Industrial Strength COMPASS: A Comprehensive Algorithm and Software for Optimization via Simulation

JIE XU and BARRY L. NELSON
Northwestern University
and
L. JEFF HONG
The Hong Kong University of Science & Technology

Industrial Strength COMPASS (ISC) is a particular implementation of a general framework for optimizing the expected value of a performance measure of a stochastic simulation with respect to integer-ordered decision variables in a finite (but typically large) feasible region defined by linear-integer constraints. The framework consists of a global-search phase, followed by a local-search phase, and ending with a "clean-up" (selection of the best) phase. Each phase provides a probability 1 convergence guarantee as the simulation effort increases without bound: Convergence to a globally optimal solution in the global-search phase; convergence to a locally optimal solution in the local-search phase; and convergence to the best of a small number of good solutions in the clean-up phase. In practice, ISC stops short of such convergence by applying an improvement-based transition rule from the global phase to the local phase; a statistical test of convergence from the local phase to the clean-up phase; and a ranking-and-selection procedure to terminate the clean-up phase. Small-sample validity of the statistical test and ranking-and-selection procedure is proven for normally distributed data. ISC is compared to the commercial optimization via simulation package OptQuest on five test problems that range from 2 to 20 decision variables and on the order of 10^4 to 10^{20} feasible solutions. These test cases represent response-surface models with known properties and realistic system simulation problems.

Categories and Subject Descriptors: G.3 [**Mathematics of Computing**]: Probability and Statistics—*Experimental Design*; I.6.6 [**Simulation and Modeling**]: Simulation Output Analysis

General Terms: Algorithms, Experimentation

Additional Key Words and Phrases: Optimization via simulation, random search, ranking and selection

This research was partially supported by National Science Foundation grant number DMI-0217690 and Hong Kong Research Grants Council grant numbers CERG 613305 and 613706. Author Addresses: J. Xu, Department of Industrial Engineering & Management Sciences, Northwestern University, Evanston, IL 60208-3119; email: jiexu@iems.northwestern.edu; B. L. Nelson, Department of Industrial Engineering & Management Sciences, Northwestern University, Evanston, IL 60208-3119; email: nelsonb@northwestern.edu; L. J. Hong, Department of Industrial Engineering and Logistics Management, The Hong Kong University of Science and Technology, Clear Water Bay, Hong Kong; email: hongl@ust.hk.

1. INTRODUCTION

The popularity of stochastic simulation for system design and analysis has been driven by a sequence of key advances: Implementation of intuitive process-interaction (network) modeling paradigms; the development of graphical user interfaces for model development; convenient animation of simulation results; interapplication communication between simulation and other software; and integrated toolkits for optimization of simulated system performance (see Nance and Sargent [2002] for a more comprehensive overview of the evolution of discrete-event, stochastic simulation). Only this last advance—which we call *optimization via simulation (OvS)*—is an analysis capability (analysis methods have certainly been incorporated into simulation software, but are probably not responsible for its popularity). Although not every simulation problem requires optimization, it is rare to find an application where the analyst is uninterested in "the best" settings for the simulated system, and in many cases finding a good system design is the reason the simulation was constructed.

As noted by a number of authors [e.g., Fu 2002], there has been a disconnect between *research* on OvS and the *practice* of OvS, as represented by the commercial OvS products. Simply stated, the impact of published research on commercial software has been limited. Nevertheless, commercial OvS software has been successful because (a) there was and is a substantial market for actually doing OvS in practice, (b) the products are able to handle realistic problems with complex objectives and constraints, while delivering results in a timely manner, and (c) they are integrated into simulation modeling software. The research community, on the other hand, has focused on OvS *correctness*, as quantified by convergence or "correct selection" guarantees. These properties are easiest to prove (which is not to say easy to prove) for simple, elegant algorithms that leave a host of implementation issues unresolved (e.g., complex constraints and stopping rules). Correctness, in this formal sense, is not a feature of commercial products. Correctness matters, however, because when stochastic noise is present an inferior solution may be selected, and its actual performance may be poorly estimated, in the absence of correctness guarantees.

In the last decade there has been significant research activity aimed at bridging this divide, and we believe it has reached a level of maturity that supports a *first step* toward developing OvS software that offers correctness guarantees while also being competitive with the features provided by commercial products. This paper reports one such step, which we call *Industrial Strength COMPASS (ISC)*. The name is derived from the Convergent Optimization via Most Promising Area Stochastic Search algorithm of Hong and Nelson [2006], which is the core of ISC. ISC is a specific instance of a high-level framework for OvS algorithms that consists of three phases: Global, Local and Clean Up. Section 3 defines the properties that we desire for each phase. While we are guilty of drawing heavily on our own research in turning this framework into a specific algorithm, our approach is also strongly influenced by the foundational work of Andradóttir [1995, 1999].

How can we establish that we are "competitive with the features provided by commercial products?" We will do so by comparing ISC to OptQuest (OptTek Systems, Inc.). If OptQuest is not the best of the commercial products, it is certainly a good representative of them and it is very widely used (it is integrated into 13 simulation products according to `www.opttek.com/simulation.html`). OptQuest implements robust metaheuristics; refer to the product web site for more details. Because OptTek Systems, Inc. provided a standalone version of the OptQuest engine, our comparisons are not tied to a particular simulation modeling product.

We use OptQuest to establish a competitive benchmark for optimization performance as a function of simulation effort, and we count on OptQuest to deliver good solutions quickly (which in our experience it does). *We do not expect to beat OptQuest in any comprehensive sense.* OptQuest has had years of development, and its algorithms are smart and efficient. *We consider ISC to be a success if it can deliver as good or better solution quality as OptQuest without expending substantially more simulation effort.* This constitutes "success" because ISC provides convergence guarantees and inference that OptQuest does not. Our only potential advantage comes from how we deal with the stochastic aspect of the problem, which is fundamentally different from any of the commercial products.

Since ISC is a first step, it has limitations:

Objective. Our objective is to maximize or minimize the expected value of a simulation output random variable whose distribution depends on a finite-dimensional vector of controllable decision variables. This is also the implicit objective in OptQuest. ISC does not support multiple objectives.

Decision Variables. We only consider integer-ordered decision variables. Continuous-valued decision variables can be handled by discretizing them, but ISC does not exploit the fact that they are continuous valued nor is discretizing likely to be very efficient. Categorical decision variables are not considered at all, but if there are only a small number of categories then ISC can be run separately for each category. This is more limited than OptQuest.

Constraints. We only consider deterministic constraints, specifically linear-integer inequality constraints. The methodology employed in ISC works for any convex feasible region, but we have only implemented the methods for linear-integer inequality constraints. OptQuest can also include a useful form of stochastic constraint.

Problem Size. While there is no conceptual limit to the problem size that ISC can tackle, its performance will be affected more by the number of decision variables than by the number of feasible solutions that they imply. This is due to the way we define a locally optimal solution as being better than all of its $2d$ immediate neighboring solutions, where d is the number of decision variables (see §3). Therefore, to declare a solution "optimal" at least $2d + 1$ solutions must be simulated extensively. OptQuest is less restrictive on problem size but can provide no guarantee of optimality without simulating *all* feasible solutions.

We will show that ISC achieves our goals: In some cases (noisy problem, multimodal response surface) demonstrating superior performance to OptQuest; in others (low noise, more regular response surface) being beaten by OptQuest in terms of relative effort expended; and in yet others having performance almost indistinguishable from OptQuest. Comparison of COMPASS-based approaches with other convergent algorithms can be found in Hong and Nelson [2006]. The ISC software and all supporting papers may be found at `www.ISCompass.net`.

In the next section we give a more precise statement of the OvS problem, review relevant literature, precisely state our objectives and set up the remainder of the paper.

2. BACKGROUND

We are interested in solving the following problem:

$$\text{Minimize } g(\mathbf{x}) = \mathrm{E}\left[G(\mathbf{x})\right] \quad \text{subject to} \quad \mathbf{x} \in \Theta = \Phi \cap \mathcal{Z}^d \tag{1}$$

where $\mathbf{x}$ is a vector of d integer-ordered decision variables in a feasible region $\Phi \subset \Re^d$, possibly defined by a set of constraints and $\mathcal{Z}^d$ denotes all d-dimensional vectors with integer components. We assume that Φ is compact and convex so that $|\Theta| < \infty$ (but probably quite large). The random variable $G(\mathbf{x})$ typically has no closed form, but can be observed through simulation experiments at $\mathbf{x}$. We assume that $\mathrm{Var}[G(\mathbf{x})] < \infty$ for all $\mathbf{x} \in \Theta$, and that we can simulate independent and identically distributed replications, $G_1(\mathbf{x}), G_2(\mathbf{x}), \ldots$ at any $\mathbf{x}$. Problem (1) is called a discrete optimization-via-simulation (DOvS) problem, and we refer to any $\mathbf{x}$ as a potential "feasible solution."

DOvS problems arise in many areas of operations research and management science. For instance, the following problems can all be modeled as DOvS problems: capacity planning, where the capacities of all workstations need to be determined; call center staffing, where the agents are allocated to different departments and different time periods; and supply-chain management, where inventory order-up-to levels are critical decisions. For reviews of the theory and practice of optimization via simulation, see Fu [2002] and Fu, Glover and April [2005].

A number of methods have been proposed in the research literature to solve DOvS problems, including globally convergent random search (GCRS) algorithms, locally convergent random search (LCRS) algorithms, ranking and selection (R&S) and ordinal optimization (OO). GCRS algorithms converge to the set of global optimal solutions as the simulation effort goes to infinity. They typically achieve global convergence by making sure that all feasible solutions are simulated infinitely often. GCRS algorithms include the stochastic ruler algorithm of Yan and Mukai [1992], the simulated annealing algorithm of Alrefaei and Andradóttir [1999], the stochastic comparison algorithm of Gong et al. [1999] and the nested partitions algorithm of Shi and Ólafsson [2000] and Pichitlamken and Nelson [2003].

LCRS algorithms include the COMPASS algorithm [Hong and Nelson 2006] and the revised COMPASS algorithm [Hong and Nelson 2007a] that converge to a locally optimal solution as the simulation effort goes to infinity. Hong and Nelson [2007a] show that local convergence can be achieved by simulating only a locally optimal solution and its neighbors infinitely often. Compared with GCRS, LCRS algorithms generally converge much faster in practice, but they can be trapped in inferior solutions if there exist multiple locally optimal solutions.

R&S procedures select the best solution by simulating all of them and making statistical inference in the form of a probability of correct selection (PCS). They include the indifference-zone procedures [e.g., Nelson et al. 2001], the Bayesian procedures [e.g., Chick and Inoue 2001], and the optimal computing budget allocation procedures [OCBA, e.g., Chen et al. 2000]. When the set of feasible solutions of Problem (1) is small, R&S procedures can be directly applied to find the best solution. When the set is large, special R&S procedures have been developed to facilitate the optimization process [e.g., Hong and Nelson 2007b] and to "clean up" after the optimization process [Boesel et al. 2003b].

OO softens the goal of finding an optimal solution to finding a top-n solution [Ho et al. 1992]; it becomes a DOvS algorithm by first picking (often by blind picking) a set of k solutions and then finding the best among the k solutions, often by using OCBA. The critical issue is to determine k, which OO does by achieving a certain level of the "alignment probability," which is the probability that at least one of the k solutions is a top-n solution.

The algorithms proposed in the research community typically have good convergence or statistical properties. However, they are often too simple or inefficient to solve practical problems. Therefore, they are seldom used in practice. On the other hand, all of the major commercial simulation modeling products have simulation optimization toolkits that use sophisticated metaheuristics designed for challenging deterministic optimization problems, but that often take a simplistic approach to handling the noise of stochastic problems. Therefore, the algorithms work well when the level of noise in the simulation output is low, but they can be significantly misled when it is not.

In this paper, we modify several existing algorithms in the research literature and combine them together under a carefully designed algorithm framework. We show that the resulting new algorithm, called ISC, has nice convergence and statistical properties and also has competitive performance compared to the commercial package OptQuest. The algorithm framework includes three phases: Global, Local and Clean Up. The Global Phase explores the whole feasible region and identifies several good solution seeds; the Local Phase takes one seed at a time and finds a locally optimal solution; and the Clean-up Phase selects the best from the set of solutions identified in the Local Phase and also estimates its expected performance.

ISC is obtained by making specific choices for each phase. For the Global Phase, we adapt a niching genetic algorithm (NGA) [Miller and Shaw 1995, Sareni and Krahenbuhl 1998] that can quickly identify several clusters of good solutions. We show that our version of the NGA can be made to converge to a globally optimal solution as the simulation effort goes to infinity. For the Local Phase, we use a modified version of the COMPASS algorithm that converges to a locally optimal solution. A stopping test is used to terminate the COMPASS algorithm when a locally optimal solution is identified with probability at least $1-\alpha_L$. Therefore, as $1-\alpha_L \to 1$, the COMPASS algorithm can be shown to converge with probability 1 to a locally optimal solution. For the Clean-up Phase, we apply a R&S procedure to select the best of the solutions identified in the Local Phase with probability greater than or equal to $1-\alpha_C$, and also provide a fixed-width $\pm\delta_C$ confidence interval for the expected value of the performance of the selected solution. As $1-\alpha_C \to 1$ the best of these solutions will be identified with probability 1. Throughout the paper we use the subscripts G, L and C to denote parameters associated with the Global, Local and Clean-up Phases, respectively.

An alterative framework that has been proposed is the balanced explorative and exploitative search with estimation scheme (BEESE) [Prudius and Andradóttir 2004, 2006]. BEESE integrates the global search, local search and estimation together, and it converges to a globally optimal solution as the simulation effort goes to infinity. A framework with similar objectives is also proposed by Lin and Lee [2006]. Compared to these frameworks, our framework is designed to find good locally optimal solutions, and stop with a statistical guarantee that we have done so, without the need to simulate all feasible solutions. However, our approach requires transitions between phases which these frameworks avoid.

The remainder of the paper is organized as follows: In Section 3 we introduce the three-phase framework and discuss how to transition between phases. A high-level description of the ISC algorithm is presented in Section 4 along with its convergence and statistical properties. In Section 5 we evaluate the performance of ISC through a number of examples and compare it to OptQuest. The paper is concluded in Section 6. Proofs and the details of the ISC algorithm are provided in the appendices, some of which are electronic (online-only).

3. DOVS FRAMEWORK

Our DOvS framework has three phases, Global, Local and Clean Up. In this section, we discuss the desired properties of each phase.

In the Global Phase we want to quickly identify a number of solution seeds that may lead to competitive locally optimal solutions in the second phase, and also facilitate a quick start for the local search of the second phase. To ensure that the algorithm used in this phase has good large-sample properties, we require it to be globally convergent if the simulation effort of this phase goes to infinity.

Although the algorithm of the first phase should be globally convergent, it will transition to the second phase in practice. The transition rules can be effort based or quality based. An effort-based rule transitions from the first phase to the second phase after the simulation budget for the first phase is consumed; a quality-based rule transitions if it is clear that the seeds can lead to high-quality solutions in the second phase. Effort-based rules are often arbitrary and therefore to be avoided; but they are sometimes necessary because it is difficult to predict in advance how long it might take a particularly difficult problem to transition based on quality rules. The detailed transition rules used in ISC are introduced in the next section.

