_n^0)$ `/get the set of sample-path` $\mathcal{N}_a$ `-LWEP's in` $\hat{\mathcal{A}}_n^0$

3 **if** $\hat{\mathcal{A}}_n^{\text{w}} = \emptyset$ **then** $\hat{\mathcal{A}}_n^{\text{w}} \leftarrow \hat{\mathcal{A}}_n^0$ `/Min update timed out, no other sample-path` $\mathcal{N}_a$ `-LWEP's exist`

4 Initialize: $c^0 \leftarrow |\hat{\mathcal{A}}_n^{\text{w}}|$ `/set` ε `'s using sample-path` $\mathcal{N}_a$ `-LWEP's`

5 **for** $k^* = 1, 2$ **do** `/determine objective to minimize,` k^*

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

7 Sort $\bar{\mathbf{G}}_n(\hat{\mathcal{A}}_n^{\text{w}})$ on k^{con} to get $(\mathbf{X}_{(1)}^{\text{w}}, \ldots, \mathbf{X}_{(c^0)}^{\text{w}})$ where $\bar{G}_{k^{\text{con}},n}(\mathbf{X}_{(1)}^{\text{w}}) \leq \ldots \leq \bar{G}_{k^{\text{con}},n}(\mathbf{X}_{(c^0)}^{\text{w}})$

8 Set constraint lower bound $L_{k^*} \leftarrow \bar{G}_{k^{\text{con}},n}(\mathbf{X}_{(1)}^{\text{w}}) + \hat{f}_{k^{\text{con}}}(\mathbf{X}_{(1)}^{\text{w}}, \beta_\varepsilon)$

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

10 $\varepsilon_{k^*}^{\text{L}}(i) \leftarrow \bar{G}_{k^{\text{con}},n}(\mathbf{X}_{(i)}^{\text{w}}) - \hat{f}_{k^{\text{con}}}(\mathbf{X}_{(i)}^{\text{w}}, \beta_\varepsilon)$ and $\varepsilon_{k^*}^{\text{U}}(i) \leftarrow \bar{G}_{k^{\text{con}},n}(\mathbf{X}_{(i)}^{\text{w}}) + \hat{f}_{k^{\text{con}}}(\mathbf{X}_{(i)}^{\text{w}}, \beta_\varepsilon)$

11 $\mathcal{C}_{k^*} \leftarrow \{\varepsilon_{k^*}^{\text{L}}(i) \colon i \in \{2, \ldots, c^0\},\ L_{k^*} < \varepsilon_{k^*}^{\text{L}}(i),\ \varepsilon_{k^*}^{\text{L}}(i) \notin (\varepsilon_{k^*}^{\text{L}}(i'), \varepsilon_{k^*}^{\text{U}}(i')] \text{ for all } i' \in \{2, \ldots, c^0\}\}$

12 $C_{k^*} \leftarrow |\mathcal{C}_{k^*}|$

13 $K^* \leftarrow \arg\min\{C_{k^*} \colon k^* \in \{1, 2\}\}$ `/choose` ***least*** ε `-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 $\mathcal{C}_{K^*}$ in ascending order to get the ordered list $(\varepsilon_1, \ldots, \varepsilon_C)$

17 **for** $j = 1, 2, \ldots, C$ **do** `/partition space`

18 $\varepsilon_j^{\text{L}} \leftarrow \max(L_{K^*}, \max\{\varepsilon_{K^*}^{\text{U}}(i) \colon \varepsilon_{K^*}^{\text{U}}(i) < \varepsilon_j, i \in \{2, \ldots, c^0\}\})$ `/get traceback lower bound`

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

20 **while** $\varepsilon_j^{\text{L}} < \varepsilon_{\text{new}}$ **do** `/find new sample-path` $\mathcal{N}_a$ `-LWEP's`

21 $\mathbf{X}^0 \leftarrow \arg\min\{\bar{G}_{K^*,n}(\mathbf{X}) \colon \mathbf{X} \in \mathcal{T} \cup \hat{\mathcal{A}}_n^0,\ \bar{G}_{K^{\text{con}},n}(\mathbf{X}) \leq \varepsilon_{\text{new}}\}$

22 $[\mathbf{X}_n^{\text{w}}, \mathcal{T}', \mathcal{N}(\mathbf{X}_n^{\text{w}})] = \text{SPLINE}(K^*, \mathbf{X}^0, n, b, \varepsilon_{\text{new}})$ `/SEARCH: solve` ε `-constrained`

23 $\hat{\mathcal{A}}_j \leftarrow \hat{\mathcal{A}}_j \cup \{\mathbf{X}_n^{\text{w}}\}$ and $\mathcal{T} \leftarrow \mathcal{T} \cup \mathcal{T}'$

24 $\varepsilon_{\text{new}} \leftarrow \bar{G}_{K^{\text{con}},n}(\mathbf{X}_n^{\text{w}}) - f_{K^{\text{con}}}(\mathbf{X}_n^{\text{w}}, \beta_\varepsilon)$ `/traceback: set new` ε `to disqualify`

25 $\hat{\mathcal{A}}_{\text{LWEP}} = \cup_{j=1}^{C} \hat{\mathcal{A}}_j$ `/collect new sample-path` $\mathcal{N}_a$ `-LWEP's`

26 $\hat{\mathcal{A}}_{\text{new}} = \text{RemoveDominated}(\bar{\mathbf{G}}_n(\hat{\mathcal{A}}_n^{\text{w}} \cup \hat{\mathcal{A}}_{\text{LWEP}} \cup \{\mathbf{x}_0\}))$ `/remove dominated, do no harm`

points on a high standard error objective. To determine the set of ε-constraint values for each objective, first, in Step 2, PE creates an initial set of known sample-path $\mathcal{N}_a$-LWEP's, $\hat{\mathcal{A}}_n^{\text{w}}$, from the points in $\hat{\mathcal{A}}_n^0$ using the RemoveNonLWEP function (discussed in §6.2). If there are no such points in $\hat{\mathcal{A}}_n^0$ in Step 3, the search budget value b must have been binding in the Min algorithm. Then PE sets ε values based on $\hat{\mathcal{A}}_n^0$. Since b is non-binding in the limit, for large enough RA iteration number ν, all ε values will be set based on known sample-path $\mathcal{N}_a$-LWEP's. In Steps 5 through 12, PE sorts the initial points and partitions the objective space by placing ε 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 $\hat{\mathbf{f}}$ from §3.2. For all known sample-path $\mathcal{N}_a$-LWEP's, denoted $\mathbf{X}_n^{\text{w}}$, the PE algorithm does not place any new ε values in the interval $\bar{G}_{k^{\text{con}},n}(\mathbf{X}_n^{\text{w}}) \pm \hat{f}_{k^{\text{con}}}(\mathbf{X}_n^{\text{w}}, \beta_\varepsilon)$,

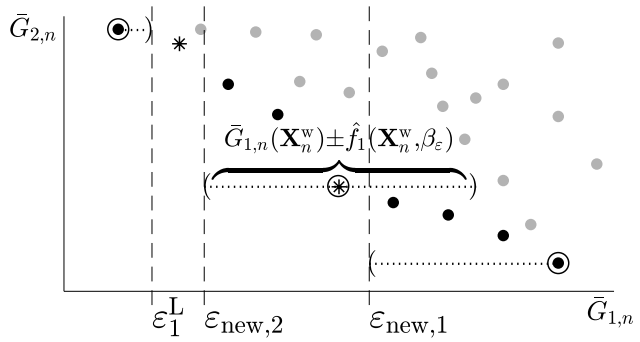


Figure 2 **The figure shows part of a "traceback" from PE Steps 20–24, assuming objective 1 is constrained and that in PE Step 4, $\hat{\mathcal{A}}_n^{\mathrm{w}}$ has exactly two sample-path local minimizers: one on each objective, represented by outlined solid black dots. The traceback places the first ε-constraint at $\varepsilon_{\mathrm{new},1}$, thus retrieving the outlined black star point. Then, it places $\varepsilon_{\mathrm{new},2}$ and retrieves the black star point, and so on, until a value of $\varepsilon_{\mathrm{new}} < \varepsilon_1^{\mathrm{L}}$.**

where $\beta_\varepsilon \in (0, \infty)$ is a parameter. The β_ε parameter controls how large the standard error intervals are; a smaller β_ε value results in wider intervals. Once both sets of ε-constraint values have been determined, Steps 13 and 14 select the objective with the least ε's to minimize, where ties are broken randomly.

If the objective chosen to minimize results in one or more ε-constraint problems, Steps 15 through 25 solve these problems. First, PE partitions the objective space based on the ε values. Then, each ε-constraint problem is solved using the SPLINE algorithm (see §5.3). Within a partition, once a new sample-path $\mathcal{N}_a$-LWEP is found, PE performs what we call a *traceback* in Step 24 by attempting to place a new ε value that is both within the current partition and that sets as infeasible the newly-found sample-path $\mathcal{N}_a$-LWEP, as shown in Figure 2. If the new ε value is outside the partition, the search ends; otherwise, the new ε-constraint problem is solved, and this process repeats until no more ε 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 $\nu \to \infty$ in R-PERLE, under appropriate regularity conditions, the PE algorithm solves as many ε-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

Algorithm 3: $\hat{\mathcal{A}}_n^* = \text{Min}(\hat{\mathcal{A}}_{\text{old}}, n, b)$

Input: a set of feasible points $\hat{\mathcal{A}}_{\text{old}} \subset \mathcal{X}$; sample size, n; limit on SPLINE calls, b
Output: $\hat{\mathcal{A}}_n^*$, a candidate ALES with updated sample-path local minimizers at sample size n

for $k = 1, 2, \ldots, d$ **do**
  $\mathbf{X}_{\text{k,old}}^{\min} \leftarrow \arg\min\{\bar{G}_{k,n}(\mathbf{X}) \colon \mathbf{X} \in \hat{\mathcal{A}}_{\text{old}}\}$
  $[\mathbf{X}_{k,n}^{\min}, \sim, \mathcal{N}(\mathbf{X}_{k,n}^{\min})] = \text{SPLINE}(k, \mathbf{X}_{\text{k,old}}^{\min}, n, b, \sim)$ `/SEARCH: sample-path` $\mathcal{N}_a$`-local minimizer`
$\overline{\mathcal{M}}_n \leftarrow \cup_{k=1}^{d} \{\mathbf{X}_{k,n}^{\min}\}$ `/points in` $\bar{\mathbf{G}}_n(\overline{\mathcal{M}}_n)$ `may dominate other points in` $\bar{\mathbf{G}}_n(\overline{\mathcal{M}}_n)$
$\hat{\mathcal{A}}_n^* = \text{RemoveDominated}(\bar{\mathbf{G}}_n(\hat{\mathcal{A}}_{\text{old}} \cup \overline{\mathcal{M}}_n \cup \{\mathbf{x}_0\}))$

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 $\hat{\mathcal{A}}_n^*$.

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 ε-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 $\tilde{\mathcal{X}} \subseteq \mathcal{X}$ 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 $\tilde{\mathcal{X}}$. Our ability to pass additional bound constraints to SPLINE enables us to restrict the search space when solving ε-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: $[\mathbf{X}^*, \mathcal{T}, \mathcal{N}(\mathbf{X}^*)] = \text{SPLINE}(k, \mathbf{X}_0, n, b, \mathcal{G})$

Input: objective k; initial point $\mathbf{X}_0 \in \tilde{\mathcal{X}} \subseteq \mathcal{X}$; sample size n; limit on search oracle calls, b; optional bound constraint values $\mathcal{G}$ that define $\tilde{\mathcal{X}} \subseteq \mathcal{X}$
Output: local solution $\mathbf{X}^*$ on $\tilde{\mathcal{X}} \subseteq \mathcal{X}$; sample-path search set, $\mathcal{T}$; neighborhood points, $\mathcal{N}(\mathbf{X}^*)$

Initialize: search oracle calls spent so far $b^* \leftarrow 0$, $\mathbf{X}_{\text{NE}} \leftarrow \mathbf{X}_0$, and $\mathcal{T} \leftarrow \{\mathbf{X}_0\}$
repeat
  $[b', \mathbf{X}_{\text{SPLI}}, \bar{\mathbf{G}}_n(\mathbf{X}_{\text{SPLI}})] = \text{SPLI}(k, \mathbf{X}_{\text{NE}}, n, b, \mathcal{G})$ `/SEARCH: line search with interpolation`
  if $\bar{G}_{k,n}(\mathbf{X}_{\text{SPLI}}) > \bar{G}_{k,n}(\mathbf{X}_{\text{NE}})$ **then** $\mathbf{X}_{\text{SPLI}} \leftarrow \mathbf{X}_{\text{NE}}$ `/SPLI cannot cause harm`
  $[b'', \mathbf{X}_{\text{NE}}, \bar{\mathbf{G}}_n(\mathbf{X}_{\text{NE}}), \mathcal{N}(\mathbf{X}_{\text{NE}})] = \text{NE}(k, \mathbf{X}_{\text{SPLI}}, n, \mathcal{G})$ `/neighborhood enumeration`
  $\mathcal{T} \leftarrow \mathcal{T} \cup \{\mathbf{X}_{\text{SPLI}}, \mathbf{X}_{\text{NE}}\}$ `/update trajectory`
  $b^* \leftarrow b^* + b' + b''$ `/update oracle calls expended`
until $\bar{G}_{k,n}(\mathbf{X}_{\text{NE}}) = \bar{G}_{k,n}(\mathbf{X}_{\text{SPLI}})$ *or* $b^* > b$ `/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 $\mathcal{N}_a$-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 $\mathcal{N}_a$-LWEP's may "get stuck" by returning points that do not belong to the same, or to any, sample-path $\mathcal{N}_a$-local weakly efficient set. For example, in Figure 1, the PE algorithm may return $\mathcal{S} = \{\mathbf{x}^{\min}_{1,\mathrm{f}}, \mathbf{x}^{*}_{\mathrm{g}}, \mathbf{x}^{\min}_{2,\mathrm{e}}\}$, which is a set of $\mathcal{N}_a$-LWEP's containing a minimum on each objective but that is not an $\mathcal{N}_a$-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 $\mathcal{S} \subseteq \mathcal{X}$ such that none of the estimated images of points in $\mathcal{S}$ dominate the estimated images of other points in $\mathcal{S}$. The non-conforming neighborhood of $\mathcal{S}$ is the set of points in the deleted neighborhood of $\mathcal{S}$ that prevent it from being an ALES.

