Available online at www.sciencedirect.com

SCIENCE DIRECT®

European Journal of Operational Research 169 (2006) 450–476

EUROPEAN JOURNAL OF OPERATIONAL RESEARCH

www.elsevier.com/locate/ejor

# Continuous scatter search: An analysis of the integration of some combination methods and improvement strategies ☆

F. Herrera *, M. Lozano, D. Molina

*Department of Computer Science and A.I., University of Granada, ETS de Ingeniera Informatica, Avda. Andalucia 38, Granada 18071, Spain*

Received 28 July 2003; accepted 5 May 2004
Available online 7 October 2004

**Abstract**

Scatter search is an evolutionary method that shares with genetic algorithms, a well-known evolutionary approach, the employment of a combination method that combines the features of two parent vectors to form several offspring. Furthermore, it uses improvement strategies to efficiently produce the local tuning of the solutions. An important aspect concerning scatter search is the trade-off between the exploration abilities of the combination method and the exploitation capacity of the improvement mechanism.

In this paper, we deal with a continuous version of the scatter search, which works directly with vectors of real components. Our objective is to study the balance between the reliability induced by the combination method and the accuracy levels provided by the improvement mechanism in continuous scatter search. To do this, we analyse two combination methods that may be applied to continuous scatter search: (1) the BLX-$\alpha$ operator, which is one of the most effective combination methods for real-coded genetic algorithms; and (2) the average combination method, which is the classical combination method for continuous scatter search. We investigate the interrelations that exist between these combination methods and two improvement mechanisms, the Solis and Wets' algorithm and the Nelder–Mead simplex algorithm, which are well-known continuous local searchers. In addition, we also perform a comparison among continuous scatter search and other continuous optimization algorithms presented in the literature.

*Keywords:* Scatter search; Continuous optimization; Combination method; Local searcher

## 1. Introduction

*Scatter search* (SS) [15,19,31,32]) is a population-based meta-heuristic method that uses a *reference set* to combine its solutions and construct others. It generates a reference set from a population of

☆ This research has been supported by project TIC-2002-0436-C05-01 and Network HEUR TIC-2002-10886E.

* Corresponding author. Tel.: +34 58 244019; fax: +34 58 243317.

*E-mail addresses:* herrera@decsai.ugr.es (F. Herrera), lozano@decsai.ugr.es (M. Lozano), dmolina@decsai.ugr.es (D. Molina).

solutions. Then the solutions in this reference set are combined to get starting solutions to run an *improvement procedure*, whose result may indicate an updating of the reference set and even an updating of the population of solutions. From the standpoint of meta-heuristic classification, SS may be viewed as an *evolutionary algorithm* (EA) [1,14,33] because it builds, maintains and evolves a set of solutions throughout the search.

An important point in common between SS and another well-known evolutionary approach, genetic algorithms (GAs) [22,28] is that both algorithms apply a *crossover* or combination mechanism to create new solutions [16]. The solution combination mechanism is a method for sharing information between solutions. Generally, it combines features of two parent vectors to form several offspring, with the possibility that good solutions may generate better ones. It has always been regarded as the main search operator in both GAs [9,29] and SS [20], because it manages the available information in previous samples to influence future searches.

In SS, the importance of the combination method could be even greater, due to the differences in the method used to select the solutions to combine. In traditional EAs, such as GAs, parents are chosen following a random sampling scheme. By contrast, in SS, the selection of the parents is made using a deterministic method called Subset Generation Method. This method generates all subsets of size 2, skipping subsets for which both elements have not changed from previously iterations.

The possible improvement solution methods applied to the solutions range from simple local searches to a very specialized search. An example of a very simple procedure is a local search based on basic moves in which the best improving move or the first improving move found is selected. The procedure must allow the use of tools like recent or intermediate memory, variable neighborhoods, or hashing scanning methods of the neighborhood. Among the methods to apply there are the Tabu Search [18], a Variable Neighborhood Search [23] or any sophisticated hybrid heuristic search [21]. In this sense, SS procedures can be classified as well as memetic algorithms (MAs). MAs [34,35] are EAs that apply a separate local search process to refine individuals. An important aspect concerning MAs is the trade-off between the exploration abilities of the EA, and the exploitation abilities of the local search mechanism used [30].

The combination method and the improvement method are two of the main components of SS that decisively affect the trade-off between the exploration and the exploitation maintained by this search procedure. The combination method has a strong influence on the exploration because population diversity is generated by creating new solutions. Therefore, its goal is to induce *reliability* in the search process. The improvement method is applied with the aim of exploiting the diversity provided by the combination method. In order to do this, an effective refinement on the individuals returned by this method is produced. Thus, its principal objective is to obtain the best possible *accuracy* levels.

Originally, SS was introduced as a heuristic to obtain a near optimal solution to an integer-programming problem [15]. It was also used to generate both starting as well as trial solutions. Recently, the SS approach was refined and used for both discrete and continuous optimization problems [13,16,17,31,42,43]. Continuous SS (CSS) directly handles vectors of real components and combines these vectors by linear combinations to produce new ones through successive generations. In addition, it uses continuous local searchers, such as the Solis and Wets' algorithm [39] and the Nelder and Mead's simplex method [36], as improvement procedures. This paper deals with CSS algorithms.

The objective of this paper is to study the effects of the balance between the exploration induced by the combination method and the exploitation introduced by the improvement mechanism on the CSS performance. To do this, we undertake the analysis of two combination methods that may be applied to continuous scatter search:

- *BLX-$\alpha$ operator*, which is one of the most effective combination methods for real-coded GAs (RCGAs) [25]. RCGAs are specific GA implementations that were provided to deal with continuous problems. They are based on real number representation of the solutions. Most RCGA research has been focused on the

developing of effective real-parameter combination operators, and as a result, many different possibilities have been proposed [7,27]. In [27], we carry out an empirical study of different combination method instances for RCGAs, which offers some clues as to the key features that have a positive influence on the combination method behaviour. Our experiences on the application of solution combination methods to RCGAs and the similarity between RCGAs and CSS make us believe that their implementation in the CSS framework may enhance the operation of this search algorithm.
We have considered BLX-$\alpha$ as a combination method for CSS due to three reasons: (1) its includes randomness, which can be effective [31], (2) it favors the production of diversity in the population of an EA, which may improve the CSS reliability, and finally, (3) BLX-$\alpha$ has a *self-adaptive* behaviour [3].
- *Average combination method*, which is the classical combination method for CSS.

