

# Comparison of deterministic and stochastic approaches to global optimization

Leo Liberti<sup>a</sup> and Sergei Kucherenko<sup>b</sup>

<sup>a</sup>*DEI, Politecnico di Milano, P.zza L. da Vinci 32, 20133 Milano, Italy,*

<sup>b</sup>*CPSE, Imperial College London, London SW7 2BY, UK*

*E-mails: liberti@elet.polimi.it [Liberti]; s.kucherenko@imperial.ac.uk [Kucherenko]*

Received 14 December 2004; received in revised form 9 February 2005; accepted 22 February 2005

---

## Abstract

In this paper, we compare two different approaches to nonconvex global optimization. The first one is a deterministic spatial Branch-and-Bound algorithm, 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 generally faster, although in some instances the Branch-and-Bound algorithm outperforms it.

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

---

## 1. Introduction

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

$$\begin{aligned} \min_x & f(x) \\ l & \leq g(x) \leq u, \\ x^L & \leq x \leq x^U, \end{aligned} \tag{1}$$

where  $x \in \mathbb{R}^n$ ,  $f: \mathbb{R} \rightarrow \mathbb{R}$ ,  $g: \mathbb{R}^n \rightarrow \mathbb{R}^m$ ,  $l, u \in \mathbb{R}^m$  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 and continuous.

Global optimization algorithms are usually broadly divided into deterministic and stochastic. Deterministic methods provide a theoretical guarantee of locating the  $\varepsilon$ -global optimum, i.e. a local optimum whose objective function value differs by at most  $\varepsilon$  from the global one for a given  $\varepsilon > 0$ . Stochastic methods only offer a guarantee in probability: their convergence proofs usually state that the global optimum will be identified in infinite time with probability 1. Stochastic methods, moreover, 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 (GNC) with different versions of simulated annealing stochastic algorithms was performed in Blake (1989) (however, it must be pointed out that GNC is a deterministic heuristic method, rather than an exact method); the author's conclusion was that on the particular class of problems considered in the paper, stochastic algorithms are less efficient. This result can be partially explained by the inefficiency of the chosen stochastic algorithms: Decker and Aarts (1991), for example, found that the stochastic multi level single linkage (MLSL) algorithm is more efficient than simulated annealing algorithms. Dixon et al. (1975) compared the deterministic interval method with the stochastic MLSL algorithm. Their conclusion was that the stochastic algorithm was at least eight times faster than the deterministic one. Zabinsky (1998) reported results with several deterministic methods, including a branch-and-bound scheme coupled with gradient search local optimization methods. The 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 as the only viable alternative for global optimization. Good results were obtained with the Improving Hit-and-Run method, which is a version of sequential random search methods. Of course, nowadays we have deterministic methods that can find the global optimum of practical problems with nearly 200 variables (see the Molecular Distance Geometry Problem (MDGP) instances (Törn et al., 1999)), so a new assessment as regards the comparison between deterministic and stochastic methods for global optimization is in order.

In this paper, we compare the performances of a deterministic spatial Branch-and-Bound (sBB) method and a Quasi Monte Carlo (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, Adjiman et al., 1998; Adjiman, 1998; Epperly and Pistikopoulos, 1997) 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  $\varepsilon$  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  $\varepsilon$ , a branching variable and a branching point are chosen and the region is split into 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 Smith (1996), Smith and Pantelides (1997) and Smith and Pantelides (1999). In sBB algorithms, the time taken to find the solution 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 Rinnooy-Kan and Timmer (1987a, b), Schoen (2002) and Törn and Žilinskas (1989). One of the most efficient clustering methods is the MLSL algorithm developed by Rinnooy-Kan and Timmer (1987a, b). 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 Kucherenko and Sytsko (2005) that application of LDS can significantly increase the efficiency of MLSL methods.

## 2. Deterministic sBB algorithm

The sBB algorithm used in these tests (based on Smith and Pantelides, 1999) 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  $\varepsilon$  and the best objective function value  $U := +\infty$ . Optionally perform pre-processing steps (such as optimization- and feasibility-based bounds tightening; Smith, 1996).

3. *Choice of Region.* If the list of regions is empty, terminate the algorithm with solution  $U$ . Otherwise, choose a region (the *current region*) and remove it from the list. Optionally perform feasibility-based bounds tightening on the current region.
4. *Lower Bound.* Generate a (linear) convex relaxation of the original problem subject to the variable ranges given by the current region 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 and 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. To this effect, in step 8 the lower bound  $l$  for the current region is set as the initial lower bound for the two “children” subregions.

