



# A direct stochastic algorithm for global search

B. Raphael<sup>\*</sup>, I.F.C. Smith

*IMAC, EPFL-Federal Institute of Technology, Lausanne CH-1015, Switzerland*

---

## Abstract

This paper presents a new algorithm called probabilistic global search lausanne (PGSL). PGSL is founded on the assumption that optimal solutions can be identified through focusing search around sets of good solutions. Tests on benchmark problems having multi-parameter non-linear objective functions revealed that PGSL performs better than genetic algorithms and advanced algorithms for simulated annealing in 19 out of 23 cases studied. Furthermore as problem sizes increase, PGSL performs increasingly better than these other approaches. Empirical evidence of the convergence of PGSL is provided through its application to Lennard–Jones cluster optimisation problem. Finally, PGSL has already proved to be valuable for engineering tasks in areas of design, diagnosis and control.

© 2002 Elsevier Inc. All rights reserved.

*Keywords:* Global optimisation; Stochastic search; Random search

---

## 1. Introduction

Search methods are gaining interest with the increase in activities related to modelling complex systems. Although many methods have been proposed, difficulties related to computation time and reliability remain. Often methods do not scale up well when applied to full scale practical applications.

Probabilistic methods have been successfully applied to complex engineering and scientific tasks where near optimal solutions are sufficient. Well known methods that have been applied to complex tasks include

---

<sup>\*</sup> Corresponding author.

*E-mail addresses:* [benny.raaphael@epfl.ch](mailto:benny.raaphael@epfl.ch) (B. Raphael), [ian.smith@epfl.ch](mailto:ian.smith@epfl.ch) (I.F.C. Smith).

- Simulated annealing.
- Genetic algorithms.
- Adaptive random search.
- Multiple random starts with local search.

Methods that make use of gradients are not included in this list since most practical applications involve objective functions which cannot be expressed in explicit mathematical forms and their derivatives cannot be easily computed. Similarly methods that approximate objective functions using surrogate models are also excluded since these approximations work only in ideal conditions.

We are interested in direct search methods [1] as defined below:

A direct method for numerical optimisation is any algorithm that depends on the objective function only through ranking a countable set of function values.

Direct methods do not compute or approximate values of derivatives. They use the value of the objective function only to determine whether a point ranks higher than other points.

This paper proposes a new direct search method that performs better than others for difficult bench mark problems that have been published from 1995 to 2000. Section 2 contains a review of existing search techniques. The following section describes the details of the PGSL algorithm. In Section 4, performance is compared with other algorithms. Finally, Section 5 contains a discussion of limitations and practical applications where improved performance has already been achieved.

## 2. Existing search techniques

The most widely used search methods in engineering applications are simulated annealing [2] and genetic algorithms [3]. Since these are well known, they are not described here. The following paragraphs contain brief summaries of selected search methods.

*Adaptive random search:* Pure random search procedures have been used for optimisation problems as early as 1958 [4]. These techniques are attractive due to their simplicity. However, they converge extremely slowly to a global optimum in parameter spaces of many dimensions. In order to improve convergence, “random creep” procedures are used in which exploratory steps are limited to a hyper-sphere centred about the latest successful point. Masri and Beki [5] have proposed an algorithm called adaptive random search (ARS) in which the step size of the random search procedure is optimised periodically

throughout the search process. Controlled random search (CRS) [6,7] is another search method that samples points in the neighbourhood of the current point through the use of a probability density function (PDF).

*Multiple random starts with local search:* Local search techniques involve iteratively improving upon a solution point by searching in its neighbourhood for better solutions. If better solutions are not found, the process terminates; the current point is taken as a locally optimal solution. Since local search performs poorly when there are multiple local optima, a modification to this technique has been suggested in which local search is repeated several times using randomly selected starting points. This process is computationally expensive because after every iteration, the search re-starts from a point possibly far away from the optimum. Also search might converge to the same point obtained in a previous iteration. Furthermore, no information that has been obtained from previous iterations is reused. Random bit climbing (RBC) [8] is a form of local search in which neighbouring points are randomly evaluated and the first move producing an improvement is accepted for the next stage.

*Improving hit and run:* The basic structure of improving hit and run (IHR) [9] is to generate a random direction followed by a candidate point that is along a random step in that direction. A positive definite matrix  $\mathbf{H}$  in the algorithm controls the direction distribution. If the matrix  $\mathbf{H}$  is the identity matrix, then the direction distribution is uniform on a hyper-sphere. In practice,  $\mathbf{H}$  is locally estimated in a similar way to derivative-based local search procedures.

IHR has polynomial complexity with respect to the number of variables for the class of elliptical programs. This complexity is attainable for strictly convex quadratic programs by choosing  $\mathbf{H}$  to be the Hessian of the objective function. IHR is an approximation of pure adaptive search (PAS). PAS is an idealistic procedure used to model realistic algorithms and to analyse their complexity [12]. PAS involves generating a sequence of improving points in the feasible region with the property that each point is uniformly distributed in the level set corresponding to its predecessor. Zabinsky and Smith [10] have shown that under certain conditions the number of iterations required grows only linearly with respect to the number of variables. The main challenge associated with implementing PAS is the difficulty of generating a point in each iteration that is uniformly distributed in the improving region. Recently, algorithms such as random ball walk Markov chain sampling have been used to generate nearly uniform points in a convex region [11]. Uniform covering by probabilistic rejection [12] is another algorithm that aims to realise PAS. Hesitant adaptive search (HAS) [13] is an extension of PAS that allows hesitation or pausing at the current level as the algorithm progresses. HAS is capable of modelling random search algorithms such as simulated annealing better than PAS.

### 3. Probabilistic global search lausanne

The PGSL algorithm was developed starting from the observation that optimally directed solutions can be obtained efficiently through carefully sampling the search space without using special operators. The principal assumption is that better points are likely to be found in the neighbourhood of families of good points. Hence, search is intensified in regions containing good solutions.

The search space is sampled by means of a PDF defined over the entire search space. Each axis is divided into a fixed number of intervals and a uniform probability distribution is initially assumed. As search progresses, intervals and probabilities are dynamically updated so that sets of points are generated with higher probability in regions containing good solutions. The search space is gradually narrowed down so that convergence is achieved.

The algorithm includes four nested cycles:

- Sampling cycle.
- Probability updating cycle.
- Focusing cycle.
- Subdomain cycle.

In the sampling cycle (innermost cycle) a certain number of samples,  $NS$ , are generated randomly according to the current PDF. Each point is evaluated by the user-defined objective function and the best point is selected. In the next cycle, probabilities of regions containing good solutions are increased and probabilities decreased in regions containing less attractive solutions. In the third cycle, search is focused on the interval containing the best solution after a number of probability updating cycles, by further subdivision of the interval. In the subdomain cycle, the search space is progressively narrowed by selecting a subdomain of smaller size centred on the best point after each focusing cycle.

Each cycle serves a different purpose in the search for a global optimum. The sampling cycle permits a more uniform and exhaustive search over the entire search space than other cycles. Probability updating and focusing cycles refine search in the neighbourhood of good solutions. Convergence is achieved by means of the subdomain cycle.

#### 3.1. Terminology

The following definitions are used in the description of PGSL:

*Solution point:* A point consists of a set of values for each variable.

*Search space:* The set of all potential solution points. It is an  $N$ -dimensional space with an axis corresponding to each variable.  $N$  denotes the total number

of variables. The user defines the minimum and maximum values along each axis. A subset of the search space is called a subdomain.

*Axis width:* The difference between the minimum and the maximum along an axis of the search space or a subdomain.

*Cost function:* A user-supplied function to evaluate a solution point. The value of the cost function for a given point is called the *cost* or *evaluation* of the solution point.

*Probability density function, PDF:* The PDF of a variable is defined in the form of a histogram. The axis represented by the variable is discretised into a fixed number of intervals, NINTERVALS. Uniform probability distribution is assumed within each interval. The cumulative distribution function (CDF) is obtained by integrating the PDF.

Important parameters involved in the algorithm are listed below:

*Number of samples, NS:* The number of samples evaluated in the sampling cycle.

*Iterations in the probability updating cycle, NPUC:* The number of times the sampling cycle is repeated in a probability updating cycle.

*Iterations in the focusing cycle, NFC:* The number of times the probability updating cycle is repeated in a focusing cycle.

*Iterations in the subdomain cycle, NSDC:* The number of times the focusing cycle is repeated in a subdomain cycle.

