Comparison of Deterministic and Stochastic Approaches to Global Optimization

Leo Liberti*

DEI, Politecnico di Milano, P.zza L. da Vinci 32, 20133 Milano, Italy,
**Corresponding author: E-mail:* `liberti@elet.polimi.it`

Sergei Kucherenko

CPSE, Imperial College London, SW7 2BY, UK,
E-mail: `s.kucherenko@imperial.ac.uk`

Abstract

In this paper we compare two different approaches to nonconvex global optimization. The first one is a deterministic spatial Branch-and-Bound algorithm (sBB), whereas the second approach is a quasi Monte Carlo (QMC) variant of a stochastic multi level single linkage (MLSL) algorithm. Both algorithms apply to problems in a very general form and are not dependent on problem structure. The test suite we chose is fairly extensive in scope, in that it includes constrained and unconstrained problems, continuous and mixed-integer problems. The conclusion of the tests is that in general the QMC variant of the MLSL algorithm is more efficient, although in some instances the Branch-and-Bound algorithm is capable of locating the global optimum of hard problems in just one iteration.

Keywords: global optimization, convex envelope, bilinear programming, spatial Branch-and-Bound, multi level single linkage, low discrepancy sequences

1 Introduction

The nonlinear optimization problems (NLPs) considered in this paper are of the form:

$$\left.\begin{array}{rcl} \min_x & f(x) & \\ l & \leq g(x) \leq & u \\ x^L & \leq x \leq & x^U \end{array}\right\} \qquad (1)$$

where $x \in \mathbb{R}^n$, $f : \mathbb{R} \to \mathbb{R}$, $g : \mathbb{R}^n \to \mathbb{R}^m$, $l, u \in \mathbb{R}^m$ are the lower and upper bounds of the constraints, and $x^L, x^U \in \mathbb{R}^n$ are the lower and upper bounds to the variables. The functions f and g are, in general, nonconvex.

Global optimization algorithms are usually broadly divided into deterministic and stochastic. Deterministic methods provide a theoretical guarantee of locating the global minimum, or at least a local minimum whose objective function value differs by at worst ε from the global one for a given $\varepsilon > 0$. Stochastic methods only offer a guarantee in probability. On the other hand, stochastic methods are usually faster in locating a global optimum than deterministic ones. Moreover, stochastic methods adapt better to black-box formulations and extremely ill-behaved functions, whereas deterministic methods usually rest on at least some theoretical assumptions about the problem formulation and its analytical properties. Comparisons between deterministic and stochastic global optimization methods are scarce in the literature. A comparison of the deterministic graduated nonconvexity algorithm with different versions of simulated annealing stochastic algorithms was performed in [5]. Author's conclusion was that stochastic algorithms are less efficient. Partly this result can be explained by the inefficiency of the chosen stochastic algorithms: *e.g.* Dekkers and Aarts found that the stochastic MLSL algorithm is more efficient than simulated annealing algorithms [7]. Dixon and Jha compared the deterministic interval method with the stochastic MLSL algorithm. Their conclusion was that the stochastic algorithm was at least 8 times faster than the deterministic one [8]. Zabinsky reported results with several deterministic methods, including a branch and bound scheme coupled with gradient search local optimization methods [34]. Limitations of the deterministic approach such as the loose bounds and impossibility of solving practical problems with more than 10 variables led the author to consider stochastic methods for global optimization. Good results were obtained with the Improving Hit-and-Run method which is a version of sequential random search methods.

In this paper, we compare the performances of a deterministic sBB method and a QMC variant of a stochastic MLSL method on several classes of problems. Formulation (1) shows that the type of problems we aim to solve are very general. The algorithms being tested are not tailored to any particular problem structure (but see Section 2.1 about the treatment of bilinear terms in the sBB algorithm).

In deterministic global optimization, when the form of the problem is not known *a priori*, Branch-and-Bound algorithms seem to be the most promising tool for finding the global solution efficiently. Many variants of these algorithms (see for example [1], [2], [9]) are used in this area. The search space is the whole definition range of all variables. At each step, the range of one of the variables is selected. Lower and upper bounds to the objective function are then calculated relative to the current region. If these bounds come within ε distance of each other, the upper bound is accepted as the best minimum in the current region: if it is better than the current overall best minimum, it replaces it, otherwise the region is discarded. If the bounds are farther apart than ε, a branching variable and a branching point are chosen and the region is split in smaller subregions; the original region is then discarded. The algorithm terminates when there are no more regions to examine. The deterministic sBB algorithm implementation we used is based on the sBB algorithm with symbolic reformulation presented in [26, 29, 30]. In sBB algorithms, the complexity of the problem may grow exponentially as a function of the number of variables. For high dimensional problems it results in prohibitively large computational time.

The simplest form of a stochastic approach for global optimization is called Pure Random Search (PRS). In PRS an objective function $f(x)$ is evaluated at N randomly chosen points and the smallest value of $f(x)$ is taken as the global minimum. The PRS approach is not very efficient because the expected number of iterations for reaching a specified tolerance grows exponentially in the dimension n of the problem.

Advanced stochastic techniques use stochastic methods to search for the location of local minima and then employ deterministic methods to solve a local minimization problem. Two phases are considered: global and local. In the global phase, the function is evaluated in a number of randomly sampled points from a uniform distribution over a unit hypercube H_n. In the local phase the sample points are used as starting points for a local minimization search. The efficiency of the multistage methods depends both on the performance of the global stochastic and the local minimization phases.

In the most basic form of the multistage approach, a local search is applied to every sample point. Inevitably, some local minima are found many times. Since the local search is the most CPU-time consuming stage, ideally it should start just once in every region of attraction: this is the idea behind various versions of the so-called clustering methods. Extensive reviews on this subject can be found in [22, 23, 25] and [32]. One of the most efficient clustering methods is the MLSL algorithm developed by Rinnooy-Kan and Timmer in [22, 23].

The efficiency of stochastic methods depends on the quality of sampled points. It has been recognized through theory and practice that uniformly distributed deterministic sequences provide more accurate results than purely random sequences. The low-discrepancy sequences (LDS) are designed specifically to place sample points as uniformly as possible. Unlike random numbers, successive low discrepancy points "know" about the position of their predecessors and fill the gaps left previously. Methods based on LDS are known as QMC methods. QMC methods have superior performance to that of MC methods in almost all applications. Improvement in time-to-accuracy using QMC methods can be as large as several orders of magnitude. It was shown in [13] that application of LDS can significantly increase the efficiency of MLSL methods.