In addition, we investigate the interrelations that exist between these combination methods and two improvement mechanisms, the Solis and Wets' algorithm and the Nelder–Mead simplex algorithm, which are well-known continuous local searchers. The performance of the improvement mechanism depends on the number of iterations assigned to this method (*local search depth*). Obviously, the greater the local search depth is, the more refinement on the solutions will be accomplished. In order to study the impact of this parameter on the exploration/exploitation balance, different values have been used when carrying out our experiments.

The paper is set up as follows. In Section 2, we present the combination methods used in this work. In Section 3, we describe the local searchers. In Section 4, we provide details of the implementation considered for CSS. In Section 5, we describe the experiments carried out in order to determine the suitability of CSS instances designed with different combination methods, local searchers, and local search depth values. In Section 6, we provide a comparison among our best performing CSS instances and different optimization procedures proposed in the literature to deal with continuous domains. Finally, we draw our conclusions in Section 6. In addition, in Appendix A, we include the features of the test suite used for the experimental study and in Appendix B, we introduce all the results of our experiments.

## 2. Combination methods for continuous problems

The combination operator is a fundamental search operator because it exploits information about the search space that is currently available in the population. Much effort has been put in developing sophisticate combination methods for RCGAs, and as a result, many different instances has been proposed [7,26,27]. Real-coding of solutions for numerical problems offers the possibility of defining a wide variety of special real-parameter combination methods which can take advantage of its numerical nature.

Let us assume that $X = (x_1, \ldots, x_n)$ and $Y = (y_1, \ldots, y_n)$ $(x_i, y_i \in [a_i, b_i] \subset \Re,\quad i = 1, \ldots, n)$ are two real-coded vectors selected to be combined. Below, we describe the operation of the combination methods considered in this paper, and show their effects in graph form.

**Average combination method.** It returns an offspring: $Z_1 = (z_1, \ldots, z_i, \ldots, z_n)$ with $z_i = \frac{1}{2} \cdot x_i + \frac{1}{2} \cdot y_i$ (Fig. 1).

**BLX-$\alpha$ combination method** [11]. In this case, an offspring is generated: $Z = (z_1, \ldots, z_i, \ldots, z_n)$, where $z_i$ is a randomly (uniformly) chosen number of the interval $[c_{\min} - I \cdot \alpha, c_{\max} + I \cdot \alpha]$, where $c_{\max} = \max\ (x_i, y_i)$, $c_{\min} = \min(x_i, y_i)$, and $I = c_{\max} - c_{\min}$ (Fig. 2).

Investigations have reported that BLX-$\alpha$ with $\alpha = 0.5$ performs better than BLX-$\alpha$ operator with any other $\alpha$ value [8].

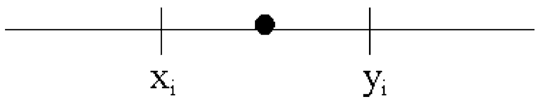

Fig. 1. Average combination method.

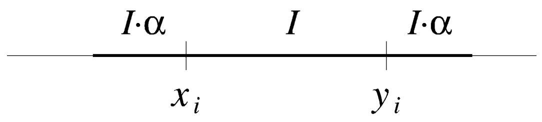

Fig. 2. BLX-α combination method.

We consider BLX-α a convenient combination method for CSS because of the following reasons:

- Combination methods that include randomness can be effective [31]. BLX-α is a variable-wise operator that determines the values for each decision variable of the offspring by extracting values from intervals defined in neighborhoods associated with the decision variables of the parents using uniform probability distributions. Thus, it is a uniformly emphasized combination operator.
- Population diversity is crucial to an EA's ability to continue the fruitful exploration of the search space. If the lack of population diversity takes place too early, a premature stagnation of the search is caused. Under these circumstances, the search is likely to be trapped in a local optimum before the global optimum is found. This problem, called *premature convergence*, has long been recognized as a serious failure mode for EAs [10]. In addition, in the MA literature, keeping population diversity while using local search together with an EA is always an issue to be addressed, either implicitly or explicitly [24,30].
  Nomura et al. [37] have demonstrated theoretically that BLX-α has the ability to promote diversity in the population of an EA. In particular, these authors provide a formalisation of this operator to analyse the relationship between the solution probability density functions before and after its application, assuming an infinite population. They state that BLX-α spreads the distribution of the chromosomes when $\alpha > \frac{\sqrt{3}-1}{2}$, reducing it otherwise. This property was verified through simulations. In particular, Nomura et al. observed that BLX-0.0 makes the variances of the distribution of the chromosomes decrease, reducing the distribution, whereas BLX-0.5 makes the variances of the distribution increase, spreading the distribution.
  In this way, BLX-0.5 arises as a useful tool to enhance the global search (exploration) capabilities of an MA (and in particular of CSS). In the case of the CSS, this aspect becomes even more important, because it uses reference sets with very few individuals, thus increasing the risk of premature convergence. We introduce BLX-0.5 in the CSS algorithm to induce reliability in the search process, ensuring that different promising search zones are the focus of the improvement method.
  Furthermore, Nomura et al. offer theoretical properties of the average combination method and conclude that it decreases the variances and the covariances between the different coordinates of the solutions. Thus, repeated applications of average crossover reduce the distribution of the population.
- BLX-α has a *self-adaptive* nature in that it can generate offspring adaptively according to the distribution of parents without any adaptive parameter.
  Beyer et al. [3] argue that a variation operator that harness the difference of the parents in the search space is essential for the resulting EA to exhibit self-adaptive behaviour on the population level. BLX-α uses probability distributions that are calculated according to the distance between the decision variables in the parents ($x_i$ and $y_i$). If the parents are located closely to each other, the offspring generated by this combination method might distribute densely around the parents. On the other hand, if the parents are located far away from each other, then the offspring will be sparsely distributed around them. An emergent property of this setting is that BLX-α allows the EA to convergence, divergence, or adapt to changing objective function landscapes without incurring into extra parameters or mechanisms to achieve the mentioned behaviour. In fact, in the recent past, RCGAs with BLX-α have been demonstrated to exhibit self-adaptive behaviour similar to that observed in evolution strategies and evolutionary programming approaches [3].