The Local Phase starts with the best solution seed identified in the first phase and uses efficient local search to find a locally optimal solution; it then goes on to the second-best solution seed and so on until exhausting all solution seeds. To ensure good performance of the algorithm used in this phase, we require it to be locally convergent as the simulation effort goes to infinity regardless of the quality of the seeds that are provided. We define a *local minimum* as follows (this definition is also used in Hong and Nelson [2006, 2007a]):

Definition: Let $\mathcal{N}(\mathbf{x}) = \{\mathbf{y} : \mathbf{y} \in \Theta \text{ and } \|\mathbf{x} - \mathbf{y}\| = 1\}$ be the local neighborhood of $\mathbf{x} \in \Theta$, where $\|\mathbf{x} - \mathbf{y}\|$ denotes the Euclidean distance between $\mathbf{x}$ and $\mathbf{y}$. Then $\mathbf{x}$ is a *local minimum* if $\mathbf{x} \in \Theta$ and either $\mathcal{N}(\mathbf{x}) = \emptyset$ or $g(\mathbf{x}) \leq g(\mathbf{y})$ for all $\mathbf{y} \in \mathcal{N}(\mathbf{x})$. Let $\mathcal{M}$ denote the set of local minimizers of the function g in Θ.

Since the quality of the solutions found in the Local Phase is critical to the performance of the algorithm, in practice we require each solution obtained in this phase to be a local minimum with probability at least $1 - \alpha_L$. Therefore, as the

PCS $1 - \alpha_L \to 1$, the effort goes to infinity and the algorithm is guaranteed to be locally convergent.

The transition between the second and third phases can be either effort based or quality based. For the effort-based rule, transition happens when the simulation budget for the second phase is exhausted. If there is no effort limit, the algorithm transitions once all solutions seeds are searched, which is what we prefer. The Local Phase returns a set of solutions $\mathcal{L}$, and is designed to give high confidence that $\mathcal{L} \subset \mathcal{M}$.

In the Clean-up Phase we want to select the best solution from among the locally optimal solutions $\mathcal{L}$ found in the second phase, and the actual value of the selected solution also needs to be estimated to within $\pm\delta_C$, all with confidence level $\geq 1-\alpha_C$, where $\delta_C > 0$ and $0 < \alpha_C < 1$ are set by the user. As $1 - \alpha_C \to 1$ the best solution is therefore selected with probability 1.

4. METHODOLOGY

Our goal is to produce a DOvS algorithm that satisfies the requirements set by Andradóttir and Nelson [2004]: Provable asymptotic performance, competitive finite-time performance and valid statistical inference at termination. We achieve this by using the framework in §3. To a certain extent, we exploit existing technologies to fill in each phase. However, the requirement that we achieve "competitive finite-time performance" requires us to invent a number of enhancements that improve performance without affecting convergence guarantees. These enhancements range from essential speed-ups to minor tweaks, and from rigorously justified to "makes sense and seems to work well in experiments." In this section we provide a high-level review of the existing technologies we incorporate and the details of only the most critical enhancements. In the electronic appendix we fully document the entire algorithm; Figure 1 is an overview of the approach.

A few basic terms and concepts are critical to an understanding of ISC:

—To "sample a solution" means to select a solution randomly, either from Θ or a designated subregion of it. The methods we use to sample solutions guarantee that all solutions in the region of interest have a (nearly) uniform probability of being selected.

—To "evaluate a solution" means to obtain one or more independent replications of its performance and to average those results. The set of "visited solutions" are those that have been evaluated at least once. We always accumulate performance data on visited solutions, so that if a solution is evaluated more than once its performance is estimated by the cumulative sample mean of all replications.

—Convergence guarantees are always as the simulation effort goes to infinity. To keep the effort from going to infinity in ISC, we use transition rules in the Global Phase, and statistical tests in the Local and Clean-up Phases. The small-sample validity of these tests depends on the simulation output data being normally distributed, or on the sample size being large enough that Central Limit Theorem considerations make normality a good approximation for the cumulative average. The nature of the COMPASS algorithm, which concentrates effort on apparently locally optimal solutions and their neighbors, tends to generate enough replications of the tested solutions that approximate normality can be anticipated. The

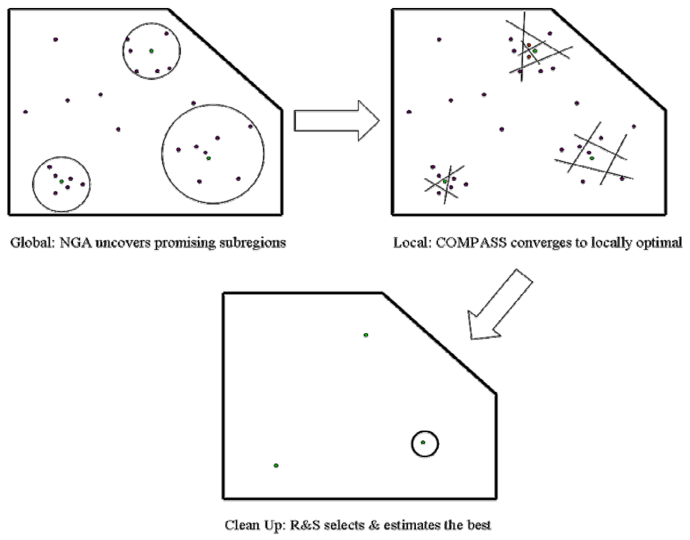
Global: NGA uncovers promising subregions
Local: COMPASS converges to locally optimal
Clean Up: R&S selects & estimates the best

Fig. 1. Industrial Strength COMPASS implementation of the optimization framework.

convergence guarantees of the Local and Clean-up Phases do not depend on the small-sample validity of these tests, however.

4.1 Global Phase

The role of the NGA is to serve as our global search engine, and we have extended the basic NGA technology to make it globally convergent (if desired) for the class of DOvS problems considered here (see the Appendix). As noted in §2, there are already globally convergent DOvS algorithms, so why do we need another? In practice, we will transition from Global to Local phases long before achieving anything like global convergence, and the Local Phase will then pursue promising subregions more intensely. Therefore, we need an algorithm that can achieve global convergence, in a limiting sense, but does so by simultaneously identifying and investigating one or more promising subregions of Θ that will be seeds for the locally convergent second phase. COMPASS, which we use for the Local Phase, converges particularly quickly given a good starting solution and a cluster of solutions around it. The NGA, which is designed for multimodal functions, accomplishes this.

We chose genetic-algorithm technology because it is a population-based, as opposed to a point-to-point, search that considers many good solutions in parallel and therefore tends to be somewhat robust to stochastic noise [Jin and Branke 2005]. That GAs converge slowly is not an issue since we depend on the rapid convergence of COMPASS to close in on locally optimal solutions. What is far more important is that we explore the feasible region broadly and identify collections of solutions in promising subregions.

4.1.1 *Niching GA.* GAs, by their nature, are complicated algorithms to describe. Here we present a high-level overview of our version of a NGA; complete details for each step below may be found in the electronic appendix. When we stop the NGA, based on a transition rule, it delivers one or more "niche centers" (good solutions or seeds) and a cluster of other near-by solutions, as defined by a "niche radius;" see Figure 1. The Local Phase attacks each niche in turn looking for a locally optimal solution.

Niching GA

Initialization. Randomly sample m_G solutions $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_{m_G}$ from Θ and evaluate them.

Evolution. Execute a generation of the NGA to evolve the current population of m_G solutions into a new population of m_G solutions, maintaining a niche structure around apparently good solutions:

(1) Form niches: Form up to q niches by clustering solutions within radius r of good solutions.

(2) Selection: Select $m_G/2$ solutions randomly, but with replacement, from the population using a linear ranking scheme that gives preference to solutions with better estimated performance (we use the approach described in Boesel [1999] and Boesel et al. [2003a]).

(3) Mating: For each solution selected in Step 2, use a mating restriction scheme to select its partner.

(4) Crossover: Randomly create a new pair of solutions that are geometrically between each pair of solutions.

(5) Mutation: Randomly perturb the coordinates $x_{i1}, x_{i2}, \ldots, x_{id}$ of each new solution $\mathbf{x}_i$.

Evaluate Solutions. Evaluate all solutions in the current population.

Next Generation. Replace solutions in the population with newly generated solutions.

Go to Evolution.

Key input parameters for the NGA are the number of niches to form, q, and the niche radius, r, which defines how close a solution must be to a niche center (good solution) to be clustered in the same niche. Because our algorithm is intended to be general purpose, we have no way to know these values in advance, nor should we expect the user to be able to provide them. To solve this problem, we enhance the *Initialization* and *Evolution* steps so that on each iteration of the NGA we form a rough response surface model over Θ from which we extract estimates of q and r; therefore, q and r will change from iteration to iteration of the NGA. The modified steps are as follows:

Enhanced Initialization. Estimate the number of regions containing local minimums, q, and half the shortest distance between any of these two local minimums, r, which will then serve as the number of niches and the niche radius, respectively, for the NGA.

(1) Randomly sample m_G solutions $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_{m_G}$ from Θ. The value of m_G is set as at least 50, since this guarantees that, with a probability greater than 90%, at least one of the m_G solutions are among the best 5% of all solutions in Θ based on calculation of the alignment probability in ordinal optimization (see, for instance, Ho et al. [1992]).

(2) Evaluate each solution by obtaining n_0 replications and averaging.

(3) For each $\mathbf{x} \in \Theta$, let its value be the sample average of the closest visited solution to it; this defines a piecewise constant response surface over Θ. On this response surface, let q be the number of local minimums (niche centers) and let r be half the distance between the closest pair of local minimums.

Enhanced Evolution. Execute a generation of the NGA to evolve the current population of m_G solutions into a new population of m_G solutions, maintaining a niche structure:

(1) For each $\mathbf{x} \in \Theta$, let its value be the sample average of the closest visited solution to it; this defines a piecewise constant response surface over Θ. On this response surface, let q be the number of local minimums (niche centers) and let r be half the distance between the closest pair of local minimums.

(2) Continue with the Evolution step as before....

Like most GAs, NGAs were designed for deterministic optimization problems. In our stochastic context, to "evaluate" a solution means to simulate it and estimate its value. The second key enhancement relates to how we do this so that we obtain a global convergence guarantee (this step, along with our implementation of mutation, guarantee that all solutions will be simulated infinitely often if the NGA is never terminated, so that the sample averages will converge to their true expectations with probability 1; see Appendix A):

Evaluation of Solutions. For all solutions in the current population that have been evaluated previously, generate Δn additional replications and average. For all solutions in the population that have not yet been evaluated, generate n_0 replications and average.

4.1.2 *Transition Rules.* In practice, we terminate the Global Phase when conditions indicate that we have uncovered fertile regions for seeking locally optimal solutions. We believe that there are at least four categories of transition rules:

Budget Rule. The user may specify a budget (in terms of number of replications) for the Global Phase so that a transition occurs when the budget is exhausted if it has not already occurred.

Niche Rule. If at any time there is only one niche, then transition to the Local Phase.

Improvement Rule. If there is no apparent improvement in solution quality in T_G generations, then transition to the Local Phase. The value of T_G may be user specified.

Dominance Rule. If the solutions in one niche statistically dominate the solutions in all other niches, then transition to the Local Phase.

Our implementation of ISC incorporates all of these rules, but in the empirical evaluation we did not set a budget limit.

4.2 Local Phase

The role of the COMPASS algorithm is to start with a niche of solutions as input and rapidly converge to a locally optimal solution. COMPASS is guaranteed to do this as the number of iterations goes to infinity even if the niche does not surround a locally optimal solution [Hong and Nelson 2006]. However, the hope is that the NGA has identified small subregions containing local optima so that COMPASS can converge very quickly.

4.2.1 *COMPASS.* In the COMPASS algorithm, we use $\mathcal{V}_k$ to denote the set of all solutions visited through iteration k of a COMPASS run, and use $\widehat{\mathbf{x}}_k^*$ to denote the solution with the smallest aggregated sample mean among all $\mathbf{x} \in \mathcal{V}_k$; that is, $\widehat{\mathbf{x}}_k^*$ is the sample best solution through iteration k. When COMPASS is run after the transition from the Global Phase, the set $\mathcal{V}_0$ contains the solutions from the niche being explored.

At the end of iteration k, we construct

$$\mathcal{C}_k = \{\mathbf{x} : \mathbf{x} \in \Theta \text{ and } \|\mathbf{x} - \widehat{\mathbf{x}}_k^*\| \le \|\mathbf{x} - \mathbf{y}\|,\ \forall \mathbf{y} \in \mathcal{V}_k \text{ and } \mathbf{y} \ne \widehat{\mathbf{x}}_k^*\}$$

the *most promising area* at iteration k. The set $\mathcal{C}_k$ includes all feasible solutions that are at least as close to $\widehat{\mathbf{x}}_k^*$ as to other solutions in $\mathcal{V}_k$. At iteration $k+1$, COMPASS samples m_L solutions (nearly) uniformly from $\mathcal{C}_k$. Notice that $\mathcal{C}_k$ is never empty since $\widehat{\mathbf{x}}_k^*$ is always in $\mathcal{C}_k$, and we do not require the m_L solutions to be unique. As we show below, the most promising area is defined by a collection of linear constraints (including the original constraints that defined the feasible region).

The COMPASS algorithm uses a simulation-allocation rule (SAR) to evaluate solutions in $\mathcal{V}_k$ at each iteration. The rules we employ, which are described in the electronic appendix, allocate replications either based on a fixed schedule or using OCBA ideas. Let $a_k(\mathbf{x})$ be the additional observations allocated to $\mathbf{x}$ at iteration k as determined by the SAR; $a_k(\mathbf{x})$ may depend on all past information such as $\mathcal{V}_k$ and $G_i(\mathbf{x})$, $i = 1, 2, \ldots$, for all $\mathbf{x} \in \mathcal{V}_k$. Then $N_k(\mathbf{x}) = \sum_{i=0}^k a_i(\mathbf{x})$ denotes the total number of observations on solution $\mathbf{x}$ at iteration k for every $\mathbf{x} \in \mathcal{V}_k$. We use $\bar{G}_k(\mathbf{x})$ to denote the sample mean of all $N_k(\mathbf{x})$ observations of $G(\mathbf{x})$ at iteration k. The basic COMPASS algorithm is as follows [Hong and Nelson 2006]:

COMPASS

Step 0. Set iteration counter $k = 0$ and form $\mathcal{C}_0$ based on the solutions in $\mathcal{V}_0$.

Step 1. Let $k = k + 1$. Sample $\mathbf{x}_{k1}, \mathbf{x}_{k2}, \ldots, \mathbf{x}_{km_L}$ uniformly and independently from $\mathcal{C}_{k-1}$. Let $\mathcal{V}_k = \mathcal{V}_{k-1} \cup \{\mathbf{x}_{k1}, \mathbf{x}_{k2}, \ldots, \mathbf{x}_{km_L}\}$. Determine $a_k(\mathbf{x})$ according to the SAR for every $\mathbf{x}$ in $\mathcal{V}_k$. For all $\mathbf{x} \in \mathcal{V}_k$, evaluate $\mathbf{x}$ by taking $a_k(\mathbf{x})$ observations and updating $N_k(\mathbf{x})$ and $\bar{G}_k(\mathbf{x})$.

Step 2. Let $\widehat{\mathbf{x}}_k^* = \operatorname{argmin}_{\mathbf{x} \in \mathcal{V}_k} \bar{G}_k(\mathbf{x})$. Construct $\mathcal{C}_k$ and go to Step 1.

The COMPASS algorithm converges to a locally optimal solution as $k \to \infty$ under very mild conditions, essentially that each sample mean $\bar{G}_k(\mathbf{x})$ satisfies a strong law of large numbers, the SAR guarantees that the number of replications allocated to all visited solutions goes to infinity, and Θ is finite [Hong and Nelson 2006]. Stronger conditions allow COMPASS to converge when Θ is countably infinite; see Hong and Nelson [2006] and also Andradóttir [2006]. Since ISC assumes that observations of a solution are obtained from independent and identically distributed replications with finite variance, the first condition is easily satisfied. However, as noted in Hong and Nelson [2007a], the latter condition—that the number of replications that *all* visited solutions receive goes to infinity—is inefficient and much stronger than necessary. In fact, only those solutions in $\mathcal{V}_k$ that are necessary to define $\mathcal{C}_k$ need to receive additional replications on iteration k, and only solutions in the neighborhood of the locally optimal solution identified by COMPASS need to receive an infinite number of replications in the limit.