2 Deterministic Spatial Branch-and-Bound Algorithm

The sBB algorithm used in these tests (based on [30]) is outlined below:

1. (Preprocessing) Generate *reduction constraints*. These help tighten the feasible region significantly in the presence of linear equality constraints and bilinear terms (see section 2.1).
2. (Initialization) The list of regions to be explored initially contains a single region comprising the whole set of variable ranges. Set the convergence tolerance ε and the best objective function value $U := \infty$. Optionally perform pre-processing steps (like *e.g.* optimization-based bounds tightening).
3. (Choice of Region) If the list of regions is empty, terminate the algorithm with solution U. Otherwise, choose a region from the list. Delete this region from the list. Optionally perform feasibility-based bounds tightening on this region.
4. (Lower Bound) Generate a (linear) convex relaxation of the original problem and solve it to obtain an underestimation l of the objective function for the current region. If $l > U$ or the relaxed problem is infeasible, go back to step 3.
5. (Upper Bound) Attempt to solve the original (generally nonconvex) problem to obtain a locally optimal objective function value u. If this fails, set $u := +\infty$.
6. (Pruning) If $U > u$, set $U := u$. Delete all regions in the list that have lower bounds greater than U, as they cannot possibly contain the global minimum.
7. (Check Region) If $u - l \leq \varepsilon$, accept u as the global minimum for this region and return to step 3. Otherwise, we may not yet have located the global minimum for the current region, so we proceed to the next step.
8. (Branching) Apply a branching rule to the current region to split it into subregions. Add these to the list of regions, assigning to them an (initial) lower bound of l. Go back to step 3.

The region selection at step 3 follows the simple policy of "choose the region with lowest lower objective function bound" as the most promising for further consideration.

In the rest of this section we shall briefly discuss the most important steps of the sBB algorithm given above.

2.1 Reduction Constraints

The full theory of reduction constraints is explained elsewhere [14, 15, 17]. Here, only a short introduction is given. Let $n \in \mathbb{N}$, $x_1, \ldots, x_n, w_1, \ldots, w_n, y$ be problem variables and suppose that the feasible region of an optimization problem is defined by a set of constraints that include the following:

$$\begin{array}{rcl} w_1 & = & x_1 y \\ & \vdots & \\ w_n & = & x_n y \\ \sum_{i=1}^{n} a_i x_i & = & b \end{array}$$

where $a_1, \ldots, a_n, b$ are real parameters and $a_i \neq 0$ for all $i \leq n$. We can multiply the linear constraint above by the variable y to obtain

$$\sum_{i=1}^{n} a_i x_i y = by$$

and since $w_i = x_i y$ for all $i \leq n$ we get

$$\sum_{i=1}^{n} a_i w_i = by \tag{2}$$

i.e. a linear relation between the w variables and y. Constraint (2) is called a *reduction constraint* because it reduces the number of bilinear constraints $w_i = x_i y$ required to define the same feasible region. If we discard one of the bilinear constraints, say $w_n = x_n y$, from the formulation of the problem, we shall show that we can use the (linear) reduction constraint $\sum_{i=1}^{n} a_i w_i = by$ instead.

For all $i \leq n$ let $z_i = w_i - x_i y$. Notice that since $b = \sum_{i=1}^{n} a_i x_i$ we can re-write the reduction constraint as

$$\begin{array}{rc} & \sum_{i=1}^{n} a_i w_i - y \sum_{i=1}^{n} a_i x_i = 0 \quad \Rightarrow \\ \Rightarrow & \sum_{i=1}^{n} a_i (w_i - x_i y) = 0 \\ \Rightarrow & \sum_{i=1}^{n} a_i z_i = 0 \end{array}$$

Since the formulation of the problem includes the $n-1$ bilinear constraints $w_i = x_i y$ for all $i \leq n-1$ (we have discarded the last), it follows that $\forall i \leq n-1$ we have $z_i = 0$, so the linear relationship above between the z variables is reduced to $a_n z_n = 0$. We had assumed $a_n \neq 0$, hence $z_n = 0$, i.e. $w_n = x_n y$, as claimed.

The reasoning above is the essence of the usefulness of reduction constraints: they are linear constraints that can replace bilinear terms in a nonlinear problem without modifying the feasible region. There are many ways in which this concept can be generalized; see the cited papers for further details.

Creation of reduction constraints, as a pre-processing step to the Branch-and-Bound algorithm, often produces much tighter convex relaxations. Instead of using McCormick's convex relaxation of bilinear terms (Eq. (10)-(13)), which is usually very loose, we employ linear reduction constraints that do not need to be relaxed at all. The solution times are therefore dramatically reduced.

2.2 Standard Form

The convex relaxation solved at step 4 of the algorithm aims to find a guaranteed lower bound to the objective function value. In the sBB algorithm, the convex relaxation is calculated automatically for a problem in form (1). The problem is first reduced to a *standard form* where nonlinear terms of the same type are collected in lists. Secondly, each nonlinear term is replaced by the corresponding convex under- and overestimators.

The standard form is as follows:

$$\begin{array}{rcll} \min x_{obj} & & & (3) \\ l \leq & Ax & \leq u & (4) \\ x_k & = & x_i x_j \quad \forall (i,j,k) \in \mathcal{M} & (5) \\ x_k & = & \dfrac{x_i}{x_j} \quad \forall (i,j,k) \in \mathcal{D} & (6) \\ x_k & = & x_i^n \quad \forall (i,k) \in \mathcal{P} & (7) \\ x_k & = & f_i(x_i) \quad \forall (i,k) \in \mathcal{U} & (8) \\ x^L \leq & x & \leq x^U & (9) \end{array}$$

where $x = (x_1, \ldots, x_n)$ are the problem variables, A is a matrix of linear constraints, l, u are the linear constraint bounds, x^L, x^U are the variable bounds, and the constraints (5)–(8) are the defining constraints of the standard form; $\mathcal{M}, \mathcal{D}$ are sets of triplets of variable indices which define the bilinear and linear fractional terms while $\mathcal{P}, \mathcal{U}$ are sets of variable index pairs which define power terms and terms involving univariate functions f_i. Each defining constraint has one of the forms:

$$\begin{aligned} \textit{added variable} &= \textit{operand (binary operator) operand}, \\ \textit{added variable} &= \textit{(unary operator) operand}, \end{aligned}$$

where *operand* is an original or an added variable (added variables are all variables added to the original variable set by the standardization procedure). The objective function has been replaced by the added variable x_{obj} and a constraint of the form

$$x_{obj} = objective\ function$$

has been added to the problem constraints.