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

(a) $\exists \tilde{\mathbf{x}} \in \mathcal{S}$ such that $\mathbf{x} \in \mathcal{N}_a(\tilde{\mathbf{x}})$ and $\bar{\mathbf{G}}_n(\mathbf{x}) < \bar{\mathbf{G}}_n(\tilde{\mathbf{x}})$, or

(b) (i) $\nexists \tilde{\mathbf{x}} \in \mathcal{S}$ such that $\bar{\mathbf{G}}_n(\tilde{\mathbf{x}}) \leqq \bar{\mathbf{G}}_n(\mathbf{x})$, and (ii) $\nexists \tilde{\mathbf{x}} \in \mathcal{S}$ such that $(\bar{\mathbf{G}}_n(\mathbf{x}) \leq \bar{\mathbf{G}}_n(\tilde{\mathbf{x}})$ and $\bar{\mathbf{G}}_n(\tilde{\mathbf{x}}) - \boldsymbol{\delta}(\tilde{\mathbf{x}}) \leqq \bar{\mathbf{G}}_n(\mathbf{x}) + \boldsymbol{\delta}(\mathbf{x}))$, and (iii) $\exists \tilde{\mathbf{x}} \in \mathcal{S}$ such that $\bar{\mathbf{G}}_n(\mathbf{x}) \leqq \bar{\mathbf{G}}_n(\tilde{\mathbf{x}})$, or $\nexists \tilde{\mathbf{x}} \in \mathcal{S}$ such that $\bar{\mathbf{G}}_n(\tilde{\mathbf{x}}) - \boldsymbol{\delta}(\tilde{\mathbf{x}}) \leqq \bar{\mathbf{G}}_n(\mathbf{x}) + \boldsymbol{\delta}(\mathbf{x})$ or $\bar{\mathbf{G}}_n(\mathbf{x}) - \boldsymbol{\delta}(\mathbf{x}) \leqq \bar{\mathbf{G}}_n(\tilde{\mathbf{x}}) + \boldsymbol{\delta}(\tilde{\mathbf{x}})$.

First, Definition 8(a) adds $\mathbf{x} \in \mathcal{N}'_a(\mathcal{S}) \cap \mathcal{X}$ to the NCN if it prevents a point in $\mathcal{S}$ from being a sample-path $\mathcal{N}_a$-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 $\mathcal{S}$, and (ii) does not dominate any points in $\mathcal{S}$ by less than a certain amount, and (iii) weakly dominates a point in $\mathcal{S}$, or would not either weakly dominate or be weakly dominated by a point in $\mathcal{S}$ if both were moved by a certain amount.

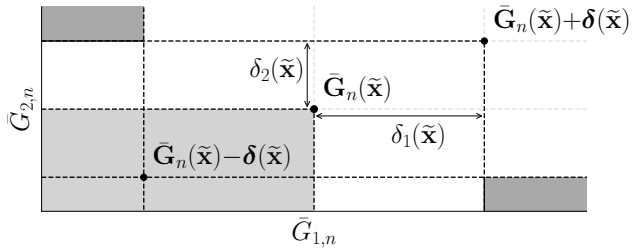


Figure 3 **Let $\mathcal{S}=\{\tilde{\mathbf{x}}\}$ with two objectives, and let $\mathbf{x}\in\mathcal{N}_a'(\mathcal{S})\cap\mathcal{X}$ be in the deleted neighborhood of $\mathcal{S}$. Def. 8(a) adds $\mathbf{x}$ to the NCN if $\mathbf{x}\in\mathcal{N}_a(\tilde{\mathbf{x}})$ and $\bar{\mathbf{G}}_n(\mathbf{x})$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

Given a single point $\mathcal{S}=\{\tilde{\mathbf{x}}\}\subseteq\mathcal{X}$, Figure 3 shows regions of the objective space that correspond to a feasible point in the deleted neighborhood of $\mathcal{S}$, $\mathbf{x}\in\mathcal{N}_a'(\mathcal{S})\cap\mathcal{X}$, being declared a member of the NCN. Definition 8(a) implies $\mathbf{x}$ is in the NCN if it is an $\mathcal{N}_a$-neighbor of $\tilde{\mathbf{x}}$ and $\bar{\mathbf{G}}_n(\mathbf{x})$ is in the light gray region of Figure 3. If $\mathbf{x}$ does not meet the requirements of Definition 8(a), Definition 8(b) implies that $\mathbf{x}$ is in the NCN if its entire "$\boldsymbol{\delta}$ box," defined by the corners $\bar{\mathbf{G}}_n(\mathbf{x})\pm\boldsymbol{\delta}(\mathbf{x})$, is completely contained in the dark gray shaded region of Figure 3. Thus there is no overlap between the $\boldsymbol{\delta}$ boxes of $\mathbf{x}$ and $\tilde{\mathbf{x}}$ on any objective. Henceforth, we use $\hat{\boldsymbol{\delta}}(\mathbf{x})=\hat{\mathbf{f}}(\mathbf{x},\beta_\delta)$ for all $\mathbf{x}\in\mathcal{X}$ (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 $\mathcal{S}$ whose estimated objective vectors are dominated by the estimated objective vectors of other points in $\mathcal{S}$. 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.

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,

Algorithm 5: $\hat{\mathcal{A}}_{\text{ALES}} = \text{RLE}(\mathcal{S}, n, b, \beta_\delta)$

Input: set of points $\mathcal{S} \subset \mathcal{X}$; limit on search oracle calls, b; ALES completeness parameter β_δ
Output: $\hat{\mathcal{A}}_{\text{ALES}}$, which is an ALES for the sample-path problem with sample size n

1 $\mathcal{S} = \text{RemoveDominated}(\bar{\mathbf{G}}_n(\mathcal{S} \cup \{\mathbf{x}_0\}))$
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 $[b', \mathcal{N}^{\text{w}}, \mathcal{N}_2^*] = \text{RemoveNonLWEP}(\mathcal{N}^{\text{nc}})$
6 $\mathcal{S} \leftarrow \mathcal{S} \cup \mathcal{N}^{\text{w}}$ `/add neighborhood sample-path` $\mathcal{N}_a$`-LWEP's to` $\mathcal{S}$
7 **if** $\mathcal{N}^{\text{w}} = \emptyset$ **then** `/the neighbors are dominated by their neighbors:` $\mathcal{N}_2^* \neq \emptyset$
8 Initialize: $\mathcal{X}^{\text{new}} \leftarrow \mathcal{N}_2^*$, $\mathcal{X}^{\text{w}} \leftarrow \emptyset$, and inner search oracle calls spent so far $b^{**} \leftarrow 0$
9 **while** $b^{**} \leq b$ *and* $\mathcal{X}^{\text{w}} = \emptyset$ **do** `/SEARCH: traverse dominating chains`
10 $[b''', \mathcal{X}^{\text{w}}, \mathcal{N}_2^*] = \text{RemoveNonLWEP}(\mathcal{X}^{\text{new}})$
11 $\mathcal{X}^{\text{new}} \leftarrow \mathcal{N}_2^*$ and $b^{**} \leftarrow b^{**} + b'''$
12 $\mathcal{S} \leftarrow \mathcal{S} \cup \mathcal{X}^{\text{w}}$
13 **if** $\mathcal{X}^{\text{w}} = \emptyset$ **then** $\mathcal{S} \leftarrow \mathcal{S} \cup \mathcal{X}^{\text{new}}$ `/keep progress if search times out`
14 $\mathcal{S} = \text{RemoveDominated}(\bar{\mathbf{G}}_n(\mathcal{S} \cup \{\mathbf{x}_0\}))$
15 $[b'', \mathcal{N}^{\text{nc}}] = \text{GetNCN}(\mathcal{S}, \beta_\delta)$ `/get non-conforming neighborhood`
16 $b^* \leftarrow b^* + b' + b''$
17 **return** $\hat{\mathcal{A}}_{\text{ALES}} \leftarrow \mathcal{S}$

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 ν, 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

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

Input: initial point $\mathbf{x}_0 \in \mathcal{X}$; 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, β_δ

Initialize: $\hat{\mathcal{A}}_0 = \{\mathbf{x}_0\}$ and set $\mathbf{x}_0$ as a global variable

for $\nu = 1, 2, \ldots$ *with CRN* **do**

  $\hat{\mathcal{A}}_{\min} = \text{Min}(\hat{\mathcal{A}}_{\nu-1}, m_\nu, b_\nu)$ `/update the sample-path` $\mathcal{N}_a$`-local minimizers`

  $\hat{\mathcal{A}}_\nu = \text{RLE}(\hat{\mathcal{A}}_{\min}, m_\nu, b_\nu, \beta_\delta)$ `/guarantee the returned set is an ALES`

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 β_ε 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 $\overline{M}_{d,m_\nu}$ 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 β_δ 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 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 $\mathcal{N}_a$-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 $\hat{\boldsymbol{\delta}} = \hat{\mathbf{f}}(\mathbf{x}, \beta_\delta)$ 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 ν large enough, under our regularity conditions. Thus we ignore issues related to binding budget sequences for small ν.

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 $\mathbf{x} \in \mathcal{X}$ and $k \in \{1, \ldots, d\}$, there exists $\alpha_k > 0$, dependent on $\mathbf{x}$, such that $\mathcal{S}_k(\mathbf{x}, \alpha_k) := \{\mathbf{x}' \in \mathcal{X} \colon g_k(\mathbf{x}') \leq g_k(\mathbf{x}) + \alpha_k\}$ is finite.*

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

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

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

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 $\mathfrak{W}_a$ and $\mathfrak{L}_a$ 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 $\mathfrak{L}_a \subseteq \mathfrak{W}_a$. Since the global weakly efficient set is also an $\mathcal{N}_a$-local weakly efficient set for $a \geq 1$, notice that $|\mathfrak{W}_a| \geq 1$ 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, $\mathcal{E}^{\mathrm{w}} \subseteq \mathcal{X}$, 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 $\mathcal{W}_a \in \mathfrak{W}_a$.

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 $\sigma_k^2(\mathbf{x})$ is $\mathbb{V}(G_k(\mathbf{x}, \xi))$ for each $\mathbf{x} \in \mathcal{X}$ 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 $\{\bar{G}_{k,m_\nu}(\mathbf{x}) - g_k(\mathbf{x})\}$ is governed by a large-deviation principle with rate function $I_{k,\mathbf{x}}(s)$ (Dembo and Zeitouni 1998); (b) each $I_{k,\mathbf{x}}(s)$ is such that for any $\epsilon > 0$, $\inf_{\mathbf{x}\in\mathcal{X},\, k\in\{1,\ldots,d\}} \min(I_{k,\mathbf{x}}(\epsilon), I_{k,\mathbf{x}}(-\epsilon)) = \eta > 0$; and (c) the sequence of sample sizes $\{m_\nu\}$ increases faster than logarithmically, that is, $\limsup_{\nu\to\infty} (m_\nu)^{-1} (\log \nu)^{1+\Delta_1} = 0$ 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 $\{\bar{G}_{k,m_\nu}(\mathbf{x})\}$ for each $\mathbf{x} \in \mathcal{X}$, that is, $\sqrt{m_\nu}(\sigma_k(\mathbf{x}))^{-1}(\bar{G}_{k,m_\nu}(\mathbf{x}) - g_k(\mathbf{x})) \Rightarrow Z$, where $\sigma_k(\mathbf{x}) > 0$ satisfies $\sup_{\mathbf{x}\in\mathcal{X}} \sigma_k^2(\mathbf{x}) < \infty$, and (b) as $\nu \to \infty$, $\sup_y |F_{k,\mathbf{x},m_\nu}(y) - \Phi(y)| = O(1/\sqrt{m_\nu})$ for all $\mathbf{x} \in \mathcal{X}$, where $F_{k,\mathbf{x},m_\nu}(\cdot)$ denotes the cumulative distribution function of the random variable $\sqrt{m_\nu}(\sigma_k(\mathbf{x}))^{-1}(\bar{G}_{k,m_\nu}(\mathbf{x}) - g_k(\mathbf{x}))$, and (c) the sequences of sample sizes $\{m_\nu\}$ satisfies $\limsup_{\nu\to\infty} (m_\nu)^{-1} \nu^{2+\Delta_2} = 0$ 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 $\mathcal{S}(\mathbf{x}, \boldsymbol{\alpha}) = \cup_{k=1}^d \mathcal{S}_k(\mathbf{x}, \alpha_k)$ is finite, and define $\hat{\mathcal{S}}_\nu(\mathbf{x}) := \cup_{k=1}^d \hat{\mathcal{S}}_{k,\nu}(\mathbf{x})$ 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* $\hat{\mathcal{S}}_{k,\nu}(\mathbf{x})$ *converge almost surely into the set* $\mathcal{S}_k(\mathbf{x}, \alpha_k)$*, that is,* $\mathbb{P}\{\hat{\mathcal{S}}_{k,\nu}(\mathbf{x}) \not\subseteq \mathcal{S}_k(\mathbf{x}, \alpha_k)\ i.o.\} = 0$;

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

Finally, we need our last required assumption, Assumption 3, because our ε-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* $\mathbf{g}(\mathbf{x}) = \mathbf{g}(\mathbf{x}')$*, 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* $\mathcal{N}_a$*-LWEP's are global efficient points and there exists exactly one* $\mathcal{N}_a$*-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 $\mathcal{N}_a$-local efficient set within each $\mathcal{N}_a$-local weakly efficient set. We present the result without proof; intuitively, it follows because Assumption 4 prevents the points in the $\mathcal{N}_a$-local weakly efficient set from having identical objective vector values. Thus the set must contain $\mathcal{N}_a$-LEP's, from which the $\mathcal{N}_a$-local efficient set can be constructed.