*Subdomain scale factors, SDSF1, SDSF2:* The default factors for scaling down the axis width in the subdomain cycle. SDF1 is used when there is an improvement and SDF2 if there is no improvement.

### 3.2. Algorithm details

The algorithm is illustrated in the form of a flowchart in Figs. 1–3 and is explained in more detail next.

#### 3.2.1. Initialisation

The search space is defined by reading the minimum and maximum values for each variable given by the user. The PDF of each variable is created by assuming a uniform distribution over the entire domain. All PDFs have intervals of constant width in the beginning.

#### 3.2.2. Sampling cycle

NS points are generated randomly by generating a value for each variable according to its PDF. This is similar to the sampling in the Monte Carlo technique. Each point is evaluated and the point having the minimum cost, best sample (BS), is selected.

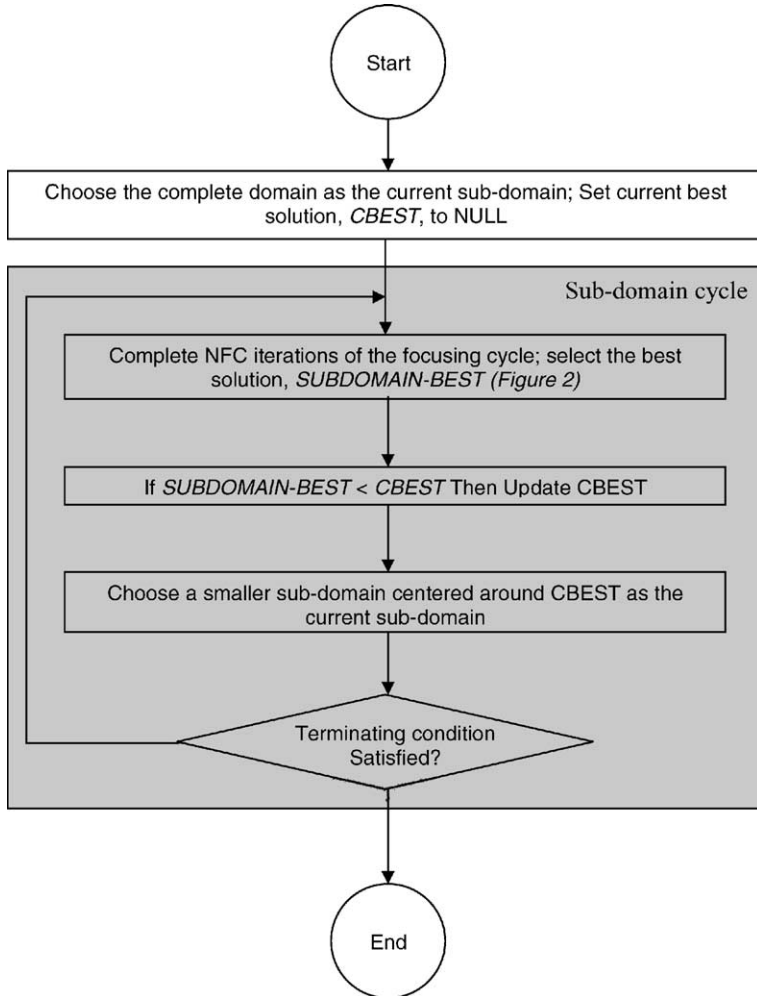


Fig. 1. The terminating condition for all cycles, except the sampling cycle, is the completion of the specified number of iterations or the value of the objective function becoming smaller than a user-defined threshold.

### 3.2.3. Probability updating cycle

The sampling cycle is invoked NPUC times. After each iteration, the PDF of each variable is modified using the probability-updating algorithm. This ensures that the sampling frequencies in regions containing good points are increased. The evolution of the PDF for a variable after several sampling cycles is illustrated in Fig. 4.

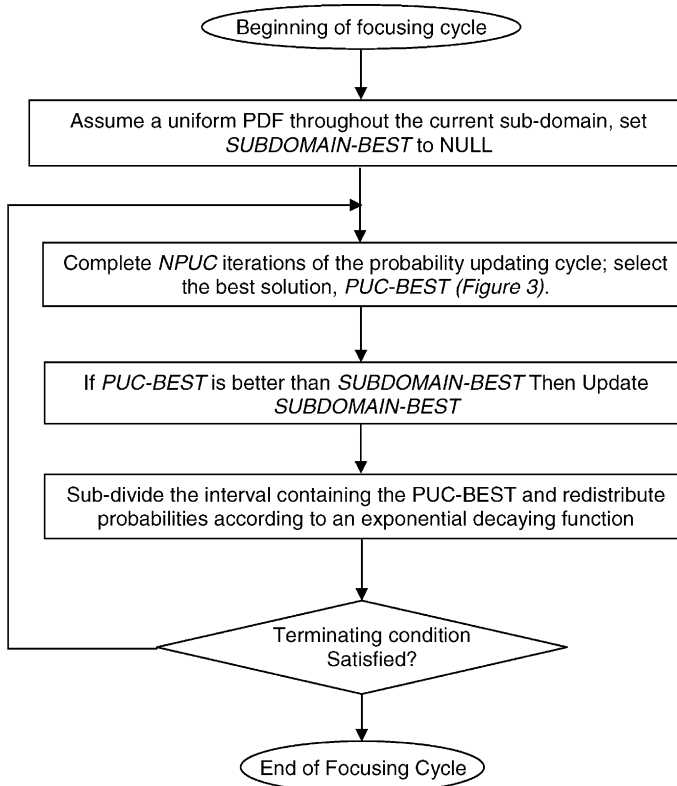


Fig. 2. The focusing cycle.

### 3.2.4. Probability-updating algorithm

The PDF of a variable is updated through these steps:

- Locate the interval containing the value of the variable in BS.
- Multiply the probability of the interval by a factor ( $>1$ ).
- Normalise the PDF.

### 3.2.5. Focusing cycle

The probability updating cycle is repeated NFC times. After each iteration, the search is increasingly focused on the interval containing the current best point, CBEST. This is done by subdividing the interval containing the value of each variable in CBEST. The evolution of the PDF after several probability updating cycles is illustrated in Fig. 5.

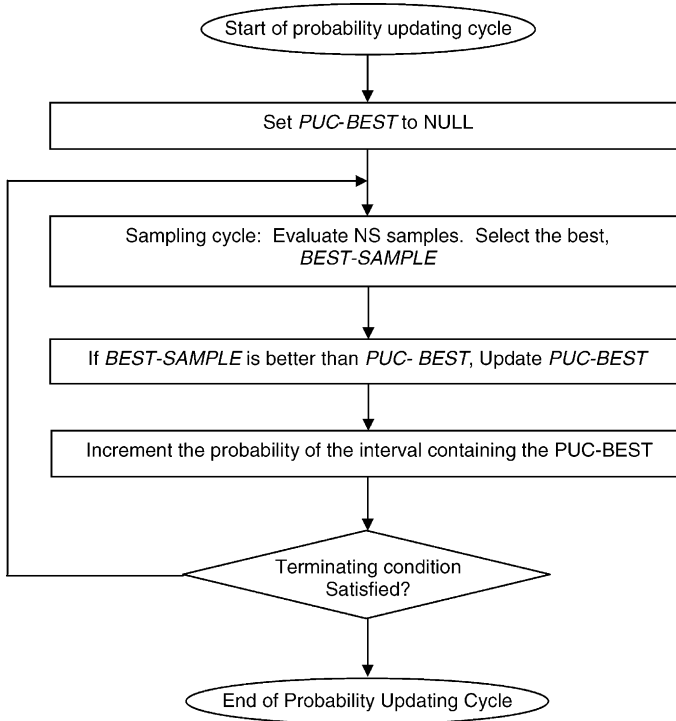


Fig. 3. The probability updating cycle.

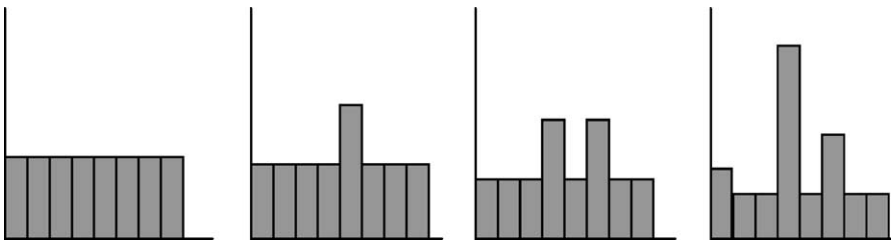


Fig. 4. Evolution of the PDF of a variable after several probability updating cycles.