## 3. Continuous local searchers

In this section we present a detailed description of two well-known continuous local searchers, the Solis and Wets' algorithm [39] and the Nelder and Mead's simplex method [36], that have been considered as improvement methods for CSS. They are random hill-climbing that search efficiently only for a local optimum. Next, we present a detailed description of these procedures.

**Solis and Wets' algorithm.** This local search algorithm is a randomized hill-climber with an adaptive step size. Each step starts at a current point $x$. A deviate $d$ is chosen from a normal distribution whose standard deviation is given by a parameter $\rho$. If either $x + d$ or $x - d$ is better, a move is made to the better point and a success is recorded. Otherwise, a failure is recorded. After several successes in a row, $\rho$ is increased to move more quickly. After several failures in a row, $\rho$ is decreased to focus the search. Additionally, a bias term is included to put the search momentum in directions that yield success. See [39] for details.

**Nelder–Mead simplex algorithm.** This is a classical and very powerful local descent algorithm, which makes no use of the objective function derivatives. A simplex is a geometrical figure consisting, in $n$ dimensions, of $n+1$ points $s_0, \ldots, s_n$. When a point of a simplex is taken as the origin, the $n$ other points are used to define vector directions that span the $n$-dimension vector space. Thus, if we randomly draw an initial starting point $s_0$, then we generate the other $n$ points $s_i$ according to the relation $s_i = s_0 + \lambda e_j$, where the $e_j$ are $n$ unit vectors, and $\lambda$ is a constant which is typically equal to one (but may be adapted to the problem characteristics).

Through a sequence of elementary geometric transformations (reflection, contraction, expansion and multi-contraction), the initial simplex moves, expands or contracts. To select the appropriate transformation, the method only uses the values of the function to be optimized at the vertices of the simplex considered. After each transformation, a better vertex replaces the current worst one.

At the beginning of the algorithm, only the point of the simplex where the objective function is worst is replaced, and another point, the image of the worst one, is generated. This being the reflection operation. If the reflected point is better than all other points, the method expands the simplex in this direction, otherwise, if it is at least better than the worst one, the algorithm performs again the reflection with the new worst point. The contraction step is performed when the worst point is at least as good as the reflected point, in such a way that the simplex adapts itself to the function landscape and finally surrounds the optimum. If the worst point is better than the contracted point, the multi-contraction is performed. At each step we check that the generated point is not outside the allowed reduced solution space.

## 4. Continuous scatter search: Components and implementation

In this section, we describe the implementation of CSS used for the experimental part of this paper (Section 5). It is based on the basic procedure outline in Fig. 3 [31], where $P$ denotes the set of solutions generated with the diversification generation method, *RefSet* is the set of solutions in the reference set and *Pool* is the set of trial solutions constructed with the combination and improvement methods. *Psize* is the size of $P$ ($Psize = 100$), $b_1$ is the initial number of high-quality in the reference set ($b_1 = 10$), $b_2$ is the initial number of diverse solutions in the reference set ($b_2 = 10$), and $d(\cdot,\cdot)$ is a dissimilarity measure between two solutions (we have employed the Euclidean distance).

The main method involved in the implementation of CSS are the following:

**Diversification generation method.** This method employs controlled randomization and frequency memory to generate a set of diverse solutions. We divide the range of each variable into four equal size sub-range. Afterwards, a solution is constructed on two step. In the first step, one of the sub-range is randomly selected, being the probability of selection inversely proportional to its

```
Continuous Scatter Search
  1. $P = \emptyset$.
  2. Repeat until $|P| = PSize$
     2.1. Use the Diversification Generation Method to obtain a solution $x$.
     2.2. Apply the Improvement Method to $x$ to generate the improved solution $x^*$.
     2.3. If $x^* \notin P$ then $P = P \cup \{x^*\}$ .
  3. Order the solutions in $P$ according to the objective function (the best overall
     solution is first on the list). Add the first $b_1$ solutions to $RefSet$ and delete these
     solutions from $P$.
  4. For $Iter = 1$ to $MaxIter$ Do
     4.1. Repeat $b_2$ times.
          * Select the solution $x' \in P - RefSet$ that maximises $d_{min}(x) = \min_{x \in RefSet} d(x, y)$.
          * $RefSet = RefSet \cup \{x'\}$ and $P = P - \{x'\}$
     4.2. $NewElements = TRUE$ and $Pool = \emptyset$.
     4.3. While ($NewElements$) Do
          * Generate $SubSets$ with the Subset Generation Method.
          * $NewElements = FALSE$.
          * While ($SubSets \neq \emptyset$) Do
            · Select the next subset $s \in SubSets$.
            · Apply the Solution Combination Method to $s$ to obtain a new
              solution, $x$.
            · Apply the Improvement Method to the solution $x$ to obtain the
              solution $x^*$.
            · $Pool = Pool \cup \{x^*\}$ and $SubSets = SubSets - \{s\}$.
          * Update $RefSet$ by selecting the best $b$ solutions in $RefSet \cup Pool$
          * If ($RefSet$ has at least one new solution) Then $NewElements = TRUE$
     4.4. If ($Iter < MaxIter$) Then
          * Delete the last (worst) $b_2$ solutions from $RefSet$.
          * Build a new set $P$ using the Diversification Generation Method.
```

Fig. 3. Pseudocode algorithm for CSS.

frequency count. in the second step, a value within the selected sub-range is randomly generated.

**Improvement method.** We have considered the *Solis and Wets' algorithm* and the *Nelder–Mead simplex algorithm* (Section 3).

**Subset generation method.** This method produces a subset of the solutions of the reference set as a basis for creating combined solutions. In particular, it generates all subsets of size 2, skipping subsets for which both elements have not changed from previously iterations.

**Solution combination method.** We use the BLX-$\alpha$ ($\alpha = 0.5$) operator and the average combination method (Section 2).