The algorithm for reducing a MINLP to the standard form is described in detail in [26, 30]. It works by recursively tackling nonlinear terms in the problem and forming linear and defining constraints as it goes along.

2.3 Convex Relaxation

This is the second stage of the process where the actual convex relaxation of the original problem is built. The algorithm for convexification is entirely symbolic (as opposed to numeric) and hence performs very efficiently even in presence of very complicated mathematical expressions. Having reduced a problem to the standard form, we replace every nonlinear equation of the kind $x_i = \nu(x_j, x_k)$ with a convex relaxation consisting of convex over- and underestimating inequality constraints. The rules we follow to build over- and underestimators are as follows:

1. $x_i = x_j x_k$ is replaced by four linear inequalities (McCormick's envelopes, [20]):

$$\begin{aligned} x_i &\geq x_j^L x_k + x_k^L x_j - x_j^L x_k^L && (10) \\ x_i &\geq x_j^U x_k + x_k^U x_j - x_j^U x_k^U && (11) \\ x_i &\leq x_j^L x_k + x_k^U x_j - x_j^L x_k^U && (12) \\ x_i &\leq x_j^U x_k + x_k^L x_j - x_j^U x_k^L && (13) \end{aligned}$$

2. $x_i = x_j / x_k$ is reformulated to $x_i x_k = x_j$ and the rules for multiplication are applied.

3. $x_i = f(x_j)$ where f is concave univariate is replaced by two inequalities: the function itself and the secant.

$$\begin{aligned} x_i &\leq f(x_j) && (14) \\ x_i &\geq f(x_j^L) + \frac{f(x_j^U) - f(x_j^L)}{x_j^U - x_j^L}(x_j - x_j^L) && (15) \end{aligned}$$

4. similarly, $x_i = f(x_j)$ where f is convex univariate is replaced by

$$\begin{aligned} x_i &\leq f(x_j^L) + \frac{f(x_j^U) - f(x_j^L)}{x_j^U - x_j^L}(x_j - x_j^L) && (16) \\ x_i &\geq f(x_j) && (17) \end{aligned}$$

5. $x_i = x_j^q$ where $0 < q < 1$ the function is concave univariate and falls into category 3 above.

6. $x_i = x_j^{2m}$ for each $m \in \mathbb{N}$ is convex univariate and falls into category 4 above.

7. $x_i = x_j^{2m+1}$ can be convex, concave, or piecewise convex and concave with a turning point at 0. If the range of x_j does not include 0, the function is convex or concave and falls into a category described above. A convex relaxation for the case where the range includes 0 is given in [16].

We deliberately chose to make the convex relaxation linear. Although this may generate looser lower bounds than with a nonlinear relaxation, and hence more Branch-and-Bound iterations, the cost of solving a linear problem at each iteration will be significantly less than that of a nonlinear problem.

3 Stochastic Multilevel Single Linkage Method

In the simplest variant of a multistage method, a small number of random points are sampled and then a deterministic local search procedure is applied to all these points. All located stationary points are sorted and the one with the lowest value of the objective function is taken as a global minimum. One problem with this technique is that the same local minimum may be located several times. Ideally, the local search should be started only once in every region of attraction. The region of attraction of a local minimum x^* is defined as the set of points starting from which a given local search procedure converges to x^*.

Among various versions of clustering methods, the MLSL method [22, 23] is one of the most efficient. In this method only those sample points whose function values are small enough are chosen as starting points. Points are grouped into clusters. A cluster is initiated by a seed point. The seed point is normally a local minimum $x^* \in X^*$ that has already been found (where X^* is the set of all local minima of the problem). All sample points within a critical distance are assigned to the same cluster. The critical distance r_k is given by:

$$r_k = \left(\frac{m(B)}{\omega_n}\frac{\sigma\log(kN)}{kN}\right)^{\frac{1}{n}}. \tag{18}$$

where

$$\omega_n = \frac{\pi^{\frac{n}{2}}}{\Gamma\left(1+\frac{n}{2}\right)}$$

is the volume of the n-dimensional unit ball, $m(B)$ is the Lebesgue measure of a feasible region B (if $B = H_n$ then $m(B) = 1$), k is an iteration index, s is a known parameter, [22, 23]. In our calculations s was equal to 2.

The improvement in efficiency can be archived by selecting points with the lowest values of an objective function and discarding the rest of the sampled points. Let X be a set of N sampled points, and let $\langle f(x_i)|x_i \in X\rangle$ be the sequence of corresponding objective function values ordered so that $f(x_i) \le f(x_{i+1})$ for all $i < N$. The reduced sample set is defined as:

$$X_r = \{x_i \in X \mid i = 1, \ldots, N_r\}, \tag{19}$$

where $N_r = \alpha N$ with $0 < \alpha < 1$. We note, that some local minima can be discarded without affecting the global minimum search.

Reliable termination criteria of the global stage was developed in [6]. It is based on Bayesian estimates for the number of real minima not yet identified and the probability that the next local search will locate a new local minimum. An optimal Bayesian stopping rule is defined as follows: if W different local minima have been found after N local searches started in uniformly distributed points, then the expectation of the number of local minima is

$$W_{exp} = \frac{W(N-1)}{N-W-2}, \tag{20}$$

provided that $N > W + 2$. The searching procedure is terminated if $W_{exp} < W + 0.5$.

3.1 The MLSL algorithm

The general scheme of the MLSL algorithm is outlined below:

1. Set $W = 0$, $k = 0$. Set k_{max} to the maximum allowed number of iterations. Initialize the list of local minima X^* to the empty list.
2. Set $k = k + 1$, $i = 0$.
3. Sample a set X of N points from a uniform distribution over H_n.
4. Evaluate an objective function on a set X, build the sequence $\langle f(x_i)\rangle$ sorted in order of increasing function values, and select a reduced set X_r according to (19).
5. Set $i = i + 1$ and take $x_i \in X_r$.
6. Assign the sample point x_i to some cluster C_l if $\exists\ x_j$, $x_j \in C_l$ such that $\varrho(x_i, x_j) \leq r_k$, $f(x_j) \leq f(x_i)$, where r_k is a critical distance given by (18). If x_i is not assigned to any cluster yet, then start a local search at x_i to yield a local minimum x^*. If $x^* \notin X^*$, then add x^* to X^*, set $W = W + 1$ and initiate the W-th cluster with x^*. Assign x_i to this cluster.
7. If $i < N_r$ go to step (5).
8. If $k = k_{max}$ or the stopping criterion (20) is satisfied, then terminate. Else go to step (2).