LEMMA 4. *Under Assumptions 1 and 4, given* $a \in (0, \infty]$*, all* $\mathcal{N}_a$*-local weakly efficient sets contain an* $\mathcal{N}_a$*-local efficient set; that is, for each* $\mathcal{W}_a \in \mathfrak{W}_a$*,* $\exists \mathcal{L}_a^* \in \mathfrak{L}_a$ *such that* $\mathcal{W}_a \supseteq \mathcal{L}_a^*$.

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 $\mathcal{N}_a$-LWEP is an $\mathcal{N}_a$-LEP, and every $\mathcal{N}_a$-local weakly efficient set is an $\mathcal{N}_a$-local efficient set. Assumption 6 is required for the convergence of R-PE, and stipulates that every $\mathcal{N}_a$-LWEP is a global efficient point, and there exists exactly one $\mathcal{N}_a$-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 $\mathcal{N}_a$-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 $\mathbf{x}_0 \in \mathcal{X}$, ε-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 $\{\hat{\mathcal{A}}_\nu\}$ such that*

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

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

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

(d) under Assumptions 5 and 6, $\{\hat{\mathcal{A}}_\nu\}$ converges to the global efficient set almost surely, that is, $\mathbb{P}\{\hat{\mathcal{A}}_\nu \neq \mathcal{E} \; i.o.\} = 0$.

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 $\mathcal{N}_a$-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 $\mathcal{N}_a$-local minimizers produced by PE in Step 1 across RA iterations, defined as $\{\overline{\mathcal{M}}_\nu, \nu = 1, 2, \ldots\}$ where $\overline{\mathcal{M}}_\nu = \{\mathbf{X}^{\min}_{1,m_\nu}, \mathbf{X}^{\min}_{2,m_\nu}\}$ for all $\nu = 1, 2, \ldots$, converges into the set of all true $\mathcal{N}_a$-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 $\mathbf{x}_0 \in \mathcal{X}$, and ε-placement rule, across RA iterations, PE Step 1 generates a sequence of sample-path $\mathcal{N}_a$-local minimizers $\{\overline{\mathcal{M}}_\nu\}$ that converges into $\mathcal{M}_a^*$ almost surely, that is, $\mathbb{P}\{\overline{\mathcal{M}}_\nu \not\subseteq \mathcal{M}_a^* \; i.o.\} = 0$.*

THEOREM 2. *Under Assumptions 1–6, for $d=2$, any neighborhood size $a \in [1,\infty)$, initial point $\mathbf{x}_0 \in \mathcal{X}$, and ε-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 $\mathbb{P}\{\hat{\mathcal{A}}_\nu \neq \mathcal{E} \; i.o.\} = 0$.*

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 $\mathcal{X}^{\mathrm{w}}$ denote the set of all $\mathcal{N}_1$-LWEP's for Problem M_d, and let $\bar{\mathcal{X}}_\nu^{\mathrm{w}}$ denote the set of all sample-path $\mathcal{N}_1$-LWEP's on the νth RA iteration. Further, let $\hat{\mathcal{A}}_\nu$ denote the solution returned on the νth RA iteration of R-PERLE ($d=2$), R-PE ($d=2$), or R-MinRLE ($d \geq 2$) for any $\mathbf{x}_0 \in \mathcal{X}$, ε-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 $\max_{k\in\{1,\ldots,d\}} \sup_{\mathbf{x}\in\mathcal{X}} \sigma_k^2(\mathbf{x}) < \infty$. For all objectives $k \in \{1,\ldots,d\}$, let the sequence of random variables $\{\bar{G}_{k,m_\nu}(\mathbf{x}) - g_k(\mathbf{x})\}$ be governed by a large-deviation principle with rate function $I_{k,\mathbf{x}}(s)$, as stipulated in Lemma 2. Then the following hold:*

(a) $\mathbb{P}\{\bar{\mathcal{X}}_\nu^{\mathrm{w}} \not\subseteq \mathcal{X}^{\mathrm{w}}\} = O(e^{-\gamma m_\nu})$ 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 $\limsup_{\nu\to\infty} m_\nu^{-1}\nu < \infty$, then

(i) $\mathbb{P}\{\hat{\mathcal{A}}_\nu \not\subseteq \mathcal{X}^{\mathrm{w}}\} = O(e^{-\gamma m_\nu})$ for some $\gamma > 0$,

(ii) under Assumption 5 and 6, $\mathbb{P}\{\hat{\mathcal{A}}_\nu \neq \mathcal{E}\} = O(e^{-\gamma m_\nu})$ 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 $\boldsymbol{\beta} = (\beta_\varepsilon, \beta_\delta)$ 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 $(\mathbf{x}, \bar{\mathbf{G}}_n(\mathbf{x}), \widehat{\text{s.e.}}(\bar{\mathbf{G}}_n(\mathbf{x})))$ 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 $\mathcal{N}_a$-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, $\mathbf{x}_0$, to a sample-path $\mathcal{N}_a$-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 ν, the sample size limit b_ν 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 ε values in PE and the completeness of the ALES returned by RLE using the parameters β_ε and β_δ. 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 $\mathbf{x}_0$ 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

$$\text{Problem } T_A\text{:} \qquad \text{minimize}_{\mathbf{x}\in\mathcal{X}} \begin{cases} g_1(\mathbf{x}) = \mathbb{E}[(x_1/10 - 2\xi_1)^2 + (x_2/10 - \xi_2)^2] \\ g_2(\mathbf{x}) = \mathbb{E}[x_1^2/100 + (x_2/10 - 2\xi_3)^2] \end{cases}$$

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 ξ_i 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 $i \in \{1, 2, 3\}$. Thus the random objective values returned by the simulation oracle are independent for each $\mathbf{x} \in \mathcal{X}$. 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

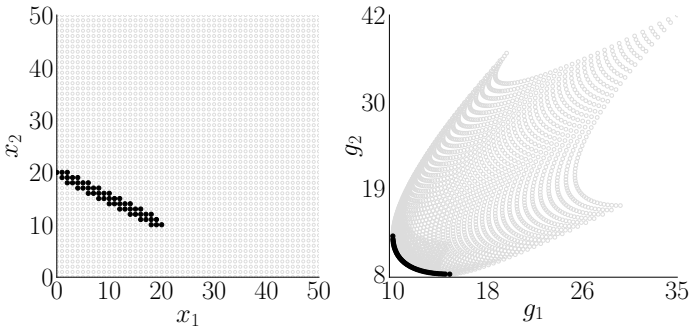


Figure 4 **Problem T_A: The black dots represent points in the global efficient set (left) and their images (right). The global efficient set is the only $\mathcal{N}_1$-local efficient set.**

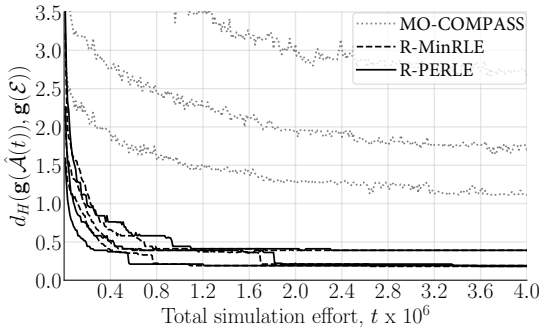


Figure 5 **Problem T_A: Sample quantiles (.25, .5, .75) of the coverage error across 1,000 independent runs per algorithm.**

distance between the image of the set returned by the algorithm and the image of the true efficient set as a function of t, $d_H(\mathbf{g}(\hat{\mathcal{A}}(t)), \mathbf{g}(\mathcal{E}))$, where $\hat{\mathcal{A}}(t)$ 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

$$\text{Problem } T_B\text{:} \qquad \text{minimize}_{\mathbf{x}\in\mathcal{X}} \begin{cases} g_1(\mathbf{x}) = \mathbb{E}\left[\xi_1 h_1(x_1)\right] \\ g_2(\mathbf{x}) = \mathbb{E}\left[\xi_1\xi_2 f(x_2) h_2(h_1(x_1), f(x_2))\right] \end{cases}$$

where $\mathcal{X} = \tilde{\mathcal{X}}_{B1} \times \tilde{\mathcal{X}}_{B2}$ and $\tilde{\mathcal{X}}_{B1} = \tilde{\mathcal{X}}_{B2} = \{0, 1, \ldots, 100\}$, $|\mathcal{X}| = 10,201$, $h_1(x_1) = 4x_1/100$, and $h_2(h_1, f)$ and $f(x_2)$ are defined as

$$h_2(h_1, f) = \begin{cases} 1 - (h_1/f)^\alpha \text{ if } h_1 \le f, \\ 0 \qquad\qquad \text{otherwise;} \end{cases} \qquad f(x_2) = \begin{cases} 4 - 3\exp\left\{-\left(\frac{x_2-20}{2}\right)^2\right\} \text{ if } 0 \le x_2 \le 40, \\ 4 - 2\exp\left\{-\left(\frac{x_2-70}{20}\right)^2\right\} \text{ if } 40 < x_2 \le 100; \end{cases}$$

and $\alpha = 0.25 + 3.75(f(x_2) - 1)$. As in the previous test problem, ξ_1 and ξ_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 $\mathcal{N}_1$-local weakly efficient sets, one of which is the global weakly efficient

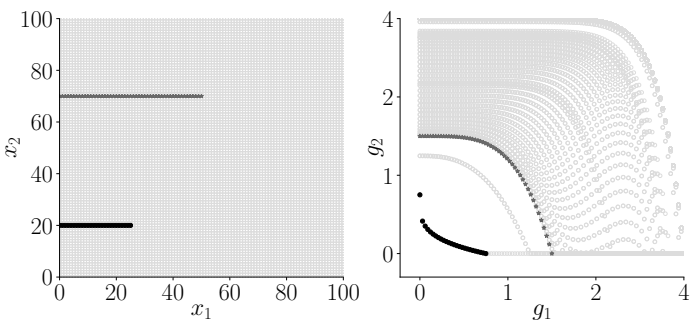


Figure 6 **Problem T_B: The black dots and gray stars represent points in the global efficient set and the $\mathcal{N}_1$-local efficient set, respectively (left) and their images (right).**

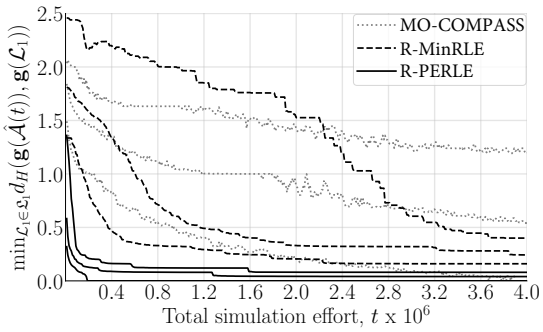


Figure 7 **Problem T_B: Sample quantiles (.25, .5, .75) of the local coverage error across 1,000 independent runs per algorithm.**

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 $\mathcal{N}_1$-local weakly efficient set almost surely, and contain an $\mathcal{N}_1$-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 $\mathbf{g}(\hat{\mathcal{A}}(t))$ to the nearest $\mathcal{N}_1$-local Pareto set as a function of the total simulation work done, $\min_{\mathcal{L}_1 \in \mathfrak{L}_1} d_H(\mathbf{g}(\hat{\mathcal{A}}(t)), \mathbf{g}(\mathcal{L}_1))$. 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 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

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

where $\mathcal{X} = \tilde{\mathcal{X}}_{C1} \times \tilde{\mathcal{X}}_{C2} \times \tilde{\mathcal{X}}_{C3}$, $\tilde{\mathcal{X}}_{Ci} = \{-5, -4.5, -4.0, -3.5, \ldots, 5\}$ for all $i \in \{1, 2, 3\}$, $|\mathcal{X}| = 9,261$, and ξ_1, ξ_2, and ξ_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.

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 $\min_{\mathcal{W}_1 \in \mathfrak{W}_1} d_H(\mathbf{g}(\hat{\mathcal{A}}(t)), \mathbf{g}(\mathcal{W}_1))$. 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.