Our primary enhancement to COMPASS is solution pruning, which means that at the end of iteration k we determine which solutions in $\mathcal{V}_k$ generate active constraints for the most promising area; stated differently, we find the smallest set of solutions that define $\mathcal{C}_k$. Then only the solutions that define $\mathcal{C}_k$, the current sample best solution, and the newly sampled solutions $\mathbf{x}_{k1}, \mathbf{x}_{k2}, \ldots, \mathbf{x}_{km_L}$ are evaluated according to the SAR. The trick is figuring out if $\mathbf{x} \in \mathcal{V}_k$ is an active solution, which is the enhancement we present here.

In COMPASS, each visited solution $\mathbf{x}_i \in \mathcal{V}_k$ that is not the current sample best solution $\widehat{\mathbf{x}}_k^*$ defines a constraint plane that is halfway between $\mathbf{x}_i$ and $\widehat{\mathbf{x}}_k^*$. The most promising area consists of the solutions closer to $\widehat{\mathbf{x}}_k^*$ than to $\mathbf{x}_i$; that is, on the $\widehat{\mathbf{x}}_k^*$ side of the planes. These constraints take the form

$$(\widehat{\mathbf{x}}_k^* - \mathbf{x}_i)' \left(\mathbf{x} - \frac{\widehat{\mathbf{x}}_k^* + \mathbf{x}_i}{2} \right) \geq 0.$$

Removing redundant solutions is equivalent to finding all solutions that do not define active constraints. We can decide if a solution $\mathbf{x}_i \in \mathcal{V}_k$ is redundant by solving a linear program (LP) where we maximize the constraint of interest subject to the other constraints (see for instance Telgen [1983]). Therefore, to decide if solution $\mathbf{x}_i$ is redundant we solve

$$\begin{aligned} &\min_{\mathbf{x}} \ (\widehat{\mathbf{x}}_k^* - \mathbf{x}_i)' \left(\mathbf{x} - \frac{\widehat{\mathbf{x}}_k^* + \mathbf{x}_i}{2} \right) \\ &\text{s.t.} \qquad (\widehat{\mathbf{x}}_k^* - \mathbf{x}_j)' \left(\mathbf{x} - \frac{\widehat{\mathbf{x}}_k^* + \mathbf{x}_j}{2} \right) \geq 0 \qquad \forall \mathbf{x}_j \in \mathcal{V}_k, j \neq i. \end{aligned}$$

If the objective function is nonnegative, then the constraint is redundant and the solution is not active. After doing it the first time we only need to apply pruning to the currently active solutions and the newly sampled solutions, not all visited solutions. Therefore, the number of active solutions in COMPASS can be kept small by occasionally doing this pruning, and since solving small LPs is fast, the overhead is more than offset by avoiding the need to apply the SAR to an ever-increasing number of solutions. Further, it turns out that having a small number of COMPASS constraints also speeds the random sampling of new solutions from the most promising area.

A second, less-critical enhancement to COMPASS is to locate the constraint plane based on how much improvement we predict in moving from $\mathbf{x}_i$ to $\widehat{\mathbf{x}}_k^*$ rather than always using half the distance. This modification keeps COMPASS from closing in too quickly when the current sample best is not conclusively better than the other active solutions, and is described in the electronic appendix.

4.2.2 *Transition Rule.* The evolution of the COMPASS algorithm guarantees that eventually the most promising area is a single solution $\widehat{\mathbf{x}}^*$, and that only the solutions in its neighborhood $\mathcal{N}(\widehat{\mathbf{x}}^*)$ will be active. At this point it is reasonable to stop COMPASS and test if the single solution is superior to all of its neighbors. We propose the following hypothesis test:

$$H_0: \; g(\widehat{\mathbf{x}}^*) \leq \min_{\mathbf{y} \in \mathcal{N}(\widehat{\mathbf{x}}^*)} g(\mathbf{y}) \quad \text{vs.} \quad H_1: \; g(\widehat{\mathbf{x}}^*) > \min_{\mathbf{y} \in \mathcal{N}(\widehat{\mathbf{x}}^*)} g(\mathbf{y}).$$

We set the Type I error to α_L and we want the power to be at least $1 - \alpha_L$ if $g(\widehat{\mathbf{x}}^*) \geq \min_{\mathbf{y} \in \mathcal{N}(\widehat{\mathbf{x}}^*)} g(\mathbf{y}) + \delta_L$, where δ_L is a tolerance that the user chooses (that defaults to δ_C defined in the next section). If $\widehat{\mathbf{x}}^*$ passes the test, then we stop COMPASS and declare $\widehat{\mathbf{x}}^*$ to be locally optimal; otherwise, the test gives a solution in $\mathcal{N}(\widehat{\mathbf{x}}^*)$ that is better than $\widehat{\mathbf{x}}^*$, which enables COMPASS to continue. Notice that, if we drive $1 - \alpha_L$ to 1, then this drives the sample size to infinity and we are guaranteed to find a locally optimal solution.

We can rewrite this test in the following form:

$$\begin{aligned} \Pr\{\text{declare } \widehat{\mathbf{x}}^* \text{ locally optimal}\} &\geq 1 - \alpha_L \quad \text{if } g(\widehat{\mathbf{x}}^*) \leq \min_{\mathbf{y} \in \mathcal{N}(\widehat{\mathbf{x}}^*)} g(\mathbf{y}) \\ \Pr\{\text{declare } \widehat{\mathbf{x}}^* \text{ not locally optimal}\} &\geq 1 - \alpha_L \quad \text{if } g(\widehat{\mathbf{x}}^*) \geq \min_{\mathbf{y} \in \mathcal{N}(\widehat{\mathbf{x}}^*)} g(\mathbf{y}) + \delta_L. \end{aligned}$$

This test can be viewed as a special case of comparisons with a standard (see, for instance, Nelson and Goldsman [2001] and Kim [2005]), with $\widehat{\mathbf{x}}^*$ being the standard. Therefore, we can use a highly efficient comparison-with-a-standard procedure to carry out the test. In the ISC algorithm we use the sequential procedure of Kim [2005]. This procedure takes one simulation observation at a time from all solutions still in contention and determines if any of them can be eliminated. It stops when only one solution remains and selects that solution. The procedure typically needs fewer simulation observations to make the selection decision than a two-stage procedure, for instance, the procedures in Nelson and Goldsman [2001]. Moreover, it can be shown to guarantee the Type I error and power requirement if the simulation observations are normally distributed.

If there is no budget limit in the Local Phase (and we did not impose one in our empirical evaluation), then ISC transitions to the Clean-up Phase when all niches identified in the Global Phase have been explored by COMPASS. Therefore, the first two phases of the ISC algorithm attempt to ensure that the search space is both fully explored and thoroughly searched.

4.3 Clean-up Phase

When we transition to the Clean-up Phase, we have a set $\mathcal{L}$ of solutions that have been declared, with high statistical confidence, to be locally optimal ($\mathcal{L} \subset \mathcal{M}$). If $|\mathcal{L}| = 1$, then we simply need to make sure we have estimated the value of this solution with sufficiently high precision; if $|\mathcal{L}| > 1$ then we want to select the best of these local optima, and estimate its value with sufficiently high precision.

We measure "precision" by a user-specified parameter $\delta_C > 0$, which indicates that we want confidence $\geq 1 - \alpha_C$ that we have identified the best in $\mathcal{L}$ when its true mean is at least δ_C better than all of the others. The user should set δ_C to the smallest difference that it is worth detecting relative to the decision that the simulation model is required to support. The smaller δ_C is the more simulation effort (replications) that will be required to deliver the correct-selection guarantee. We also want to estimate the value of the selected solution to within $\pm\delta_C$ with high confidence. If we drive $1 - \alpha_C \to 1$ then this drives the sample size to infinity and we guarantee to both select the best and estimate its value perfectly. To accomplish this we invoke the "clean up" technology of Boesel et al. [2003b] to select the best, and enhance their procedure with a $\pm\delta_C$ confidence interval on the value of the selected solution. Here we present a high-level overview of the Clean-up Phase.

Clean Up

Screening. Using whatever data are already available on the solutions in $\mathcal{L}$, discard any solutions that can be shown to be statistically inferior to one or more of the others. Let $\mathcal{L}_C$ be the surviving solutions.

Selection. Acquire enough additional replications on the solutions in $\mathcal{L}_C$ to select the best with the desired guarantees. Let $\mathbf{x}_B$ be the selected solution.

Estimation. With confidence level $\geq 1 - \alpha_C$, $\mathbf{x}_B$ is the best, or within δ_C of the best, of the solutions in $\mathcal{L}$ and $g(\mathbf{x}_B) \in \bar{G}(\mathbf{x}_B) \pm \delta_C$.

A proof of the validity of the confidence interval for normally distributed data is in Appendix B.

5. EVALUATION

The "no free lunch theorems" of Wolpert and Macready [1997] imply that we cannot expect any optimization algorithm to dominate all others, and they support the intuitive notion that, with essentially no restriction on the function $g(\mathbf{x})$ and the noisiness of $G(\mathbf{x})$, we can always invent a problem to make any approach look bad. This fact is a strong argument in favor of correctness guarantees, because even in a problem designed to make, say, ISC perform poorly, we nevertheless have a guarantee to get a locally optimal solution eventually. In any event, no statements about either ISC or OptQuest that apply to all possible problems can be made based on the five test cases we present here.

To evaluate the performance of ISC we have selected both known response-surface functions to which we add noise, and realistic DOvS problems that have a physical interpretation. The response-surface functions were chosen to facilitate control of the surface, noise properties and problem dimension to see how they affect ISC. The realistic problems were selected to provide some sense of how ISC can be expected to perform in practice. The problems were not selected to be particularly difficult for OptQuest, and in fact for one of the response-surface functions OptQuest is significantly better than ISC, using our performance metrics.

We made three important decisions about how ISC would be evaluated with respect to OptQuest:

(1) For each DOvS problem considered, we make multiple, independent trials of each optimization algorithm. For ISC, this means that both the random sampling that is internal to ISC, and the simulation outputs themselves, are independent from trial to trial. For OptQuest, only the simulation outputs are independent from trial to trial since we have no control over what, if any, randomness is internal to OptQuest.

(2) We focus on average performance over the trials as a function of the simulation effort (although we also look at the quality of the final solution from individual

runs). This raises two issues:

(a) We measure "effort" only in terms of the simulation effort, and specifically the number of replications consumed. In doing so, we are assuming that (i) the effort per replication is roughly the same at all solutions $\mathbf{x} \in \Theta$, which is reasonable for our test problems, and (ii) that the optimization overhead for both ISC and OptQuest is small relative to the cost of running simulations. Assumption (ii) is not strictly true. For our response-surface functions the simulations are essentially instantaneous, so algorithm overhead is probably relatively significant. And even though OptQuest is a finely tuned algorithm that typically seemed to impose little overhead, on our flowline design and inventory management test problems it did execute substantially slower than ISC while on the high-dimensional response-surface problem it was significantly faster. Solving the LPs to do solution pruning and randomly sampling the most-promising area in the Local Phase are the primary overhead in ISC, and both can be slow but could be made faster with continued development.

(b) We judged performance, when possible, using the *true* expected value of the solution that each algorithm currently thinks is the best, rather than by its *estimated* value. Of course this is impossible to do in real problems for which $g(\mathbf{x})$ is not known, but it is the appropriate metric for an evaluation since $g(\widehat{\mathbf{x}}^*)$ will be the long-run performance we get from the selected solution, not $\bar{G}(\widehat{\mathbf{x}}^*)$.

(3) We first ran trials of ISC, because ISC has the advantage of knowing when to stop. OptQuest can stop in two ways: either the computation budget is exhausted or no improvement in the estimated objective value is obtained within a certain number of iterations. To make sure that we did not stop OptQuest prematurely, we gave it a budget substantially larger than the maximum number of replications any ISC trial used for the first three and the fifth test problems, and the average number of replications ISC used for the fourth test problem. We did not have OptQuest stop based on lack of improvement.

5.1 Test Problems

The first test problem is a modification of the multimodal function F_2 used in Miller and Shaw [1995]. We rescale F_2, change its sign (since it represents a maximization problem) and then add up two copies of it and call the resulting function $g_1(x_1, x_2)$:

$$\begin{aligned} F_2(x) &= \frac{\sin^6(0.05\pi x)}{2^{2(\frac{x-10}{80})^2}}, 0 \le x \le 100 \\ g_1(x_1, x_2) &= -[F_2(x_1) + F_2(x_2)],\ 0 \le x_1, x_2 \le 100. \end{aligned} \tag{2}$$

The function F_2 has 5 local optima with a global optimum at $x = 10$. The magnitudes of the local optima decrease exponentially: $F_2(10) = 1, F_2(30) = 0.917, F_2(50) = 0.7071, F_2(70) = 0.4585$ and $F_2(90) = 0.25$. Since there is no interaction between the two variables, g_1 has 25 local optima and a global optimum at $(10, 10)$ with objective value -2 (see Figure 2). This function represents a response surface with many, widely spaced local optima.

Normally distributed noise with zero mean and standard deviation of 0.3 was added to g_1 to make it a DOvS problem. Considering the difference between the global minimum (-2) and the second best solution (-1.917), the noise is quite

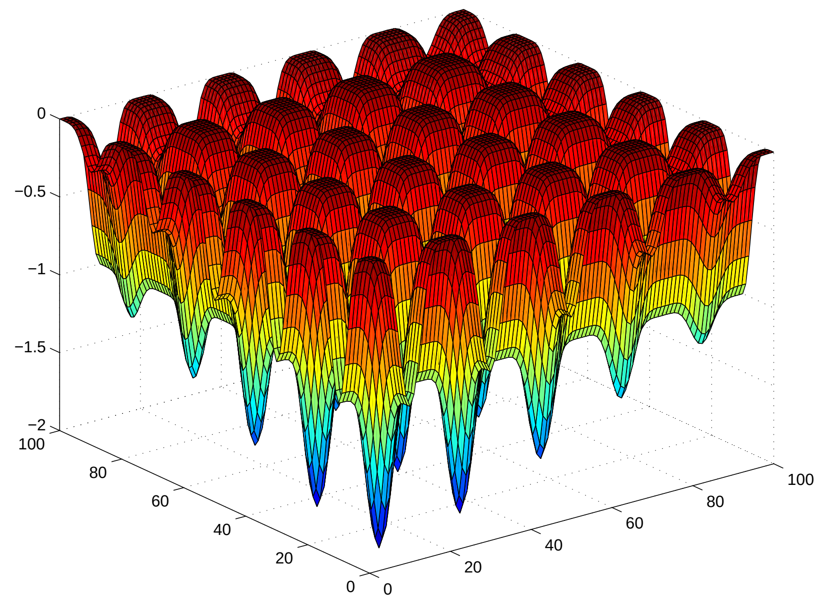

Fig. 2. Test function g_1 ($0 \leq x_1, x_2 \leq 100$).

significant. The feasible solution space is $0 \leq x_1 \leq 100, 0 \leq x_2 \leq 100$ and $x_1, x_2 \in \mathcal{Z}^+$.