### 3.2.6. Interval subdivision

The following steps are used for subdividing intervals in the focusing cycle:

Locate the interval (called BESTINTERVAL) containing the value of the variable in CBEST.

Divide the interval into NDIV uniform subintervals.

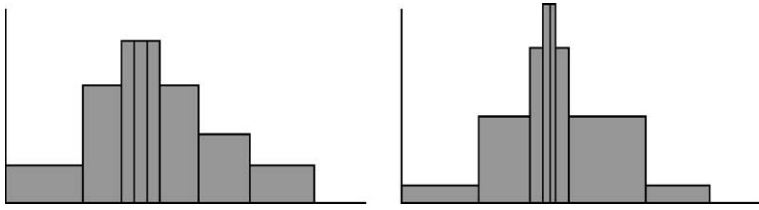


Fig. 5. Evolution of the PDF of a variable after several focusing cycles.

Assign 50% probability to BESTINTERVAL., (so that half of the points generated will be in this interval). Divide this probability uniformly to its subintervals.

Calculate the number of intervals into which the remainder of the domain should be divided so that the total number of intervals remain constant.

Distribute the remaining probability to the region outside the BESTINTERVAL so that the PDF decays exponentially away from the BESTREGION.

After subdivision, intervals no longer have the same width and probabilities are heavily concentrated near the current best.

### 3.2.7. Subdomain cycle

In the subdomain cycle, the focusing cycle is repeated NSDC times and at the end of each iteration, the current search space is modified. In the beginning, the entire space (the global search space) is searched, but in subsequent iterations a subdomain is selected for search. The size of the subdomain decreases gradually and the solution converges to a point. A subdomain is selected by changing the minimum and maximum of each variable.

While choosing the next subdomain, certain precautionary measures are taken to avoid premature convergence. Firstly, a higher scale factor is used after an iteration that does not produce a better cost. This avoids rapid reduction of the axis width after several unsuccessful iterations. Secondly, the statistical variations of the values of the variable in previous iterations are considered in determining the new minimum and maximum. If the value of the variable fluctuates by a large amount the convergence is slowed down.

The method to compute the new values of minimum and maximum for each variable is explained in pseudo-code below:

Let  $XP$  = the value of the variable in CBEST.

Let  $DX = (\text{Current axis width})/2$ .

Let  $GX1$  = Minimum of the axis in the global search space.

Let  $GX2$  = Maximum of the axis in the global search space.

Let  $STDEV$  be the standard deviation of the value of the variable (that is under consideration) in the previous 5 iterations.

If there has been an improvement in cost in the current iteration, Then the scale factor,  $SCF = SDF1$ , else  $SCF = SDF2$ .

The new half width,  $NDX = DX * SCF$ .

If  $NDX < STDEV$   $NDX = STDEV$ .

The new minimum of the axis,  $X1 = XP - NDX$ .

The new maximum of the axis  $X2 = XP + NDX$ .

If  $X1 < GX1$  then  $X1 = GX1$ .

If  $X2 > GX2$  then  $X2 = GX2$ .

### 3.3. Choosing values for parameters

Values of parameters that have been empirically found to be insensitive to the problem-type are given below:

Number of intervals in the PDF,  $NINTERVALS = 20$ .

The number of subintervals,  $NDIV = 6$ .

Subdomain scale factor  $SDSF2 = 0.96$ .

Subdomain scale factor,  $SDSF1$ .

Problem dependent parameters include:

- Number of samples,  $NS$ .
- Iterations in the probability updating cycle,  $NPUC$ .
- Iterations in the focusing cycle,  $NFC$ .
- Iterations in the subdomain cycle,  $NSDC$ .

It is found that for reasonably smooth problems, the values of  $NS$  and  $NPUC$  can be taken as 2 and 1 respectively. Increasing these values produces no improvement in most situations. However, for very irregular domains higher values should be used. Best results are obtained when these values are proportional to the number of regular subregions within the space. However, even for highly non-linear problems, the default values of 2 and 1 work well; they were used in all the benchmark problems listed in the next section.

The most effective values of  $NFC$  are between  $10N$  and  $20N$ , where,  $N$  is the number of variables in the problem. Particularly hard problems require higher values of  $NFC$ .

The value of  $SDSF1$  should be between 0.5 and 0.99. A lower value results in rapid reduction in the sizes of subdomains and may cause premature convergence. A high value slows down convergence and it may take much longer to find the optimum. The following empirical formula is found to produce good results:

$$SDSF1 = N^{(-1/N)}$$

The value of NSDC controls the precision of results and is dependent on the scale factors. A low value results in a large axis width of the subdomain after completing all iterations. The length of search (the number of evaluations) can be modified by adjusting the values of SDSF1 and NSDC.

### 3.4. Similarities with existing random search methods

A common feature that PGSL shares with other random search methods such as ARS and CRS is the use of a PDF. However, this similarity is only superficial. The following is a list of important differences between PGSL and other random methods.

1. Most random methods follow a “creep” procedure similar to simulated annealing. They aim for a point to point improvement by restricting search to a small region around the current point. The PDF is used to search within the neighbourhood. On the other hand, PGSL works by global sampling. There is no point to point movement.
2. The four nested cycles in PGSL have no similarities with characteristics of other algorithms.
3. Representation of probabilities is different. Other methods make use of a mathematical function with a single peak (e.g. Gaussian) for the PDF. PGSL uses a histogram—a discontinuous function with multiple peaks. This allows for fine control over probabilities in small regions by subdividing intervals.
4. Probabilities are updated differently (more details in Section 3.4.1). The primary mechanism for updating probabilities in other methods is to change the standard deviation. In PGSL, the entire shape and form of the PDF can be changed by subdividing intervals as well as through directly increasing probabilities of intervals.

In spite of the apparent similarities with other random search methods, there are significant differences in the performance of PGSL. There is no evidence that random search methods such as ARS and CRS perform as well as genetic algorithms or simulated annealing for large problems. However, PGSL performs as well or better than these algorithms—results from the benchmark tests are presented in the next section.

#### 3.4.1. Updating PDF in PGSL

The most significant difference between PGSL and other random search methods is in the procedure used to update the PDF. The objective of updating the PDF is to generate more points in the region containing the best point without totally excluding other regions. The PDF is updated differently in the focusing cycle and the probability updating cycle. Modifications made in the focusing cycle result in a predominantly single peak function. Since 50% of

the probability is assigned to the best interval, roughly 50% of variables of every point that is generated lies in the best interval. Due to the exponential decay of probability, about 3% of variables lie in the farthest interval. Minor variations to these probabilities are effected in the probability updating cycle and might temporarily contain multiple peaks. This results in increased exploration of values of variables in previous best points.

At the end of each iteration in the subdomain cycle, the PDF tends to peak around a local optimum. The probability of finding a better local optimum increases as the subdomain is narrowed down (considerably reducing the search space). The process of narrowing down is slowed if better points are found far away from the current best point (Section 3.2.7). At the end of all iterations the PDF has its peak around a near global optimum.

#### 4. Comparison with other algorithms

The performance of PGSL is evaluated using several benchmark problems. Results are compared with those collected from two different sources in three series of tests. In the first series of tests, PGSL is compared with three versions of genetic algorithms: simple genetic algorithm (ESGAT), steady state genetic algorithm [14] (Genitor) and CHC [15]. CHC stands for Cross generational elitist selection, Heterogeneous recombination (by incest prevention) and Cataclysmic mutation. In the second series of tests, results from three algorithms for global search are used to evaluate the performance of PGSL. In the third series of tests, a difficult global optimisation problem (Lennard–Jones cluster optimisation) is chosen to test whether PGSL is able to find reported global optima without the use of special heuristics or domain knowledge.