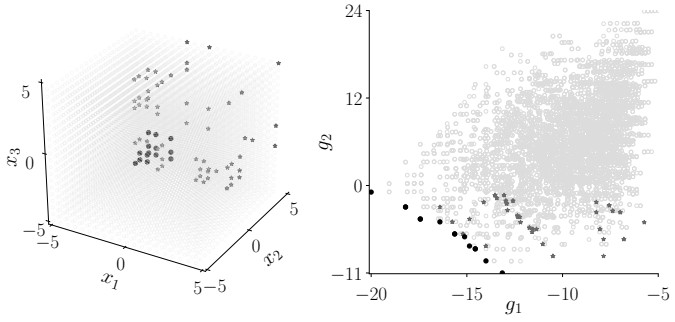


Figure 8 **Problem T_C: Black circles and gray stars represent points in the global weakly efficient set and the $\mathcal{N}_1$-local weakly efficient set members, respectively (left) and their images (right).**

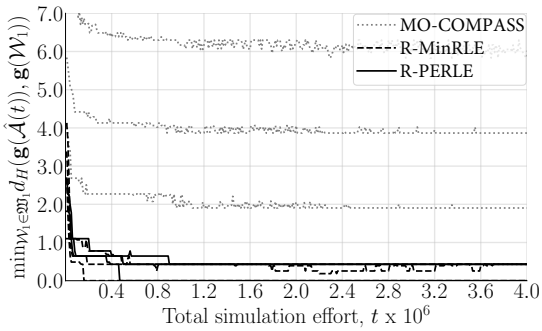


Figure 9 **Problem T_C: Sample quantiles (.25, .5, .75) of the local weakly coverage error across 1,000 independent runs per algorithm.**

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 $\boldsymbol{\beta}$ Values

We explore R-PERLE's performance on our test problems across a variety of $\boldsymbol{\beta} = (\beta_\varepsilon, \beta_\delta)$ values. Recall that for PE, smaller β_ε values result in solving fewer sample-path ε-constraint problems, and larger β_ε values correspond to solving more sample-path ε-constraint problems. For the RLE algorithm, smaller β_δ implies less crawling and a less-complete ALES, and larger β_δ 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 $\boldsymbol{\beta}$ values.

At total simulation budget $t = 0.4 \times 10^6$ on Problem T_A, there seems to be a "sweet spot" for setting β_ε 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

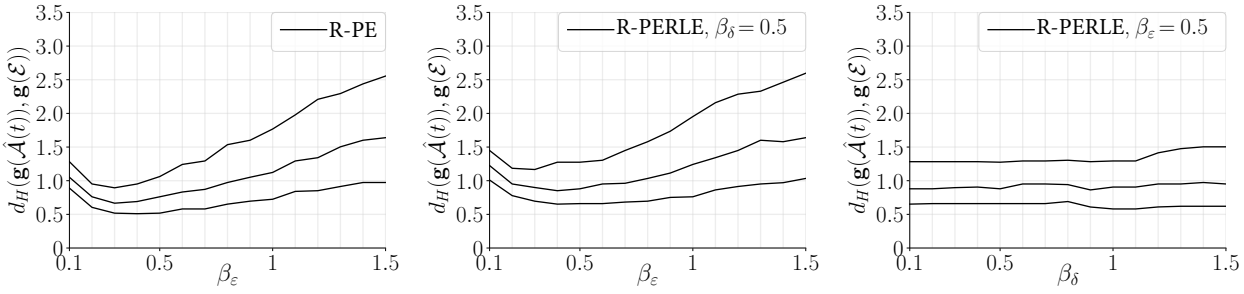


Figure 10 **Problem T_A: Sample quantiles (0.25, 0.50, 0.75) of the coverage error at $t = 0.4 \times 10^6$ across 1,000 independent runs of R-PE (left), R-PERLE, $\beta_\delta = 0.5$ (center), R-PERLE, $\beta_\varepsilon = 0.5$ (right).**

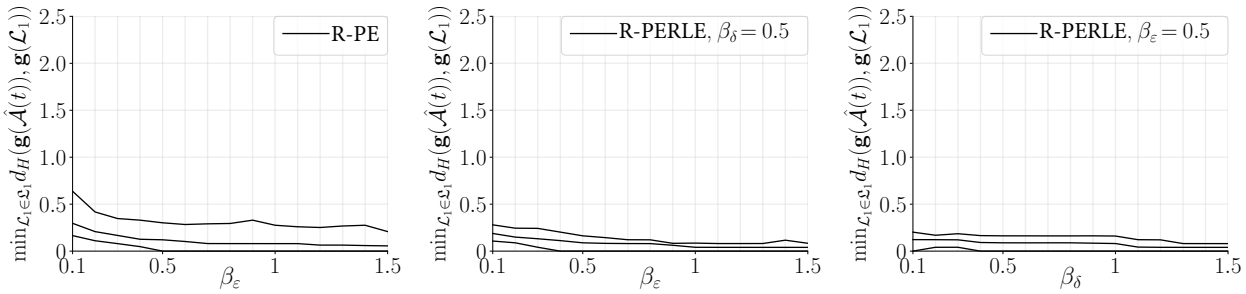


Figure 11 **Problem T_B: Sample quantiles (0.25, 0.50, 0.75) of the local coverage error at $t = 0.4 \times 10^6$ across 1,000 independent runs of R-PE (left), R-PERLE, $\beta_\delta = 0.5$ (center), R-PERLE, $\beta_\varepsilon = 0.5$ (right).**

R-PERLE performance in the right panel of Figure 10 is fairly robust to different values of β_δ. Notice that with $\beta_\varepsilon = 0.5$, for small values of β_δ, R-PE and R-PERLE return similar sets.

On Problem T_B, however, solving more ε-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 β_ε. 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 ε-constraint problems (smaller β_ε) and let RLE do the work of finding the disconnected $\mathcal{N}_1$-local weakly efficient set members, with the sweet spot for β_δ 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 β_ε and β_δ values seem reasonable.

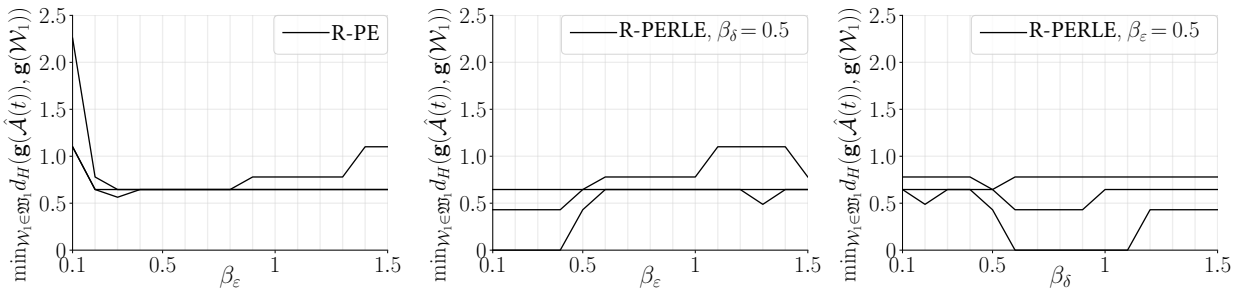


Figure 12 **Problem T_C: Sample quantiles (0.25, 0.50, 0.75) of the local weakly coverage error at $t = 0.4 \times 10^6$ across 1,000 independent runs of R-PE (left), R-PERLE, $\beta_\delta = 0.5$ (center), R-PERLE, $\beta_\varepsilon = 0.5$ (right).**

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

Andersson M, Grimm H, Persson A, Ng A (2007) A web-based simulation optimization system for industrial scheduling. Henderson SG, Biller B, Hsieh MH, Shortle J, Tew JD, Barton RR, eds., *Proceedings of the 2007 Winter Simulation Conference*, 1844–1852 (Piscataway, NJ: IEEE).

Applegate EA, Feldman G, Hunter SR, Pasupathy R (2019) Multi-objective ranking and selection: Optimal sampling laws and tractable approximations via SCORE. *Journal of Simulation* URL `http://dx.doi.org/10.1080/17477778.2019.1633891`.

Audet C, Hare W (2017) *Derivative-Free and Blackbox Optimization*. Springer Series in Operations Research and Financial Engineering (Switzerland: Springer).

Bertsimas D, Farias VF, Trichakis N (2013) Fairness, efficiency, and flexibility in organ allocation for kidney transplantation. *Operations Research* 61(1):73–87, URL `http://dx.doi.org/10.1287/opre.1120.1138`.

Branke J, Zhang W (2015) A new myopic sequential sampling algorithm for multi-objective problems. Yilmaz L, Chan WKV, Moon I, Roeder TMK, Macal C, Rossetti MD, eds., *Proceedings of the 2015 Winter Simulation Conference*, 3589–3598 (Piscataway, NJ: IEEE).

Branke J, Zhang W, Tao Y (2016) Multiobjective ranking and selection based on hypervolume. Roeder TMK, Frazier PI, Szechtman R, Zhou E, Huschka T, Chick SE, eds., *Proceedings of the 2016 Winter Simulation Conference*, 859–870 (Piscataway, NJ: IEEE).

Chen T, Wang C (2016) Multi-objective simulation optimization for medical capacity allocation in emergency department. *Journal of Simulation* 10(1):50–68, URL `http://dx.doi.org/10.1057/jos.2014.39`.

Chew EP, Lee LH, Teng S, Koh CH (2009) Differentiated service inventory optimization using nested partitions and MOCBA. *Computers & Operations Research* 36(5):1703–1710, URL http://dx.doi.org/10.1016/j.cor.2008.04.006.

Conn AR, Scheinberg K, Vicente LN (2009) *Introduction to Derivative-Free Optimization*. MOS-SIAM Series on Optimization (Philadelphia, PA: Society for Industrial and Applied Mathematics and Mathematical Programming Society), URL http://dx.doi.org/10.1137/1.9780898718768.

Cooper K, Hunter SR (2019) PyMOSO: Software for multi-objective simulation optimization with R-PERLE and R-MinRLE. *INFORMS Journal on Computing* URL http://dx.doi.org/10.1287/ijoc.2019.0902.

Cooper K, Hunter SR, Nagaraj K (2017) An epsilon-constraint method for integer-ordered bi-objective simulation optimization. Chan WKV, D'Ambrogio A, Zacharewicz G, Mustafee N, Wainer G, Page E, eds., *Proceedings of the 2017 Winter Simulation Conference*, 2303–2314 (Piscataway, NJ: IEEE), URL http://dx.doi.org/10.1109/WSC.2017.8247961.

Custódio AL, Madeira JFA, Vaz AIF, Vicente LN (2011) Direct multisearch for multiobjective optimization. *SIAM Journal on Optimization* 21(3):1109–1140, URL http://dx.doi.org/10.1137/10079731X.

Dembo A, Zeitouni O (1998) *Large Deviations Techniques and Applications* (New York: Springer), 2nd edition.

Feldman G, Hunter SR (2018) SCORE allocations for bi-objective ranking and selection. *ACM Transactions on Modeling and Computer Simulation* 28(1):7:1–7:28, URL http://dx.doi.org/10.1145/3158666.

Fu MC (2002) Optimization for simulation: theory vs. practice. *INFORMS Journal on Computing* 14:192–215, URL http://dx.doi.org/10.1287/ijoc.14.3.192.113.

Fu MC, ed. (2015) *Handbook of Simulation Optimization*, volume 216 of *International Series in Operations Research & Management Science* (New York: Springer).

Gropp W, Snir M (2013) Programming for exascale computers. *Computing in Science & Engineering* 15(6):27–35, URL http://dx.doi.org/10.1109/MCSE.2013.96.

Henderson SG, Pasupathy R (2019) Simulation optimization library. URL http://www.simopt.org.

Hong LJ, Nelson BL (2006) Discrete optimization via simulation using COMPASS. *Operations Research* 54(1):115–129, URL http://dx.doi.org/10.1287/opre.1050.0237.

Huang H, Zabinsky ZB (2014) Multiple objective probabilistic branch and bound for Pareto optimal approximation. Tolk A, Diallo SY, Ryzhov IO, Yilmaz L, Buckley S, Miller JA, eds., *Proceedings of the 2014 Winter Simulation Conference*, 3916–3927 (Piscataway, NJ: IEEE), URL http://dx.doi.org/10.1109/WSC.2014.7020217.

Hunter SR, Applegate EA, Arora V, Chong B, Cooper K, Rincón-Guevara O, Vivas-Valencia C (2019) An introduction to multi-objective simulation optimization. *ACM Transactions on Modeling and Computer Simulation* 29(1):7:1–7:36, URL http://dx.doi.org/10.1145/3299872.

Hunter SR, McClosky B (2016) Maximizing quantitative traits in the mating design problem via simulation-based Pareto estimation. *IIE Transactions* 48(6):565–578, URL http://dx.doi.org/10.1080/0740817X.2015.1096430.

Kim S, Pasupathy R, Henderson SG (2015) A guide to sample average approximation. Fu M, ed., *Handbook of Simulation Optimization*, volume 216 of *International Series in Operations Research & Management Science*, 207–243 (New York: Springer-Verlag), URL http://dx.doi.org/10.1007/978-1-4939-1384-8_8.