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 (Liberti, 2004, 2005; Liberti and Pantelides, 2005). Here, only a short introduction is given. Let  $n \in \mathbb{N}$ ,  $x_1, \dots, x_n$ ,  $w_1, \dots, w_n$ ,  $y$  be problem variables and suppose that the feasible region of an optimization problem is defined by a set of constraints that include the following:

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

where  $a_1, \dots, a_n, b$  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

$$\sum_{i=1}^n a_i w_i = b y \quad (2)$$

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  $\sum_{i=1}^n a_i w_i = b y$  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

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

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 (Equations (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:

$$\min x_{\text{obj}}, \quad (3)$$

$$l \leq Ax \leq u, \quad (4)$$

$$x_k = x_i x_j, \quad \forall (i, j, k) \in \mathcal{M}, \quad (5)$$

$$x_k = \frac{x_i}{x_j}, \quad \forall (i, j, k) \in \mathcal{D}, \quad (6)$$

$$x_k = x_i^n, \quad \forall (i, k) \in \mathcal{P}, \quad (7)$$

$$x_k = f_i(x_i), \quad \forall (i, k) \in \mathcal{U}, \quad (8)$$

$$x^L \leq x \leq x^U, \quad (9)$$

where  $x = (x_1, \dots, x_n)$  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

added variable = operand (binary operator) operand,

added variable = (unary operator) operand,

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  $x_{\text{obj}}$ , and a constraint of the form

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

has been added to the problem constraints.

The algorithm for reducing a Mixed-integer nonlinear programming problem (MINLP) to standard form is described in detail in Smith (1996) and Smith and Pantelides (1999). 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  $x_i = v(x_j, x_k)$  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 envelope, McCormick, 1976)

$$x_i \geq x_j^L x_k + x_k^L x_j - x_j^L x_k^L, \quad (10)$$

$$x_i \geq x_j^U x_k + x_k^U x_j - x_j^U x_k^U, \quad (11)$$

$$x_i \leq x_j^L x_k + x_k^U x_j - x_j^L x_k^U, \quad (12)$$

$$x_i \leq x_j^U x_k + x_k^L x_j - x_j^U x_k^L, \quad (13)$$

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

$$x_i \leq f(x_j), \quad (14)$$

$$x_i \geq f(x_j^L) + \frac{f(x_j^U) - f(x_j^L)}{x_j^U - x_j^L} (x_j - x_j^L) \quad (15)$$

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

$$x_i \leq f(x_j^L) + \frac{f(x_j^U) - f(x_j^L)}{x_j^U - x_j^L} (x_j - x_j^L) \quad (16)$$

$$x_i \geq f(x_j), \quad (17)$$

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 Liberti and Pantelides (2003).

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 multi level 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 (Rinnooy-Kan and Timmer, 1987a, b) 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:

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

where

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

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 and  $\sigma$  is a known parameter (Rinnooy-Kan and Timmer, 1987a, b). In our calculations  $\sigma$  was equal to 2.

Some improvement in efficiency can be achieved by selecting points with the lowest objective function values 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  $f(x_i) \leq f(x_{i+1})$  for all  $i < N$ . The reduced sample set is defined as:

$$X_r = \{x_i \in X | i = 1, \dots, N_r\}, \quad (19)$$

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 Boender (1984). 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

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

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

### 3.1. The MLSL algorithm

The general scheme of the MLSL algorithm is outlined below:

1. Set  $W = 0$ ,  $k = 0$ . Set  $k_{\text{max}}$  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  $q(x_i, x_j) \leq r_k, f(x_j) \leq f(x_i)$ , 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 (Sobol', 1967, 1998). 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 Törn et al. (1999) 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 Törn et al. (1999).

### 4.1. Six-hump camel back function

The “six-hump camel back function”, introduced in Dixon et al. (1975), is a well known test for global optimization algorithms.

$$\min_{-1 \leq x_1, x_2 \leq 4} 4x_1^2 - \frac{21}{10}x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4. \quad (21)$$

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 Törn et al. (1999), this problem belongs to a class of “easy” (E1) problems.

Table 1  
Problem statistics

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
knp24 (32)	97	300
knp25 (32)	101	325
mdgp90 (33)	90	0
mdgp192 (33)	192	0

#### 4.2. Schaffler function

This test function was introduced in Schaffler (1994).

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

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

problem:

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

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

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

having an objective function value 0.3794.