The second test problem is the singular function of Hong [2004]. To facilitate the use of a log scale in our performance plots, we add one to the original singular function so now it is

$$g_2(x_1, x_2, x_3, x_4) = (x_1 + 10x_2)^2 + 5(x_3 - x_4)^2 + (x_2 - 2x_3)^4 + 10(x_1 - x_4)^4 + 1. \quad (3)$$

When only integer solutions are considered, this function has three local minima: $(0, 0, 0, 0)$ with $g_2(0, 0, 0, 0) = 1$; $(1, 0, 0, 1)$ with $g_2(1, 0, 0, 1) = 7$; and $(-1, 0, 0, -1)$ with $g_2(-1, 0, 0, -1) = 7$. This test problem represents a response surface with a small number of tightly clustered local optima.

We consider two versions of the stochastic noise for this surface: (a) normally distributed noise with zero mean and standard deviation $\sqrt{g_2(x_1, x_2, x_3, x_4)}$ (noise decreases as we approach the global optimum and (b) normally distributed noise with zero mean and standard deviation 30 (substantial noise near the global optimum). The feasible solution space is $-100 \leq x_i \leq 100, x_i \in \mathcal{Z}^+,\ i = 1, 2, \ldots, 4$.

The third test problem is a three-stage flow line with finite buffer storage space in front of stations 2 and 3 (denoted by x_4 and x_5) and an infinite number of jobs in front of station 1 (see Buzacott and Shantikumar [1993], Pichitlamken and Nelson [2003]). There is a single server at each station, and the service time at station i is exponentially distributed with service rate $x_i, i = 1, 2, 3$. If the buffer of station i is full, then station $i-1$ is blocked and a finished job cannot be released from station $i - 1$. The total buffer space and the service rates are limited. The objective is to find a buffer allocation and service rates such that the steady-state throughput is maximized. The constraints are $x_1 + x_2 + x_3 \leq 20, x_4 + x_5 = 20, 1 \leq x_i \leq 20$ and

$x_i \in \mathcal{Z}^+$ for $i = 1, 2, \ldots, 5$. This gives $21,660$ feasible solutions. The local optima happen to also be global optima: $(6, 7, 7, 12, 8)$ and $(7, 7, 6, 8, 12)$ with steady-state throughput 5.776. The average throughput from time 50 to time 1000 constitutes a replication. This test problem represents a realistic DOvS problem where we actually know the answer.

The fourth test problem is an inventory management problem with dynamic consumer substitution. The model that we optimize is adapted from Mahajan and Van Ryzin [2001] and it represents a difficult research problem for which the optimal solutions are not known.

The model considers a one-shot inventory stocking decision faced by a retailer for v product variants at the beginning of a season; no inventory replenishment happens in the model, and there is no salvage value for the products. It is assumed that each individual consumer chooses an available product with the highest utility, which may be a no-purchase option. Pricing is assumed to be an exogenous decision. Variant j's unit price is given as p_j and the unit cost is c_j.

The number of customers is Poisson with mean 1000, and the customer's choice behavior is modeled by the widely used Multinomial Logit model (MNL). Briefly, variant j's utility to customer t is $U_{tj} = a_j - p_j + \xi_{tj}$, $j = 1, 2, \ldots, v$, where a_j is variant j's quality factor, and ξ_{tj} is a random variable having an extreme value (Type I) distribution. See Mahajan and Van Ryzin [2001] for complete details.

In Mahajan and Van Ryzin [2001], a continuous relaxation of the problem was numerically solved via a sample path algorithm, which is essentially a continuous-variable OvS method. Although the continuous version of this problem is of some practical interest for commodities like gasoline, it is not appropriate for some situations. We apply ISC and OptQuest to the original integer-ordered problem. There are 6 products and one no-purchase option.

The original problem was unconstrained. However, since demands are not infinite, there exist reasonable upper bounds that we may impose on decision variables without worrying about cutting out globally optimal solutions. In the numerical experiment we ran, which is based on Example 1 in Mahajan and Van Ryzin [2001], there are 6 integer variables, each ranging from 0 to 500. Therefore, the size of the feasible solution space is $500^6 \approx 1.6 \times 10^{16}$ solutions.

The fifth test problem was designed to illustrate the impact of dimension. Let

$$g_5(x_1, x_2, \ldots, x_d) = -\beta \exp\left\{ -\gamma \sum_{j=1}^{d} j(x_j - \xi^\star)^2 \right\}$$

where we set $\gamma = 0.001$, $\beta = 10000$ and $\xi^\star = 0$. This gives a surface shaped like an inverted multivariate normal density function with a single globally optimal solution at $\mathbf{x} = (\xi^\star, \xi^\star, \ldots, \xi^\star)$ having value -10,000. The feasible region is the hyperbox defined by

$$x_j \in \left\{ -\frac{m^{1/d}}{2}, \frac{m^{1/d}}{2} \right\}$$

for $j = 1, 2, \ldots, d$ with $m = 10^{20}$, where we round the bounds to the nearest integer if necessary. Defining the feasible region in this way keeps the number of feasible solutions (nearly) the same as dimension changes, allowing us to isolate the impact of dimension from that of number of feasible solutions. To make the problem stochastic we added normally distributed noise with standard deviation $0.3 \times |g_5(\mathbf{x})|$ so that noise increases near the optimal solution.

5.2 Tuning the DOvS Algorithms

Both OptQuest and ISC have options or settings that can enhance or degrade their performance for any particular problem. While developing ISC we tried to identify good general-purpose choices. Those that seemed to have the most impact are as follows:

—In the Global Phase, the key settings are related to the transition rules. We found that a good choice for the number of NGA generations without an improvement to trigger a transition from Global to Local Phases was $T_G = 3$ generations, and we used that in all experiments reported here. We also found that the dominant niche rule—which triggers a transition if there appears to be only a single niche—could have a substantial impact on performance. Therefore, we report results both with and without this rule.

—In the Local Phase, we found that an adaptive SAR (based on OCBA ideas, see the electronic appendix) typically worked better than a fixed incremental allocation rule for dimension $d \leq 10$, so for the first four test problems we use the adaptive rule. We also tested a modification of COMPASS that invokes a sequential statistical test whenever the data indicate that COMPASS should backtrack from the current most promising area to a surrounding area. This test attempts to prevent COMPASS from incorrectly terminating progress toward a locally optimal solution, but it does so at the cost of substantial additional sampling to control the power of the test. We report results without the backtracking test since it did not appear to be helpful.

Note that none of these settings affects the asymptotic convergence guarantees offered by ISC, and we did not tune the parameters for the test problems reported here.

For OptQuest, the most critical setting is how it allocates replications to solutions. In all experiments reported here we used both the default setting and the adaptive allocation option: The default setting allocates 3 replications per solution and no more. The adaptive setting allocates at least 3, but no more than 100 replications to each solution. OptQuest stops adding replications within this range to a solution $\mathbf{x}$ when the length of a 95% confidence interval for $g(\mathbf{x})$ is less than 5% of the sample mean, or if the confidence interval for $g(\mathbf{x})$ does not overlap the 95% confidence interval for the current sample best solution's mean. For the inventory management problem we raised OptQuest's minimum number of replications from 3 to 20 for both the default and adaptive settings because the problem was simply too noisy for OptQuest to make any progress with 3 replications per solution.

After terminating its search, OptQuest includes an option to take the three solutions with the sample best performance and apply a R&S procedure to discover which of them is the true best, with high confidence. This R&S procedure plays a different role than the Clean-up Phase of ISC, since it is not used to terminate OptQuest, and the three solutions chosen for evaluation are not necessarily expected to be locally optimal. For these reasons we do not report results from using

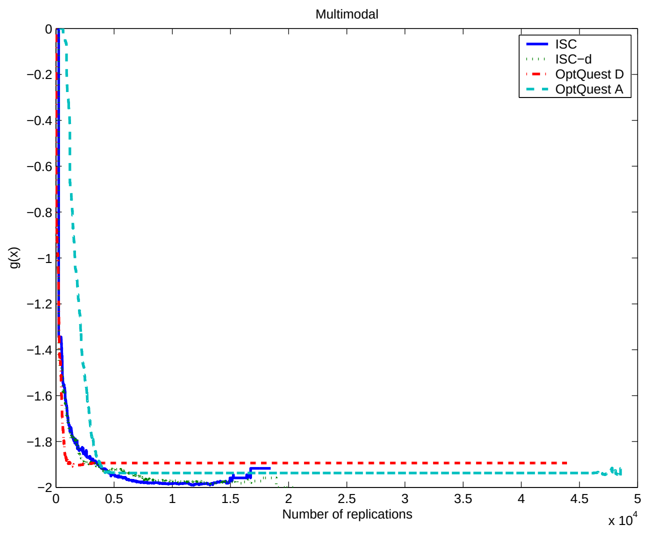

Fig. 3. Performance plot for the multimodal function.

the R&S option of OptQuest.

5.3 Empirical Results

For the first two test problems the performance plot is the average for fifty trials of the true expected value of the solution that each DOvS algorithm thinks is the best solution found so far as a function of the number of replications consumed. Recall that OptQuest is given a budget substantially larger than the maximum number of replications consumed by any trial of ISC. In the legends on the plots, "d" refers to ISC with the dominant niche transition rule turned on, while "D" and "A" are versions of OptQuest with the default and adaptive sample allocation rules, respectively.

A performance plot for the multimodal function is shown in Figure 3. Clearly OptQuest with the default rule makes more rapid initial progress than ISC, but then becomes trapped in inferior solutions. Notice also that since ISC is self stopping, and stops after different numbers of replications on each of the fifty trials, fewer trial results are being averaged when moving from left to right on the replications axis. This is why the performance of ISC can appear to get worse or better near the end of the plot.

A performance plot for the singular function with large noise is shown in Figure 4. Here OptQuest with the adaptive allocation rule is clearly superior, while OptQuest with the default allocation rule is inferior, to ISC. In this problem ISC tends to terminate at or near one of the local optima, but not always the global optimum. Recall that this problem had three locally optimal solutions in close proximity that are likely not separated by the NGA. When noise that decreases near the optimal solution is added, OptQuest's performance is even better.

A performance plot for the flowline design problem is shown in Figure 5. Because this problem executed much more slowly, and the overhead for both ISC

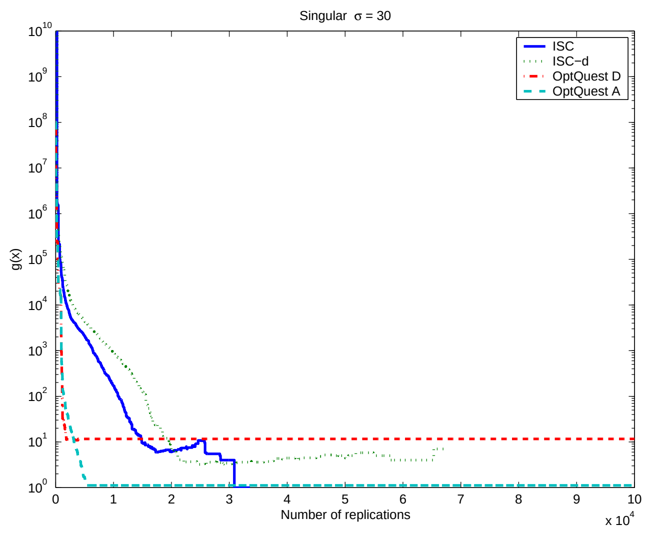

Fig. 4. Performance plot for the singular function.

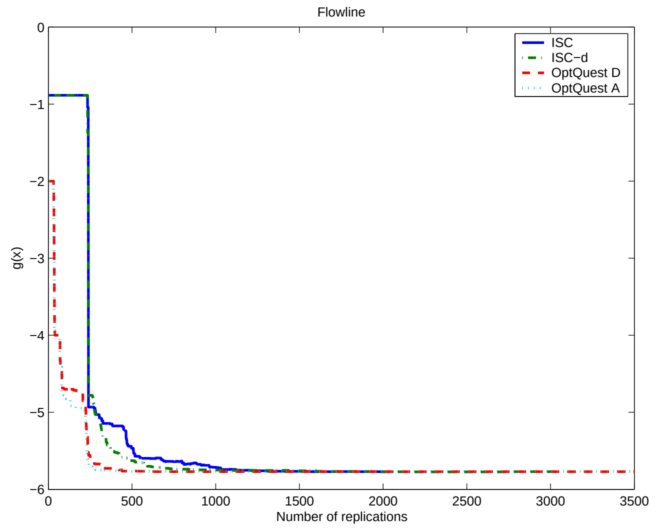

Fig. 5. Performance plot for the flowline problem.

and OptQuest was more substantial, the performance curves are the average of 25 and 10 trials for ISC and OptQuest, respectively. For this realistic problem OptQuest makes faster initial progress, but both DOvS algorithms converge to a global optimal solution, with ISC knowing when to stop.

While the flowline design problem is a relatively easy DOvS problem, the inventory management problem is a particularly difficult one: The response is quite noisy, and $g(\mathbf{x})$ appears to be relatively flat near the good solutions (recall that

Table I. Ten Trials of ISC and OptQuest on the Inventory Management Problem and the Average Deviation of the Estimated Solution Value from "True"

	ISC			OptQuest: Default			OptQuest: Adaptive		
Trial	$\bar{G}(\widehat{\mathbf{x}}^*)$	"True"	SE	$\bar{G}(\widehat{\mathbf{x}}^*)$	"True"	SE	$\bar{G}(\widehat{\mathbf{x}}^*)$	"True"	SE
1	1891	1874	1.6	1921	1858	1.5	1948	1874	1.5
2	1892	1873	1.6	1950	1875	1.6	1942	1868	1.6
3	1894	1870	1.4	1934	1870	1.6	1962	1877	1.7
4	1877	1871	1.7	1950	1873	1.6	1935	1871	1.4
5	1882	1869	1.4	1933	1859	1.3	1938	1859	1.6
6	1878	1860	1.7	1964	1852	1.8	1964	1869	1.5
7	1896	1876	1.6	1926	1861	1.4	1956	1872	1.5
8	1895	1876	1.5	1958	1872	1.6	1950	1863	1.7
9	1892	1878	1.5	1929	1855	1.7	1945	1867	1.6
10	1890	1876	1.4	1940	1867	1.6	1956	1866	1.6
Ave Dev	16			76			81		

the true response surface is not known). To compare ISC and OptQuest, we ran each for 10 trials, and at the end of each trial did further extensive simulation of its chosen best solution to estimate $g(\widehat{\mathbf{x}}^*)$ very precisely. For this problem, we only ran ISC without the dominant niche test, and OptQuest was given a budget equal to the average number of replications that ISC used (approximately 100,000).

Table I shows the results. For each algorithm, the $\bar{G}(\widehat{\mathbf{x}}^*)$ column shows the estimated value of the sample best solution when the algorithm terminated, while the column labeled "True" is an estimate of this solution's true expected value based on 10,000 replications (the standard error of this estimate is shown in the SE column). While ISC found slightly better solutions—bigger is better in this problem, and the average solution values for ISC, OptQuest (default allocation) and OptQuest (adaptive allocation) were $1872, 1864$ and 1869, respectively—of equal importance is that ISC's estimate of the performance of the chosen solution is much closer to its "True" value: Both ISC and OptQuest tend to overestimate the true objective function value, as expected in a maximization problem, but ISC's estimate is on average only $16 < \delta_C = 20$ greater, as guaranteed by ISC (only one deviation was greater than 20, and that was 24). This demonstrates the benefit of controlling the precision of the final estimate in the Clean-up Phase.

Statistically guaranteed convergence becomes more and more difficult to obtain as problems dimensionality d increases, and therefore the efficiency of ISC is affected more by dimension than by the number of feasible solutions. The fifth test problem illustrates this issue. Although we report results for a specific test problem, some general lessons learned from looking at a number of high-dimensional test problems are the following:

—The overhead for constraint pruning in the Local (COMPASS) Phase becomes significant as dimension increases, although constraint pruning is still very important. For dimension $d \geq 10$ we recommend pruning less frequently than the ISC default value of 5.