Kim S, Ryu J (2011) The sample average approximation method for multi-objective stochastic optimization. Jain S, Creasey RR, Himmelspach J, White KP, Fu M, eds., *Proceedings of the 2011 Winter Simulation Conference*, 4026–4037 (Piscataway, NJ: IEEE), URL http://dx.doi.org/10.1109/WSC.2011.6148092.

Larson J, Menickelly M, Wild SM (2019) Derivative-free optimization methods. *arXiv* URL https://arxiv.org/abs/1904.11585.

Law AM (2015) *Simulation Modeling and Analysis* (New York: McGraw Hill Education), 5 edition.

Le Digabel S, Wild SM (2015) A taxonomy of constraints in simulation-based optimization. *arXiv* URL https://arxiv.org/abs/1505.07881.

Lee LH, Chew EP, Teng S, Goldsman D (2010) Finding the non-dominated Pareto set for multi-objective simulation models. *IIE Transactions* 42:656–674, URL http://dx.doi.org/10.1080/07408171003705367.

Li H, Lee LH, Chew EP, Lendermann P (2015a) MO-COMPASS: A fast convergent search algorithm for multi-objective discrete optimization via simulation. *IIE Transactions* 47(11):1153–1169, URL http://dx.doi.org/10.1080/0740817X.2015.1005778.

Li H, Zhu Y, Chen Y, Pedrielli G, Pujowidianto NA, Chen Y (2015b) The object-oriented discrete event simulation modeling: a case study on aircraft spare part management. Yilmaz L, Chan WKV, Roeder TMK, Macal C, Rosetti M, eds., *Proceedings of the 2015 Winter Simulation Conference*, 3514–3525 (Piscataway, NJ: IEEE).

Li J, Liu W, Pedrielli G, Lee LH, Chew EP (2018) Optimal computing budget allocation to select the non-dominated systems – a large deviations perspective. *IEEE Transactions on Automatic Control* 63(9):2913–2927, URL http://dx.doi.org/10.1109/TAC.2017.2779603.

Liuzzi G, Lucidi S, Rinaldi F (2018) An algorithmic framework based on primitive directions and non-monotone line searches for black box problems with integer variables. *Optimization Online* URL http://www.optimization-online.org/DB_HTML/2018/02/6471.html.

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

Nsoesie EO, Beckman RJ, Shashaani S, Nagaraj KS, Marathe MV (2013) A simulation optimization approach to epidemic forecasting. *PloS one* 8(6):e67164.

Osorio C, Bierlaire M (2013) A simulation-based optimization framework for urban transportation problems. *Operations Research* 61(6):1333–1345, URL http://dx.doi.org/10.1287/opre.2013.1226.

Pasupathy R (2010) On choosing parameters in retrospective-approximation algorithms for stochastic root finding and simulation optimization. *Operations Research* 58(4):889–901, URL http://dx.doi.org/10.1287/opre.1090.0773.

Pasupathy R, Ghosh S (2013) Simulation optimization: a concise overview and implementation guide. Topaloglu H, ed., *TutORials in Operations Research*, chapter 7, 122–150 (Catonsville, MD: INFORMS), URL http://dx.doi.org/10.1287/educ.2013.0118.

Pasupathy R, Henderson SG (2006) A testbed of simulation-optimization problems. Perrone LF, Wieland FP, Liu J, Lawson BG, Nicol DM, Fujimoto RM, eds., *Proceedings of the 2006 Winter Simulation Conference*, 255–263 (Piscataway, NJ: IEEE), URL http://dx.doi.org/10.1109/WSC.2006.323081.

Pasupathy R, Henderson SG (2011) SimOpt: A library of simulation optimization problems. Jain S, Creasey RR, Himmelspach J, White KP, Fu M, eds., *Proceedings of the 2011 Winter Simulation Conference*, 4075–4085 (Piscataway, NJ: IEEE), URL http://dx.doi.org/10.1109/WSC.2011.6148097.

Powers MJ, Sanchez SM, Lucas TW (2012) The exponential expansion of simulation in research. Laroque C, Himmelspach J, Pasupathy R, Rose O, Uhrmacher AM, eds., *Proceedings of the 2012 Winter Simulation Conference*, 1552–1563 (Piscataway, NJ: IEEE), URL http://dx.doi.org/10.1109/WSC.2012.6465125.

Ralphs TK, Saltzman MJ, Wiecek MM (2006) An improved algorithm for solving biobjective integer programs. *Annals of Operations Research* 147(1):43–70, ISSN 0254-5330, URL http://dx.doi.org/10.1007/s10479-006-0058-z.

Royset JO, Szechtman R (2013) Optimal budget allocation for sample average approximation. *Operations Research* 61(3):762–776, URL http://dx.doi.org/10.1287/opre.2013.1163.

Ryu J, Kim S (2014) A derivative-free trust-region method for biobjective optimization. *SIAM J. Optim.* 24(1):334–362, URL http://dx.doi.org/10.1137/120864738.

Shapiro A, Dentcheva D, Ruszczyński A (2009) *Lectures on Stochastic Programming: Modeling and Theory*. MPS-SIAM Series on Optimization (Philadelphia, PA: Society for Industrial and Applied Mathematics).

Singh A, Minsker BS (2008) Uncertainty-based multiobjective optimization of groundwater remediation design. *Water Resources Research* 44, URL http://dx.doi.org/10.1029/2005WR004436.

Wang H, Pasupathy R, Schmeiser BW (2013) Integer-ordered simulation optimization using R-SPLINE: Retrospective Search using Piecewise-Linear Interpolation and Neighborhood Enumeration. *ACM Transactions on Modeling and Computer Simulation* 23(3), URL http://dx.doi.org/10.1145/2499913.2499916.

Xu J, Nelson BL, Hong LJ (2010) Industrial Strength COMPASS: A comprehensive algorithm and software for optimization via simulation. *ACM Transactions on Modeling and Computer Simulation* 20:1–29.

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 $\boldsymbol{\alpha}$, $\mathbf{x}$, and the random realization, such that for all $\nu \geq \tilde{\nu}_k$, $\hat{\mathcal{S}}_{k,\nu}(\mathbf{x}) \subseteq \mathcal{S}_k(\mathbf{x}, \alpha_k)$ w.p.1. Let $\tilde{\nu} := \max_k\{\tilde{\nu}_k\}$, so that for all $\nu \geq \tilde{\nu}$, $\hat{\mathcal{S}}_\nu(\mathbf{x}) \subseteq \mathcal{S}(\mathbf{x}, \boldsymbol{\alpha})$ w.p.1. $\square$

B. Proof of Theorem 1

Proof of Theorem 1 Part (a). For every ν, R-PERLE and R-MinRLE return a set $\hat{\mathcal{A}}_\nu$ in finite time. Thus both algorithms produce an infinite sequence of solutions $\{\hat{\mathcal{A}}_\nu\}$. Further, notice that R-PERLE and R-MinRLE never return a set $\hat{\mathcal{A}}_\nu$ containing a point whose estimated objective vector is dominated by $\bar{\mathbf{G}}_{m_\nu}(\mathbf{x}_0)$ (see Algorithm 5, RLE Steps 1 and 14). Now consider the union of the level sets corresponding to the starting point $\mathbf{x}_0$, $\mathcal{S}(\mathbf{x}_0, \boldsymbol{\alpha})$. By Lemma 3, there exists $\tilde{\nu}$ such that for all $\nu \geq \tilde{\nu}$, $\hat{\mathcal{A}}_\nu \subseteq \hat{\mathcal{S}}_\nu(\mathbf{x}_0) \subseteq \mathcal{S}(\mathbf{x}_0, \boldsymbol{\alpha})$ w.p.1. Since $\mathcal{S}(\mathbf{x}_0, \boldsymbol{\alpha})$ is finite, then any sequence of estimated efficient points $\{\mathbf{X}^*_\nu : \mathbf{X}^*_\nu \in \hat{\mathcal{A}}_\nu \text{ for all } \nu = 1, 2, \ldots\}$ 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 $\{\hat{\mathcal{A}}_\nu\}$ converges into an $\mathcal{N}_a$-local weakly efficient set w.p.1.

Since $\mathcal{S}(\mathbf{x}_0, \boldsymbol{\alpha})$ is finite and $a \in (0, \infty)$, then $\mathcal{N}_a(\mathcal{S}(\mathbf{x}_0, \boldsymbol{\alpha})) \cap \mathcal{X}$ is also finite. Thus for all $k \in \{1, \ldots, d\}$, $\bar{G}_{k,m_\nu}(\cdot)$ uniformly converges to $g_k(\cdot)$ w.p.1 as $\nu \to \infty$ on the set $\mathcal{N}_a(\mathcal{S}(\mathbf{x}_0, \boldsymbol{\alpha})) \cap \mathcal{X}$. Let the set $\mathcal{D}_k := \{(\mathbf{x}, \mathbf{x}') : \mathbf{x}, \mathbf{x}' \in \mathcal{N}_a(\mathcal{S}(\mathbf{x}_0, \boldsymbol{\alpha})) \cap \mathcal{X}, g_k(\mathbf{x}') \neq g_k(\mathbf{x})\}$ 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 $\kappa_1 = \min_{k\in\{1,\ldots,d\}} \inf_{\mathcal{D}_k} |g_k(\mathbf{x}) - g_k(\mathbf{x}')| > 0$ be the smallest difference in objective values across these pairs; Assumption 1 implies $\kappa_1 > 0$. Then w.p.1, there exists ν' (dependent on neighborhood size a, initial point $\mathbf{x}_0$, the constants κ_1 and $\boldsymbol{\alpha}$, and the random realization) such that for all $\nu \geq \nu'$, $\max_{k\in\{1,\ldots,d\}} |\bar{G}_{k,m_\nu}(\mathbf{x}) - g_k(\mathbf{x})| < \kappa_1/4$ for all $\mathbf{x} \in \mathcal{N}_a(\mathcal{S}(\mathbf{x}_0, \boldsymbol{\alpha})) \cap \mathcal{X}$. Since $\beta_\delta \in (0, \infty]$, the ALES completeness function $\hat{\delta}_k(\cdot) = \hat{f}_k(\cdot, \beta_\delta) = \hat{\sigma}_{k,m_\nu}(\cdot)/m_\nu^{\beta_\delta}$ uniformly converges to zero w.p.1 on the finite set $\mathcal{N}_a(\mathcal{S}(\mathbf{x}_0, \boldsymbol{\alpha})) \cap \mathcal{X}$ as $\nu \to \infty$ for all $k \in \{1, \ldots, d\}$. Then w.p.1, there exists ν'' (dependent on the same quantities as ν' and dependent on β_δ) such that for all $\nu \geq \nu''$, $\max_{k\in\{1,\ldots,d\}} \hat{\delta}_k(\mathbf{x}) < \kappa_1/4$ for all $\mathbf{x} \in \mathcal{N}_a(\mathcal{S}(\mathbf{x}_0, \boldsymbol{\alpha})) \cap \mathcal{X}$.

Henceforth, let $\nu \geq \max\{\nu', \nu''\}$. Combining the above results, for all $\mathbf{x} \in \mathcal{N}_a(\mathcal{S}(\mathbf{x}_0, \boldsymbol{\alpha})) \cap \mathcal{X}$, $\max_{k\in\{1,\ldots,d\}} \big||\bar{G}_{k,m_\nu}(\mathbf{x}) - \hat{\delta}_k(\mathbf{x})| - g_k(\mathbf{x})\big| < \kappa_1/2$ w.p.1. Thus for all $k \in \{1, \ldots, d\}$ and for all $\mathbf{x}, \mathbf{x}' \in \mathcal{N}_a(\mathcal{S}(\mathbf{x}_0, \boldsymbol{\alpha})) \cap \mathcal{X}$, the following hold:

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

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

Further, for all $\mathbf{x}, \mathbf{x}' \in \mathcal{N}_a(\mathcal{S}(\mathbf{x}_0, \boldsymbol{\alpha})) \cap \mathcal{X}$, if $\bar{\mathbf{G}}_{m_\nu}(\mathbf{x}) \nleqq \bar{\mathbf{G}}_{m_\nu}(\mathbf{x}')$, then $\exists k \in \{1, \ldots, d\}$ such that $\bar{G}_{k,m_\nu}(\mathbf{x}') < \bar{G}_{k,m_\nu}(\mathbf{x})$, implying that $g_k(\mathbf{x}') \leq g_k(\mathbf{x})$ w.p.1. This result, along with results R1 and R2 above, imply that for all $\mathbf{x}, \mathbf{x}' \in \mathcal{N}_a(\mathcal{S}(\mathbf{x}_0, \boldsymbol{\alpha})) \cap \mathcal{X}$,

R3. if $\bar{\mathbf{G}}_{m_\nu}(\mathbf{x}) \nleqq \bar{\mathbf{G}}_{m_\nu}(\mathbf{x}')$, then $\mathbf{g}(\mathbf{x}) \not< \mathbf{g}(\mathbf{x}')$ w.p.1;

R4. if $\bar{\mathbf{G}}_{m_\nu}(\mathbf{x}) + \hat{\boldsymbol{\delta}}(\mathbf{x}) \leqq \bar{\mathbf{G}}_{m_\nu}(\mathbf{x}') - \hat{\boldsymbol{\delta}}(\mathbf{x}')$, then $\mathbf{g}(\mathbf{x}) \leqq \mathbf{g}(\mathbf{x}')$ w.p.1.