## 5. Experiments

In this section, we present the computational experiments carried out. Our main objective is to analyse the behaviour of CSS with different combination methods and local searchers (with different local search depth values).

In Section 5.1, we summarize the main characteristics of the test problems used for the experimentation. In Section 5.2, we provide details of the CSS instances executed and explain the choices of the parameter values. In Section 5.3, we explain the performance measure used to compare the results of the different CSS instances. In Section 5.4, we pay close attention to the local searchers. In Section 5.5, we detect whether the performance of CSS is improved using the BLX-0.5 operator instead of the average combination. Finally, we draw some preliminary conclusions in Section 5.6.

### 5.1. Test problems

The viability of a new feature in EAs is typically assessed by testing it on a number of problems. Here we restrict our attention to continuous optimization problems (minimization) and execute the experiments on a test suite that consists of 45 test functions (F1–F45) and three real-world problems (F46, F47, and F48). Table 1 summarizes their main features. A full description for these test problems is presented in Appendix A.

In order to facilitate the study of the results of the experiments, we have classified all the test problems into two groups, *simple* problems and *complex* problems. Table 2 outlines this classification. The first group includes those problems with a number of variables less or equals than three and some well-known functions that traditionally have been considered as easy problems. Examples are the Sphere function (F39 and F41) and the Trid problem (F25). The second group comprises the problems with more than three variables and those that are not categorized in the group of simple problems.

### 5.2. CSS Instances and parameter settings

We have implemented several CSS instances that are distinguished by the combination method, the improvement method, and the number of iterations (local search depth) ($n_I$) assigned to this method ($n_I = 0, 10, 50, 100$) (in the case of $n_I = 0$ no improvement method is applied). They follow the CSS framework presented in Section 4 with the following values for the CSS parameters: $Psize = 100$, $b_1 = 10$, and $b_2 = 10$.

The CSS instances based on BLX-$\alpha$ are denoted as "B-SW $n_I$" (with Solis and Wets' algorithm) and "B-S $n_I$" (with Nelder-Mead simplex algorithm), whereas the ones based on the average combination are denoted as "A-SW $n_I$" and "A-S $n_I$", respectively. We call "B" and "A" the CSS instances that do not use the improvement method.

There exist at least two ways to study the performance of search algorithms:

- The first one studies it from the point of view of the *efficacy* of the algorithms. This quality determines the accuracy levels of the solutions returned by the algorithm, without taking into account the time needed to obtain them. Effective algorithms are recommendable for real problems where the objective function may be evaluated very quickly and a precise solution is needed. To study the efficacy, we assign enough objective function evaluations to the algorithms and analyze the quality of the final solutions achieved.
- The second view concentrates on of the *efficiency* of the algorithms. Search algorithms are efficient when they return solutions with an 'acceptable' quality requiring few objective function evaluations. They are advisable for real-problem with a time-consuming objective function and where obtaining the exact solution is not the crucial objective. We access the efficiency of the algorithms by running them with a small number of objective function evaluations and analyzing the solutions obtained.

In our experiments, we are interested in investigating both the efficacy and the efficiency of the CSS instances. In order to do this:

- We carried out all the algorithms with different values for the maximum of evaluations ($n_{ev}$), 25,000, 50,000, 100,000, and 500,000.
- They were executed 50 times for every $n_{ev}$ value.

The results obtained are shown in Appendix B.

Table 1
Test problems

| Dimensions | Functions | Name and parameters | Range | $f(x^*)$ |
|---|---|---|---|---|
| 2 | F1 | Branin | [−5, 15] | 0.397887 |
| | F2 | B2 | [−50, 100] | 0 |
| | F3 | Easom | [−100, 100] | −1 |
| | F4 | Goldstein and Price | [−2, 2] | 3 |
| | F5 | Shubert | [−10, 10] | −186.7309 |
| | F6 | Beale | [−4.5, 4.5] | 0 |
| | F7 | Booth | [−10, 10] | 0 |
| | F8 | Matyas | [−5, 10] | 0 |
| | F9 | SixHumpCamelBack | [−5, 5] | −1.0316285 |
| | F10 | Schwefel(2) | [−500, 500] | 0 |
| | F11 | Rosenbrock(2) | [−10, 10] | 0 |
| | F12 | Zakharov(2) | [−5, 10] | 0 |
| 3 | F13 | DeJoung | [−2.56, 5.12] | 0 |
| | F14 | Hartmann(3,4) | [0, 1] | 0 |
| 4 | F15 | Colville | [−10, 10] | 0 |
| | F16 | Shekel(5) | [0, 10] | −10.1532 |
| | F17 | Shekel(7) | [0, 10] | −10.4029 |
| | F18 | Shekel(10) | [0, 10] | −10.53641 |
| | F19 | Perm(4,0.5) | [−4, 4] | 0 |
| | F20 | Perm0(4,10) | [−4, 4] | 0 |
| | F21 | PowerSum(8,18,44,114) | [0, 4] | 0 |
| 6 | F22 | Hartmann(6,4) | [0, 1] | 0 |
| | F23 | Schwefel(6) | [−500, 500] | 0 |
| | F24 | Trid(6) | [−100, 100] | −50 |
| 10 | F25 | Trid(10) | [−100, 100] | −210 |
| | F26 | Rastrigin(10) | [−2.56, 5.12] | 0 |
| | F27 | Griewank(10) | [−300, 600] | 0 |
| | F28 | Sum Squares(10) | [−5, 10] | 0 |
| | F29 | Rosenbrock(10) | [−10, 10] | 0 |
| | F30 | Zakharov(10) | [−5, 10] | 0 |
| 20 | F31 | Rastrigin(20) | [−2.56, 5.12] | 0 |
| | F32 | Griewank(20) | [−300, 600] | 0 |
| | F33 | Sum Squares(20) | [−5, 10] | 0 |
| | F34 | Rosenbrock(20) | [−10, 10] | 0 |
| | F35 | Zakharov(20) | [−5, 10] | 0 |
| >20 | F36 | Powell(24) | [−4, 5] | 0 |
| | F37 | Dixon and Price(25) | [−10, 10] | 0 |
| | F38 | Levy(30) | [−10, 10] | 0 |
| | F39 | Sphere(30) | [−2.56, 5.12] | 0 |
| | F40 | Ackley(30) | [−15, 30] | 0 |
| | F41 | Sphere(25) | [−4, 6] | 0 |
| | F42 | Rosenbrock(25) | [5.12, 5.12] | 0 |
| | F43 | Schewefel(25) | [−35, 95] | 0 |
| | F44 | Rastrigin(25) | [−8.5, 1.5] | 0 |
| | F45 | Griewank(25) | [−200, 1000] | 0 |
| | F46 | System Linear Equation(25) | [−127, 127] | 0 |
| | F47 | Frequency Modulation Sounds(25) | [−6.4, 6.35] | 0 |
| | F48 | Polynomial Fitting(25) | [−512, 512] | 0 |