—As dimension increases it improves efficiency if the Local (COMPASS) Phase defers adaptively allocating replications until an apparently locally optimal solution is discovered, at which point the stopping test takes over sample allocation. Therefore, we recommend turning off OCBA so that the sample allocation scheme increases sampling just fast enough to guarantee convergence when $d \geq 10$.

—ISC is guaranteed to stop eventually. However, including a maximum budget is important in high dimension since it is impossible to know in advance how long it might take to stop and at present it is not possible to gracefully interrupt ISC.

For ISC, we set $\delta_C = 10 \approx \beta(1 - e^{-\gamma})$, the performance gap between the optimal solution and its neighbors; did constraint pruning only every 50 iterations; turned off the OCBA adaptive sample allocation; and otherwise used default ISC settings. OptQuest was run using its adaptive sample allocation setting. We discovered that OptQuest always starts by evaluating the feasible solution at the center of the feasible region, which means in our problem it starts with the optimal solution. Therefore, for OptQuest we shifted the position of the optimal solution a bit so that at least one iteration would be required to find it by chance alone. We ran 5 macroreplications of each algorithm for dimensions $d = 5, 10, 15$ and 20. We again ran ISC first, then gave OptQuest a budget larger than the largest number of replications used by ISC. Here is what we observed dimension by dimension:

$d = 5$. Both ISC and OptQuest quickly found the optimal solution.

$d = 10$. ISC found the optimal solution on all 5 trials, taking approximately 1 minute to do so and a maximum of 27,966 replications. OptQuest used a budget of 30,000 replications in approximately 0.2 minutes (showing lower algorithm overhead than ISC). In 3 of 5 trials it found the optimal solution, and in the other two trials it selected another good solution (within 1% of optimal).

$d = 15$. ISC's algorithm overhead becomes substantial, taking approximately 20 times longer to exhaust a maximum of 172,348 replications than the 180,000 budget given to OptQuest. ISC found the optimal in all 5 trials; OptQuest found it in 2 trials and otherwise chose solutions within about 1% of optimal.

$d = 20$. ISC spent about 30 times as long as OptQuest to exhaust a maximum of 359,054 replications; OptQuest was given a budget of 360,000. ISC again found the optimal solution on each trial while OptQuest discovered it on 2 of them and otherwise chose solutions within 1% of optimal.

Some general statements about the results at all dimensions are that (a) ISC's terminal confidence interval on the value of the selected solution (which was always the optimal solution) always contained its true expected value; (b) OptQuest was distracted from the optimal solution when it visited an inferior solution that got a particularly favorable sample, and as a result OptQuest's estimate of the value of its selected solution was better than its true value, often by several times δ_C; and (c) OptQuest appeared to make good use of insight it gained about the shape of the response surface (which is particularly nice in this example), a feature that would help ISC.

Across these five cases, ISC demonstrated that it could be competitive with OptQuest, despite enforcing finite-sample and limiting guarantees. *This performance encourages us to believe that provable guarantees and practical usefulness can be compatible.* Of course, there will be problems on which a severe price is paid to attain provable convergence, and robust metaheuristics that "break the rules" will provide much better solutions. We believe that this will most often be the case when the available time for optimization is tight or dimension is high so that rapid solution improvement is essential. Convergence tends to slow progress.

6. CONCLUSIONS

We have presented Industrial Strength COMPASS, an algorithm and software for DOvS that has provable asymptotic performance, competitive finite-time performance and valid statistical inference at termination. Our focus in this paper has been on using ISC in the same way that the commercial products are typically used: with no information about the problem at hand other than the definition of the objective function, decision variables and constraints, and a link to the simulation model. Only a single parameter, the error tolerance on the estimate of the optimal solution δ_C, *must* be set by the user; all of the other parameters have default values that we obtained during development of ISC (the electronic appendix contains tables of all of the parameters and their default values).

Like OptQuest, ISC has parameters that can be adjusted in ways that may make it more effective on a particular problem. Of these tunable parameters, the most critical determine the aggressiveness of the Global Phase in allocating replications to distinguish good solutions from bad, and how demanding the Local Phase is before declaring a solution to be locally optimal. Our defaults lean toward a relatively passive Global Phase that explores broadly and applies little effort to each solution visited, and a relatively stringent test for local optimality in the Local Phase. These choices could waste time exploring the entire surface too broadly when a little additional effort might uncover a friendly (e.g., unimodal) response surface quickly, or by repeatedly applying time-consuming local optimality tests in an area with many close solutions. What we would like to have are not better ways for the user to set these parameters, but rather adaptive methods for tuning them during the optimization run as more and more is learned about the particular DOvS problem at hand. This is a fruitful area for future research.

Certain types of a priori information can be exploited by the user to make ISC more effective, however:

(1) If the function $g(\mathbf{x})$ is known or strongly suspected to be unimodal, then convergence is likely to be more rapid if the Global (NGA) Phase is skipped, going directly to the Local (COMPASS) Phase from an initial sample of solutions. COMPASS alone converges very quickly for unimodal functions.

(2) If the simulation budget is tight and the goal is to find a pretty good solution quickly, then the Local (COMPASS) Phase could be skipped, using an effort-based transition rule to terminate the NGA, which explores the feasible region broadly, then moving directly to Clean Up to obtain statistical confidence in the selected solution. This is similar to the algorithm in Boesel et al. [2003a].

(3) Known good starting solutions or a systematic experimental design can be used to seed the NGA or COMPASS.

(4) For problems with dimension greater than 10, pruning constraints less frequently and using the base (not OCBA) sample allocation rule is recommended.

An overriding objective in the development of ISC was to have an algorithm that could stop on its own with well-defined guarantees. This objective led us to search for locally optimal solutions and use R&S procedures to establish and compare them. ISC is not likely to work well under a very strict and tightly time-constrained budget—since the user would have to provide arbitrary effort-based transition rules for each phase—nor does it exploit an essentially unlimited budget—the NGA is probably not the choice of globally convergent algorithm we would make if that had been our goal. We contend that the time required to develop a detailed simulation model, and the impact of the decision that will be based on it, argue in favor of a liberal, but not infinite, computing budget in most situations.

In our effort to bridge the gap between research and practice, we attempted to resolve key implementation issues often overlooked in the research literature. Of these, we can claim some success in incorporating constraints, developing meaningful transition and stopping rules, and adaptively allocating simulation effort without giving up convergence or statistical guarantees. Open issues not addressed include incorporating stochastic constraints, multiple objectives, and mixed (integer, continuous and categorical) decision variables. All of these need solutions before we can claim a truly general-purpose OvS algorithm.

APPENDIX

In this Appendix we prove any new results not available in the literature (and therefore cited in the paper).

A. GLOBAL CONVERGENCE OF THE NGA

We require that the NGA satisfies the following requirements, where the specific values may differ from problem to problem:

(1) In the Selection and Mating steps of iteration k, any solution sampled in iteration $k-1$ has a probability at least $p_1 > 0$ of being selected for crossover.

(2) In the Crossover step, any parent solution has a probability at least $p_2 > 0$ of being generated as an offspring.

(3) In the Mutation step, any offspring generated in the crossover has a probability at least $p_3 > 0$ of being selected for mutation.

(4) In the Mutation step, each coordinate may be selected with a probability at least $p_4 > 0$, and each feasible solution along the coordinate may be selected with a probability at least $p_5 > 0$.

In this subsection, we show that the NGA converges to a global optimal solution as the number of iterations goes to infinity.

The NGA allocates at least $\Delta n \geq \log_{10}(\text{number of iterations})$ observations to any previously simulated solution and $n_0 \geq 1$ observations to any new solution in the Evaluate Solutions step. If we can show that all feasible solutions are visited infinitely often as the number of iterations goes to infinity, then the global convergence follows directly from the strong law of large numbers.

For any two solutions $\mathbf{x}_1, \mathbf{x}_2 \in \Theta$, there exists a path in Θ, denoted $\mathbf{x}_1, \mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_{|\Theta|-1}, \mathbf{x}_2$, such that any two consecutive solutions in the path differ at most by one coordinate. Let S_k denote the set of solutions sampled in iteration k. If $\mathbf{x}_1 \in S_k$, then it is selected for crossover in iteration $k+1$ with a probability at least p_1. If it is selected for crossover, it is generated as an offspring with a probability at least

p_2; then it is selected for mutation with a probability at least p_3. If it is selected for mutation, then $\mathbf{y}_1$ will be sampled in iteration $k+1$ with a probability at least $p_4 p_5$. Therefore, $\Pr\{\mathbf{y}_1 \in S_{k+1} | \mathbf{x}_1 \in S_k\} \geq \prod_{i=1}^5 p_i$. Then

$$\begin{aligned}
&\Pr\{\mathbf{x}_2 \in S_{k+|\Theta|} | \mathbf{x}_1 \in S_k\} \\
&= \Pr\{\mathbf{y}_1 \in S_{k+1} | \mathbf{x}_1 \in S_k\} \Pr\{\mathbf{y}_2 \in S_{k+2} | \mathbf{y}_1 \in S_{k+1}\} \\
&\quad \times \cdots \times \Pr\{\mathbf{x}_2 \in S_{k+|\Theta|} | \mathbf{y}_{|\Theta|-1} \in S_{k+|\Theta|-1}\} \\
&\geq \left(\prod_{i=1}^5 p_i \right)^{|\Theta|} . \qquad (4)
\end{aligned}$$

Since Equation (4) holds for any pair of solutions $\mathbf{x}_1, \mathbf{x}_2 \in \Theta$, then any solution may be visited in any $|\Theta|$ consecutive iterations with a positive probability that is bounded away from 0. Therefore, all solutions in Θ will be visited infinitely often as the number of iterations goes to infinity. Therefore, the NGA is globally convergent.

B. CLEAN-UP INFERENCE

Let $\tilde{\mathbf{x}}^* = \operatorname{argmin}_{\mathbf{x} \in \mathcal{L}} g(\mathbf{x})$. ISC applies the "clean-up" procedure of Boesel et al. [2003b] to the solutions in $\mathcal{L}$, starting with the data that have already been obtained prior to the Clean-up Phase. This procedure obtains additional replications as needed to guarantee to select $\tilde{\mathbf{x}}^*$ with probability $\geq 1 - \alpha_C$ whenever the data are normally distributed, the solutions are simulated independently and $g(\tilde{\mathbf{x}}^*) - \min_{\mathbf{x} \in \mathcal{L}, \mathbf{x} \neq \tilde{\mathbf{x}}^*} g(\mathbf{x}) \leq -\delta_C$ where $1/2 < 1 - \alpha_C < 1$ and $\delta_C > 0$ are user-specified parameters ($1 - \alpha_C$ defaults to 0.95 in ISC). We prove our results under the independence assumption because the complex sequential sampling that occurs in the Global and Local Phases makes it difficult to synchronize random numbers so as to effectively use common random numbers [Law and Kelton 2000], so we believe this is a reasonable approximation even if common random numbers are used.

Let $\mathbf{x}_B$ be the solution selected by this procedure. Then we will show that, whatever the values of the $g(\mathbf{x}), \mathbf{x} \in \mathcal{L}$, $\Pr\left\{ g(\mathbf{x}_B) \in \bar{G}(\mathbf{x}_B) \pm \delta_C \right\} \geq 1 - \alpha_C/2$. This is a new result, not proven in Boesel et al. [2003b].

Let $b = \delta_C t / h$ where h is the sampling constant used in the clean-up procedure (specifically $h = h(2, (1 - \alpha_C/2)^{1/(|\mathcal{L}|-1)}, \underline{n})$ is Rinott's constant in the special case of 2 solutions, confidence level $(1 - \alpha_C/2)^{1/(|\mathcal{L}|-1)}$ and $\underline{n}$ degrees of freedom; see Boesel et al. [2003b]), $t = t_{1-(1-\frac{1}{2}(1-\alpha_C)^{1/\mathcal{L}}), \underline{n}-1}$, and $\underline{n}$ is the minimum number of replications any solution in $\mathcal{L}$ received prior to transition to the Clean-up Phase. Notice that this t value is not the t-value used in the screening procedure in Boesel et al. [2003b], but rather is an artifact used to show that the final confidence interval is valid.

The clean-up procedure requires each solution $\mathbf{x} \in \mathcal{L}$ to receive

$$N_C(\mathbf{x}) = \min \left\{ N_C^{(0)}(\mathbf{x}), \left\lceil \frac{h^2 S^2(\mathbf{x})}{\delta_C^2} \right\rceil \right\}$$

replications, where $N_C^{(0)}(\mathbf{x})$ is the number of replications solution $\mathbf{x}$ received prior to the Clean-up Phase and $S^2(\mathbf{x})$ is the sample variance of these replications. Notice that

$$\begin{aligned}\Pr\left\{g(\mathbf{x}_B) \in \bar{G}(\mathbf{x}_B) \pm b\right\} &\geq \Pr\left\{g(\mathbf{x}) \in \bar{G}(\mathbf{x}) \pm b, \forall \mathbf{x} \in \mathcal{L}\right\} \\ &\geq \prod_{\mathbf{x}\in\mathcal{L}} \Pr\left\{g(\mathbf{x}) \in \bar{G}(\mathbf{x}) \pm b\right\} \qquad (5)\end{aligned}$$

where (5) follows because the solutions are simulated independently and the probability on the right-hand side is with respect to an experiment in which all solutions in $\mathbf{x} \in \mathcal{L}$ receive $N(\mathbf{x})$ samples. But notice that

$$\Pr\left\{g(\mathbf{x}) \in \bar{G}(\mathbf{x}) \pm b\right\} = \Pr\left\{\frac{\bar{G}(\mathbf{x}) - g(\mathbf{x})}{S(\mathbf{x})/\sqrt{N(\mathbf{x})}} \in \pm\frac{\sqrt{N(\mathbf{x})}}{S(\mathbf{x})}b\right\}$$

and

$$\frac{\sqrt{N(\mathbf{x})}}{S(\mathbf{x})}b \geq \frac{hS(\mathbf{x})}{\delta}\frac{b}{S(\mathbf{x})} = t.$$

Therefore, $\Pr\left\{g(\mathbf{x}) \in \bar{G}(\mathbf{x}) \pm b\right\} \geq (1-\alpha_C/2)^{1/k}$ and $\Pr\left\{g(\mathbf{x}_B) \in \bar{G}(\mathbf{x}_B) \pm b\right\} \geq 1-\alpha_C/2$. The result that we want then follows by noting that $t/h \leq 1$, so $b \leq \delta_C$.

ELECTRONIC APPENDIX

The electronic appendix for this article can be accessed in the ACM Digital Library by visiting the following URL: `http://www.acm.org/pubs/citations/journals/jn/20YY-V-N/p1-URLend`.

ACKNOWLEDGMENTS

The authors gratefully acknowledge OptTek Systems, Inc. for providing the OptQuest engine to use in our research, and the clarifying comments of the Editor, Area Editor, Associate Editor and referees.

References

Alrefaei, M. H., and Andradóttir, S. 1999. A simulated annealing algorithm with constant temperature for discrete stochastic optimization. *Management Science 45*, 748–764.

Andradóttir, S. 1995. A method for discrete stochastic optimization. *Management Science 41*, 1946–1961.

Andradóttir, S. 1999. Accelerating the convergence of random search methods for discrete stochastic optimization. *ACM TOMACS 9*, 349–380.

Andradóttir, S. 2006. Simulation optimization with countably infinite feasible regions: Efficiency and convergence. *ACM TOMACS 16*, 357–374.