Now let $\nu \geq \max\{\tilde{\nu}, \nu', \nu''\}$. Then by Lemma 3, the set of decision points $\hat{\mathcal{A}}_\nu$ returned by each algorithm lie in $\mathcal{S}(\mathbf{x}_0, \boldsymbol{\alpha})$ w.p.1. We now consider all parts of the definition of an ALES (Definition 7). First, Algorithm 5 ensures no points in $\bar{\mathbf{G}}_n(\hat{\mathcal{A}}_\nu)$ dominate other points in $\bar{\mathbf{G}}_n(\hat{\mathcal{A}}_\nu)$ (RLE Steps 1 and 14). Thus result R3 above implies that no points in $\mathbf{g}(\hat{\mathcal{A}}_\nu)$ strictly dominate other points in $\mathbf{g}(\hat{\mathcal{A}}_\nu)$. Second, Algorithm 5 ensures each point in $\mathbf{X}_\nu^{\mathrm{w}} \in \hat{\mathcal{A}}_\nu$ 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 $\hat{\mathcal{A}}_\nu$ are $\mathcal{N}_a$-LWEP's. Third, Algorithm 5 ensures that the NCN of $\hat{\mathcal{A}}_\nu$ is empty (RLE Steps 2 and 15). Then applying results R3 and R4 above, for all $\mathbf{X} \in \mathcal{N}_a'(\hat{\mathcal{A}}_\nu) \cap \mathcal{X}$, (i) $\exists \mathbf{X}_\nu^{\mathrm{w}} \in \hat{\mathcal{A}}$ such that $\mathbf{g}(\mathbf{X}_\nu^{\mathrm{w}}) \leqq \mathbf{g}(\mathbf{X})$ w.p.1, or (ii) $\exists \mathbf{X}_\nu^{\mathrm{w}} \in \hat{\mathcal{A}}$ such that $(\mathbf{g}(\mathbf{X}) \leqq \mathbf{g}(\mathbf{X}_\nu^{\mathrm{w}})$ and $\bar{\mathbf{G}}_{m_\nu}(\mathbf{X}_\nu^{\mathrm{w}}) - \hat{\boldsymbol{\delta}}(\mathbf{X}_\nu^{\mathrm{w}}) \leqq \bar{\mathbf{G}}_{m_\nu}(\mathbf{X}) + \hat{\boldsymbol{\delta}}(\mathbf{X}))$ w.p.1, which happens with probability zero unless $\mathbf{g}(\mathbf{X}) = \mathbf{g}(\mathbf{X}_\nu^{\mathrm{w}})$, or (iii) employing the complements of the previous two conditions, $\forall \mathbf{X}_\nu^{\mathrm{w}} \in \hat{\mathcal{A}}$, $\mathbf{g}(\mathbf{X}_\nu^{\mathrm{w}}) \nleqq \mathbf{g}(\mathbf{X}) \nleqq \mathbf{g}(\mathbf{X}_\nu^{\mathrm{w}})$, and $\exists \tilde{\mathbf{X}}_\nu^{\mathrm{w}} \in \hat{\mathcal{A}}$ such that $\bar{\mathbf{G}}_{m_\nu}(\tilde{\mathbf{X}}_\nu^{\mathrm{w}}) - \hat{\boldsymbol{\delta}}(\tilde{\mathbf{X}}_\nu^{\mathrm{w}}) \leqq \bar{\mathbf{G}}_{m_\nu}(\mathbf{X}) + \hat{\boldsymbol{\delta}}(\mathbf{X})$ or

$\bar{\mathbf{G}}_{m_\nu}(\mathbf{X}) - \hat{\boldsymbol{\delta}}(\mathbf{X}) \leqq \bar{\mathbf{G}}_{m_\nu}(\tilde{\mathbf{X}}^{\mathrm{w}}_\nu) + \hat{\boldsymbol{\delta}}(\tilde{\mathbf{X}}^{\mathrm{w}}_\nu)$ w.p.1, which happens with probability zero. To see this, let $\mathbf{X} \in \mathcal{N}'_a(\hat{\mathcal{A}}_\nu) \cap \mathcal{X}$ and notice that the condition $\forall \mathbf{X}^{\mathrm{w}}_\nu \in \hat{\mathcal{A}}$, $\mathbf{g}(\mathbf{X}^{\mathrm{w}}_\nu) \nleqq \mathbf{g}(\mathbf{X}) \nleqq \mathbf{g}(\mathbf{X}^{\mathrm{w}}_\nu)$ w.p.1 implies that $\forall \mathbf{X}^{\mathrm{w}}_\nu \in \hat{\mathcal{A}}$, $\exists k_1, k_2 \in \{1, \ldots, d\}$ such that $g_{k_1}(\mathbf{X}^{\mathrm{w}}_\nu) > g_{k_1}(\mathbf{X})$ and $g_{k_2}(\mathbf{X}^{\mathrm{w}}_\nu) < g_{k_2}(\mathbf{X})$ w.p.1, and hence by result R1 above, $\bar{G}_{k_1,m_\nu}(\mathbf{X}^{\mathrm{w}}_\nu) - \hat{\delta}_{k_1}(\mathbf{X}^{\mathrm{w}}_\nu) > \bar{G}_{k_1,m_\nu}(\mathbf{X}) + \hat{\delta}_{k_1}(\mathbf{X})$ and $\bar{G}_{k_2,m_\nu}(\mathbf{X}) - \hat{\delta}_{k_2}(\mathbf{X}) > \bar{G}_{k_2,m_\nu}(\mathbf{X}^{\mathrm{w}}_\nu) + \hat{\delta}_{k_2}(\mathbf{X}^{\mathrm{w}}_\nu)$ w.p.1.

Therefore when $\nu \geq \max\{\tilde{\nu}, \nu', \nu''\}$, R-PERLE and R-MinRLE certify that w.p.1, all points in $\hat{\mathcal{A}}_\nu$ are $\mathcal{N}_a$-LWEP's, no points in $\mathbf{g}(\hat{\mathcal{A}}_\nu)$ strictly dominate other points in $\mathbf{g}(\hat{\mathcal{A}}_\nu)$, and for all $\mathbf{X} \in \mathcal{N}'_a(\hat{\mathcal{A}}_\nu) \cap \mathcal{X}$, $\exists \mathbf{X}^{\mathrm{w}}_\nu \in \hat{\mathcal{A}}$ such that $\mathbf{g}(\mathbf{X}^{\mathrm{w}}_\nu) \leqq \mathbf{g}(\mathbf{X})$. Thus by Definition 5, $\hat{\mathcal{A}}_\nu$ is an $\mathcal{N}_a$-local weakly efficient set w.p.1. Further, each $\hat{\mathcal{A}}_\nu$ is such that there does not exist a pair of points $\mathbf{X}^*_{\nu-1} \in \hat{\mathcal{A}}_{\nu-1}$ and $\mathbf{X}^*_\nu \in \hat{\mathcal{A}}_\nu$ such that $\mathbf{g}(\mathbf{X}^*_{\nu-1}) < \mathbf{g}(\mathbf{X}^*_\nu)$ w.p.1. (The second part follows because we carry forward the points from $\hat{\mathcal{A}}_{\nu-1}$ into the νth iteration, and we ensure that no points in $\bar{\mathbf{G}}_n(\hat{\mathcal{A}}_\nu)$ dominate other points in $\bar{\mathbf{G}}_n(\hat{\mathcal{A}}_\nu)$ in Algorithm 5, RLE Steps 1 and 14.) Therefore if $\nu \geq \max\{\tilde{\nu}, \nu', \nu''\}$, there exists $\mathcal{W}_a \in \mathfrak{W}_a$ such that R-PERLE and R-MinRLE returns $\hat{\mathcal{A}}_\nu \subseteq \mathcal{W}_a \subseteq \mathcal{S}(\mathbf{x}_0, \boldsymbol{\alpha})$ w.p.1.

Proof of Theorem 1 Part (b). Let $\nu \geq \max\{\tilde{\nu}, \nu', \nu''\}$. By the proof of Theorem 1 Part (a), $\hat{\mathcal{A}}_\nu$ is an $\mathcal{N}_a$-local weakly efficient set w.p.1. Under Assumption 4, by Lemma 4, there exists $\mathcal{L}_a \in \mathfrak{L}_a$ such that $\hat{\mathcal{A}}_\nu \supseteq \mathcal{L}_a$ w.p.1, and the result holds.

Proof of Theorem 1 Part (c). Assumption 5 implies that all $\mathcal{N}_a$-local weakly efficient sets are $\mathcal{N}_a$-local efficient sets. Thus $\mathfrak{W}_a = \mathfrak{L}_a$, and the result follows from Theorem 1 Parts (a) and (b).

Proof of Theorem 1 Part (d). Assumptions 5 and 6 imply that the $\mathcal{N}_a$-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 ν, R-PE returns a set $\hat{\mathcal{A}}_\nu$ in finite time, thus producing an infinite sequence of solutions $\{\hat{\mathcal{A}}_\nu\}$. Further, R-PE never returns a set $\hat{\mathcal{A}}_\nu$ containing a point whose estimated objective vector is dominated by $\bar{\mathbf{G}}_{m_\nu}(\mathbf{x}_0)$ (see Algorithm 2, PE Step 26). Recall that for all $\nu \geq \tilde{\nu}$, $\hat{\mathcal{A}}_\nu \subseteq \hat{\mathcal{S}}_\nu(\mathbf{x}_0) \subseteq \mathcal{S}(\mathbf{x}_0, \boldsymbol{\alpha})$ w.p.1, and since $\mathcal{S}(\mathbf{x}_0, \boldsymbol{\alpha})$ is finite, then any sequence of estimated efficient points $\{\mathbf{X}^*_\nu : \mathbf{X}^*_\nu \in \hat{\mathcal{A}}_\nu \text{ for all } \nu = 1, 2, \ldots\}$ is bounded w.p.1.

As in the proof of Theorem 1 Part (a), for all $k \in \{1, 2\}$, $\bar{G}_{k,m_\nu}(\cdot)$ uniformly converges to $g_k(\cdot)$ w.p.1 as $\nu \to \infty$ on the finite set $\mathcal{N}_a(\mathcal{S}(\mathbf{x}_0, \boldsymbol{\alpha})) \cap \mathcal{X}$. Since there are only two objectives and $\beta_\varepsilon \in (0, \infty)$, the maximum ε-placement distance $\max_{k^{\mathrm{con}} \in \{1,2\}} \hat{f}_{k^{\mathrm{con}}}(\cdot, \beta_\varepsilon) =$

$\max_{k^{\text{con}}\in\{1,2\}} \hat{\sigma}_{k^{\text{con}},m_\nu}(\cdot)/m_\nu^{\beta_\varepsilon}$ also uniformly converges to zero w.p.1 as $\nu\to\infty$ on $\mathcal{N}_a(\mathcal{S}(\mathbf{x}_0,\boldsymbol{\alpha}))\cap\mathcal{X}$. Let $\kappa>0$ be as in Assumption 5. Then w.p.1, there exists ν'_{PE} (dependent on $a,\mathbf{x}_0,\kappa,\boldsymbol{\alpha}$, and the random realization) such that for all $\nu\geq\nu'_{\text{PE}}$ and all $\mathbf{x}\in\mathcal{N}_a(\mathcal{S}(\mathbf{x}_0,\boldsymbol{\alpha}))\cap\mathcal{X}$, we have $\max_{k\in\{1,2\}}|\bar{G}_{k,m_\nu}(\mathbf{x})-g_k(\mathbf{x})|<\kappa/4$ w.p.1. Also w.p.1, there exists ν''_{PE} (dependent on the same quantities as ν'_{PE} and dependent on β_ε) such that for all $\nu\geq\nu''_{\text{PE}}$, $\max_{k^{\text{con}}\in\{1,2\}}\hat{f}_{k^{\text{con}}}(\mathbf{x},\beta_\varepsilon)<\kappa/4$ for all $\mathbf{x}\in\mathcal{N}_a(\mathcal{S}(\mathbf{x}_0,\boldsymbol{\alpha}))\cap\mathcal{X}$. Combining the above results, if $\nu\geq\max\{\nu'_{\text{PE}},\nu''_{\text{PE}}\}$, then for all $\mathbf{x}\in\mathcal{N}_a(\mathcal{S}(\mathbf{x}_0,\boldsymbol{\alpha}))\cap\mathcal{X}$, $\max_{k^{\text{con}}\in\{1,2\}}\big||\bar{G}_{k^{\text{con}},m_\nu}(\mathbf{x})-\hat{f}_{k^{\text{con}}}(\mathbf{x},\beta_\varepsilon)|-g_{k^{\text{con}}}(\mathbf{x})\big|<\kappa/2$ w.p.1. Henceforth, let $\nu\geq\max\{\nu'_{\text{PE}},\nu''_{\text{PE}}\}$. Then for all $\mathbf{x},\mathbf{x}'\in\mathcal{N}(\mathcal{S}(\mathbf{x}_0,\boldsymbol{\alpha}))\cap\mathcal{X}$, the following hold w.p.1:

R5. $\forall k\in\{1,2\}$, $g_k(\mathbf{x})<g_k(\mathbf{x}')$ if and only if $\bar{G}_{k,m_\nu}(\mathbf{x})+\hat{f}_k(\mathbf{x},\beta_\varepsilon)<\bar{G}_{k,m_\nu}(\mathbf{x}')-\hat{f}_k(\mathbf{x}',\beta_\varepsilon)$;

R6. if $\bar{\mathbf{G}}_{m_\nu}(\mathbf{x})\not\leq\bar{\mathbf{G}}_{m_\nu}(\mathbf{x}')$, then $\mathbf{g}(\mathbf{x})\not\leqq\mathbf{g}(\mathbf{x}')$, that is, $\exists k\in\{1,2\}$ such that $g_k(\mathbf{x}')<g_k(\mathbf{x})$.