##### 4.1. Test suite 1

De Jong [16] first proposed common test functions (F1–F5) with multiple optima to be used for evaluating genetic algorithms. However, it has been shown that local search can identify global optima of some functions[8]. More difficult test functions have been proposed [17]. These have higher degrees of non-linearity than F1–F5 and can be scaled to a large number of variables. Some of these functions are used for testing the performance of the PGSL algorithm. A short description of the test functions that have been used in the benchmark studies are given below:

*F8 (Griewank's function)*: It is a scalable, non-linear, and non-separable function given by

$$f(x_{i|i=1,N}) = 1 + \sum_{i=1}^N \frac{x_i^2}{4000} - \prod_{i=1}^N (\cos(x_i/\sqrt{i}))$$

*Expanded functions:* Expanded functions [17] are constructed by starting with a primitive non-linear function in two variables,  $F(x, y)$ , and scaling to multiple variables using the formula,

$$EF(x_{i|i=1,N}) = \sum_{j=1}^N \sum_{i=1}^N F(x_i, x_j)$$

The expanded functions are no longer separable and introduce non-linear interactions across multiple variables. An example is the function EF10 which is created using the primitive function F10 shown below:

$$F10(x, y) = (x^2 + y^2)^{0.25} [\sin^2(50(x^2 + y^2)^{0.1}) + 1]$$

*Composite functions:* A composite function can be constructed from a primitive function  $F(x_1, x_2)$  and a transformation function  $T(x, y)$  using the formula

$$EF(x_{i|i=1,N}) = F(T(x_n, x_1)) + \sum_{i=1}^{N-1} F(T(x_i, x_{i+1}))$$

The composite function  $EF8_{avg}$  is created from *Griewank's function*,  $F8$ , using the transformation function  $T(x, y) = (x + y)/2$ .

The composite test function  $EF8_{F2}$  is created from *Griewank's function*,  $F8$  using the De Jong function  $F2$  as the transformation function.  $F2$  is defined as

$$F2(x, y) = 100(x^2 - y^2) + (1 - y^2)$$

The composite functions are known to be much harder than the primitive functions and are resistant to hill climbing [17].

#### 4.1.1. Description of the tests

Four test functions are used for comparison:  $F8$ ,  $EF10$ ,  $EF8_{AVG}$  and  $EF8_{F2}$ . All these test functions have a known optimum (minimum) of zero. It is known that local search techniques perform poorly in solving these problems [17]. Results from PGSL are compared with those reported for three programs based on genetic algorithms, namely, ESGAT, CHC and Genitor [17]. All versions of genetic algorithms used 22 bit gray scale encoding. For  $EF10$ , variables are in the range  $[-100,100]$ . For  $F8$ ,  $EF8_{AVG}$  and  $EF8_{F2}$  the variable range is  $[-512,511]$ .

Results are summarised in Tables 1–4. Thirty trial runs were performed for each problem using different seed values for random numbers. In each trial., a maximum of 500,000 evaluations of the objective function is allowed. Performance is compared using three criteria.

Table 1  
Results for Simple F8 test function

	Number of variables			
	10	20	50	100
<i>Successes</i>				
ESGAT	6	5	0	0
CHC	30	30	29	20
Genitor	25	17	21	21
PGSL	28	29	30	30
<i>Mean solution</i>				
ESGAT	0.0515	0.0622	0.0990	0.262
CHC	0.0	0.0	0.00104	0.0145
Genitor	0.00496	0.0240	0.0170	0.0195
PGSL	0.0007	0.0002	0.0	0.0
<i>Mean number of evaluations</i>				
ESGAT	354 422	405 068		
CHC	51 015	50 509	182 943	242 633
Genitor	92 239	104 975	219 919	428 321
PGSL	283 532	123 641	243 610	455 961

PGSL is compared with results reported in [17]. The global minimum is 0.0 for all instances.

Table 2  
Results for the extended function EF10

	Number of variables		
	10	20	50
<i>Successes</i>			
ESGAT	25	2	0
CHC	30	30	3
Genitor	30	4	0
PGSL	30	30	27
<i>Mean solution</i>			
ESGAT	0.572	1.617	770.576
CHC	0.0	0.0	7.463
Genitor	0.0	3.349	294.519
PGSL	0.0	0.0	0.509639
<i>Mean number of evaluations</i>			
ESGAT	282 299	465 875	
CHC	51 946	139 242	488 966
Genitor	136 950	339 727	
PGSL	61 970	119 058	348 095

The global minimum is 0.0 for all instances.

1. Succ, the success rate (the number of trials in which the global optimum was found).

2. The mean solution obtained in all the trials. The closer the mean solution is to zero (the global optimum), the better the algorithm’s performance.
3. The mean number of evaluations of the objective function required to obtain the global optimum (only for trials in which the optimum was found).

4.1.1.1. *Simple F8 test function.* Results for simple F8 test function are given in Table 1. Thirty trial runs were performed on problems with 10, 20, 50 and 100 variables. PGSL has a success rate of 100% for 50 and 100 variables; no version of GA is able to match this. (Surprisingly, the success rate is slightly lower for fewer variables.) However, the mean number of evaluations to obtain the optimum is higher than CHC and Genitor for this problem.

Table 3  
Results for EF8<sub>AVG</sub> test function

	Number of variables		
	10	20	50
<i>Successes</i>			
ESGAT	0		
CHC	10		
Genitor	5		
PGSL	9		
<i>Mean solution</i>			
ESGAT	3.131	8.880	212.737
CHC	1.283	8.157	83.737
Genitor	1.292	12.161	145.362
PGSL	0.0151	0.1400	1.4438
<i>Mean number of evaluations</i>			
ESGAT			
CHC	222 933		
Genitor	151 369		
PGSL	212 311		

The global minimum is 0.0 for all instances.

Table 4  
Results for EF8<sub>F2</sub> test function

	Number of variables		
	10	20	50
<i>Mean solution</i>			
ESGAT	4.077	47.998	527.1
CHC	1.344	5.63	75.0995
Genitor	4.365	21.452	398.12
PGSL	0.123441	0.4139	1.6836

The global minimum is 0.0 for all instances.

*4.1.1.2. EF10 test function.* Results for the extended function EF10 are summarised in Table 2. PGSL has a success rate of 27 out of 30 runs even for 50 variables. For all criteria, PGSL performs better than all versions of GAs.

*4.1.1.3. EF8<sub>AVG</sub> test function.* Results for the composite function EF8<sub>AVG</sub> are summarised in Table 3. For 20 and 50 variables, none of the algorithms is able to find the exact global optimum. For 10 variables the performance of CHC is comparable with that of PGSL. In terms of the mean value of the optimum, PGSL outperforms all other algorithms.

*4.1.1.4. EF8<sub>F2</sub> test function.* Results for the composite function EF8<sub>F2</sub> are given in Table 4. None of the algorithms is able to find the global optimum for this problem. However, in terms of the quality of the mean solution, PGSL fares better than the rest.

#### *4.1.2. Summary of comparisons*

For the test functions F8 and EF10, PGSL enjoys a near 100% success rate in locating the global optimum even for large instances with more than 50 variables (Tables 1 and 2). No other algorithm is able to match this. For the other two test functions (Tables 3 and 4), none of the algorithms is able to locate the global optima for instances larger than 20 variables. However, mean value of the minima identified by PGSL is much less than those found by other algorithms.

Among the three implementations of GAs considered in this section, CHC performs better than the rest. In most cases, the quality of results produced by PGSL is better than CHC in terms of success rate and mean value of optima. However, PGSL requires a greater number of evaluations than CHC—especially for small problems. Therefore, the overall performance of PGSL is comparable to CHC.

#### *4.1.3. Effect of problem size*

An important criterion for evaluating the robustness of an algorithm is how well it performs when the number of variables is increased. Deterministic algorithms perform well for small problems, but fail to provide reasonable solutions when the number of variables is increased. Although stochastic algorithms perform well in higher dimensions, there is a wide variation in their relative performance. In order to study the effect of problem size, the success rate is plotted against the number of variables in Fig. 6. The functions used for scaling up are described in Section 4.1.1. The performance of PGSL does not deteriorate as rapidly as other algorithms. Another indication of performance degradation in higher dimensions is the mean number of evaluations required to find the global optimum. This is plotted in Fig. 7. The number of evaluations increases almost linearly with the number of variables.