3.2 Deterministic Sequences

As in other cases of transition from MC to QMC algorithms, a significant improvement in efficiency can be achieved simply by substituting random points with LDS. Central to the QMC approach is the choice of LDS. Different principles were used for constructing LDS by Holton, Faure, Sobol', Niederreiter and others. Many well-known LDS were constructed mainly upon asymptotic considerations, as a result they do not perform well in real practical tests. Points generated by the Sobol' LDS produce a very uniform filling of the space even for a rather small number of points N, which is a very important case in practice ([27], [28]). Many practical studies have proven that Sobol' LDS is superior to other LDS in many respects. This is the reason why Sobol's LDS were used in the present study.

4 Description of Test Suite

In this section we give a brief description of each of the test problems that we have solved. A summary of problem statistics is reported in Table 1.

As stated in [31] the choice of test problems should be systematic, so that they represent different types of problems ranging from easy to difficult to solve. In our work we tried to follow this recommendation. Some of the problems we used are well known tests for global optimization. For some of these problems we used a classification into degrees of difficulty suggested in [31].

Problem	*Variables*	*Constraints*
`sixhump` (21)	2	0
`schaffler1` (22)		
$n = 3$	3	0
$n = 5$	5	0
$n = 10$	10	0
$n = 30$	30	0
$n = 40$	40	0
$n = 50$	50	0
`schaffler2` (23)		
$n = 3$	3	0
$n = 4$	4	0
$n = 5$	5	0
$n = 6$	6	0
$n = 7$	7	0
$n = 8$	8	0
$n = 9$	9	0
$n = 10$	10	0
`griewank1` (24)		
$n = 2$	2	0
$n = 10$	10	0
`griewank2` (25)		
$n = 2$	2	0
$n = 10$	10	0
`schubert1` (26)	3	0
`schubert2` (26)	5	0
`blending1` (28)	21	30
`blending2` (28)	25	42
`simpleminlp` (29)	6	5
`yuan1` (30)	5	5
`yuan2` (31)	9	7

Table 1: Problem statistics.

4.1 Six-hump Camel Back Function

The "six hump back camel function", introduced n [8], is a well known test for global optimization algorithms.

$$\min_{-1\le x_1,x_2\le 4} 4x_1^2 - \frac{21}{10}x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4. \tag{21}$$

The global optimum is located at $x = (-0.0883, 0.7125)$ and has an objective function value of -1.03136. According to the classification of problems into the degrees of difficulty suggested in [31] this problem belongs to a class of "easy" (E1) problems.

4.2 Schaffler Function

This test function was introduced in [24]. The formulation below depends on an integer n, thus it is actually a family of different problems.

$$\min_{-1.05\le x_i\le 2.95} 1 + 590\sum_{i=2}^{n}(x_i - x_{i-1})^2 + 6x_1^2 - \cos(12x_1). \tag{22}$$

By inspection it is easy to note that, for all $n \in \mathbb{N}$, the global optimum is at $x = (0, \ldots, 0)$ with an objective function value of 0. This ceases to be apparent if one solves the following modified problem:

$$\min_{-1.05 \leq x \leq 2.95} 1 + 590 \sum_{i=2}^{n} (x_i - x_{i-1})^2 + 6x_1^2 + \cos(12x_1). \tag{23}$$

The plus sign in front of the cosine term complicates the geometrical properties of the problem. In this case there are many symmetrical global optima positioned at

$$x = (\pm 0.2414, \ldots, \pm 0.2414)$$

having an objective function value 0.3794.

4.3 Griewank Function

This test function was introduced in [32]. The formulation below depends on an integer n that can take two values (2 and 10).

$$\min_{-r \leq x \leq r} 1 + \frac{1}{d} \sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos(\frac{x}{\sqrt{i}}), \tag{24}$$

where $d = 200$, $r = 100$ for $n = 2$, and $d = 4000$, $r = 600$ for $n = 10$. Both problems belong to the class of "moderate" (M2) [31].

As in problem (22), here it is again evident by inspection that the global optimum is at $x = (0, \ldots, 0)$ with objective function value 0. Again, this ceases to be the case if one considers the following modified problem:

$$\min_{-r \leq x \leq r} 1 + \frac{1}{d} \sum_{i=1}^{n} x_i^2 + \prod_{i=1}^{n} \cos(\frac{x}{\sqrt{i}}), \tag{25}$$

where we have replaced the minus sign in front of the cosine product term with a plus sign. See [19] for more details about the Griewank function. For $n = 2$ the global optimum for (25) is at $x = (3.1104, 0.0)$ and has an objective function value 0.0488. For $n = 10$ the global optimum is at $x = (\pm 3.1400, 0.0, \ldots, 0.0)$ and has an objective function value 0.0024. The trend, for increasing values of n and d, is for x_1 to tend to π so that the sign of the cosine product is again turned to "minus". Obviously this has a positive cost on the sum of squares, which becomes negligible because of the d factor in the denominator.

4.4 Schubert Functions

These test functions were introduced in [7]. The first is defined as

$$\min_{-10 \leq x \leq 10} \frac{\pi}{n} \left(10 \sin^2(\pi \frac{x_1 + 5}{4}) + \sum_{i=1}^{n-1} (\frac{1}{4} x_i + 1)^2 (1 + 10 \sin^2(\pi \frac{x_{i+1} + 5}{4})) + (\frac{1}{4} x_n + 1)^2 \right) \tag{26}$$

with $n = 3$; the second is defined as

$$\min_{-5 \leq x \leq 5} \frac{1}{10} \left(\sin^2(3\pi x_1) + \sum_{i=1}^{n-1} (x_i - 1)^2 (1 + \sin^2(3\pi x_{i+1})) + (x_n - 1)^2 (1 + \sin^2(2\pi x_n)) \right) \tag{27}$$

with $n = 5$.

The global optima of these problems are located at $x = (1, \ldots, 1)$ with an objective function value of 0. Both problems belong to the class of "easy" (E2) problems [31].

4.5 Adhya's Multi-quality Blending Problems

These two blending problems from the petrolchemical industry are taken from [3], where they were named example 1 and example 2 respectively. Various types of crude oils of different qualities and coming from different sources are mixed together to produce several end-products subject to certain quality requirements and demands. These are bilinear problems with many local optima. The solution of this type of blending problem is important because of the direct application to the industrial world as well as for the mathematical challenges it poses. The literature on pooling and blending bilinear problems is vast [3, 10, 33].