#### 4.3. Griewank function

This test function was introduced in Törn and Žilinskas (1989). The formulation below depends on an integer  $n$  that can take two values (2 and 10).

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

where  $d = 200$ ,  $r = 100$  for  $n = 2$ , and  $d = 4000$ ,  $r = 600$  for  $n = 10$ . Both instances belong to the class of “moderate” (M2) problems (Törn et al., 1999).

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

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

where we have replaced the minus sign in front of the cosine product term with a plus sign. See Locatelli (2003) 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  $x = (\pm 3.1400, 0.0, \dots, 0.0)$  and has an objective function value 0.0024. The trend, for increasing values of  $n$  and  $d$ , is for  $x_1$  to tend to  $\pi$  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 Decker and Aarts (1991). The first is defined as

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

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

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

with  $n = 5$ .

The global optima of these problems are located at  $x = (1, \dots, 1)$  with an objective function value of 0. Both problems belong to the class of “easy” (E2) problems (Törn et al., 1999).

#### 4.5. Adhya's multi-quality blending problems

These two blending problems from the petrochemical industry are taken from Adhya et al. (1999), where they were named `example1` and `example2`, 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 (Adhya et al., 1999; Foulds et al., 1992; Visweswaran and Floudas, 1996).

Here, we use the general blending problem formulation found in Adhya et al. (1999, p. 1958):

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

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 Adhya et al. (1999, 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:

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

whereas the values of the quality variables  $q$  change:

$$q' = (3.1708, 2.382, 3.2764, 1.5854, 2.278, 2.8917, 3.361, 1.2166),$$

$$q'' = (3.1708, 2.382, 3.2764, 1.5854, 4.2763, 5.382, 2.278, 2.8917, 3.361, 1.2166, 3, 5.083),$$

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 Liberti (2005). 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  $x_5^2 = x_5$ .

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

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 Floudas et al. (1999). 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  $y$ .

$$\begin{aligned}
& \min_{x,y} 2x_1 + 3x_2 + \frac{3}{2}y_1 + 2y_2 - \frac{1}{2}y_3, \\
& y_1 + x_1^2 = \frac{5}{4}, \\
& \frac{3}{2}y_2 + x_2^{\frac{3}{2}} = 3, \\
& y_1 + x_1 \leq \frac{8}{5}, \\
& y_2 + \frac{4}{3}x_5 \leq 3, \\
& -y_1 - y_2 + y_3 \leq 0, \\
& 0 \leq x \leq 10, y \in \{0, 1\}^3.
\end{aligned} \tag{30}$$

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

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

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.

#### 4.8. The kissing number problem

The kissing number problem in  $D$  dimensions consists in finding the maximum number of balls of like radius that can be arranged around a central ball of the same radius. This problem has a noble pedigree, having been first stated by Newton. See Liberti et al. (2005) for some historical remarks and a more complete discussion. The surrounding balls must touch the central balls but must not overlap among them; this arrangement is termed a “kissing configuration”. The problem can be formulated as a nonconvex constrained problem with continuous variables that finds an arrangement of  $K$  kissing balls in  $D$  dimensions. The arrangement is feasible/infeasible according to the globally optimal value of a feasibility parameter  $\alpha$  in the objective function. By a simple bisection method, with a few global optimization runs, we arrive at a number  $K$  such that it is possible to arrange  $K$  balls in  $D$  dimensions in a kissing configuration, but it is not possible to arrange  $K+1$  balls. Thus, we can determine the kissing number  $K$ . The model, for balls of radius 1 centered in  $x^i = (x_1^i, \dots, x_D^i)$ , is as follows:

$$\begin{aligned} \max \quad & \alpha \\ \text{s.t.} \quad & \|x^i\|^2 = 4, \quad \forall i \leq K \\ & \|x^i - x^j\|^2 \geq 4\alpha, \quad \forall i < j \leq K. \end{aligned} \quad (32)$$

Thus, if the optimal value for  $\alpha$  is at least 1, then there is a kissing configuration of  $K$  balls, otherwise there is not. It is known that 24 is a lower bound for four dimensions and 25 is an upper bound. In order to determine the kissing numbers for  $D=4$ , therefore, we must solve two problems. The optimal value  $\alpha$  for  $K=24$  is 1, for  $K=25$  it is 0.92357 (from this it follows that the kissing number in four dimensions is 24).