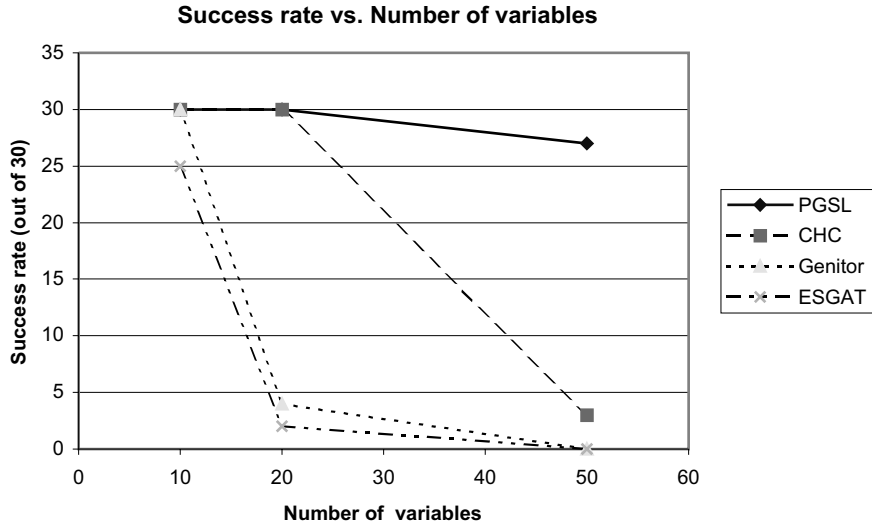


Fig. 6. Effect of problem size. PGSL enjoys a high success rate in finding the global minimum even for 50 variables. For other algorithms, the success rate drops drastically as the problem size is increased beyond 20 variables.

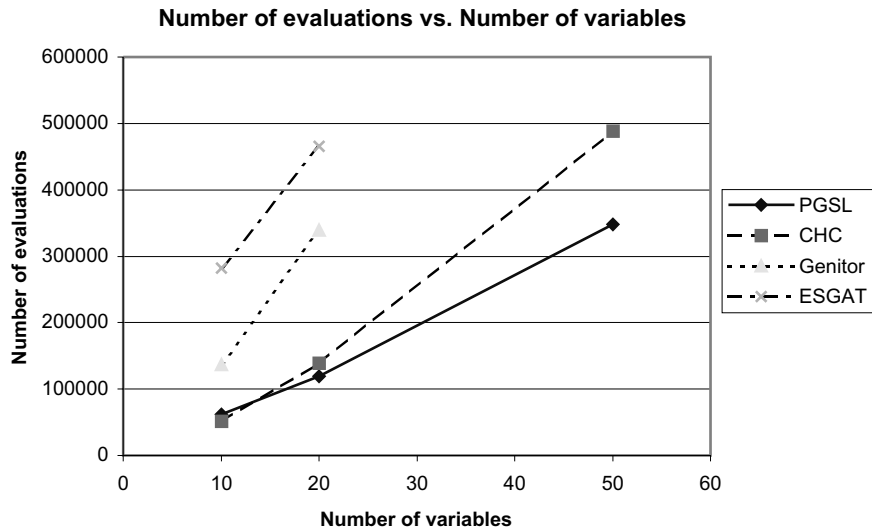


Fig. 7. This figure shows the number of evaluations required to find the global minimum for different problem sizes. With PGSL, the required number of evaluations grows more slowly than the increase observed for the other algorithms.

#### 4.2. Test suite 2

Mongeau et al. [18] have evaluated the performance of six public domain software implementations for global search. Three that have given best results are used to evaluate the relative performance of PGSL. These are

- ADA: adaptive simulated annealing.
- GAS: an implementation of genetic algorithm.
- INTGLOB: integral global optimisation.

Six test problems belonging to two categories have been chosen. They are (a) least median of squares and (b) multi-dimensional scaling. Some problems involve non-differentiable and discontinuous objective functions. All have multiple local minima. A short description of these problems is given below.

##### 4.2.1. Problem descriptions

*Least median of squares:* The objective is to perform linear regression by minimising the median of errors (instead of the conventional root mean square error). This procedure ensures that at least half the points lie as close as possible to the resulting hyperplane. The optimisation problem can be stated mathematically as

$$\text{Minimise median of } r_i^2(\theta, b)$$

where  $r_i$  is the error in each point given by

$$r_i(\theta, b) = y_i - x_{i1}\theta_1 - \dots - x_{ip}\theta_p - b$$

and where  $\theta, b$  are the coefficients to be estimated with  $(y_i, x_i)$  as the given set of points.

Since there is no closed-form mathematical expression to compute the median, the objective function is non-differentiable. In the present study, regression has been performed on the following five sets of data: lms1a, lms1b, lms2, lms3 and lms5. These sets contain one to five variables.

*Multi-dimensional scaling:* The objective is to compute the coordinates of  $n$  points such that their distances are as close as possible to specified values. Mathematically, the problem can be stated as

$$\text{Minimise } \sum_{i < j}^n w_{ij}(\delta_{ij} - \|x_i - x_j\|)^2$$

where  $w_{ij}, \delta_{ij}$  are the given sets of weights and distances between each pair of points having coordinates  $x_i$  and  $x_j$  respectively. One instance of this problem (ms2) involving 10 points in two dimensions (having a total of 20 variables) is considered in the present study.

Table 5  
Results of comparison of test suite 2

Problem instance	N	Optimum found				Evaluations required			
		PGSL	ASA	GAS	INT-GLOB	PGSL	ASA	GAS	INT-GLOB
<i>Least median of squares</i>									
lms1a	1	0.0074	0.0074	0.0074	0.0074	369	190	700	300
lms1b	1	0.00676	0.0068	0.0068	0.0068	1632	700	700	200
lms2	2	0.576	0.576	0.591	0.576	1556	350	2000	2000
lms3	3	0.145	0.15	0.15	0.14	2759	485	4000	1090
lms5	5	0.033	0.02	0.045	0.034	11986	2580	14840	4260
<i>Multi-dimensional scaling</i>									
ms2	20	11.82	12.16	12.7	11.73	10387	19918	25081	10000

4.2.2. Results

Results are summarised in Table 5. PGSL was able to find the reported minimum point for all problem instances except lms5 and ms2. In the case of ms2, the minimum found by PGSL is very close to that found by INTGLOB. For small problems, PGSL takes more evaluations to find the minimum point. However, for the 20 variable problem, the number of evaluations taken by PGSL is much less than ASA and GAS.

4.3. Test suite 3: Lennard–Jones cluster optimisation

The global optimisation of Lennard–Jones clusters is a very simple yet reasonably accurate mathematical model of a real physical system, namely that of low temperature micro-clusters of heavy rare gas atoms such as argon, krypton or xenon. The objective function is non-convex and the number of energetically distinct local optima is believed to grow at least exponentially with the number of atoms [19]. Also, multiple discontinuities exist in the domain since the energy tends to infinity when atoms approach each other. Many of the global optima have been found fairly recently [20,21]. Most successful algorithms use domain specific knowledge (heuristics) to obtain reasonable performance. For example, Wolf and Landman [20] use a version of GA in which the starting population consists of a special configuration of atoms and each offspring produced is relaxed to the nearest local minimum through conjugate-gradient minimisation.

The configuration of atoms according to the Lennard–Jones model is determined by minimising the total energy of given by

$$\sum_{i < j}^n r(d_{ij})$$

where  $d_{ij}$  is the distance between the atoms  $i$  and  $j$ . The function

$$r(s) = s^{-12} - 2s^{-6}$$

is the Lennard–Jones energetic model. The total energy can be evaluated if the coordinates of all atoms are known. If there are  $n$  atoms, there are  $3n$  unknown variables which are the  $x, y, z$  coordinates of each atom. It is possible to reduce the number of variables to  $(3n - 6)$  through defining the coordinate system using the positions of the first two atoms. However, this has not been done in the PGSL implementation since there is no significant improvement.