Here, we use the general blending problem formulation found in [3], p. 1958:

$$\left.\begin{array}{rl} \min_{f,q,x} \sum_{j=1}^{p} \sum_{i=1}^{n_j} c_{ij} f_{ij} - \sum_{k=1}^{r} d_k \sum_{j=1}^{p} x_{jk} & \\ \sum_{i=1}^{n_j} f_{ij} - \sum_{k=1}^{r} x_{jk} = 0 & \forall j \le p \\ q_{jw} \sum_{k=1}^{r} x_{jk} - \sum_{i=1}^{n_j} \lambda_{ijw} f_{ij} = 0 & \forall j \le p \; \forall w \le l \\ \sum_{j=1}^{p} x_{jk} \le S_k & \forall k \le r \\ \sum_{j=1}^{p} q_{jw} x_{jk} - z_{kw} \sum_{j=1}^{p} x_{jk} \le 0 & \forall k \le r \; \forall w \le l \\ f^L \le f \le f^U, q^L \le q \le q^U, x^L \le x \le x^U, & \end{array}\right\} \quad (28)$$

where f_{ij} is the flow of input stream i into pool j, x_{jk} is the total flow from pool j to product k and q_{jw} is the w-th quality of pool j; p is the number of pools, r the number of products, l the number of qualities, n_j the number of streams; $c_{ij}, d_k, S_k, Z_{kw}, \lambda_{ijw}$ are parameters (their values are reported in [3], p. 1967).

The objective function value at the global optimum is -549.8031 in both cases. The values of the variables f, x are also the same in both cases:

$$\begin{array}{rcl} f & = & (7.5443, 19.752, 0, 4.9224, 2.7812) \\ x & = & (0, 19.2097, 0, 8.0866, 0, 5.7903, 0, 1.9133) \end{array}$$

whereas the values of the quality variables q change:

$$\begin{array}{rcl} q' & = & (3.1708, 2.382, 3.2764, 1.5854, 2.278, 2.8917, 3.361, 1.2166) \\ q'' & = & (3.1708, 2.382, 3.2764, 1.5854, 4.2763, 5.382, 2.278, 2.8917, 3.361, 1.2166, 3, 5.083) \end{array}$$

where q' are the values of q at the global optimum of example 1, and q'' are the corresponding values for example 2.

4.6 A Simple MINLP Example

The following example was taken from [14]. It is an example of a nonlinear mixed-integer problem having one binary variable, x_5. The solution techniques (both the deterministic and stochastic methods) make use of a continuous reformulation of the problem with the addition of an integrality enforcing constraint $x_5^2 = x_5$.

$$\left.\begin{array}{ll} \min_x & x_1^2 + x_1x_2 - x_1x_3 - 2x_1x_4 + x_2^2 + 3x_2x_4 - x_2x_5 + x_3x_4 + 3x_4^2 + 2x_4x_5 - \\ & \qquad -x_1 - x_4 - x_6 + e^{-x_2x_3} \\ & x_1 + x_2 - x_3 + x_4 + x_5 = 1 \\ & x_2 - x_4 - x_5 = -1 \\ & x_1 + 2x_2 - 2x_3 \ge 0 \\ & 2x_1 + 7x_2 - x_3 \le 0 \\ & e^{x_3x_4} - \log(x_6) - x_2x_6 \le 1 \\ & \forall i \le 4 \; x_i \in [0, 10] \\ & x_5 \in \{0, 1\} \\ & 1 \le x_6 \le 2 \end{array}\right\} \quad (29)$$

The global solution is at $x = (0, 0, 0, 0, 1, 2)$, with an objective function value -1.

4.7 Yuan MINLPs

The following examples come from [11]. They are nonlinear mixed-integer problems. The solution techniques make use of a continuous reformulation of the problem including the integrality enforcing constraints $y = y^2$ for each binary variable y.

$$\left.\begin{array}{ll}\min_{x,y} & 2x_1 + 3x_2 + \frac{3}{2}y_1 + 2y_2 - \frac{1}{2}y_3 \\ & y_1 + x_1^2 = \frac{5}{4} \\ & \frac{3}{2}y_2 + x_2^{\frac{3}{2}} = 3 \\ & y_1 + x_1 \le \frac{8}{5} \\ & y_2 + \frac{4}{3}x_5 \le 3 \\ & -y_1 - y_2 + y_3 \le 0 \\ & 0 \le x \le 10 \\ & y \in \{0,1\}^3 \end{array}\right\} \qquad (30)$$

The global solution of problem (30) is at $x = (1.12, 1.31)$, $y = (0, 1, 1)$ with an objective function value 7.6672.

$$\left.\begin{array}{ll}\min_{x,y} & (y_1-1)^2 + (y_2-2)^2 + (y_3-1)^2 - \log(y_4+1) + (x_1-1)^2 + (x_2-2) + (x_3-3)^2 \\ & y_1 + y_2 + y_3 + x_1 + x_2 + x_3 \le 5 \\ & y_3^2 + x_1^2 + x_2^2 + x_3^2 \le \frac{11}{2} \\ & x_1 + y_1 \le \frac{6}{5} \\ & x_2 + y_2 \le \frac{9}{5} \\ & x_3 + y_3 \le \frac{5}{2} \\ & y_4 + x_1 \le \frac{6}{5} \\ & y_2^2 + x_2^2 \le \frac{41}{25} \\ & y_3^2 + x_3^2 \le \frac{17}{4} \\ & y_2^2 + x_3^2 \le \frac{116}{25} \\ & 0 \le x \le 10 \\ & y \in \{0,1\}^4 \end{array}\right\} \qquad (31)$$

The global solution of problem (30) is at $x = (0.2, 0.8, 1.908)$, $y = (1, 1, 0, 1)$ with an objective function value 4.5796.

5 Implementation of the Benchmark: $oo\mathcal{OPS}$

Both algorithms have been tested and benchmarked with the help of $oo\mathcal{OPS}$, a software framework for optimization [18]. $oo\mathcal{OPS}$ is an integrated object-oriented optimization system. It consists of a parser (that reads an algebraic formulation of an optimization problem), a symbolic computational module (that carries out elementary algebraic simplifications and computes derivatives symbolically), and a set of solver wrappers for interfacing with different numerical solvers. At the time of writing $oo\mathcal{OPS}$ interfaces to two global optimization solvers (the two methods explained in this paper) and three local solvers (SNOPT [12], the NAG `e04vcf` nonlinear local solver [21], and LPSOLVE [4], a freely distributable code for large sparse linear systems).

Both SNOPT and the NAG solver were used as nonlinear local search procedures in both the deterministic and the stochastic codes. The solution of the (linear) convex relaxation of the problem, within the sBB algorithm, was carried out using both SNOPT and LPSOLVE. In terms of performance comparison, it was observed that on small and medium-sized problems, such as those proposed in this paper, SNOPT and the NAG solver are roughly equivalent in terms of numerical accuracy and computational efficiency.