#### 4.9. MDGP

The MDGP is the problem of determining the three-dimensional structure of a molecule where a subset of the atomic distances is known. Formally, we need to find vectors  $x_1, \dots, x_n \in \mathbb{R}^3$ , which describe the three-dimensional position of each atom in the molecule, such that:

$$\forall \{i, j\} \in S \quad (\|x_i - x_j\| = d_{ij}),$$

where  $S$  is the subset of pairs of atoms  $\{i, j\}$  whose distances  $d_{ij}$  are known. We address the problem in terms of finding the global minimizer of the function

$$f(x_1, \dots, x_n) = \sum_{\{i, j\} \in S} (\|x_i - x_j\|^2 - d_{ij}^2)^2. \quad (33)$$

It is easy to verify that  $x_1, \dots, x_n \in \mathbb{R}^3$  solve the problem if and only if  $f(x_1, \dots, x_n) = 0$ . See Lavor et al. (2005) for more information about the MDGP and about the instances used (in particular, the two instances `mdgp90` (with 30 atoms and 90 variables) and `mdgp192` (with 64 atoms and 192 variables) have been generated automatically with different methods).

## 5. Implementation of the benchmark: *ooOPS*

Both algorithms have been tested and benchmarked with the help of *ooOPS*, a software framework for optimization (Liberti et al., 2001). *ooOPS* 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 *ooOPS* interfaces to three global optimization solvers (the two methods explained in this paper and a Variable Neighbourhood Search solver) and three local solvers (SNOPT (Gill, 1999)), the NAG e04vcf nonlinear local solver (Numerical Algorithms Group, 2002) and LPSOLVE (Berkelaar et al., 2004), 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 that only apply to one method or the other, but not to both. Thus, we shall present the results in three tables. In Table 2, we have listed the common performance criteria; Table 3 refers to the deterministic sBB algorithm, while Table 4 refers to the QMC variant of the MLSL algorithm. All of the tests have been carried out under the *ooOPS* framework running under Linux on a Pentium III 850 MHz PC with 384 MB 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)

Table 2

Performance comparison of the two methods ([hh:]mm:ss)

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	> 10 h	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
knnp24 (32)	NC	Yes	> 10 h	5:34	–	512
knnp25 (32)	NC	Yes	> 10 h	6:10	–	512
mdgp90 (33)	Yes	Yes	26:54	1:29:52	92	320
mdgp192 (33)	Yes	Yes	2:03:04	2:32:04	120	320

NC, method did not converge within the time specified in the CPU time column; Det., deterministic; Stoch., stochastic; NLP, nonlinear programming problems.

with  $n = 3, 4, 5, 6$ . The deterministic method showed better performance than the stochastic method in the following instances: schaffler1 with  $n = 30, 40, 50$  (22), schubert2 (27), and mdgp90, mdgp192 (33). For all other cases the stochastic method showed a superior performance. There were also four 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.

Table 3

Results obtained with the deterministic branch-and-bound method

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
$n = 8$	14,883	1	0
$n = 9$	35,167	1	0
$n = 10$	71,707	1	0
griewank1 (24)			
$n = 2$	11	7	0
$n = 10$	25	15	0
griewank2 (25)			
$n = 2$	61,345	24	0
$n = 10$	> 360,000	14,124	0
schubert1 (26)	> 105,000	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
knp24 (32)	NA	—	0
knp25 (32)	NA	—	0
mdgp90 (33)	94	94	0
mdgp192 (33)	122	122	0

NA, not applicable.

### 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  $\varepsilon$  tolerance (see step 7 of the sBB algorithm in Section 2), which was set to  $1 \times 10^{-6}$  for the whole test suite. It is a typical feature of Branch-and-Bound algorithms 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, because of 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 and 10 are those where the local optimization procedure exhibited the highest instability. We were, however, not able to ascertain the exact cause of this occurrence. Note that in the instance  $n = 3$ , a local optimum was located rather than the global one (thus the column “Glob. Opt. Iteration” is marked as “not applicable” (NA)).

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 because of 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 because of reduction constraints pre-processing and/or effective bounds tightening techniques (steps 1 and 3 of the sBB algorithm in Section 2).

Unfortunately, none of the *knp* instances was solved by the sBB solver in the time allowed (10h), because of the fact that the convex relaxation was extremely slack.

For the *mdgp* we have the unusual situation where the sBB solver is faster than the MLSL solver. This is mainly because of a very effective fathoming procedure based on the fact that the convexification always attains the optimum with an objective function value of 0. In other words, once the upper bound becomes 0 in a region, immediately all other regions are discarded (this explains the fact that the number of iterations is equal to the iteration counter at which the global optimum is found).