#### 4.3.1. Evaluation using small instances

In the first set of comparison, four instances of the problem are considered with the number of atoms ranging from 3 to 6. These have been designated as pf3, pf4, pf5 and pf6. Results are compared with four global optimisation algorithm implementations reported in [18]. Known values of global optima for these functions are  $-3$ ,  $-6$ ,  $-9.1$  and  $-12.71$ . According to Mongeau et al. [18], ASA and GAS are unable to find the global optimum for pf4 and pf5. INT-GLOB is not able to find the global optimum for pf6. PGSL is able to find the global minimum for all instances with the recommended values for parameters. The best-so-far curves are shown in Figs. 8–11 in order to compare the

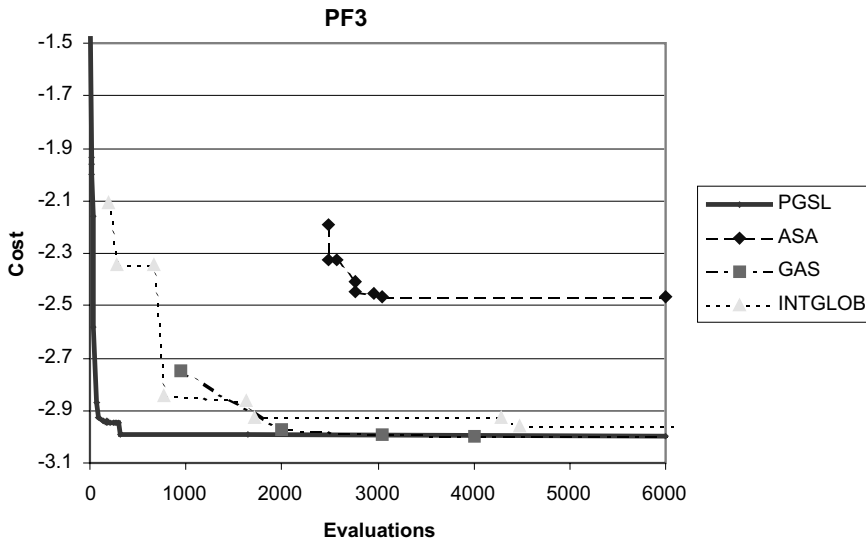


Fig. 8. The best-so-far curves for Lennard–Jones cluster optimisation problem with 3 atoms (9 variables). PGSL converges to the global optimum faster than all other algorithms.

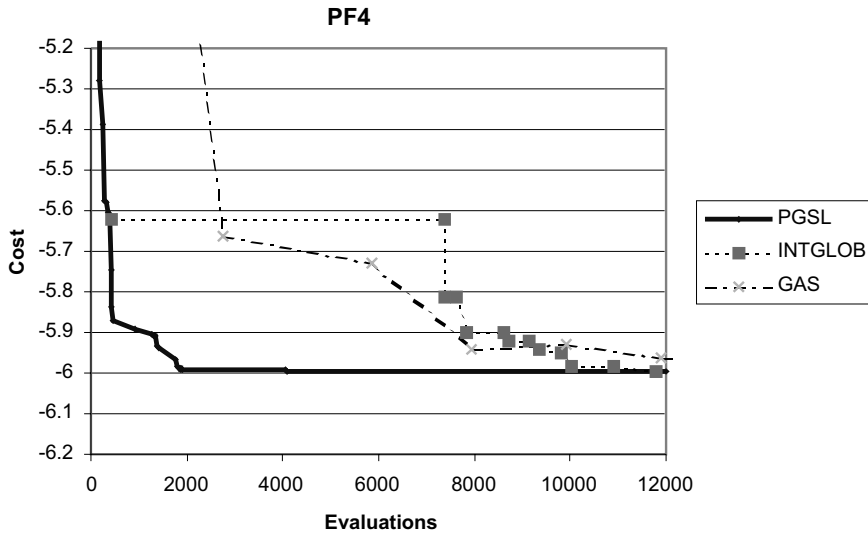


Fig. 9. The best-so-far curves for Lennard–Jones cluster optimisation problem with 4 atoms (12 variables). PGSL converges to the global optimum faster than all other algorithms.

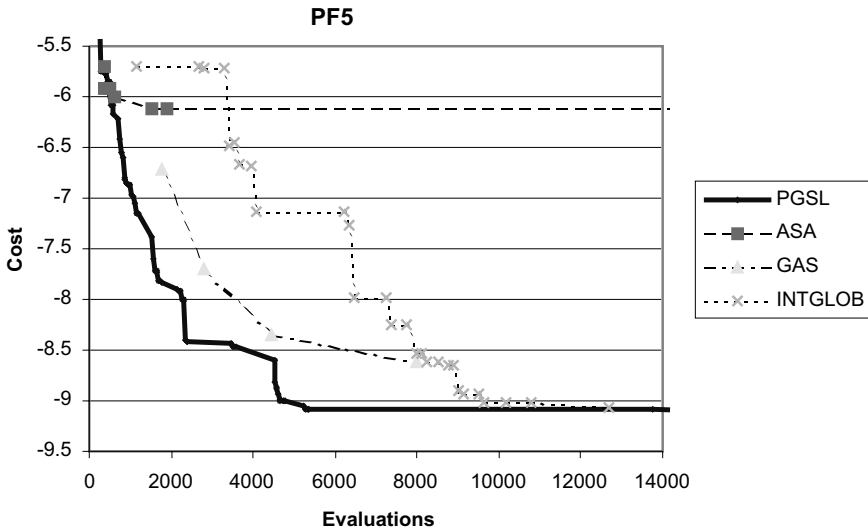


Fig. 10. The best-so-far curves for Lennard–Jones cluster optimisation problem with 5 atoms (15 variables). ASA and GAS are unable to find the global optimum for this problem.

performance of different algorithms. The  $y$  axis is the best value of the objective function found so far, and  $x$  axis is the number of function evaluations.

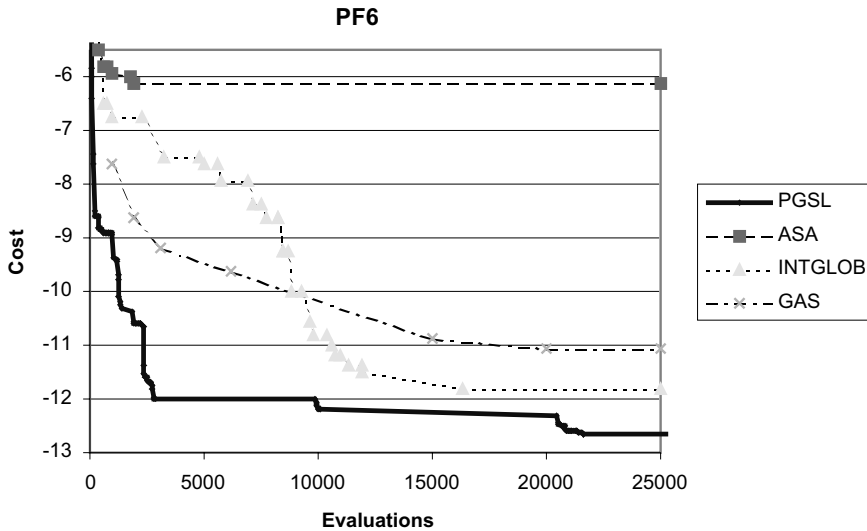


Fig. 11. The best-so-far curves for Lennard–Jones cluster optimisation problem with 6 atoms (18 variables). ASA, GAS and INTGLOB are unable to find the global optimum for this problem.

#### 4.3.2. Evaluation using larger instances

Information related to the number of evaluations of the objective function required to find the global optima are not available for large problem sizes since most successful implementations use special heuristics for initial starting points and for subsequent exploration. For example, Deaven and Ho [21], Wolf and Landman [20], and Hartke [22] use special crossover and mutation operators that act on clusters in the configuration space. These tailor-made operators produce best results for molecular cluster optimisation since they consider special characteristics of the problem. Most of the implementations combine global and local search methods through the use of gradients. Leary [19] applied a two stage descent procedure, an initial fixed length steepest descent followed by conjugate gradient descent.

Our objective in this study is to examine whether PGSL is able to find the global optimum for large problem sizes without using any problem specific information and without computing gradients. The following procedure was followed:

Start with default values of PGSL parameters, that is,  $NS = 2$ ,  $NPUC = 1$ ,  $NFC = 20 * N$  and  $NSDC = 40$ .

Execute PGSL, If the known global solution is found STOP. Otherwise increase NSDC by one and repeat PGSL with a different random seed.

Using this procedure<sup>1</sup> (referred to as “PGSL alone” formulation), PGSL was executed for problems of size 7–25 atoms (21–75 variables). The cumulative total number of evaluations of the objective functions required to find the global optima (including all restarts) are shown in Table 6. These results show that PGSL is able to identify global minima for complex objective functions without any domain dependent information or the use of gradients.

It is possible to improve the efficiency of PGSL by incorporating local search and domain dependent heuristics. In order to illustrate this point, two different formulations of the Lennard–Jones problem were used. In the first formulation, the objective function of a point is evaluated as the energy value of the nearest local minimum. A gradient descent is performed starting from the point to be evaluated. In the second formulation, the global optimum is located in two stages as described below.

Stage 1: Perform a global search by treating all coordinates as variables. Let the best point obtained be  $P_{best}$  and the value of the objective function be  $y_{best}$ .