6 Results of the Tests

Most of the run-time parameters of the deterministic and stochastic solver modules are very different. This is also true of the performance criteria. It turns out that the most significant comparisons (apart from the obvious indication on whether the method actually located the global optimum or not) are based on the following criteria:

1. CPU time;
2. The number of calls to the nonlinear local solver procedure (NLP).

The second performance criterion refers to the fact that usually (not always) most of the CPU-time, in both algorithms, is spent finding local minima.

There are other performance criteria which only apply to one method or the other, but not to both. Thus, we shall present the results in three tables. In Table 2 given in Section 6.1 we have listed the common performance criteria; Table 3 refers to the deterministic sBB algorithm, whilst Table 4 given in Section 6.2 refers to the QMC variant of the MLSL algorithm. All of the tests have been carried out under the $oo\mathcal{OPS}$ framework running under Linux on a Pentium III 850MHz PC with 256MB RAM.

In the first column of table 2 we list the names of the problems as assigned throughout this paper. Each of the other three columns ("Global Opt.?", "CPU Time", "Local NLP Calls") lists the performance criteria for the method comparisons: whether a global optimum has been found, the CPU time, and the number of calls to the nonlinear local optimization procedure. These columns are divided in two subcolumns containing the data from the deterministic and stochastic methods.

Small differences in overall performances of the two algorithms were revealed for `griewank1` function (24) with $n = 10$, `schaffler1` function (22) with $n = 5$ and `schaffler2` function (23) with $n = 3, 4, 5, 6$. For the `schaffler1` function (22) with $n = 30, 40, 50$ and `schubert2` function (27), the deterministic method showed better performance than SobolOpt while for all other cases SobolOpt showed a superior performance. There were also two problems for which the deterministic method failed to find a global minimum.

In the following two subsections we present tests results specific to each of the algorithms.

6.1 Tests with the sBB Algorithm

Table 3 lists the numerical results obtained with the deterministic sBB algorithm.

- The column *Problem* lists the names of the problems as assigned throughout this paper.
- The column *Iterations* lists the number of main algorithmic iterations taken. This is equal to the number of Branch-and-Bound nodes examined.
- The column *Glob. Opt. Iteration* lists the iteration at which the global optimum was located (this is useful to have a rough idea about the quality of the best current optimum when the sBB algorithm is used heuristically — that is, stopped after a fixed amount of time processing instead of satisfying the termination conditions).
- The column *N. Red. Constr.* (Number of Reduction Constraints) lists the number of reduction constraints generated for the problem (see Section 2.1).

Our implementation of the sBB algorithm does not depend on many numerical parameters. The main parameter affecting convergence speed is the ε tolerance (see step 7 of the sBB algorithm in Section 2),

Problem	*Global Opt?*		*CPU time*		*Local NLP Calls*	
	Det.	Stoch.	Det.	Stoch.	Det.	Stoch.
`sixhump` (21)	yes	yes	0:01	0:01	6	6
`schaffler1` (22)						
$n = 3$	local	yes	0:04	0:00	8	2
$n = 5$	yes	yes	0:00	0:00	1	3
$n = 10$	yes	yes	2:20	0:00	56	11
$n = 30$	yes	yes	0:01	0:33	1	220
$n = 40$	yes	yes	0:01	20.35	1	4028
$n = 50$	yes	yes	0:01	2:17:30	1	16206
`schaffler2` (23)						
$n = 3$	yes	yes	0:02	0:00	8	3
$n = 4$	yes	yes	0:01	0:00	2	3
$n = 5$	yes	yes	0:05	0:10	4	8
$n = 6$	yes	yes	0:15	0:11	12	6
$n = 7$	yes	yes	0:38	0:13	2	13
$n = 8$	yes	yes	1:52	0:37	5	23
$n = 9$	yes	yes	7:22	0:50	38	26
$n = 10$	yes	yes	26:06	0:27	2	16
`griewank1` (24)						
$n = 2$	yes	yes	0:00	0:02	16	23
$n = 10$	yes	yes	0:01	0:01	36	18
`griewank2` (25)						
$n = 2$	yes	yes	16:57	0:02	224	28
$n = 10$	NC	yes	>10h	0:01	-	16
`schubert1` (26)	yes	yes	2:45:00	0:00	6	13
`schubert2` (27)	yes	yes	0:00	0:02	1	68
`blending1` (28)	yes	yes	4:20	0:01	160	32
`blending2` (28)	yes	yes	13:55	0:01	307	16
`simpleminlp` (29)	yes	yes	0:15	0:00	9	4
`yuan1` (30)	yes	yes	0:13	0:00	485	5
`yuan2` (31)	yes	yes	1:05	0:00	1241	15

Table 2: Performance comparison of the two methods ([hh:]mm:ss). NC = method did not converge within the time specified in the *CPU time* column.

which was set to 1×10^{-6} for the whole test suite. It is a typical feature of Branch-and-Bound algorithms, due to its exponential behaviour, to either converge to the global optimum very quickly (in less than 10 iterations) or to take a long time and computational effort to find it. The results of Table 3 confirm this.

The sBB algorithm on the `schaffler1` test function (22) generally exhibited better performance than the MLSL algorithm in high dimensional cases. The main reason for this is the tightness of the convex relaxation. In particular, the scaling factors 590 and 6 on the quadratic terms exceed by far the extent of the variation given by the cosine term. It results in the cosine term being virtually negligible, implying near convexity of the problem. In a few instances it was possible to obtain convergence in just 1 iteration of Branch-and-Bound, due to the extreme tightness of the convex relaxation. The cases $n = 3$ and $n = 10$ merit a special mention, since they are very far off the average values of the other cases. No numeric computations are ever performed in the global phase of the sBB, as all local minima are found by the local optimization procedure used as a “black box”. The cases $3, 10$ are those where the local optimization procedure exhibited the highest instability. We were, however, not able to ascertain the exact cause for this occurrence. Note that in the instance $n = 3$, a local optimum was located rather than the global one (thus the column *GOI* is marked as ”not applicable” (NA)).