## 6.2. Tests with the QMC variant of MLSL method

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

- The column *Problems* lists the names of the problems as assigned throughout this paper.

Table 4

Results obtained with the QMC variant of the MLSL method

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
$n = 40$	16,384	4096	1	4
$n = 50$	1,048,576	16,384	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)				
$n = 2$	32,768	128	2	23
$n = 10$	32,768	128	2	14
griewank2 (25)				
$n = 2$	32,768	128	5	28
$n = 10$	32,768	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
knp24 (32)	1024	256	2	512
knp25 (32)	1024	256	2	512
mdgp90 (33)	64	32	10	320
mdgp192 (33)	64	32	10	320

QMC, quasi Monte Carlo; MLSL, multi level single linkage.

- The column  $N$  lists the total number of sampled points at each iteration. For the Sobol' LDS 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 problem 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 affects only functions with all variables being equally important (Sobol', 1998), 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, and 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 Rinnooy-Kan and Timmer (1987a, b) and Törn et al. (1999). For problems with a very large number of local minima the Bayesian termination criterion does not produce reliable results (Liberti, 2004). For this reason a standard multistart method which does not use clustering and Bayesian stopping techniques was also successfully tested for location of the global minimum. Results are presented elsewhere (Kucherenko and Sytsko, 2005).

The *schubert1* function also has a very large number of local minima, however it belongs to a class of “easy” problems (Törn et al., 1999). Our results show that it was much easier to solve this problem in terms of the required  $N$  and  $N_r$  than *griewank1*,  $n = 2$ , although these two problems have a comparable number of local minima. The *schubert2* problem was more difficult to solve than the *schubert1* one: the required  $N$  and  $N_r$  were approximately one order of magnitude higher than 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, 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.

The kissing number problem in  $D = 4$  dimensions is a difficult high dimensional problem which was solved for the first time using formulation (32) by the QMC variant of MLSL method (Liberti et al., 2005).

Problems *mdgp90* and *mdgp192* belong to the class of difficult high dimensional problems with a very large number of local minima and with an objective function with equally important variables. As was mentioned before for such problems, the QMC variant of MLSL method

becomes less efficient than for other types of objective functions. Solution of such problems generally require high CPU time.

## 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 because of the structure of problems. More precisely, the sBB algorithm may be more efficient when the optimal objective function value of the convex relaxation is equal to the optimal objective function value of the original problem. Although this is a stringent requirement, important real-life problems such as the MDGP exhibit this property.

## Acknowledgments

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

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

## References

- Adhya, N., Tawarmalani, M., Sahinidis, N., 1999. A Lagrangian approach to the pooling problem. *Industrial and Engineering Chemistry Research*, 38, 1956–1972.
- Adjiman, C., 1998. *Global optimization techniques for process systems engineering*. PhD Thesis, Princeton University.
- Adjiman, C., Dallwig, S., Floudas, C., Neumaier, A., 1998. A global optimization method,  $\alpha$ BB, for general twice-differentiable constrained NLPs: I. Theoretical advances. *Computers & Chemical Engineering*, 22, 9, 1137–1158.
- Berkelaar, M., Eikland, K., Notebaert, P., 2004. lpsolve, [http://groups.yahoo.com/group/lp\\_solve](http://groups.yahoo.com/group/lp_solve)
- Blake, A., 1989. Comparison of the efficiency of deterministic and stochastic algorithms for visual reconstruction. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11, 2–12.
- Boender, C.G.E., 1984. *The generalized multinomial distribution: a Bayesian analysis and applications*. PhD Thesis, Erasmus Universiteit, Rotterdam and Centrum voor Wiskunde en Informatica, Amsterdam.
- Decker, A., Aarts, E., 1991. Global optimization and simulated annealing. *Mathematical Programming*, 50, 367–393.
- Dixon, L., Gomulka, J., Szego, G., 1975. Towards a global optimization technique. In Dixon, L., Szego, G. (eds) *Towards Global Optimization*. North-Holland, Amsterdam, pp. 29–54.