Table 2
Classification for the test problems

| Group | Problems | Number of problems in the group |
|---|---|---|
| Simple | F1, F2, F3, F4, F5, F6, F7, F8, F9, F10, F11, F12, F13, F14, F16, F24, F25, F38, F39, F41 | 20 |
| Complex | F15, F17, F18, F19, F20, F21, F22, F23, F26, F27, F28, F29, F30, F31, F32, F33, F34, F35, F36, F37, F40, F40, F42, F43, F44, F45, F46, F47, F48 | 28 |

## 5.3. Performance measure

The performance measure used to facilitate the analysis of the performance of the CSS instances is defined taking into account a criterion of reaching a successful solution and a comparison criterion among algorithms. Both criteria were presented in [5,41]:

**Successful solution.** We consider that an algorithm reached a successfully solution, $x_s$, when the following is fulfilled [5,41]:

$$|f(x_s) - F_{\text{Opt}}| \leqslant 10^{-4} \cdot F_{\text{Opt}} + 10^{-6},$$

where $F_{\text{Opt}}$ is the value for the objective function of the global optimum.

**Comparison criterion.** We have assumed that algorithm $A$ is better than $B$ when:

- $A$ successfully finds solutions in a number of runs greater than $B$, i.e., $R_s(A) > R_s(B)$.
- $R_s(A) = R_s(B)$ and to locate successfully solutions $A$ required fewer objective function evaluations than $B$, i.e., $N_s(A) < N_s(B)$. A $t$-test (at 0.05 level of significance) was applied in order to ascertain if differences in $N_s(A)$ are significant when compared with $N_s(B)$.
- $R_s(A) = R_s(B) = 0$ and the average of the best-objective function found by $A$ at the end of each run is lower than for $B$. Again, a $t$-test is applied to determine whether the differences among $A$ and $B$ with respect to this performance measure are significant.

This comparison criterion allows us to determine the best performing algorithm from a given set of CSS instances for each test problem. In particular, we may evaluate the performance of a particular CSS instance by counting the number of test problems where it arises as the best algorithm or statistically equivalent to this one. All the tables presented below to analyze the behaviour of CSS are built based on this performance measure. A distinction is made between simple and complex test problems, as stated in Section 5.1, and the results for the different values for $n_{\text{ev}}$ are shown, with the objective of facilitating the study of the efficiency and the efficacy of the algorithms.

## 5.4. Study of the local searchers

In our first empirical study of the local searchers, we investigate the influence of $n_I$ (number of iterations accomplished by the local searchers, i.e., the local search depth) on the performance of the CSS, because it has a significant effect upon the exploration/exploitation balance. Tables 3–6 show the results of CSS instances based on the same combination methods and local searchers but with different $n_I$ values ($n_I = 0, 10, 50, 100$). They follow the structure described in the previous section.

From these tables we draw the following conclusions:

- An adequate performance is achieved when the Nelder–Mead simplex algorithm is applied on the simple problems with $n_I = 100$ (see B-S 100 and A-S 100 in Tables 3 and 4, respectively). This search procedure has a very good exploitation feature. In addition, high values for $n_I$ provide an elongated operation of the local searcher. For the simple problems, these facts allowed an effective refinement of solutions to be accomplished.

Table 3
BLX-α + Nelder–Mead simplex with different $n_I$ values

| Algorithm/$n_{ev}$ | Simple (20) | | | | Complex (28) | | | |
|---|---|---|---|---|---|---|---|---|
| | 25,000 | 50,000 | 100,000 | 500,000 | 25,000 | 50,000 | 100,000 | 500,000 |
| B | 6 | 5 | 6 | 6 | 9 | 8 | 9 | 14 |
| B-S 10 | 2 | 2 | 2 | 2 | 2 | 3 | 2 | 0 |
| B-S 50 | 6 | 6 | 6 | 6 | 12 | 7 | 2 | 3 |
| B-S 100 | 17 | 17 | 17 | 16 | 14 | 20 | 21 | 15 |

Table 4
Average combination + Nelder–Mead simplex with different $n_I$ values

| Algorithm/$n_{ev}$ | Simple (20) | | | | Complex (28) | | | |
|---|---|---|---|---|---|---|---|---|
| | 25,000 | 50,000 | 100,000 | 500,000 | 25,000 | 50,000 | 100,000 | 500,000 |
| A | 0 | 0 | 0 | 1 | 2 | 3 | 4 | 6 |
| A-S 10 | 2 | 2 | 1 | 1 | 2 | 0 | 0 | 0 |
| A-S 50 | 7 | 7 | 6 | 6 | 7 | 3 | 3 | 3 |
| A-S 100 | 18 | 18 | 18 | 17 | 25 | 25 | 24 | 23 |

Table 5
BLX-α + Solis and Wets with different $n_I$ values

| Algorithm/$n_{ev}$ | Simple (20) | | | | Complex (28) | | | |
|---|---|---|---|---|---|---|---|---|
| | 25,000 | 50,000 | 100,000 | 500,000 | 25,000 | 50,000 | 100,000 | 500,000 |
| B | 2 | 4 | 4 | 4 | 2 | 0 | 0 | 2 |
| B-SW 10 | 5 | 3 | 4 | 5 | 14 | 4 | 3 | 1 |
| B-SW 50 | 6 | 7 | 7 | 7 | 10 | 22 | 13 | 11 |
| B-SW 100 | 11 | 13 | 14 | 15 | 10 | 12 | 19 | 21 |