Problem	*Iterations*	*Glob. Opt. Iteration*	*N. Red. Constr.*
`sixhump` (21)	215	1	0
`schaffler1` (22)			
$n = 3$	2199	NA	0
$n = 5$	1	1	0
$n = 10$	9913	8665	0
$n = 30$	1	1	0
$n = 40$	1	1	0
$n = 50$	1	1	0
`schaffler2` (23)			
$n = 3$	1051	1	0
$n = 4$	595	1	0
$n = 5$	1759	1	0
$n = 6$	3949	1	0
$n = 7$	7583	1	0
$n = 8$	14883	1	0
$n = 9$	35167	1	0
$n = 10$	71707	1	0
`griewank1` (24)			
$n = 2$	11	7	0
$n = 10$	25	15	0
`griewank2` (25)			
$n = 2$	61345	24	0
$n = 10$	>360000	14124	0
`schubert1` (26)	>105000	NA	0
`schubert2` (27)	1	1	0
`blending1` (28)	1257	8	8
`blending2` (28)	2283	42	12
`simpleminlp` (29)	35	3	25
`yuan1` (30)	1943	1	2
`yuan2` (31)	5495	1	0

Table 3: Results obtained with the deterministic Branch-and-Bound method (NA = not applicable).

In the `sixhump`, `schaffler2`, `schubert2`, `yuan1`, `yuan2` cases, the global optimum was located at the first iteration; however, the sBB algorithm took a considerable amount of time to make sure that the located optimum was global. This is due to the good quality of the local solver and partially also to reduction constraints pre-processing (`yuan1`).

In most other cases (`griewank1`, `blending1`, `blending2`, `simpleminlp`) the global optimum was located in an acceptably low number of sBB iterations, and the algorithm again spent most of the time proving that the located optimum was global. This is due to reduction constraints pre-processing and/or effective bounds tightening techniques (steps 1 and 3 of the sBB algorithm in Section 2).

6.2 Tests with the QMC variant of MLSL Method

Table 4 lists the numerical results obtained with the QMC variant of MLSL method.

- The column *Problems* lists the names of the problems as assigned throughout this paper.
- The column N lists the total number of sampled points at each iteration. For the Sobol' LDS

Problem	N	N_r	*Iterations*	*N. Loc. Min.*
`sixhump` (21)	256	128	1	6
`schaffler1` (22)				
$n = 3$	16	4	1	2
$n = 5$	128	8	1	2
$n = 10$	32	16	1	3
$n = 30$	8192	256	1	3
$n = 40$	16384	4096	1	4
$n = 50$	1048576	16384	1	3
`schaffler2` (23)				
$n = 3$	64	8	2	2
$n = 4$	8	4	4	2
$n = 5$	32	16	2	3
$n = 6$	128	16	2	3
$n = 7$	32	16	2	3
$n = 8$	256	64	1	3
$n = 9$	256	64	1	3
$n = 10$	64	8	4	3
`griewank1` (24)				
$n = 2$	32768	128	2	23
$n = 10$	32768	128	2	14
`griewank2` (25)				
$n = 2$	32768	128	5	28
$n = 10$	32768	128	2	16
`schubert1` (26)	32	16	10	12
`schubert2` (27)	1024	512	3	63
`blending1` (28)	64	32	1	28
`blending2` (28)	128	16	1	15
`simpleminlp` (29)	8	4	1	2
`yuan1` (30)	16	8	1	4
`yuan2` (31)	4096	32	1	10

Table 4: Results obtained with the QMC variant of MLSL method.

the equidistribution property and improved discrepancy estimates hold when N is a power of 2; therefore in all these experiments N was taken to be equal to 2^m, where $m \in \mathbb{N}$.

- The column N_r lists the reduced number of sampled points at each iteration.
- The column *Iterations* lists the number of iterations in the global stage.
- The column *N. Loc. Min.* lists the total number of located minima.

Four independent runs for each test problems were performed (for each run a different part of the Sobol' LDS was used). Values N and N_r given in Table 4 are the smallest sample sizes for which a global minimum or global minima (like *e.g.* problem `sixhump`) were found in all four runs.

For the `sixhump` test function for the presented set of parameters N/N_r all six minima were found and the number of NLP calls was equal to the number of located minima. N and N_r can be further reduced without sacrificing the probability of finding both global minima, although some local minima in this case can be missed.

Both `schaffler1` and `schaffler2` problems up to dimension n=10 were easy to solve. To solve high dimensional problems (`schaffler1` n=30, n=40, n=50) a large number of the total sampled N

and reduced N_r points were required. Increasing the dimensionality from n=30 to n=50 resulted in N increasing 128 fold, N_r increasing 64 fold and the CPU time increasing approximately 250 fold. It is known that uniformity properties of the Sobol' LDS degrade as dimensionality grows, although it effects only functions with all variables being equally important [28], which is the case for the `schaffler1` objective function: apart from variable x_1, all other variables are equally important. We can conclude that for high dimensional objective functions with equally important variables the QMC variant of MLSL method becomes less efficient than for other types of objective functions.

The `griewank1` and `griewank2` functions have a very large number of local minima. For such functions the region of attraction of the global minimum generally is very small, as a result a very large number of points must be sampled to locate it. It is interesting to note that the two-dimensional problem is more difficult to solve by using QMC based or stochastic methods than the ten-dimensional one. This is in line with the results of [22], [23] and [31]. For problems with a very large number of local minima the Bayesian termination criterion does not produce reliable results [16]. Because of this reason a standard multistart method which does not use clustering and Bayesian stopping techniques was also successfully tested for location of the the global minimum. Results are presented elsewhere [13].

The `schubert1` function has also a very large number of local minima, however it belongs to a class of "easy" problems [31]. Our results show that it was much easier to solve this problem in terms of the required N and N_r then `griewank1`, n=2, although these two problems have a comparable number of local minima. The `schubert2` problem was more difficult to solve then the `schubert1` one: the required N and N_r were approximately one order of magnitude higher then those for the `schubert1` function. It is in line with the increase in the number of local minima from 5^3 for `schubert1` to 15^5 for `schubert2`.

All mixed-integer problems problems, and the relatively high dimensional `blending1` and `blending1` were easy to solve. The `yuan2` problem was the most difficult in terms of the required N, although the presented CPU time does not show the complexity of the problem because of its low dimensionality.

7 Conclusion

In this study deterministic sBB and the QMC variant of the stochastic MLSL optimization method were compared with respect to their practical performance on a number of constrained and unconstrained continuous and mixed-integer problems of various complexity. General performance criteria such as the number of calls of a local minimizer and the CPU time were used for comparison. We can conclude that the QMC variant of stochastic MLSL method is capable of finding a global minimum with a high probability in a reasonable amount of computer time. This method is generally more efficient than the sBB method, although in some special instances the sBB method can exhibit superior performance due to the structure of problems. This work provides general guidelines with regard to the performance of the considered algorithms in global optimization.