- Epperly, T., Pistikopoulos, E., 1997. A reduced space branch and bound algorithm for global optimization. *Journal of Global Optimization*, 11, 287–311.
- Floudas, C., Pardalos, P., Adjiman, C., Esposito, W., Gumus, Z., Harding, S., Klepeis, J., Meyer, C., Schweiger, C., 1999. *Handbook of Test Problems in Local and Global Optimization*. Kluwer Academic Publishers, Dordrecht.
- Foulds, L., Haughland, D., Jornsten, K., 1992. A bilinear approach to the pooling problem. *Optimization*, 24, 165–180.
- Gill, P., 1999. *User's Guide for SNOPT 5.3*. Systems Optimization Laboratory, Department of EESOR, Stanford University, California.
- Kucherenko, S., Sytsko, Yu., 2005. Application of deterministic low-discrepancy sequences to nonlinear global optimization problems. *Computational Optimization and Applications* (accepted for publication).
- Lavor, C., Liberti, L., Maculan, N., 2005. Computational experience with the molecular distance geometry problem. In Pintér, J. (ed.) *Global Optimization: Selected Case Studies*. Kluwer, Dordrecht.
- Liberti, L., 2004. Reduction constraints for the global optimization of NLPs. *International Transactions in Operations Research*, 11, 1, 34–41.
- Liberti, L., 2005. Linearity embedded in nonconvex programs. *Journal of Global Optimization* (accepted for publication).
- Liberti, L., Pantelides, C., 2003. Convex envelopes of monomials of odd degree. *Journal of Global Optimization*, 25, 157–168.
- Liberti, L., Pantelides, C.C., 2005. An exact reformulation algorithm for large nonconvex NLPs involving bilinear terms. *Journal of Global Optimization* (submitted).
- Liberti, L., Tsiakis, P., Keeping, B., Pantelides, C., 2001. *ooCPP*. Centre for Process Systems Engineering, Chemical Engineering Department, Imperial College, London, UK, v. 1.24.
- Liberti, L., Maculan, N., Kucherenko, S., 2005. The kissing number problem: a new result from global optimization. *Electronic Notes in Discrete Mathematics*, 17.
- Locatelli, M., 2003. A note on the Griewank test function. *Journal of Global Optimization*, 25, 169–174.
- McCormick, G., 1976. Computability of global solutions to factorable nonconvex programs: part I – convex underestimating problems. *Mathematical Programming*, 10, 146–175.
- Numerical Algorithms Group 2002. NAG Library, <http://www.nag.co.uk>
- Rinnooy-Kan, A.H.G., Timmer, G.T., 1987a. Stochastic global optimization methods; part I: clustering methods. *Mathematical Programming*, 39, 27–56.
- Rinnooy-Kan, A.H.G., Timmer, G.T., 1987b. Stochastic global optimization methods; part II: multilevel methods. *Mathematical Programming*, 39, 57–78.
- Schaffler, S., 1994. *Unconstrained global optimization using stochastic integral equations*. Technical Report, Technische Universität München, Institut für Angewandte Mathematik und Statistik.
- Schoen, F., 2002. Two-phase methods for global optimization. In Pardalos, P., Romeijn, E. (eds) *Handbook of Global Optimization 2: Heuristic Approaches*, Vol. 2. Kluwer Academic Publishers, Dordrecht, pp. 151–177.
- Smith, E., 1996. *On the optimal design of continuous processes*. PhD Thesis, Imperial College of Science, Technology and Medicine, University of London.
- Smith, E., Pantelides, C., 1997. Global optimisation of nonconvex MINLPs. *Computers and Chemical Engineering*, 21, S791–S796.
- Smith, E., Pantelides, C., 1999. A symbolic reformulation/spatial branch-and-bound algorithm for the global optimisation of nonconvex MINLPs. *Computers and Chemical Engineering*, 23, 457–478.
- Sobol', I., 1967. On the distribution of points in a cube and the approximate evaluation of integrals. *Computational Mathematics and Mathematical Physics*, 7, 86–112.
- Sobol', I., 1998. On quasi-Monte Carlo integrations. *Mathematics and Computers in Simulation*, 47, 103–112.
- Törn, A., Žilinskas, A., 1989. *Global Optimization*. Springer-Verlag, Berlin.
- Törn, A., Ali, M., Vjitanen, S., 1999. Stochastic global optimization, problem classes and solution techniques. *Journal of Global Optimization*, 14, 437–447.
- Visweswaran, V., Floudas, C., 1996. New formulations and branching strategies for the gop algorithm. In Grossmann, I.E. (ed.) *Global Optimization in Engineering Design*. Kluwer Academic Publishers, Dordrecht.
- Zabinsky, Z.B., 1998. Stochastic methods for practical global optimization. *Journal of Global Optimization*, 13, 433–444.