Stage 2: Perform a sequence of optimisations  $l_1, l_2, \dots, l_i$ , with a limited number of variables. Only the coordinates of selected atoms from the current configuration,  $P_{best}$ , are treated as search variables. However, for the evaluation of the objective function, a gradient descent is performed and the value of the nearest local minimum is chosen. For the gradient descent all atoms are allowed to move instead of the selected atoms. At the end of each optimisation,  $l_i$ , the minimum point obtained  $P_i$  is chosen as  $P_{best}$  if its value is less than  $y_{best}$ . Stage 2 terminates when  $y_{best}$  becomes less than or equal to the known global minimum.

In the second formulation, a special heuristic is used to select the atom to be moved. Atoms are ordered according to their contribution to the total potential energy. The atom with the highest contribution is chosen in optimisation  $l_1$ , two atoms with the highest potential are selected in  $l_2$  and so on. The cycle starts again with a single atom, if either all atoms are completed or a better point is located. (In most PGSL runs, improvements were obtained within one or two trials.)

The use of gradients in the first formulation improves performance dramatically. This is because the local minimum is located within about 10  $N$  evaluations, where  $N$  is the total number of variables. PGSL alone requires

---

<sup>1</sup> The executable for solving Lennard–Jones cluster optimisation problem using PGSL may be downloaded from <http://imacwww.epfl.ch/imac/TEAM/raphael/LJ/index.html> (for independent evaluation).

Table 6

Comparison of the number of evaluations of the objective functions taken by PGSL to locate the known global optima for different instances of the Lennard–Jones cluster optimisation problem

Number of atoms	PGSL alone	With gradients	With heuristic
7	114 236	1619	1689
8	54 846	1711	1520
9	162 286	1991	4012
10	1 203 648	7130	5694
11	487 174	13 300	13 301
12	2 839 832	14 094	2234
13	1 992 400	6890	7065
14	1 001 022	10 936	2363
15	2 589 100	4080	8841
16	2 668 710	4713	6628
17	6 834 472	14 611	7677
18	14 930 178	24 149	39 661
19	2 224 158	12 216	24 546
20	29 259 180	10 565	22 603
21	248 825 344	31 295	9573
22	120 720 454	23 264	11 669
23	755 164 120	57 145	33 198
24	182 461 280	18 137	29 542
25	803 003 156	44 709	30 024
26		42 246	52 521
27		40 720	86 035
28		208 294	473 390
29		802 202	73 812
30		1 037 979	52 178

The first column contains the number of atoms. The other columns contain the number of evaluations required to find the global optimum through the three formulations described in 4.3.2.

many more evaluations to produce the same improvement because the direction of steepest descent is unknown. Consequently, the total number of evaluations required to locate the global minimum using gradients is reduced by about 90%. This performance could be improved further through using better methods for local optimisation such as the quasi-Newton method employed by Hartke [22].

The use of the heuristic in the second formulation improved performance further. The improvements are significant mostly for larger problems. The improvements produced by formulations 1 and 2 over black-box formulation are shown in Fig. 12. The PGSL alone formulation was not attempted for problem sizes greater than 25 atoms due to the excessive computation time that was required. Table 6 provides the number of evaluations required by the three formulations. Table 7 contains results for larger problem sizes using the third formulation. These larger problem sizes were not attempted using the second formulation.

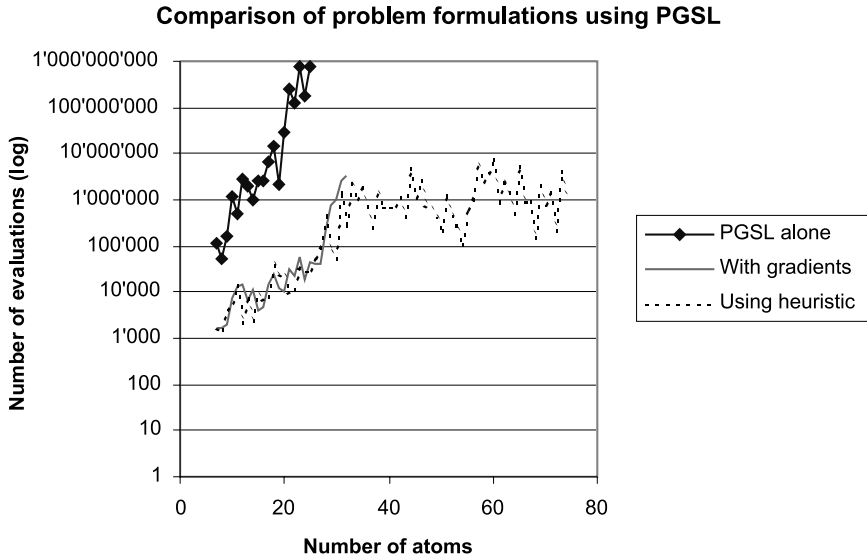


Fig. 12. The number of evaluations required by PGSL to find the global minima is reduced through incorporating local search (gradient descent) and domain dependent heuristics.

Table 7

Continued from Table 6. The number of evaluations required by PGSL to find the global minima for problem sizes from 31 to 74 using Formulation 3 described in 4.3.2

Number of atoms	Number of evaluations	Number of atoms	Number of evaluations
31	1 357 167	53	231 516
32	265 699	54	102 923
33	2 193 348	55	600 293
34	966 694	56	1 018 556
35	1 798 960	57	5 417 904
36	712 956	58	2 338 255
37	254 023	59	3 541 695
38	1 414 625	60	7 235 486
39	685 584	61	837 703
40	689 007	62	2 320 804
41	702 951	63	1 369 253
42	1 044 466	64	492 090
43	411 360	65	5 259 537
44	4 892 627	66	913 939
45	1 109 894	67	891 047
46	2 627 503	68	154 713
47	773 389	69	2 071 874
48	644 256	70	785 968
49	412 756	71	1 386 307
50	217 830	72	218 099
51	1 228 811	73	4 134 284
52	449 893	74	1 269 815

*Conclusions:* Results of PGSL alone optimisation of Lennard–Jones clusters provides empirical evidence of the convergence of the algorithm when applied to complex objective functions. The global minima were located without the use of gradients for all instances up to 25. This is remarkable considering that there are about  $10^{10}$  local optima for a cluster of 25 atoms [19]. The use of gradients and heuristics improved the performance significantly and shows that the algorithm can be combined effectively with local search techniques.

## 5. Practical applications of PGSL

PGSL has already been applied to practical engineering tasks such as design, diagnosis and control. Summaries of these applications are given below.

### 5.1. Design of timber shear wall structures

Shear-walls are the predominant method for resisting lateral forces in large timber houses, especially in the Nordic countries. The principle is to use boards connected to a timber frame by screws or nails in order to resist horizontal forces. Design of timber shear-walls involves deciding on a configuration of boards, studs and joists, as well as computing parameters such as screw distances. There is no direct method for performing this computation and variables cannot be expressed using closed-form equations. Solutions can only be generated and tested for the satisfaction of requirements.

PGSL was used to find low cost timber shear wall designs by searching through the space of possibilities [23]. Design variables are wall types, screw distances, number of extra studs and the number of special devices to resist uplifting forces. All variables including screw distances are treated as discrete. In spite of several simplifications, there are about 210 variables for a relatively simple building. The objective function consists of two parts. (a) The material and labour costs computed using a detailed procedure that was calibrated by industrial partners. (b) The penalty for violation of structural constraints. There is strong inter-relationship between variables since the form of the equations used in the objective function changes depending upon the values of variables such as wall types.

In order to test the performance of PGSL, a full-scale test case was implemented. The actual multi-storey timber building is a completed project called Winter City 2000 and was built by the industrial partner Lindbäck's Bygg AB in Sweden. This case involved 210 variables, out of which 34 variables take only two different values, others take between 5 and 17 distinct values. The most important and surprising result is that it is possible to lower the production cost in the factory for the shear-walls by up to 7%. This is in spite of the fact

that factory designs have been highly optimised over the years. Seven percent is equivalent to the average profit margin in this industry for such elements.

### *5.2. Finding the right model for bridge diagnosis*

Millions of modelling possibilities exist for modelling full-scale civil engineering structures such as bridges due to the number of possible combinations of assumptions related to their behaviour. Finding good models for explaining a given set of observations is a difficult engineering task.