8 Acknowledgments

We are grateful to Prof. Costas Pantelides for his support and interest in this work. One of the authors (SK) expresses his gratitude to I. M. Sobol' for numerous discussions and the invaluable help in improving the presented techniques.

One of the authors (SK) gratefully acknowledges the financial support of the United Kingdom's Engineering and Physical Sciences Research Council (EPSRC) under Platform Grant GR/N08636.

References

[1] C. Adjiman, S. Dallwig, C. Floudas and A. Neumaier, "A global optimization method, αBB, for general twice-differentiable constrained NLPs: I. Theoretical advances", Computers & Chemical Engineering, vol. 22(9), pp. 1137-1158, 1998.

[2] C. Adjiman, Global Optimization Techniques for Process Systems Engineering. PhD thesis, Princeton University, June 1998.

[3] N. Adhya, M. Tawarmalani, and N. Sahinidis, "A Lagrangian approach to the pooling problem," Industrial and Engineering Chemistry Research, vol. 38, pp. 1956-1972, 1999.

[4] M. Berkelaar, K. Eikland and P. Notebaert, LPSOLVE, `http://groups.yahoo.com/group/lp_solve`, 2004.

[5] A. Blake, "Comparison of the efficiency of deterministic and stochastic algorithms for visual reconstruction," IEEE Transactions on Pattern Analisys and Machine Intelligence, vol. 11, pp. 2-12, 1989.

[6] C.G.E. Boender, The Generalized Multinomial Distribution: A Bayesian Analysis and Applications, PhD thesis, Erasmus Universiteit Rotterdam and Centrum voor Wiskunde en Informatica Amsterdam, 1984.

[7] A. Decker and E. Aarts, "Global optimization and simulated annealing," Mathematical Programming, vol. 50, pp. 367-393, 1991.

[8] L. Dixon, J. Gomulka, and G. Szego, "Towards a global optimization technique," In L. Dixon and G. Szego, editors, Towards Global Optimization, pages 29-54, Amsterdam, 1975. North Holland.

[9] T. Epperly and E. Pistikopoulos, "A reduced space branch and bound algorithm for global optimization," Journal of Global Optimization, vol. 11, pp. 287-311, 1997.

[10] L. Foulds, D. Haughland, and K. Jornsten, "A bilinear approach to the pooling problem," Optimization, vol. 24, pp. 165-180, 1992.

[11] C. Floudas, P. Pardalos, C. Adjiman, W. Esposito, Z. Gumus, S. Harding, J. Klepeis, C. Meyer, and C. Schweiger, Handbook of Test Problems in Local and Global Optimization. Kluwer Academic Publishers, Dordrecht, 1999.

[12] P. Gill, User's Guide for SNOPT 5.3. Systems Optimization Laboratory, Department of EESOR, Stanford University, California, Feb 1999.

[13] S. Kucherenko and Yu. Sytsko, "Application of deterministic low-discrepancy sequences to nonlinear global optimization problems," Computational Optimization and Applications (accepted for publication), 2004.

[14] L. Liberti, "Linearity embedded in nonconvex programs," Journal of Global Optimization (accepted for publication), 2004.

[15] L. Liberti, "Reduction constraints for the global optimization of NLPs," International Transactions in Operations Research, vol. 11, no. 1, pp. 34-41, 2004.

[16] L. Liberti and C. Pantelides, "Convex envelopes of monomials of odd degree," Journal of Global Optimization, vol. 25, pp. 157-168, 2003.

[17] L. Liberti and C.C. Pantelides, "An exact reformulation algorithm for large nonconvex NLPs inppving bilinear terms," Journal of Global Optimization (submitted), 2004.

[18] L. Liberti, P. Tsiakis, B. Keeping, and C. Pantelides, *ooOPS*. Centre for Process Systems Engineering, Chemical Engineering Department, Imperial College, London, UK, v. 1.24, Jan 2001.

[19] M. Locatelli, "A note on the Griewank test function," Journal of Global Optimization, vol. 25, pp. 169-174, 2003.

[20] G. McCormick, "Computability of global solutions to factorable nonconvex programs: Part i – convex underestimating problems," Mathematical Programming, vol. 10, pp. 146-175, 1976.

[21] Numerical Algorithms Group, NAG Library, `http://www.nag.co.uk`, 2002.

[22] A.H.G. Rinnooy-Kan and G.T. Timmer, "Stochastic global optimization methods; part i: Clustering methods," Mathematical Programming, vol. 39, pp. 27-56, 1987.

[23] A.H.G. Rinnooy-Kan and G.T. Timmer, "Stochastic global optimization methods; part ii: Multilevel methods," Mathematical Programming, vol. 39, pp. 57-78, 1987.

[24] S. Schaffler, "Unconstrained global optimization using stochastic integral equations," Technical Report, Technisce Universität München, Institut für Angewandte Mathematik und Statistik, 1994.

[25] F. Schoen, "Two-phase methods for global optimization," Handbook of Global Optimization 2: Heuristic Approaches (edited by P. Pardalos and E. Romeijn), Kluwer Academic Publishers 2, pp. 151-177, 2002.

[26] E. Smith, On the Optimal Design of Continuous Processes. PhD thesis, Imperial College of Science, Technology and Medicine, University of London, October 1996.

[27] I. Sobol', "On the distribution of points in a cube and the approximate evaluation of integrals," Computational Mathematics and Mathematical Physics, vol. 7, pp. 86-112, 1967.

[28] I. Sobol', "On quasi-Monte Carlo integrations," Mathematics and Computers in Simulation, vol. 47, pp. 103-112, 1998.

[29] E. Smith and C. Pantelides, "Global optimisation of nonconvex MINLPs," Computers and Chemical Engineering, vol. 21, pp. S791-S796, 1997.

[30] E. Smith and C. Pantelides, "A symbolic reformulation/spatial branch-and-bound algorithm for the global optimisation of nonconvex MINLPs," Computers and Chemical Engineering, vol. 23, pp. 457-478, 1999.

[31] A. Törn, M. Afi and S. Vjitanen, "Stochastic global optimization, Problem Classes and Solution Techniques," Journal of Global Optimization, vol. 14, pp. 437-447, 1999.

[32] A. Törn and A. Žilinskas, Global Optimization. Springer-Verlag, Berlin, 1989.

[33] V. Visweswaran and C. Floudas, "New formulations and branching strategies for the gop algorithm." In I.E. Grossmann, editor, Global Optimization in Engineering Design, Dordrecht, 1996. Kluwer Academic Publishers.

[34] Z.B. Zabinsky, "Stochastic methods for practical global optimization," Journal of Global Optimization, vol. 13, pp. 433-444, 1998.