Under Assumption 1, for any $\mathbf{x}_0\in\mathcal{X}$, $\mathcal{E}\subseteq\mathcal{S}(\mathbf{x}_0,\boldsymbol{\alpha})$. Therefore results R5, R6 and Assumptions 5–6 imply that all points in $\mathcal{E}$ are both sample-path $\mathcal{N}_a$-LWEP's and sample-path global efficient points w.p.1. Further, all points in $\mathcal{N}'_a(\mathcal{E})$ are *not* sample-path $\mathcal{N}_a$-LWEP's w.p.1. Let $c_\varepsilon:=|\mathcal{E}|\geq 1$, and for any objective $k\in\{1,2\}$, sort the elements of $\mathcal{E}$ on objective k so that $g_k(\mathbf{x}^*_{k(1)})<\ldots<g_k(\mathbf{x}^*_{k(c_\varepsilon)})$, where $\mathbf{x}^*_{k(i)}$ denotes the ith ordered element of $\mathcal{E}$ on objective k, $i=1,\ldots,c_\varepsilon$. If $c_\varepsilon\geq 2$, then result R5 implies that w.p.1 for all $i=1,\ldots,c_\varepsilon-1$,

$$\bar{G}_{k^{\text{con}},m_\nu}(\mathbf{x}^*_{k(i)})+\hat{f}_{k^{\text{con}}}(\mathbf{x}^*_{k(i)},\beta_\varepsilon)<\bar{G}_{k,m_\nu}(\mathbf{x}^*_{(i+1)})-\hat{f}_{k^{\text{con}}}(\mathbf{x}^*_{(i+1)}). \tag{1}$$

By Lemma 5, w.p.1 there exists ν''' (dependent on the same quantities as ν'_{PE}) such that for all $\nu\geq\nu'''$, the updated sample-path $\mathcal{N}_a$-local minimizers returned as part of PE Step 1, which we call $\overline{\mathcal{M}}_\nu$, are such that $\overline{\mathcal{M}}_\nu\subseteq\mathcal{M}^*_a$. Under Assumptions 5–6, the set $\mathcal{M}^*_a=\{\mathbf{x}^{\min}_1,\mathbf{x}^{\min}_2\}$ contains the unique global minimizers for each objective $k\in\{1,2\}$.

Henceforth, let $\nu>\max\{\tilde{\nu},\nu'_{\text{PE}},\nu''_{\text{PE}},\nu'''\}$, and let $\{k^*_\nu,\nu=1,2,\ldots\}$ be any sequence of objectives minimized, where $k^{\text{con}}_\nu\neq k^*_\nu$ for each ν. Then by Lemma 3, the set of decision points $\hat{\mathcal{A}}_\nu$ returned by R-PE lie in $\mathcal{S}(\mathbf{x}_0,\boldsymbol{\alpha})$ w.p.1, as does the set of points used to set the ε values in Algorithm 2, PE Step 2, which is a set of sample-path $\mathcal{N}_a$-LWEP's we call $\hat{\mathcal{A}}^{\text{w}}_\nu$. Since all points in $\hat{\mathcal{A}}^{\text{w}}_\nu$ are sample-path $\mathcal{N}_a$-LWEP's, then results R5, R6 and Assumptions 5–6 ensure that $\hat{\mathcal{A}}^{\text{w}}_\nu\subseteq\mathcal{E}$ w.p.1; further, $\overline{\mathcal{M}}_\nu\subseteq\hat{\mathcal{A}}^{\text{w}}_\nu$, where $\overline{\mathcal{M}}_\nu=\{\mathbf{x}^{\min}_1,\mathbf{x}^{\min}_2\}$ w.p.1. If $c_\varepsilon\in\{1,2\}$, the proof is complete, since $\hat{\mathcal{A}}^{\text{w}}_\nu$ is returned as $\hat{\mathcal{A}}_\nu$ in Algorithm 2, PE Step 26, and no other points have entered the set w.p.1. Now suppose $c_\varepsilon\geq 3$. All points in $\hat{\mathcal{A}}^{\text{w}}_\nu\cup\mathcal{E}$ can be ordered on k^{con} as in line (1). Points in $\mathcal{E}\setminus\hat{\mathcal{A}}^{\text{w}}_\nu$ are retrieved by Algorithm 2, PE Steps 15–25, and

carried forward to $\hat{\mathcal{A}}_{\nu+1}$; no other points enter the set w.p.1. Then it follows that for all $\nu^* > \nu+1$, $\hat{\mathcal{A}}_{\nu^*} = \mathcal{E}$ w.p.1, and the result holds. $\square$

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 $\bar{\mathcal{B}}^*_{k,\nu}(\mathcal{D})$ denote the set of sample-path global minimizers of objective g_k, $k \in \{1,\dots,d\}$ 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 $k \in \{1,\dots,d\}$ and all $\mathcal{D} \subseteq \mathcal{X}$, there exists $\eta > 0$ such that for large enough ν,

$$\mathbb{P}\{\bar{\mathcal{B}}^*_{k,\nu}(\mathcal{D}) \not\subseteq \mathcal{B}^*_k(\mathcal{D})\} \leq |\mathcal{D}| e^{-m_\nu \eta}. \tag{2}$$

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

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

$$\begin{aligned}
\mathbb{P}\{\bar{\mathcal{X}}^{\mathrm{w}}_\nu \not\subseteq \mathcal{X}^{\mathrm{w}}\} &\leq \sum_{\mathbf{x} \in \mathcal{X} \setminus \mathcal{X}^{\mathrm{w}}} \mathbb{P}\{\mathbf{x} \in \bar{\mathcal{X}}^{\mathrm{w}}_\nu\} \\
&\leq \sum_{\mathbf{x} \in \mathcal{X} \setminus \mathcal{X}^{\mathrm{w}}} \sum_{j \in \{-q,\dots,-1,1,\dots,q\}} \sum_{k \in \{1,\dots,d\}} \mathbb{P}\{\bar{\mathcal{B}}^*_{k,\nu}(\mathcal{N}_{1,j}(\mathbf{x})) \not\subseteq \mathcal{B}^*_k(\mathcal{N}_{1,j}(\mathbf{x}))\} \\
&\leq \sum_{\mathbf{x} \in \mathcal{X} \setminus \mathcal{X}^{\mathrm{w}}} \sum_{j \in \{-q,\dots,-1,1,\dots,q\}} \sum_{k \in \{1,\dots,d\}} 2e^{-m_\nu \eta} \quad \leq \quad |\mathcal{X}| 4qd e^{-m_\nu \eta},
\end{aligned}$$

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 $\hat{\mathcal{A}}_\nu$ 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), $\mathbb{P}\{\hat{\mathcal{A}}_\nu \not\subseteq \mathcal{E}\} = O(e^{-\gamma m_\nu})$ for some $\gamma > 0$. We now consider

$\mathbb{P}\{\mathcal{E} \not\subseteq \hat{\mathcal{A}}_\nu\}$, where $\mathbb{P}\{\mathcal{E} \not\subseteq \hat{\mathcal{A}}_\nu\} \leq \sum_{\mathbf{x}\in\mathcal{E}} \mathbb{P}\{\mathbf{x} \notin \hat{\mathcal{A}}_\nu\}$. Notice that if $\mathbf{x} \in \mathcal{E}$ is not in $\hat{\mathcal{A}}_\nu$, 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 $\mathbb{P}\{\mathcal{E} \not\subseteq \hat{\mathcal{A}}_\nu\} = O(e^{-\gamma m_\nu})$ for some $\gamma > 0$, which implies the result. □

E. Finding $\mathcal{N}_1$-local weakly efficient sets in Problem T_C

To calculate the collection of all possible $\mathcal{N}_1$-local weakly efficient sets in Problem T_C, we first locate all $\mathcal{N}_1$-LWEP's; we find 512. Then, starting from each $\mathcal{N}_1$-LWEP, we run a program similar to RLE with no relaxation to find the smallest complete $\mathcal{N}_1$-local weakly efficient set that contains the $\mathcal{N}_1$-LWEP. We refer to these $\mathcal{N}_1$-local weakly efficient sets as *level-1 $\mathcal{N}_1$-local weakly efficient sets*; after removing duplicate sets, we find 39. Then, we take all possible unions of two level-1 $\mathcal{N}_1$-local weakly efficient sets, remove any dominated points, and check if this set is a new, unique $\mathcal{N}_1$-local weakly efficient set. We refer to all new, unique $\mathcal{N}_1$-local weakly efficient sets that are found by taking the union of two level-1 $\mathcal{N}_1$-local weakly efficient sets as *level-2 $\mathcal{N}_1$-local weakly efficient sets*. We repeat this process for level-3 and so on, up to level-8. We found one level-7 $\mathcal{N}_1$-local weakly efficient set and no level-8 $\mathcal{N}_1$-local weakly efficient sets. The total number of unique $\mathcal{N}_1$-local weakly efficient sets found in this manner, up to level-8, was 516. All together, these 516 $\mathcal{N}_1$-local weakly efficient sets contain just 73 points; we call these points $\mathcal{N}_1$-local weakly efficient set members in Figure 8. Recall that there are 512 $\mathcal{N}_1$-LWEP's, so not all $\mathcal{N}_1$-LWEP's are members of an $\mathcal{N}_1$-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, $g_1(\mathbf{x})$, and the passengers' expected total waiting time, $g_2(\mathbf{x})$, are minimized. The decision variable $\mathbf{x} = (x_1, x_2, \ldots, x_q)$ is an integer bus schedule, where we assume there is a no-cost bus at time 0 and a pre-scheduled bus at time τ. Scheduling one of the q total buses at time 0 or τ, or at the same time as any other bus, corresponds to not using that bus. The feasible set is $\mathcal{X} = \{0, 1, \ldots, 100\}^q$. 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 ℓ on the ith day, $\ell = 1, 2, \ldots, q+1$. On the ith 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 γ 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 jth passenger for $j = 1, 2, \ldots, N_i(\tau)$ passengers on the ith 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

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

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

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

The points that minimize the g_1 objective correspond to scheduling none of the q buses, $\mathbf{x}_1^{\min} = (0, 0, \ldots, 0)$ or $\mathbf{x}_1^{\min} = (100, 100, \ldots, 100)$. The global solution on the g_2 objective is known to be $\mathbf{x}_2^{\min} = (10, 20, 30, 40, 50, 60, 70, 80, 90)$. Thus under our approximation of the true expected cost, the ideal point is $(g_1(\mathbf{x}_1^{\min}), g_2(\mathbf{x}_2^{\min})) = (131.6,\ 5{,}000)$, and the nadir point is $(g_1(\mathbf{x}_2^{\min}), g_2(\mathbf{x}_1^{\min})) = (1{,}100,\ 50{,}000)$.

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

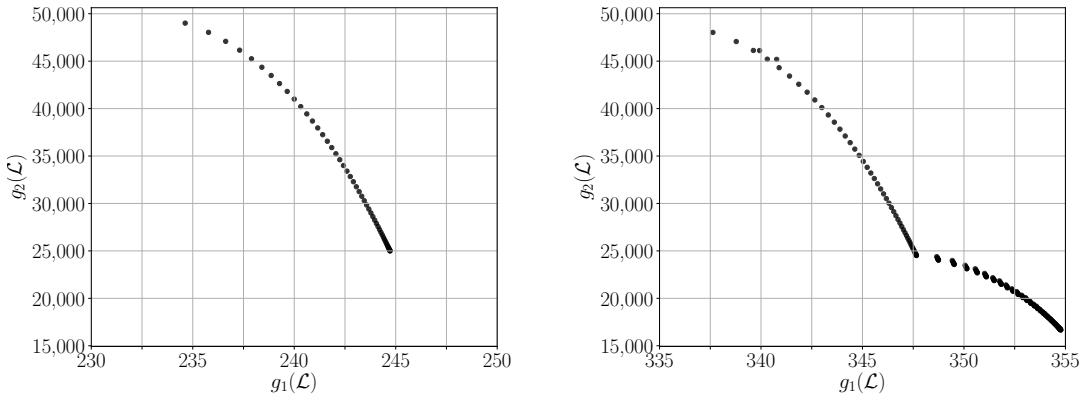


Figure 13 **The figure shows enumerated non-dominated points for the constrained 9-bus scheduling problem in which exactly one bus is scheduled at a non-null time (left) and exactly two buses are scheduled at non-null times (right). We emphasize that these sets are not necessarily $\mathcal{N}_1$-local weakly Pareto sets for the unconstrained 9-bus scheduling problem.**