PGSL was used to identify models of a full scale bridge in Switzerland [24]. Predicted behaviour using these models matched closely with measurement data. The objective function used was the root mean square error between theoretical and measured deformations, support rotations and deflections. The search procedure involved identifying the right set of values of parameters within specific models and hence the number of independent variables were limited to a maximum of 5. All variables are continuous having different ranges of values. One variable (Young's modulus of concrete) was allowed to vary from 20 to 50 (KN/mm<sup>2</sup>) whereas another variable (support rotation) varied from 0 to 1. The objective function involved complex non-linear interactions between variables. The objective function is not expressible as a closed form mathematical equation since it involves a full-scale structural analysis for each combination of values of variables.

Initial theoretical models that were constructed manually had errors up to 100% when compared to measured deformations. However, through the use of PGSL it was possible to identify models that contained less than 5% root mean square error. Although from a mathematical point of view, this is a relatively small problem, it illustrates the successful application of PGSL to practical diagnostic tasks.

### *5.3. Structural control*

Intelligent structural control involves computation of the right control movements in order to satisfy certain conditions such as stresses and deflections. The tensegrity structure constructed at EPFL—the Swiss Federal Institute of Technology in Lausanne—is equipped with actuators to tune the amount of stress in cables such that deflections are reduced. The actuators are telescopic bars whose length can be adjusted. The control task is to find out the right change in lengths of up to 15 telescopic bars. All variables are treated as discrete since lengths may only be changed in steps of 0.25 mm. Each variable takes a maximum of 84 different values (allowing movements up to 21 mm). Since changing the length of members affects the geometry of the structure as well as the pre-tension in cables, the problem is highly coupled and non-linear [25].

The performance of PGSL was compared with that of simulated annealing [25]. For small control movements (a maximum of 3 mm), both algorithms performed equally well and produced fast convergence. However, when larger movements were permitted (up to 21 mm), thereby increasing the size of the solution space, PGSL produced higher quality solutions than simulated annealing.

#### *5.4. Travelling salesman problem*

The travelling salesman problem (TSP) is a difficult combinatorial optimisation problem for search methods such as simulated annealing and genetic algorithms [26]. Although PGSL was tested on several publicly available instances of the TSP, the results are only partially encouraging. For problems consisting of up to 29 nodes, PGSL was able to find the known global optima quickly. However, it could not find global optima for larger problems. Such poor performance is thought to be related to the assumption of neighbourhoods. The PGSL algorithm is attracted towards regions that contain apparently good solutions while the global optimum for this problem exists in a different region altogether. In such situations, problem specific heuristics such as the Lin–Kernighan Heuristic [27] produce better results than generic methods.

## **6. Conclusions**

Although probabilistic methods for global search have been in use for about half a century, it is only recently that they have attracted widespread attention in the engineering and scientific communities. Considerable progress has been made in the area of direct search during the last decades. For example, the development of genetic algorithms and simulated annealing have spawned much activity. Genetic algorithms and simulated annealing are direct search methods and are well suited for practical applications where objective functions cannot be formulated as closed form mathematical expressions. PGSL is a new direct search algorithm. Its performance is comparable with, if not better than existing techniques in most situations. Bench mark tests indicate that it performs well even when objective functions are highly non-linear. Results are always better than the simple genetic algorithm and steady state genetic algorithm for the expanded test functions considered in this paper. Similar conclusions are drawn through tests comparing PGSL with other algorithms such as adaptive simulated annealing. PGSL scales up extremely well both in terms of the number of evaluations required to find good solutions as well as the quality of solutions. Results of optimisation of Lennard–Jones clusters using PGSL provides empirical evidence of the convergence of the algorithm

when applied to complex objective functions. A combination of PGSL with local search heuristics improves performance considerably, especially for hard optimisation problems such as the Lennard–Jones cluster optimisation.

## Acknowledgements

This research is funded by the Swiss National Science Foundation (NSF) and the Commission for Technology and Innovation (CTI). We would like to thank K. De Jong and K. Shea for valuable comments and suggestions. We would also like to thank Logitech SA and Silicon Graphics Incorporated for supporting this research.

## References

- [1] M.W. Trosset, I know it when i see it: Toward a definition of direct search methods, *SIAG/OPT Views-and-News*, no. 9, (Fall 1997), pp. 7–10.
- [2] S. Kirkpatrick, C. Gelatt, M. Vecchi, Optimisation by simulated annealing, *Science* 673 (1983) 220.
- [3] J. Holland, *Adaptation in natural artificial systems*, University of Michigan Press, MI, 1975.
- [4] S.H. Brooks, Discussion of random methods for locating surface maxima, *Operations Research* 6 (1958) 244–251.
- [5] S.F. Masri, G.A. Bekey, A global optimization algorithm using adaptive random search, *Applied mathematics and computation* 7 (1980) 353–375.
- [6] W.L. Price, A controlled random search procedure for global optimization, in: L.C.W. Dixon, G.P. Szego (Eds.), *Towards Global Optimization*, vol. 2, North-Holland, Amsterdam, 1978.
- [7] P. Brachetti, M.F. Ciccoli, G. Pillo, S. Lucidi, A new version of Price's algorithm for global optimisation, *Journal of Global optimisation* 10 (1997) 165–184.
- [8] L. Davis, Bit-climbing representational bias and test suite design, in: L. Booker, R. Belew (Eds.), *Proceedings of the 4th international conference on GAs*, Morgan Kaufman, 1991.
- [9] Z.B. Zabinsky, R.L. Smith, J.F. McDonald, H.E. Romeijn, D.E. Kaufman, Improving hit-and-run for global optimisation, *Journal of Global Optimization* 3 (1993) 171–192.
- [10] Z.B. Zabinsky, R.L. Smith, Pure adaptive search in global optimisation, *Mathematical Programming* 53 (1992) 323–338.
- [11] D.J. Reaume, H.E. Romeijn, R.L. Smith, Implementing pure adaptive search for global optimisation using Markov chain sampling, *Journal of Global Optimization* 20 (2001) 33–47.
- [12] E.M.T. Hendrix, O. Klepper, On uniform covering, adaptive random search and raspberries, *Journal of Global Optimization* 18 (2000) 143–163.
- [13] D.W. Bulger, G.R. Wood, Hesitant adaptive search for global optimisation, *Mathematical Programming* 81 (1998) 89–102.
- [14] G. Syswerda, A study of reproduction in generational and steady-state genetic algorithms, in: G. Rawlins (Ed.), *Foundations of Genetic algorithms*, Morgan-Kaufmann, 1991, pp. 94–101.
- [15] L. Eshelman, The CHC adaptive search algorithm, in: G. Rawlins (Ed.), *Foundations of Genetic Algorithms*, Morgan Kaufmann, 1991, pp. 256–283.
- [16] K. De Jong, Analysis of the behaviour of a class of genetic adaptive systems. Ph.D. thesis, University of Michigan, Ann Arbor, 1975.
- [17] D. Whitley, Building better test functions, in: L. Eshelman (Ed.), *Proceedings of the 6th international conference on GAs*, Morgan Kaufman, 1995.

- [18] M. Mongeau, H. Karsenty, V. Rouzé, J.-B. Hiriart-Urruty, Comparison of public-domain software for black box global optimization, *Optimization Methods and Software* 13 (3) (2000) 203–226.
- [19] R.L. Leary, Global optima of Lennard–Jones clusters, *Journal of Global Optimization* 11 (1997) 35–53.
- [20] M.D. Wolf, U. Landman, *Journal of Physical Chemistry A* (1998) 6129–6137.
- [21] D.M. Deaven, K.M. Ho, *Physical Review Letters* 75 (2) (1995) 288–291.
- [22] B. Hartke, Global cluster geometry optimization by a phenotype algorithm with niches: Location of elusive minima, and low-order scaling with cluster size, *Journal of computational chemistry* 20 (16) (1999) 1752–1759.
- [23] P. Svarerudh, B. Raphael, I.F.C. Smith, Lowering costs of timber shear-wall design using global search, *Engineering with Computers* 18 (2002) 93–108.
- [24] Y. Robert-Nicoud, B. Raphael, I.F.C. Smith, Decision support through multiple models and probabilistic search, in: *Proceedings of Construction Information Technology 2000*, Iceland building research institute, 2000.
- [25] B. Domer, B. Raphael, K. Shea, I.F.C. Smith, Comparing two stochastic search techniques for structural control, *Journal of Computing in Civil Engineering*, ASCE, accepted for publication.
- [26] O. Martin, Combining simulated annealing with local search heuristics, in: G. Laporte, I. Osman (Eds), *Metaheuristics in combinatoric optimization*, 1995.
- [27] S. Lin, B. Kernighan, An effective heuristic for the travelling salesman problem, *Operations research* 21 (1971) 498.