Table 6
Average combination + Solis and Wets with different $n_I$ values

| Algorithm/$n_{ev}$ | Simple (20) | | | | Complex (28) | | | |
|---|---|---|---|---|---|---|---|---|
| | 25,000 | 50,000 | 100,000 | 500,000 | 25,000 | 50,000 | 100,000 | 500,000 |
| A | 0 | 0 | 0 | 0 | 0 | 1 | 2 | 2 |
| A-SW 10 | 6 | 6 | 7 | 7 | 6 | 2 | 1 | 2 |
| A-SW 50 | 7 | 7 | 6 | 6 | 19 | 19 | 11 | 8 |
| A-SW 100 | 12 | 12 | 12 | 12 | 9 | 16 | 23 | 23 |

- $n_I = 100$ arises as the best choice for dealing with the complex problems when the Nelder-Mead simplex algorithm is combined with the average combination method (Table 4). Furthermore, similar results are achieved when we jointly apply this local searcher with BLX-α. However, in this case, there exists an important difference; the use of BLX-α with no local search procedure becomes sufficient to cause an effective performance for $n_{ev} = 500{,}000$. The diversity levels provided by this operator throughout these evaluations produce an adequate exploration of the search space that is well-suited for complex problems.

- In general, with $n_I = 100$ the most robust operation is obtained for the simple problems when the Solis and Wets' algorithm is applied (Tables 5 and 6). Again, an intense local search depth becomes suitable to deal with these problems.
- For the complex problems, $n_I = 50$ and $n_I = 100$ constitute the most interesting local search depths when using the Solis and Wets' algorithm. With $n_I = 50$, a remarkable efficiency is obtained (see the results for $n_{\mathrm{ev}} = 25{,}000$ and 50,000 in Tables 5 and 6), whereas with $n_I = 100$, this local searcher provides efficacy to the CSS instances (see the results for $n_{\mathrm{ev}} = 100{,}000$ and 500,000). In the initial stage of CSS, the application of the combination method is fundamental to explore the search space and locate promising zones to be refined by local searchers. Thus, when the CSS has few evaluations to carry out its operation, it is more important to perform the combination method quite a lot of times rather than wasting evaluations using deep local searches. $n_I = 50$ allows the CSS instances to execute the combination method more times than when $n_I = 100$, and therefore, it may induce better efficiency. On the other hand, when many evaluations are available, enough combination events may cover the search space to reach good zones, and deep local searches may accomplish the effective refinement of the solutions. The later explaining that $n_I = 100$ gains effectiveness as $n_{\mathrm{ev}}$ increases.

An additional objective of our experiments is to ascertain the local searcher that provides the best behaviour. In order to do this, we introduce Tables 7 and 8. All the algorithms in these tables apply the same combination method but differ in the local searcher and local search depth used.

In general, the Solis and Wets' algorithm with $n_I = 50$ and $n_I = 100$ was very useful for the complex problems, whereas the Nelder–Mead simplex algorithm with $n_I = 100$ obtained promising solutions for the simple problems. We should point out that the Solis and Wets' algorithm has explorative power whereas the Nelder–Mead simplex

Table 7
CSS instances with BLX-$\alpha$

| Algorithm/$n_{\mathrm{ev}}$ | Simple (20) | | | | Complex (28) | | | |
|---|---|---|---|---|---|---|---|---|
| | 25,000 | 50,000 | 100,000 | 500,000 | 25,000 | 50,000 | 100,000 | 500,000 |
| B | 1 | 2 | 2 | 2 | 2 | 0 | 0 | 1 |
| B-SW 10 | 5 | 3 | 3 | 4 | 12 | 4 | 2 | 1 |
| B-SW 50 | 3 | 4 | 4 | 4 | 8 | 19 | 11 | 9 |
| B-SW 100 | 2 | 4 | 5 | 6 | 9 | 12 | 16 | 18 |
| B-S 10 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 |
| B-S 50 | 3 | 3 | 3 | 4 | 1 | 0 | 0 | 0 |
| B-S 100 | 11 | 11 | 11 | 11 | 5 | 3 | 3 | 4 |

Table 8
CSS instances with average combination method

| Algorithm/$n_{\mathrm{ev}}$ | Simple (20) | | | | Complex (28) | | | |
|---|---|---|---|---|---|---|---|---|
| | 25,000 | 50,000 | 100,000 | 500,000 | 25,000 | 50,000 | 100,000 | 500,000 |
| A | 0 | 0 | 0 | 0 | 0 | 0 | 2 | 2 |
| A-SW 10 | 4 | 4 | 4 | 4 | 5 | 2 | 1 | 2 |
| A-SW 50 | 5 | 5 | 5 | 5 | 14 | 14 | 10 | 6 |
| A-SW 100 | 3 | 2 | 3 | 3 | 9 | 15 | 20 | 19 |
| A-S 10 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| A-S 50 | 4 | 4 | 4 | 4 | 1 | 1 | 0 | 0 |
| A-S 100 | 11 | 12 | 11 | 12 | 6 | 6 | 6 | 6 |

algorithm shows exploitative nature. Thus, each local searcher has the ability to adjust the search according to the peculiarities of particular problem instances.

Finally, we highlight that these tables show again the differences (regarding the efficiency and the efficacy) between the CSS instances based on the Solis and Wets' algorithm with $n_I = 50$ and $n_I = 100$ in complex problems.

### 5.5. Comparison between combination methods

We have introduced Tables 9–12 to compare the performance of the BLX-$\alpha$ operator and the average combination method. They outline the results of CSS instances based on the BLX-$\alpha$ operator and on the average combination with same conditions for the local searcher (type and number of iterations). Only the values 50 and 100 for $n_I$ have been considered.