Andradóttir, S., and Nelson, B. L. 2004. Selection error control and statistical inference for simulation optimization. *Proceedings of the 2004 NSF Design, Service, and Manufacturing Grantees and Research Conference*, Dallas, Texas.

Boesel, J. 1999. *Search and Selection for Large-Scale Stochastic Optimization.* Doctoral dissertation, Department of IEMS, Northwestern University, Evanston, IL.

Boesel, J., Nelson, B. L., and Ishii, N. 2003a. A framework for simulation-optimization software. *IIE Transactions 35*, 221–230.

Boesel, J., Nelson, B. L., and Kim, S.-H. 2003b. Using ranking and selection to 'clean up' after simulation optimization. *Operations Research 51*, 814-825.

Buzacott, J. A., and Shantikumar. J. G. 1993. *Stochastic Models of Manufacturing Systems*. Prentice-Hall, Englewood Cliffs, NJ.

Chen, C.-H., Lin, J., Yücesan, E., and Chick, S.E. 2000. Simulation budget allocation for further enhancing the efficiency of ordinal optimization. *Discrete Event Dynamic Systems, Theory and Applications 10*, 251–270.

Chick, S. E., and Inoue, K. 2001. New two-stage and sequential procedures for selecting the best simulated system. *Operations Research 49*, 732–743.

Fu, M. C. 2002. Optimization for simulation: Theory vs. practice. *INFORMS Journal on Computing 14*, 192–215.

Fu. M. C., Glover, F. W., and April, J. 2005. Simulation optimization: A review, new developments, and applications. In *Proceedings of the 2005 Winter Simulation Conference*, IEEE, 83–95.

Gong, W.-B., Ho, Y.-C., and Zhai, W. 1999. Stochastic comparison algorithm for discrete optimization with estimation. *SIAM Journal on Optimization 10*, 384–404.

Ho, Y. C., Sreenivas, R., and Vakili, P. 1992. Ordinal optimization of discrete event dynamic systems. *Journal of Discrete Event Dynamic Systems 2*, 61–88.

Hong, L. 2004. *Discrete Optimization via Simulation: Algorithms and Error Control.* Doctoral dissertation, Department of IEMS, Northwestern University, Evanston, IL.

Hong, L. J., and Nelson, B. L. 2006. Discrete optimization via simulation using COMPASS. *Operations Research 54*, 115–129.

Hong, L. J., and Nelson, B. L. 2007a. A framework for locally convergent random search algorithms for discrete optimization via simulation. *ACM TOMACS 17*, 19/1–19/22.

Hong, L. J., and Nelson, B. L. 2007b. Selecting the best system when systems are revealed sequentially. *IIE Transactions 39*, 723–734.

Jin, Y. and Branke, J. 2005. Evolutionary optimization in uncertain environments - A survey. *IEEE Transactions on Evolutionary Computation 9*, 303–317.

Kim, S.-H. 2005. Comparison with a standard via fully sequential procedures. *ACM TOMACS 15*, 1–20.

Law, A.M., and Kelton, W. D. 2000. *Simulation Modeling and Analysis*, 3rd edition. McGraw-Hill, New York.

Lin, X., and Lee, L. H. 2006. A new approach to discrete stochastic optimization problems. *European Journal of Operational Research 172*, 761–782.

Mahajan, S., and Van Ryzin, G. 2001. Stocking retail assortments under dynamic consumer substitution. *Operations Research 49*, 334–351.

Miller, B. L., and Shaw, M. J. 1995. Genetic algorithms with dynamic niche sharing for multimodal function optimization. IlliGAL Report No. 95010, Illinois Genetic Algorithms Laboratory, University of Illinois at Champaign-Urbana.

Nance, R. E., and Sargent R. G. 2002. Perspectives on the evolution of simulation. *Operations Research 50*, 161–172.

Nelson, B. L., and Goldsman, D. 2001. Comparisons with a standard in simulation experiments. *Management Science 47*, 449–463.

Nelson, B. L., Swann, J., Goldsman, D., and Song, W. 2001. Simple procedures for selecting the best simulated system when the number of alternatives is large. *Operations Research 49*, 950–963.

Pichitlamken, J., and Nelson, B. L. 2003. A combined procedure for optimization via simulation. *ACM TOMACS 13*, 155–179.

Prudius, A. A., and Andradóttir, S. 2004. Simulation optimization using balanced explorative and exploitative search. In *Proceedings of the 2004 Winter Simulation Conference*, IEEE, 545–549.

Prudius, A. A., and Andradóttir, S. 2006. Balanced explorative and exploitative search with estimation for simulation optimization. Working Paper, School of ISyE, Georgia Tech, Atlanta, GA.

Sareni, B., and Krahenbuhl, L. 1998. Fitness sharing and niching methods revisited. *IEEE Transactions on Evolutionary Computation 2*, 97–106.

Shi, L., and Ólafsson, S. 2000. Nested partitions method for stochastic optimization. *Methodology and Computing in Applied Probability 2*, 271–291.

Telgen, J. 1983. Identifying redundant constraints and implicit equalities in systems of linear constraints. *Management Science 29*, 1209–1222.

Wolpert, D. A., and Macready, W. G. 1997. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation 1*, 67–82.

Yan, D., and Mukai, H. 1992. Stochastic discrete optimization. *SIAM Journal of Control and Optimization 30*, 594–612.

Received Month Year; revised Month Year; accepted Month Year

THIS DOCUMENT IS THE ONLINE-ONLY APPENDIX TO:

Industrial Strength COMPASS: A Comprehensive Algorithm and Software for Optimization via Simulation

JIE XU and BARRY L. NELSON
Northwestern University
and
L. JEFF HONG
The Hong Kong University of Science & Technology

ACM Journal Name, Vol. V, No. N, Month 20YY, Pages 1–28.

This online supplement documents the various algorithms that constitute Industrial Strength COMPASS. The source code is available from `www.ISCompass.net`.

A. NICHING GA

The purpose of the NGA is to find promising subregions of the solution space Θ. Each subregion has a *center*, which is the sample best solution in that subregion, and a handful of other solutions around the center. In the Local Phase, COMPASS is initialized with the center solution as the current sample best, and the surrounding solutions as the active solutions defining the most-promising area.

To be consistent with the GA literature, we use the term "individual" to mean a feasible solution and its "fitness" to mean the estimated expected value of the solution. The NGA described here is designed for minimization problems with integer-ordered decision variables.

A.1 The NGA Framework in ISC

In this section, we outline the NGA framework we use in ISC. Subsequent sections elaborate on important steps in the outline.

ALGORITHM 1 THE NGA FRAMEWORK IN ISC.

Step 0. Randomly sample m_G solutions from Θ. Assign n_0 replications to evaluate each solution (§A.2).

Step 1. Identify niche radius r and number of niches q and dynamically construct niches (§A.3).

Step 2. Apply niche-based transition rule. If conditions are met, stop the NGA (§A.12).

Step 3. Apply fitness sharing (§A.4).

Step 4. Apply grouping procedure using shared fitness values. Note that here more replications may be assigned (§A.6).

Step 5. Calculate group selection probabilities for each solution (§A.6).

Step 6. Select $m_G/2$ parents via Stochastic Universal Sampling, and the other $m_G/2$ parents via a mating restriction scheme (§§A.5–A.7).

Step 7. Do arithmetical crossover for each pair of parents (§A.8).

Step 8. Apply mutation operator to each solution generated by crossover (§A.9).

Step 10. Evaluate new solutions by assigning enough replications to each solution.

Step 11. Replace old population with new population and conserve niche centers (§A.10).

Step 12. Apply transition rules to decide if the NGA should stop (§A.12). If not, go back to Step 1.

A.2 Randomly Sampling Solutions

ISC can use either algorithm RMD [Hong and Nelson 2006] or algorithm MIX-D [Pichitlamken and Nelson 2003] to randomly sample integer solutions from the appropriate feasible space. RMD, which is the default algorithm, guarantees (nearly) uniform sampling of solutions provided the space is connected in the sense that any solution can be reached from any other solution by a sequence of coordinate direction moves. MIX-D provides the same guarantees for any convex feasible region. RMD is the default in ISC because it is substantially faster.

A.3 Identifying Niche Radius and Number of Niches

Key input parameters for the NGA are the number of niches to form, q, and the niche radius, r, which defines how close a solution must be to a niche center to be clustered in the same niche. Because our algorithm is intended to be general purpose, we have no way to know these values in advance, nor should we expect the user to be able to provide them. To solve this problem, we form a rough response surface model over Θ from which we extract estimates of q and r.

To explain the approach, we define for any subset of solutions $I \subseteq \Theta$ and particular solution $\mathbf{x}^* \in I$ the region

$$\mathcal{C}(\mathbf{x}^*) = \{\mathbf{x} : \mathbf{x} \in \Theta \text{ and } \|\mathbf{x} - \mathbf{x}^*\| \le \|\mathbf{x} - \mathbf{y}\|,\ \forall \mathbf{y} \in I \text{ and } \mathbf{y} \neq \mathbf{x}^*\}$$

which is essentially a COMPASS most-promising area centered at $\mathbf{x}^*$. Further, let

$$\mathcal{A}(\mathbf{x}^*) = \{j \in I : \mathbf{x}_j \text{ defines an active (non-redundant) constraint in } \mathcal{C}(\mathbf{x}^*)\}$$

which is determined via constraint pruning.

Algorithm 2 Niche radius identification on iteration k.

```
Step 1. Reindex the solutions in the current population so that $\bar{G}_k(\mathbf{x}_1) \le \bar{G}_k(\mathbf{x}_2) \le \cdots \le \bar{G}_k(\mathbf{x}_{m_G})$. Let $I = \{1, 2, \ldots, m_G\}$, $L = \emptyset$.
Step 2. For each $i \in I$, do
(1) If $\bar{G}_k(\mathbf{x}_i) \le \bar{G}_k(\mathbf{x}_j), \forall j \in \mathcal{A}(\mathbf{x}_i)$, then $L = L \cup \{i\}, I = I \setminus \{\mathcal{A}(\mathbf{x}_i)\}$.
(2) $I = I \setminus \{i\}$. Go to Step 1.
```

Step 3. Return $r = \min_{i,j \in L, i \neq j} \frac{1}{2}\|\mathbf{x}_i - \mathbf{x}_j\|$ *and* $q = |L|$.

The niche centers are the solutions in L, and for each solution $\mathbf{x}_i \in L$ its niche are those solutions no more than r distant from $\mathbf{x}_i$. Solutions that do not fall within a niche, if there are any, form a *non-niched* group. No solution will be in more than one niche.

A.4 Fitness Sharing

The NGA explores multiple regions in the solution space in parallel through the so-called fitness sharing mechanism. "Fitness" is a term used in the GA literature to mean the value of the objective function for a solution [e.g., Miller and Shaw 1995]. Here fitness is the sample average of all of the observations obtained on a solution. The basic idea is that if a niche is populated with too many solutions, those solutions should be given less chance to reproduce than they would have had in an ordinary GA, thus allowing solutions in less populated niches to have higher probabilities of being selected to generate new solutions. This can be done by discounting the fitness value of an individual by some factor that is dependent on how many solutions are in its niche. The discounted fitness values are then used in the NGA's selection, mating, crossover and mutation steps.

If a niche has n individuals including the niche center, then the shared fitness value f_{sh} of an individual within that niche with fitness value f is calculated as the discounted fitness value $f_{sh} = f/m$ if f is negative, and $f_{sh} = fm$ if f is positive, where m is the discounting factor, and we choose to let $m = n$. In ISC, the fitness value f of solution $\mathbf{x}_i$ is its sample mean $\bar{G}_k(\mathbf{x}_i)$.

If an individual $\mathbf{x}_i$ is within the non-niched group, its shared fitness is calculated similarly, except that the discounting factor m is now calculated by using a triangular sharing function, $m = \sum_{j=1}^{N} sh(\|\mathbf{x}_i - \mathbf{x}_j\|)$, and

$$sh(\|\mathbf{x}_i - \mathbf{x}_j\|) = \begin{cases} 1 - \frac{\|\mathbf{x}_i - \mathbf{x}_j\|}{r} & \text{if } \|\mathbf{x}_i - \mathbf{x}_j\| < r, \\ 0 & \text{otherwise.} \end{cases}$$

In ISC, since the NGA operates in a noisy environment, the selection step uses a statistical grouping procedure (see §A.6) that incorporates sample variance information. Therefore, the fitness sharing step should also modify the sample variances to reflect the fact that the fitness value is now the discounted sample mean. We choose a very simple implementation: If the sample variance of solution $\mathbf{x}_i$ is $S^2(\mathbf{x}_i)$, then the sample variance used in the grouping procedure is $S^2(\mathbf{x}_i)/m$ if $f_{sh} = f/m$, or $S^2(\mathbf{x}_i)m$ if $f_{sh} = fm$.

A.5 Selection

A selection method has two components: assigning selection probabilities to individuals, and selecting individuals based on those probabilities.

In the NGA implemented for ISC, we combine linear ranking with Stochastic Universal Sampling (SUS) [Baker 1987], which is a version of stratified sampling. According to Sareni and Krahenbuhl [1998], SUS is the best for NGA among several selection methods.

A linear ranking scheme assigns the i^{th}-best solution a selection probability

$$s_i = \frac{1}{m_G}\left(\eta - 2(\eta - 1)\left(\frac{i-1}{m_G - 1}\right)\right),$$

where η is a constant between 1 and 2. The larger η is, the more likely a better solution is to be selected (in GA terminology, the higher the selection pressure). The worst solution has a selection probability of $p_1 \equiv s_{m_G} = (2-\eta)/m_G$ and thus as long as $\eta < 2$, the worst solution has a positive probability to be selected as a parent for producing new solutions as required in Appendix A of the published paper. For a DOvS problem, we suggest choosing a relatively low value for η, such as 1.5.

Given selection probabilities for each individual, SUS constructs a roulette wheel where the area for each individual is proportional to its selection probability. Then the roulette wheel is spun once, and the pointer will rest on some individual which is then selected as a parent. Other individuals are selected by advancing the pointer at a regular spacing until it wraps back to its starting point. A modification here is that instead of selecting m_G parents, only $m_G/2$ parents are selected through SUS because the second parent needs to be selected by a mating selection scheme, which will be introduced shortly. So the pointer is advanced by a spacing of $2/m_G$.

A.6 GA Selection in the Presence of Noise

To cope with the stochastic aspect of DOvS problems, we adopted the selection procedure described in Boesel [1999]. The idea is to define a metric to characterize the variability in the assigned selection probability, $\widehat{s}_i$, in a stochastic environment as a squared deviation from the desired probability s_i. The expected sum of squared deviations due to misranking is defined as

$$\mathrm{E(SSD)} = \mathrm{E}\left\{\sum_{i=1}^{N}(s_i - \widehat{s}_i)^2\right\}.$$

It is proved in Boesel [1999] that if N individuals can be divided into g groups, and that all individuals in group i are known to be superior to all solutions in group $i+1$, and within group i all within-group orderings are equally likely to be observed, then E(SSD) is minimized when all members of each group are assigned their group's average selection probability. That is, $m_j^{(i)}$, the selection probability assigned to an individual j in group i of size N_i, is given by

$$m_j^{(i)} = \frac{1}{N_i}\sum_{j=1}^{N_i} s_j^{(i)}, \qquad (6)$$

where $s_j^{(i)}$ is the selection probability of individual j in group i, as calculated by the deterministic NGA selection scheme.

In ISC, we use the grouping procedure devised by Boesel [1999]. The algorithm groups solutions that are not statistically significantly better than each other into one group. If there are too few groups formed in the first round, the algorithm identifies one group that is easiest to break into more groups, adds replications to solutions within that group, and processes that group again.