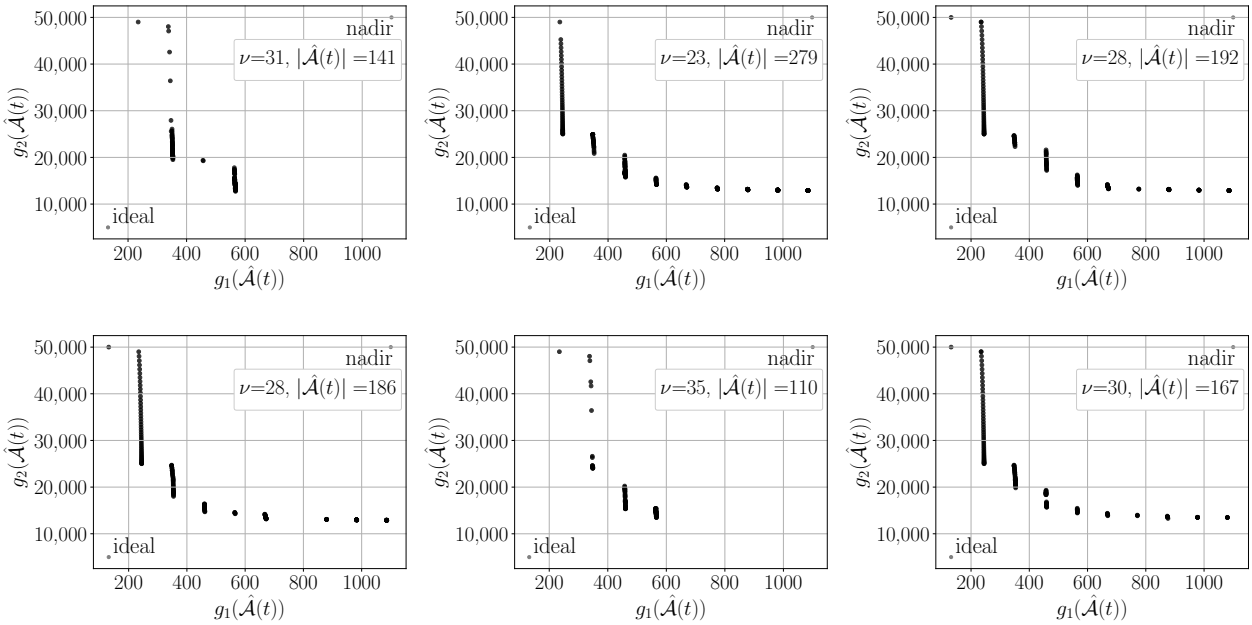
is a local minimizer for the expected cost. The point $(0,0,0,0,0,0,0,0,50)$ places the bus in the middle of the time interval and is a local minimizer for the expected total wait time. These minimizers are not unique. Then, we enumerate the points $(0,0,0,0,0,0,0,0,i)$ for $i=1,2,\ldots,50$, and plot 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 $(0,0,0,0,0,0,0,33,66)$ 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 $(0,0,0,0,0,0,0,i,j)$ 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.

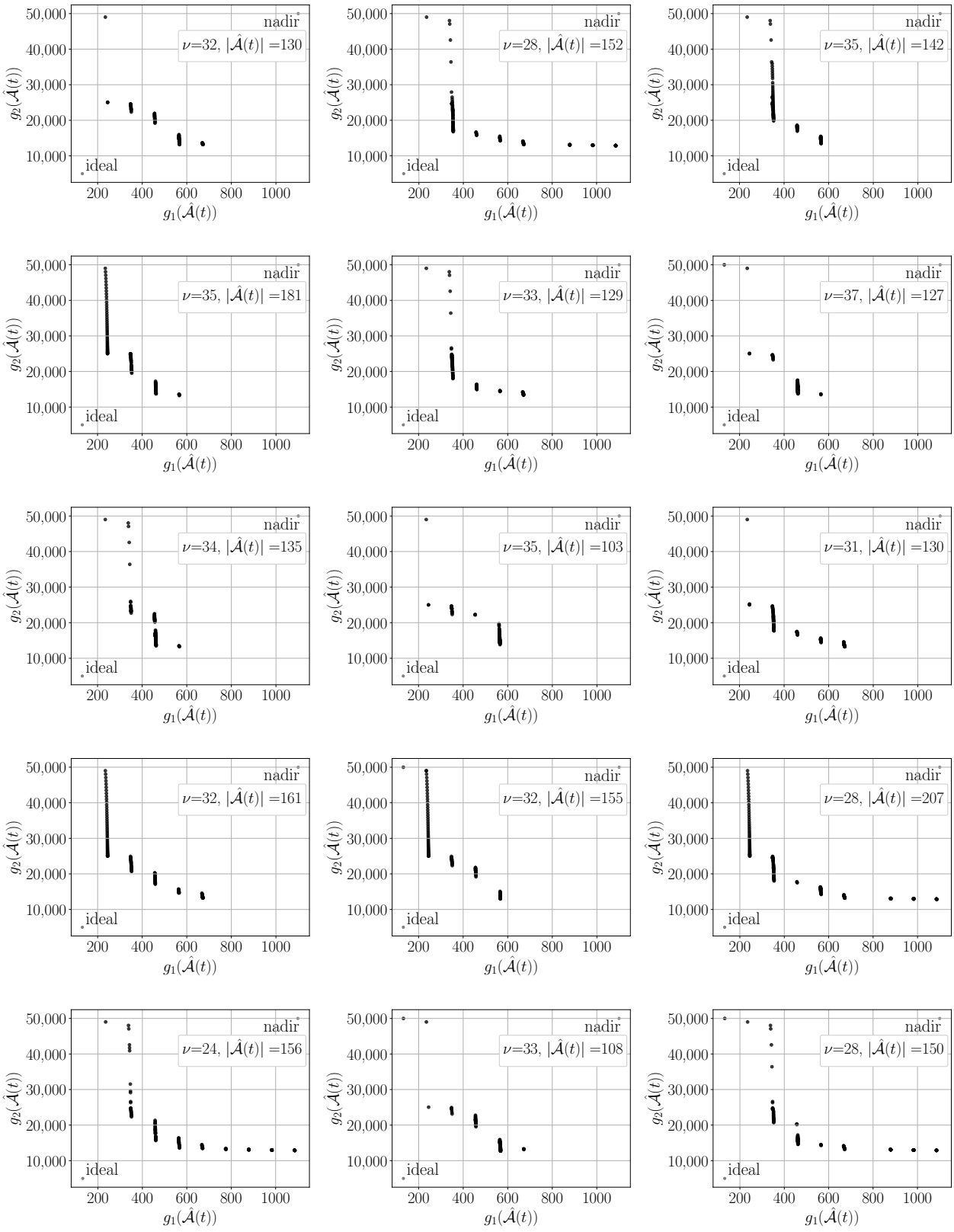
While the plots in Figure 13 and the constrained sub-problems are helpful for understanding the structure of the problem, the points $(0,0,0,0,0,0,0,0,1)$ and $(0,0,0,0,0,0,0,1,2)$ are not actually an $\mathcal{N}_1$-local minimizers on the cost objective in the unconstrained 9-bus scheduling problem — their neighbors, $(0,0,0,0,0,0,0,0,0)$ 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.

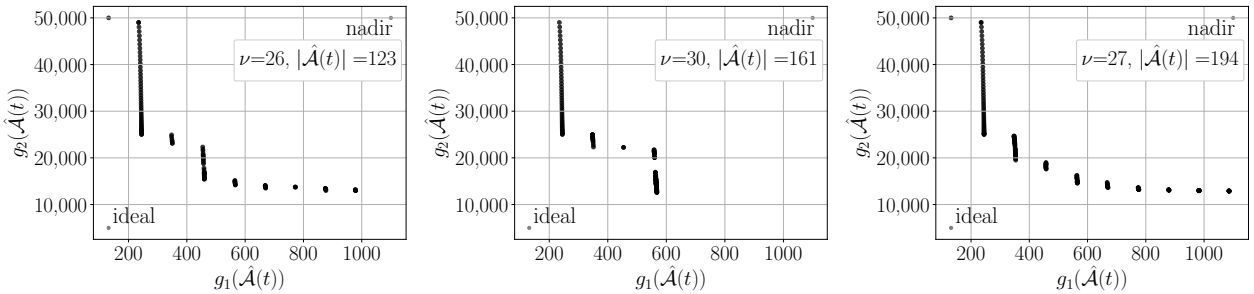
We perform 24 independent runs of R-PERLE on this difficult problem using a total simulation effort of $t = 1\times10^6$ simulation oracle calls. We configure R-PERLE as described in §9.1, and we use CRN across points visited. For consistency with the R-SPLINE implementation of the bus scheduling problem in Wang et al. (2013), we start all 24 runs from the initial feasible point $\mathbf{x}_0 = (1, 1, \ldots, 1)$, 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 $\mathcal{N}_1$-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 $\mathcal{N}_1$-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, $\mathbf{g}(\hat{\mathcal{A}}(t))$. 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.




nadir
ideal
ν=32, $|\hat{\mathcal{A}}(t)|$ =130
ν=28, $|\hat{\mathcal{A}}(t)|$ =152
ν=35, $|\hat{\mathcal{A}}(t)|$ =142
ν=35, $|\hat{\mathcal{A}}(t)|$ =181
ν=33, $|\hat{\mathcal{A}}(t)|$ =129
ν=37, $|\hat{\mathcal{A}}(t)|$ =127
ν=34, $|\hat{\mathcal{A}}(t)|$ =135
ν=35, $|\hat{\mathcal{A}}(t)|$ =103
ν=31, $|\hat{\mathcal{A}}(t)|$ =130
ν=32, $|\hat{\mathcal{A}}(t)|$ =161
ν=32, $|\hat{\mathcal{A}}(t)|$ =155
ν=28, $|\hat{\mathcal{A}}(t)|$ =207
ν=24, $|\hat{\mathcal{A}}(t)|$ =156
ν=33, $|\hat{\mathcal{A}}(t)|$ =108
ν=28, $|\hat{\mathcal{A}}(t)|$ =150
$g_1(\hat{\mathcal{A}}(t))$
$g_2(\hat{\mathcal{A}}(t))$
10,000
20,000
30,000
40,000
50,000
200
400
600
800
1000



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 4×10^6. 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

$$Q^A = d_H\big(\mathbf{g}(\hat{\mathcal{A}}(t)), \mathbf{g}(\mathcal{E})\big),\ Q^B = \min_{\mathcal{L}_1 \in \mathfrak{L}_1} d_H\big(\mathbf{g}(\hat{\mathcal{A}}(t)), \mathbf{g}(\mathcal{L}_1)\big),\ Q^C = \min_{\mathcal{W}_1 \in \mathfrak{W}_1} d_H\big(\mathbf{g}(\hat{\mathcal{A}}(t)), \mathbf{g}(\mathcal{W}_1)\big),$$

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 $\widehat{\text{s.d.}}(T) = \sqrt{(99)^{-1} \sum_{i=1}^{100} (T_i - \bar{T})^2}$ denote the estimated standard deviation of T, and let $\widehat{\text{s.e.}}(\bar{T}) = \widehat{\text{s.d.}}(T)/\sqrt{100}$ 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 $\bar{Q}^{\text{J}} := (100)^{-1} \sum_{i=1}^{100} Q_i^{\text{J}}$. Let the estimated standard deviation of Q^{J} be $\widehat{\text{s.d.}}(Q^{\text{J}}) = \sqrt{(99)^{-1} \sum_{i=1}^{100} (Q_i^{\text{J}} - \bar{Q}^{\text{J}})^2}$, and let the estimated standard error of $\bar{Q}^{\text{J}}$ be $\widehat{\text{s.e.}}(\bar{Q}^{\text{J}}) = \widehat{\text{s.d.}}(Q^{\text{J}})/\sqrt{100}$.

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.**

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

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.**

Performance Metric	Statistic	R-PERLE	R-MinRLE	MO-COMPASS
Time in Minutes	$\bar{T}$	33.13	34.03	34.72
	$\widehat{\text{s.d.}}(T)$	0.83	0.84	0.98
	$\widehat{\text{s.e.}}(\bar{T})$	0.08	0.08	0.10
Solution Quality	$\bar{Q}^{\text{B}}$	0.000	0.000	0.653
	$\widehat{\text{s.d.}}(Q^{\text{B}})$	0.000	0.000	0.680
	$\widehat{\text{s.e.}}(\bar{Q}^{\text{B}})$	0.000	0.000	0.068

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.**

Performance Metric	Statistic	R-PERLE	R-MinRLE	MO-COMPASS
Time in Minutes	$\bar{T}$	39.44	39.69	46.32
	$\widehat{\text{s.d.}}(T)$	1.04	0.81	3.31
	$\widehat{\text{s.e.}}(\bar{T})$	0.10	0.08	0.33
Solution Quality	$\bar{Q}^{\text{C}}$	0.227	0.264	4.476
	$\widehat{\text{s.d.}}(Q^{\text{C}})$	0.222	0.229	3.524
	$\widehat{\text{s.e.}}(\bar{Q}^{\text{C}})$	0.022	0.023	0.352

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.

Performance Metric	Statistic	R-PERLE	R-MinRLE	MO-COMPASS
Time in Hours	$\bar{T}$	16.43	15.25	–[a]
	$\widehat{\text{s.d.}}(T)$	0.97	0.35	–
	$\widehat{\text{s.e.}}(\bar{T})$	0.10	0.03	–

[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

Billingsley P (1995) *Probability and Measure* (New York: John Wiley and Sons), 3 edition.

Wang H, Pasupathy R, Schmeiser BW (2013) Integer-ordered simulation optimization using R-SPLINE: Retrospective Search using Piecewise-Linear Interpolation and Neighborhood Enumeration. *ACM Transactions on Modeling and Computer Simulation* 23(3), URL `http://dx.doi.org/10.1145/2499913.2499916`.