Based on these results, we would make the following comments:

- The BLX-$\alpha$ operator outperforms the average combination method when combined with the Nelder–Mead simplex algorithm with a value of $n_I = 50$ (Table 9). In this case, the relationship between the exploration supplied by

Table 9
BLX-$\alpha$ vs. average combination (Nelder–Mead simplex with $n_I = 50$)

| Algorithm/$n_{ev}$ | Simple (20) | | | | Complex (28) | | | |
|---|---|---|---|---|---|---|---|---|
| | 25,000 | 50,000 | 100,000 | 500,000 | 25,000 | 50,000 | 100,000 | 500,000 |
| A-S 50 | 14 | 13 | 12 | 14 | 14 | 6 | 7 | 12 |
| B-S 50 | 18 | 18 | 18 | 17 | 17 | 26 | 26 | 23 |

Table 10
BLX-$\alpha$ vs. average combination (Nelder–Mead simplex with $n_I = 100$)

| Algorithm/$n_{ev}$ | Simple (20) | | | | Complex (28) | | | |
|---|---|---|---|---|---|---|---|---|
| | 25,000 | 50,000 | 100,000 | 500,000 | 25,000 | 50,000 | 100,000 | 500,000 |
| A-S 100 | 19 | 18 | 17 | 18 | 20 | 15 | 11 | 12 |
| B-S 100 | 17 | 17 | 17 | 17 | 14 | 22 | 23 | 23 |

Table 11
BLX-$\alpha$ vs. average combination (Solis and Wets' with $n_I = 50$)

| Algorithm/$n_{ev}$ | Simple (20) | | | | Complex (28) | | | |
|---|---|---|---|---|---|---|---|---|
| | 25,000 | 50,000 | 100,000 | 500,000 | 25,000 | 50,000 | 100,000 | 500,000 |
| A-SW 50 | 18 | 19 | 19 | 19 | 24 | 18 | 16 | 13 |
| B-SW 50 | 16 | 15 | 15 | 15 | 8 | 17 | 18 | 20 |

Table 12
BLX-$\alpha$ vs. average combination (Solis and Wets' with $n_I = 100$)

| Algorithm/$n_{ev}$ | Simple (20) | | | | Complex (28) | | | |
|---|---|---|---|---|---|---|---|---|
| | 25,000 | 50,000 | 100,000 | 500,000 | 25,000 | 50,000 | 100,000 | 500,000 |
| A-SW 100 | 19 | 19 | 20 | 20 | 25 | 23 | 17 | 15 |
| B-SW 100 | 15 | 16 | 16 | 16 | 9 | 14 | 17 | 21 |

BLX-$\alpha$ and the exploitation of the Nelder–Mead simplex algorithm allowed promising results to be reached.

- In general, Tables 9–12 indicate that the average combination method slightly outperforms the BLX-$\alpha$ operator on the simple problems. However, for the complex problems, regarding efficiency and efficacy, there exists a compromise between these operators. In particular, the use of BLX-$\alpha$ allows the best efficacy to be obtained on this type of problems (see the results of B-S 100, B-SW 50, and B-SW 100 for $n_{ev} = 500{,}000$ in Tables 9–12), while the average combination method contributes efficiency to the CSS instances (see the performance of A-S 50, A-SW 50, and A-SW 100 for $n_{ev} = 25{,}000$).

### 5.6. Main results of the experiments

In this section, we summarize the main conclusions derived from the results of the experiments:

1. The local search method affects decisively the CSS behaviour. The Nelder–Mead simplex algorithm ($n_I = 100$) becomes determinant to achieve the best solutions for the simple problems, while for the complex ones, the Solis and Wets' algorithm is the best choice to obtain efficiency ($n_I = 100$) and reach efficacy ($n_I = 50$).
2. BLX-$\alpha$ appears to be as an attractive combination method for CSS, because it may bring together properties that are needed in an effective combination method (randomness, diversity, and self-adaptation). However, we should point out that these features have an impact when many CSS iterations have been accomplished. In the case of problems where the evaluation of the objective function takes quite a long time, the average combination method becomes recommendable, with its associated high exploitation property inducing a rapid convergence towards the initially detected promising zones. In this way, it may show an efficient behaviour because obtains good results requiring few evaluations.

## 6. Comparison of CSS with other continuous optimization algorithms

In this section, we compare CSS with other continuous optimization techniques appeared in the literature. We have considered the CSS instance based on the effective BLX-$\alpha$ operator and on the Nelder–Mead simplex algorithm (which provided an adequate performance on the simple test problems). The algorithms we are comparing CSS with are the following:

- *Continuous Hybrid Algorithm* (CHA) [6]. This algorithm performs a fast exploration step with a GA, and then a exploitation step with the Nelder-Mead simplex algorithm.
- *Enhanced Continuous Tabu Search* (ECTS) [5]. This algorithm firstly locates the most promising areas of the search space by fitting the size of the neighborhood structure to the objective function and its definition domain, and thus continues the search by *intensification* within one of these areas.
- *Continuous Genetic Algorithm* (CGA) [4]. This GA pays a special attention to the choice of the initial population. Then, it locates the most promising area of the search space, and continues the search through an intensification inside this area. The selection, the crossover and the mutation are performed by using the decimal base.
- *Enhanced Simulated Annealing* (ESA) [38]. This is a variant of the classical *Simulated Annealing*. In ESA, the original Metropolis iterative random search (that takes place in the Euclidean space, $R^n$) is replaced by another similar exploration technique, which is performed within a succession of Euclidean spaces, $R_p$, with $p \ll n$.
- *Continuous Reactive Tabu Search* (CRTS) [2]. This algorithm is a variant of the Tabu Search, which is based on a different Tabu scheme. The appropriate size of the list is learned in an automated way by reacting to the occurrence of cycles. In addition, if the search process appears to be repeating an excessive number of solutions excessively often, then the search is diversified by making a number of random moves proportional to a moving average of

the cycle length. Two versions of CRTS have been proposed, the CRTR minimum and the CRTR average.

The comparison is made on a set of eight test problems for which there exist results of the previous algorithms that are available in the corresponding publications. They include F1, F2, F4, F5, F11, F12, F14, and F16 (Table 1). All these problems were classified as simple problems (Table 2).

The continuous optimization algorithms were executed 100 times with different random seeds. These algorithms finish a run when a *successful* solution is achieved (see Section 5.3), except for CRTS, which use its own criterion [2].