Notice that the sample average $\bar{G}(\mathbf{x}_i)$ and sample variance $S^2(\mathbf{x}_i)$ are shared fitness values and sample variances. We use n_i to denote the number of replications that have been assigned to $\mathbf{x}_i$.

Algorithm 3 Grouping procedure for NGA.

Step 1. Given current population $(\bar{G}(\mathbf{x}_i), S^2(\mathbf{x}_i), n_i), i = 1, 2, \ldots, m_G$, *let* $\bar{n} = \frac{1}{m_G}\sum_{i=1}^{m_G} n_i$.

Step 2. Sort and reindex so that $\bar{G}(\mathbf{x}_1) \geq \bar{G}(\mathbf{x}_2) \geq \cdots \geq \bar{G}(\mathbf{x}_{m_G})$.

Step 3. Let

$$S^2 = \frac{1}{m_G}\sum_{i=1}^{m_G} S^2(\mathbf{x}_i)$$

$$R = Q^{1-\alpha_G}_{m_G, \sum_{i=1}^{m_G}(n_i-1)} S/\sqrt{\bar{n}},$$

where $Q^{1-\alpha_G}_{m_G, \sum_{i=1}^{m_G}(n_i-1)}$ *is the upper* $1-\alpha_G$ *quantile of the studentized range distribution [e.g., Miller 1981] with* m_G *d.f. and* $\sum_{i=1}^{m_G}(n_i-1)$ *samples.*

Step 4.

```
(1) $g = 1, i = 1$
(2) while $i < m_G$ do
    (a) $\mathcal{G}_g = \{i\}$, bottom $= i$, $i = i + 1$
    (b) while $\bar{G}(\mathbf{x}_{\text{bottom}}) - \bar{G}(\mathbf{x}_i) < R$ and $i < m_G$ do
        . $\mathcal{G}_g = \mathcal{G}_g \cup \{i\}$, $i = i + 1$
    (c) end while
    (d) $g = g + 1$
(3) end while
```

Step 5. If the number of groups formed $g < g_m$, *where* g_m *is the minimum number of groups to obtain, let*

$$\widehat{R}_i = \max_{j\in\mathcal{G}_i} \bar{G}(\mathbf{x}_j) - \min_{j\in\mathcal{G}_i} \bar{G}(\mathbf{x}_j), i = 1, 2, \ldots, g,$$

$$\widehat{m}_i = |\mathcal{G}_i|.$$

```
(1) For any group with $|\widehat{R}_i| < \delta_G$, do nothing further with $\mathcal{G}_i$.
(2) For groups with $|\widehat{R}_i| \geq \delta_G$, do the following
    (a) $\widehat{R} = \max \widehat{R}_i, \widehat{\ell} = \arg\max_{i=1,2,\ldots,g} \widehat{R}_i$.
    (b) Set
```

$$\widehat{n} = \left\lceil \frac{Q^2 S^2}{\widehat{R}^2} \right\rceil,$$

where

$$Q = Q^{1-\alpha_G}_{\widehat{m}_{\widehat{\ell}}, \sum_{j\in\mathcal{G}_{\widehat{\ell}}}(n_j-1)},$$

$$S^2 = \frac{1}{\widehat{m}_{\widehat{\ell}}} \sum_{j\in\mathcal{G}_{\widehat{\ell}}} S_j^2.$$

(*c*) *Obtain* $\max(\widehat{n} - n_j, 0)$ *more observations for* $j \in G_{\widehat{\ell}}$.
(*d*) *Apply fitness sharing.*
(*e*) *Go to Step 1 but with group* $\widehat{\ell}$ *as the population, and update* $g_m \leftarrow g_m - g + 1$.

One issue with the grouping procedure is how many groups we want to form at each iteration. Boesel [1999, pp. 66–68] pointed out that for linear ranking, each additional group formed provides less benefit and we may use as few as 3 groups.

A.7 Mating Restriction

The purpose of a mating restriction scheme is to maintain stable growth of niches during the evolution process by selecting individuals that are similar enough to each other to participate in crossover.

The mating restriction mechanism implemented here is proposed by Miller and Shaw [1995] and is known as dynamic inbreeding. Given the first parent $\mathbf{x}_i$ selected by the usual GA selection scheme, the inbreeding scheme works as follows:

ALGORITHM 4 DYNAMIC INBREEDING.

Step 1. Uniformly randomly sample m *individuals from the population.*

Step 2. From those individuals, select the best one that is within the same niche group as $\mathbf{x}_i$. *If there is no such an individual, select the one that is closest to* $\mathbf{x}_i$.

A.8 Crossover

Michalewicz [1996] described several crossover operators for GAs that optimize real-valued variables. Our experiments suggested that the arithmetical operator works best. Given two parents $\mathbf{x}_i$ and $\mathbf{x}_j$, the arithmetical crossover operator produces two offspring $\mathbf{x}'_i$ and $\mathbf{x}'_j$ by generating a number β from $U(0,1)$ distribution, and letting $\mathbf{x}'_i = \beta\mathbf{x}_i + (1-\beta)\mathbf{x}_j$, and $\mathbf{x}'_j = (1-\beta)\mathbf{x}_i + \beta\mathbf{x}_j$. Since this is an integer-ordered optimization problem, we need to round $\mathbf{x}'_i, \mathbf{x}'_j$ to integers.

One consequence of this rounding is that there is a nonzero probability that a parent will be generated as an offspring. Since the feasible solution region Θ is bounded, let $\xi > 0$ be the largest absolute value of a coordinate of any feasible solution $\mathbf{x}_j$. Then with a probability $p_2 = 1/(4\xi) > 0$, we have $\beta \le 1/(4\xi)$, in which case $\mathbf{x}'_i = \beta\mathbf{x}_i + (1-\beta)\mathbf{x}_j$ will be rounded to $\mathbf{x}_j$ (and similarly $\mathbf{x}'_j$ will be rounded to $\mathbf{x}_i$). The probability that a parent is generated as an offspring is thus at least p_2, which satisfies the requirement in our proof of the global convergence of NGA in Appendix A of the published paper.

The arithmetical operator automatically handles convex constraints. But after rounding offspring to integers, it is possible that the resulting solutions are infeasible. In that case, we simply keep the parents. Another drawback with the arithmetical operator is that there is no way to generate offspring outside of the convex combination of its two parents. To alleviate this problem, we use a relatively large mutation probability p_3.

A.9 Mutation

Two mutation operators are implemented: uniform and non-uniform [Michalewicz 1996]. When a solution is chosen to undergo a mutation according to the mutation probability p_3, a coordinate of that solution is randomly chosen with a probability of $p_4 = 1/d$ (where d is the dimension of $\mathbf{x}$), and its value is perturbed as follows:

—Uniform mutation: A line segment is drawn along the chosen coordinate direction, which hits boundaries of the feasible region at both ends. Then the new value of that coordinate is randomly chosen on that line segment. Since the feasible solution region Θ is bounded, we may assume that the largest absolute value a coordinate of any feasible solution $\mathbf{x}_j$ can have is $\xi > 0$, and thus each solution along that coordinate will be selected with a probability at least $p_5 = 1/2\xi > 0$, thus satisfying the requirement in Appendix A of the published paper.

—Non-uniform mutation: The basic idea is that as evolution continues, more fine tuning capability may be desirable to make the NGA more aggressive. Therefore, the mutated coordinate of $\mathbf{x}$, say x_i', is determined as

$$x_i' = \begin{cases} x_i + \Delta(k, \mathrm{right}(i) - x_i), & \text{with probability } 0.5, \\ x_i - \Delta(k, x_i - \mathrm{left}(i)), & \text{with probability } 0.5, \end{cases}$$

where k is the generation number, right(i) and left(i) are the end points of the line segment along the ith coordinate direction to the boundary of the feasible region, and $\Delta(k, y) = y \times U(0,1) \times (1-k/K)^b$. Here K is the maximum number of generations allowed and b is a parameter controlling the degree of non-uniformity.

Preliminary experiments showed that non-uniform mutation was better suited for our purpose. In the framework of ISC, however, it is not clear how to set K for non-uniform mutation because the NGA can be terminated at any generation. We simply chose K very large and if it does happen that the number of generations k exceeds K, we use a small number in place of $1 - k/K$. Therefore, in ISC the function $\Delta(k, y)$ for non-uniform mutation is

$$\Delta(k, y) = y \times U(0,1) \times \max\left\{0.005, \left(1 - \frac{k}{K}\right)^b\right\}.$$

A careful reader will notice that non-uniform mutation invalidates the NGA global convergence proof, so uniform mutation can be used to strictly adhere to the framework.

A.10 Replacement of Old Generation

Although it is not an essential component of the NGA, we choose to implement *elitism* as an option to speed up convergence and conserve old niche centers so as to stabilize the growth of niches; "elitism" means that the best individuals, in particular, niche centers, are always preserved through evolutions. Elitism is routinely used in GAs to speed up convergence. However, a consequence of using elitism is that it invalidates the global convergence property of the NGA, as proved in Appendix A of the published paper. If this is a concern, then the user can turn elitism off and thus preserve the global convergence property of the NGA. In our experience elitism helps, and since we transition from the Global Phase long before anything like global convergence occurs, this is a useful practical compromise.

Table II. Control Parameters for the NGA

Parameter	Functionality	Value
m_G	Population size	50
η	Linear ranking coefficient	$1 \leq \eta \leq 2$, 1.5
m	Mating restriction parameter (Algorithm 4)	10
p_3	Mutation probability	0.328
b	Nonuniform mutation parameter, the bigger b, the larger the non-uniformity	1.5
K	Used in non-uniform mutation	50
T_G	Maximum number of generations without continuous improvement for transition to local phase to occur	3
g_m	Minimum number of groups to form in Algorithm 3	3
δ_G	Minimum difference in fitness values to detect in Algorithm 3	User specified (default δ_C)
α_G	Grouping quantile parameter in Algorithm 3	0.1
n_0	Initial number of replications	User specified (default 5)
Δn	Incremental number of replications	$\log_{10}$(number of iterations)

Algorithm 5 Generation update.

Step 1. If elitism is turned on, do Step 2.

Step 2. Niche center conservation:

(1) Mark all individuals in the new population as unprocessed.

(2) For each old niche center $\mathbf{x}_i$, scan the unprocessed individuals in the new population. If the worst individual that is within the niche radius r from $\mathbf{x}_i$ is inferior to $\mathbf{x}_i$, then replace that individual with $\mathbf{x}_i$.

(3) If there is no such individual, replace the worst unprocessed individual in the new population with $\mathbf{x}_i$.

(4) Mark the modified individual as processed and go back to 2.

A.11 GA Control Parameters

Table II lists control parameters for the NGA in ISC.

A.12 Transition Rules

In practice, we terminate the NGA when conditions indicate that we have uncovered fertile regions for seeking locally optimal solutions. If the budget for the NGA is exhausted, we also terminate it and move on to the local search phase.

A.12.1 *Budget Rule.* If the user specifies a budget for the NGA in terms of number of replications, then at the beginning of each iteration the NGA checks whether that budget is exceeded. If it is, then the NGA is terminated; otherwise the NGA proceeds to generate a new population. Notice that this is a *soft* budget rule because as long as there are some replications available at the beginning of an iteration, the NGA will continue.

A.12.2 *Niche Rule.* If at any time there is only 1 niche, then go to the Local Phase. The philosophy is that when there is only one niche it suggests that there is likely only one local minimum and thus running local search alone is sufficient.

A.12.3 *Improvement Rule.* If the NGA does not achieve any improvement in T_G generations—that is, no apparently better solutions are found in any of T_G consecutive generations—then go to the Local Phase. This is similar to a regular GA stopping criterion except that we may want to set T_G smaller, say $T_G = 3$.

A.12.4 *Dominance Rule.* Let q be the number of niches and let $\mathbf{x}_{ij}$ be the j^{th} solution in niche $i, i = 1, 2, \ldots, q, j = 1, 2, \ldots, q_i$. Further let n_{ij} be the number of observations collected for $\mathbf{x}_{ij}$, let $n_i = \sum_{j=1}^{q_i} n_{ij}$, and let $\nu_i = n_i - q_i$.

If the solutions within one niche dominate all other niches (on the whole, not just the best individual), then we exploit it by going to the Local Phase.

The model used here is that the ℓ^{th} replication of solution $\mathbf{x}_{ij}$ gives a fitness value $Y_{ij\ell} = \mu_{ij} + \varepsilon_{ij\ell}$, where $\varepsilon_{ij\ell}$'s are i.i.d. and are distributed as $N(0, \sigma_i^2)$. We assume equal variance within a niche population. We denote by $t_{\nu,\beta}$ the β quantile of the t distribution with ν d.f. We further define

$$\begin{aligned} SSE_i &= \sum_{j=1}^{q_i}\sum_{\ell=1}^{n_{ij}}(Y_{ij\ell} - \bar{Y}_{ij\cdot})^2 \\ &= \sum_{j=1}^{q_i}\sum_{\ell=1}^{n_{ij}} Y_{ij\ell}^2 - \sum_{j=1}^{q_i} n_{ij} Y_{ij\ell}^2 \end{aligned}$$

where $\bar{Y}_{ij\cdot} = \sum_{\ell=1}^{n_{ij}} Y_{ij\ell}/n_{ij}$, and

$$\begin{aligned} \mu_i &= \frac{1}{q_i}\sum_{j=1}^{q_i}\mu_{ij}, \\ \widehat{\mu}_i &= \frac{1}{n_i}\sum_{j=1}^{q_i}\sum_{\ell=1}^{n_{ij}} Y_{ij\ell}. \end{aligned}$$

Let $I = \{i : \widehat{\mu}_i \geq \widehat{\mu}_j - \omega_{ij}, \forall j \neq i\}$, where

$$\begin{aligned} \omega &= \sqrt{\left(t_{\nu_i,\beta}^2 \frac{SSE_i}{\nu_i n_i} + t_{\nu_j,\beta}^2 \frac{SSE_j}{\nu_j n_j}\right)}, \\ \beta &= (1-\alpha)^{1/(q-1)}. \end{aligned}$$

Thus, the set I contains niches that are statistically better than others in terms of average quality of individuals within those niches. If $|I| = 1$, then we believe that there is a dominant niche and thus stop the NGA to move on to explore that niche with the local search algorithm.

A.13 Setting Up COMPASS After the NGA Finishes

Upon finishing the Global Phase, we have at hand a list of niche centers together with surrounding solutions. For each niche center, starting from the best one, we run COMPASS using that center as the starting point and surrounding solutions as active solutions.

B. COMPASS IN ISC

B.1 Generalized COMPASS Constraint Placement

The idea behind generalized constraint placement is to approximate the decrease in objective value as linear along the line connecting a solution $\mathbf{x}_i$ to the current sample best $\widehat{\mathbf{x}}^*$, and to make the constraint hyperplane as close to $\widehat{\mathbf{x}}^*$ as possible subject to the estimated difference being statistically significant, but no more than half way. The generalized COMPASS constraint is

$$(\widehat{\mathbf{x}}^* - \mathbf{x}_i)(\mathbf{x} - (\beta\widehat{\mathbf{x}}^* + (1-\beta)\mathbf{x}_i)) \ \geq\ 0,\ \beta \in [0, 1/2]. \tag{7}$$

Notice that when $\beta = 0$, the constraint simplifies to $(\widehat{\mathbf{x}}^* - \mathbf{x}_i)(\mathbf{x} - \mathbf{x}_i) \geq 0$, which means that $\widehat{\mathbf{x}}^*$ and $\mathbf{x}$ are on the same side with respect to $\mathbf{x}_i$ geometrically, and thus nothing is cut off. If $\beta = 1/2$, then it is the usual COMPASS constraint.

One problem with the generalized COMPASS constraint placement algorithm is that when objective values of active solutions are very close to the current sample best solution, β is very likely to be set to zero, cutting nothing off. Therefore, the algorithm would fail to declare a singleton and the stopping test procedure would not be invoked; that is, COMPASS would not terminate even upon hitting a local optimum. To circumvent this difficulty, we impose a lower bound ε on the value of β and then relate it to the dimension of the solution space. Intuitively, the higher the dimension is, the smaller β should be to avoid unnecessarily cutting off too many points. We used $\varepsilon = \eta^d$, with $\eta = 0.5$, where d is the dimension of a solution $\mathbf{x}$.

Assuming that we have independent sampling with unequal sample sizes and unequal variances, the smallest difference between $\bar{G}(\widehat{\mathbf{x}}^*)$ and $\bar{G}(\mathbf{x}_i)$ we can detect is

$$t_{1-\alpha_p,\nu} S, \tag{8}$$

where the standard error of the point estimate $S \equiv \text{s.e.}(\bar{G}(\widehat{\mathbf{x}}^*) - \bar{G}(\mathbf{x}_i))$ and approximate degrees of freedom ν are given by Banks, et al. [2004]:

$$S \ = \ \sqrt{\frac{S^2(\mathbf{x}_i)}{n_i} + \frac{S^2(\widehat{\mathbf{x}}^*)}{n}} \tag{9}$$

$$\nu \ = \ \frac{(S^2(\mathbf{x}_i)/n_i + S^2(\widehat{\mathbf{x}}^*)/n)^2}{[(S^2(\mathbf{x}_i)/n_i)^2/(n_i-1)] + [(S^2(\widehat{\mathbf{x}}^*)/n)^2/(n-1)]} \tag{10}$$

where $S^2(\mathbf{x}_i)$ and $S^2(\widehat{\mathbf{x}}^*)$, and n_i and n, are sample variances and number of observations, respectively, for $\mathbf{x}_i$ and $\widehat{\mathbf{x}}^*$. The degrees of freedom ν is rounded to an integer in the expression above. We require that $n \geq 2, n_i \geq 2$, and recommend that $n \geq 5, n_i \geq 5$.

We will set $\beta \in [\varepsilon, 1/2]$ based on the statistical test, unless the Euclidean distance between $\widehat{\mathbf{x}}^*$ and $\mathbf{x}_i$ is less than or equal to $\sqrt{d}$, in which case we set $\beta = 1/2$ and do not apply the test. Here $\sqrt{d}$ is the distance to the farthest point in the unit hypercube.

Algorithm 6 gives the procedure to determine β.

ALGORITHM 6 COMPASS CONSTRAINT PLACEMENT.

Step 0. If $\|\widehat{\mathbf{x}}^* - \mathbf{x}_i\| \le \sqrt{d}$*, set* $\beta = 1/2$ *and return; otherwise go to Step 1.*

Step 1. Calculate S *and* ν *by (9) and (10).*

Step 2. If $|\bar{G}(\mathbf{x}_i) - \bar{G}(\widehat{\mathbf{x}}^*)| < t_{1-\alpha_p,\nu}S$*, then let* $\beta = \varepsilon$*; otherwise, let*

$$\beta = \min\left\{1 - \frac{t_{1-\alpha_p,\nu}S}{|\bar{G}(\mathbf{x}_i) - \bar{G}(\widehat{\mathbf{x}}^*)|}, \frac{1}{2}\right\}.$$

C. COMPASS SAMPLE ALLOCATION RULE

The base sample allocation rule (SAR) for COMPASS is that every active solution has at least $n_0(\ln k)^{1.01}$ replications, where k is the number of COMPASS iterations; this is the rule used in Hong and Nelson [2006]. Thus, active solutions receive an infinite number of replications in the limit which guarantees the convergence of COMPASS.

However, as COMPASS progresses toward locally optimal solutions, it may be desirable to adaptively allocate additional simulation effort beyond that specified by the fixed schedule. Intuitively, any solution that is not statistically inferior to the current sample best should be allocated somewhat more replications than those that are not so that their status can become clearer. To accomplish this, we also implemented an adaptive SAR based on OCBA (see for instance Chen et al. [2000]). In COMPASS as implemented in ISC—Algorithm 8 in Section C.1 below—we apply the following OCBA heuristic in Step 1 (an option that can be turned off by the user):

Again, let $S^2(\mathbf{x}_i)$ and $S^2(\widehat{\mathbf{x}}^*)$ denote sample variances for $\mathbf{x}_i$ and $\widehat{\mathbf{x}}^*$. Let $\widehat{\delta}(\mathbf{x}) = \bar{G}(\mathbf{x}) - \bar{G}(\widehat{\mathbf{x}}^*)$. Assume that at each iteration we have ΔN additional replications to allocate to active solutions that define the most promising area. In our implementation, we fix ΔN to the size of the set of active solutions, denoted by $\mathcal{A}_{k-1}$. Note that these are *additional* replications.

Algorithm 7 describes the OCBA-based adaptive SAR.

ALGORITHM 7 ADAPTIVE SAR FOR COMPASS.

Step 0. At iteration k*, let* $\Delta N = |\mathcal{A}_{k-1}|$.

Step 1. Allocate 2 *additional replications to* $\widehat{\mathbf{x}}^*$.

Step 2. Let $\Delta N = \Delta N - 2$.

Step 3. Let $R = \sum_{\mathbf{x}\in\mathcal{A}_{k-1}} S^2(\mathbf{x})/\widehat{\delta}(\mathbf{x})$.

Step 4. Set $\Delta N(\mathbf{x}) = \frac{S^2(\mathbf{x})/\widehat{\delta}(\mathbf{x})}{R} \cdot \Delta N$*. If* $\Delta N(\mathbf{x}) < 1$*, round down to 0; otherwise, round to the nearest integer.*

C.1 COMPASS with Constraint Pruning

When the k^{th} iteration begins, we have:

—A most promising area (MPA) $\mathcal{C}_k$ defined by

  - —$\widehat{\mathbf{x}}^*_{k-1}$: The sample best solution from previous iteration, which is the user specified starting point $\mathbf{x}_0$ when $k = 1$.
  - —A set of active solutions, denoted by $\mathcal{A}(\widehat{\mathbf{x}}^*_{k-1})$, plus the original problem constraints that define the constraints actually forming the MPA. For notational simplicity, we shall use $\mathcal{A}_{k-1}$ instead of $\mathcal{A}(\widehat{\mathbf{x}}^*_{k-1})$ in the following.

—A list of visited solutions $\mathcal{V}_{k-1}$, which is $\{\mathbf{x}_0\}$ and the active solutions provided by the NGA, $\mathcal{A}_0$, when $k = 1$.

—A list of the sample best visited solutions, denoted as $\mathcal{B}_{k-1}$, in $\mathcal{V}_{k-1}$. The size of the list is usually set to the dimension of the solution space d; $\mathcal{B}_{k-1} = \{\mathbf{x}_0\}$ for $k = 1$.

Algorithm 8 COMPASS with Constraint Pruning.

Step 0. Let $k = 0$, $\mathcal{V}_0 = \{\mathbf{x}_0\}$, and $\mathcal{A}_0$ be the active solutions provided by the NGA. Let $\mathcal{C}_0 = \Theta$.

Step 1. Let $k = k + 1$. Let $\mathcal{B}_{k-1} = \emptyset$. Uniformly and independently sample m_L solutions (duplicates allowed) from $\mathcal{C}_{k-1}$ (see §A.2). Denote this set of solutions as $\mathcal{S}_k$. Apply SAR to $\widehat{\mathbf{x}}^*_{k-1}$, $\mathcal{A}_{k-1}$ and $\mathcal{S}_k$, and let $\mathcal{V}_k = \mathcal{V}_{k-1} \cup \mathcal{S}_k$.

Step 2. Identify $\mathcal{A}_k$ by solving LP programs formed by $\widehat{\mathbf{x}}^*_k$ as the center, and $\mathcal{S}_k, \mathcal{A}_{k-1}, \mathcal{B}_{k-1}$, and $\widehat{\mathbf{x}}^*_{k-1}$ (if $\widehat{\mathbf{x}}^*_{k-1} \neq \widehat{\mathbf{x}}^*_k$), as candidate active solutions, and thus obtain $\mathcal{C}_k$.

Step 3. Check if $\mathcal{C}_k$ is a singleton. If not, go to Step 1; else, apply stopping test procedure to $\widehat{\mathbf{x}}^*_k \cup \mathcal{A}_k$ using $\widehat{\mathbf{x}}^*_k$ as the standard. If $\widehat{\mathbf{x}}^*_k$ is still optimal, return $\widehat{\mathbf{x}}^*_k$ as the final result; else, let $\tilde{\mathbf{x}}$ be the new sample best, let $\widehat{\mathbf{x}}^*_{k-1} \leftarrow \widehat{\mathbf{x}}^*_k$, $\mathcal{A}_{k-1} \leftarrow \mathcal{A}_k$, $\widehat{\mathbf{x}}^*_k \leftarrow \tilde{\mathbf{x}}$, populate $\mathcal{B}_{k-1}$ with the d sample best solutions in $\mathcal{V}_k$, and go to Step 2.

C.2 COMPASS Control Parameters

Table III lists the control parameters for COMPASS.

D. CLEAN-UP

Once the COMPASS local search has exhausted all niches found by the NGA global search, or the budget has run out, ISC enters the Clean-up Phase with a collection of locally optimal solutions that we reindex as $\mathcal{L} = \{1, 2, \ldots, |\mathcal{L}|\}$ for convenience.

We want to select the best of these solutions, or one within δ_C of the best, with high confidence, and also state a $\pm\delta_C$ confidence interval on the performance of the selected solution. The user specifies parameters δ_C and α_C, but α_C defaults to 0.05.

Algorithm 9 Clean-Up Procedure.

Input. $\mathcal{L}$, the set of locally optimal solutions of size $|\mathcal{L}|$ and constants δ_C and α_C. For each $i \in \mathcal{L}$, $N_C^{(0)}(\mathbf{x}_i)$ is the number of observations, $\bar{G}(\mathbf{x}_i)$ is the sample mean, and $S^2(\mathbf{x}_i)$ is the sample variance.

Table III. COMPASS Control Parameters

Parameter	Functionality	Default value
n_0	Initial number of replications assigned to a new solution	User specified (default 5)
m_L	Within the most promising area, randomly sample with replacement m_L solutions	5
N_k	The minimum number of replications assigned to a solution at iteration k	$n_0(\ln k)^{1.01}$
α_p	$1-\alpha_p$ is the confidence level for generalized COMPASS constraint placement method	0.10
δ_L	Indifference zone parameter for the local optimality stopping test.	User specified (default δ_C)
α_L	$1-\alpha_L$ is the confidence level for the local optimality stopping test.	User specified (default 0.05)
cp	Number of COMPASS iterations between constraint prunings	User specified (default 5)

Step 1. Let

$$\begin{aligned}
\underline{n} &= \min_{i\in\mathcal{L}} N_C^{(0)}(\mathbf{x}_i),\\
t_i &= t_{(1-\alpha_C/2)^{1/(|\mathcal{L}|-1)},N_C^{(0)}(\mathbf{x}_i)-1},\\
h &= h(2,(1-\alpha_C/2)^{1/(|\mathcal{L}|-1)},\underline{n})\\
\omega_{i\ell} &= \left\{\frac{t_i^2 S^2(\mathbf{x}_i)}{N_C^{(0)}(\mathbf{x}_i)}+\frac{t_\ell^2 S^2(\mathbf{x}_\ell)}{N_C^{(0)}(\mathbf{x}_\ell)}\right\}^{1/2}, \ \forall i\neq \ell
\end{aligned}$$

where $h(2,(1-\alpha_C/2)^{1/(|\mathcal{L}|-1)},\underline{n})$ *is Rinott's constant in the special case of* 2 *solutions, confidence level* $(1-\alpha_C/2)^{1/(|\mathcal{L}|-1)}$ *and* $\underline{n}$ *degrees of freedom (see Boesel et al. [2003]).*

Step 2. Let $I=\{i:\bar{G}(\mathbf{x}_i)\leq \bar{G}(\mathbf{x}_\ell)+\omega_{i\ell},\ \forall \ell\neq i\}$.

Step 3. For all $i\in I$, *compute* $N_C(\mathbf{x}_i)=\max\{N_C^{(0)}(\mathbf{x}_i),\lceil h^2S^2(\mathbf{x}_i)/\delta^2\rceil\}$. *Collect* $N_C(\mathbf{x}_i)-N_C^{(0)}(\mathbf{x}_i)$ *more observations and update* $\bar{G}(\mathbf{x}_i)$.

Step 4. Let $B=\arg\min_{i\in I}\bar{G}(\mathbf{x}_i)$. *Report* $\mathbf{x}_B$ *as the best solution and claim* $\bar{G}(\mathbf{x}_B)\pm\delta_C$ *as the* $(1-\alpha_C/2)\times 100\%$ *confidence interval for* $g(\mathbf{x}_B)$.

E. USER-SPECIFIED PARAMETERS FOR TEST PROBLEMS

In Table IV we list user specified ISC control parameters using the notation introduced earlier. The only parameter that does not have a default is δ_C.

References

Baker, J. E. 1987. Reducing bias and inefficiency in the selection algorithm. In *Proceedings of the 2nd International Conference on Genetic Algorithms*, 14–21.

Table IV. Control Parameters used in Test Problems

	Multi-modal	Singular	Flowline	Inventory	High-d
n_0	5	5	5	20	5
δ_G	0.08	1	0.05	0.1	10
α_b	0.05	0.05	0.05	0.05	0.05
δ_b	0.08	1	0.05	20	10
α_L	0.05	0.05	0.05	0.05	0.05
δ_L	0.08	1	0.05	50	10
α_C	0.05	0.05	0.05	0.05	0.05
δ_C	0.08	1	0.05	20	10
cp	5	5	5	5	50

Banks, J., Carson, J. S., Nelson, B. L., and Nicol, D. M. 2004. *Discrete-Event System Simulation*, 4th ed., Pearson Prentice Hall, Upper Saddle River, NJ.

Boesel, J. 1999. *Search and Selection for Large-Scale Stochastic Optimization.* Doctoral dissertation, Department of IEMS, Northwestern University, Evanston, IL.

Boesel, J., Nelson, B. L., and Kim, S.-H. 2003. Using ranking and selection to 'clean up' after simulation optimization. *Operations Research 51*, 814-825.

Chen, C.-H., Lin, J., Yücesan, E., and Chick, S. E. 2000. Simulation budget allocation for further enhancing the efficiency of ordinal optimization. *Discrete Event Dynamic Systems: Theory and Applications 10*, 251–270.

Hong, L. J., and Nelson, B. L. 2006. Discrete optimization via simulation using COMPASS. *Operations Research 54*, 115–129.

Michalewicz, Z. 1996. *Genetic Algorithms + Data Structures = Evolution Algorithms*, 3rd ed., Springer, Berlin.

Miller, R. G. 1981. *Simultaneous Statistical Inference*, 2nd ed., Springer, Berlin.

Miller, B. L., and Shaw, M. J. 1995. Genetic algorithms with dynamic niche sharing for multimodal functino optimization. IlliGAL Report No. 95010, Illinois Genetic Algorithms Laboratory, University of Illinois at Champaign-Urbana.

Pichitlamken, J., and Nelson, B. L. 2003. A combined procedure for optimization via simulation. *ACM TOMACS 13*, 155–179.

Sareni, B., and Krahenbuhl, L. 1998. Fitness sharing and niching methods revisited. *IEEE Transactions on Evolutionary Computation 2*, 97–106.