The experimental results are shown in Table 13, with some results not available for some of the methods. The following information appears for each function and algorithm:

- One number only. This means that the algorithm might return successful solutions after the 50 runs, and the number outlined is the average number of evaluations accomplished to achieve successful solutions.
- Two numbers. The number in parentheses is the percentage of runs in which the algorithm found successful solutions (when this percentage is zero, we introduce a # sign). The first number is the average number of evaluations accomplished to achieve successful solutions during these runs.

We may remark that, the CSS instance outperforms most of the other algorithms on six test problems (F1, F2, F4, F11, F12, and F16). Therefore, we may conclude that CSS is very competitive with continuous optimization algorithms.

## 7. Conclusions

In this paper, we empirically studied the two main components of CSS, the combination method and the local searcher. In particular, we attempted to determine the efficiency and efficacy of two combinations methods, the BLX-$\alpha$ operator and the classical average combination method, and two local searchers, the Nelder–Mead simplex algorithm and the Solis and Wets' algorithm. In order to do this, we have considered two types of test problems, simple and complex, and ran many CSS instances with different number of evaluations.

The principal conclusions derived from the results of the experiments carried out are:

1. BLX-$\alpha$ is a suitable combination method for CSS. Its properties contribute to improve the efficacy of CSS, with respect to the one provided by the classical average combination method. However, with applications with problems with time-consuming objective functions, the later becomes the best election, because it was more efficient.
2. The exploitation properties of the Nelder–Mead simplex algorithm allow the best refined solutions to be achieved for the simple problems, while the exploration ability of the Solis and Wets' algorithm produces adequate improvements for the complex problems.

Table 13
CSS instances vs. other continuous optimization algorithms

| Problem | Algorithms | | | | | | |
|---|---|---|---|---|---|---|---|
| | B-S 100 | CHA | CGA | ECTS | CRTS min. | CRTS ave. | ESA |
| $F_1$(RC) | 65 | 295 | 620 | 245 | 41 | 38 | – |
| $F_2(B_2)$ | 108 | 132 | 320 | 210 | – | – | – |
| $F_4$(GP) | 190 | 259 | 410 | 231 | 171 | 248 | 783 |
| $F_5$(SH) | 762 | 345 | 575 | 370 | – | – | – |
| $F_{11}(R_2)$ | 292 | 459 | 960 | 480 | – | – | 796 |
| $F_{12}(Z_2)$ | 59 | 215 | 620 | 195 | – | – | 15820 |
| $F_{14}(H_{3,4})$ | # | 492 | 582 | 548 | 609 | 513 | 698 |
| $F_{16}(S_{4,5})$ | 1197 | 598 (85%) | 610 (76%) | 825 (75%) | 664 | 812 | 1137 (54%) |

3. CSS is very competitive with continuous optimization algorithms.

In essence, CSS is a very promising method and indeed worth further research. In particular, extensions of the present study may be carried out in different ways: (1) analysis of the effects of other combination methods proposed for RCGAs [27] on the CSS behaviour; (2) design adaptive techniques that select, during the CSS run, the local searcher (and the value for the local search depth) most appropriate to the problem being solved; (3) study the synergy produced by combining the different styles of the traversal of solution space associated with the different combination methods. This may be done by means of hybrid combination methods, which generate two offspring for every pair of parents, each one with a different combination method. In fact, in [31], it was suggested the use of multiple combination methods may be suitable.

## Appendix A. Test suite

The test suite that we have used for the experiments consists of 45 test functions and three real-world problems. They are described in Sections A.1 and A.2, respectively.

### A.1. Test functions

This appendix contains the description of the set of test functions. We have included the objective function, parameter values, and the bounds of each variable (Tables 14 and 17; also Tables 15, 16 and 18.

### A.2. Real-world problems

We have chosen the following three real-world problems, which, in order to be solved, are translated to optimization problems of parameters with variables on continuous domains: *Systems of*

Table 14
Functions: Part 1

| Name | $f(x)$ | Range | Dimension |
|---|---|---|---|
| Branin | $\left(x_2 - \left(\frac{5}{4\pi^2}\right)x_1^2 + \left(\frac{5}{\pi}\right)x_1 - 6\right)^2 + 10\left(1 - \frac{1}{8\pi}\right)\cos(x_1) + 10$ | $-5 \leqslant x_1, x_2 \leqslant 15$ | 2 |
| B2 | $x_1^2 + 2x_2^2 - 0.3\cos(3\pi x_1) - 0.4\cos(4\pi x_2) + 0.7$ | $-50 \leqslant x_i \leqslant 100$ | 2 |
| Eason | $-\cos(x_1)\cos(x_2)\exp(-((x_1-\pi)^2 + (x_2-\pi)^2))$ | $-100 \leqslant x_i \leqslant 100$ | 2 |
| Goldstein & Price | $\left(1 + (x_1+x_2+1)^2(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)\right)$ $\left(30 + (2x_1 - 3x_2)^2(18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)\right)$ | $-2 \leqslant x_i \leqslant 2$ | 2 |
| Shubert | $\left(\sum_{j=1}^{5} j\cos((j+1)x_1 + j)\right)\left(\sum_{j=1}^{5} j\cos((j+1)x_2 + j)\right)$ | $-10 \leqslant x_i \leqslant 10$ | 2 |
| Beale | $(1.5 - x_1 + x_1x_2)^2 + (2.25 - x_1 + x_1x_2^2)^2 + (2.625 - x_1 + x_1x_2^3)^2$ | $-4.5 \leqslant x_1, x_2 \leqslant 4.5$ | 2 |
| Booth | $(x_1 + 2x_2 - 7)^2 + (2x_1 + x_2 - 5)^2$ | $-10 \leqslant x_1, x_2 \leqslant 10$ | 2 |
| Matyas | $0.26(x_1^2 + x_2^2) - 0.48x_1x_2$ | $-5 \leqslant x_1, x_2 \leqslant 10$ | 2 |
| SixHumpCamelBack | $4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4$ | $-5 \leqslant x_1, x_2 \leqslant 10$ | 2 |
| Schwefel | $418.9829n + \sum_{i=1}^{n}\left(-x_i\sin\sqrt{\lvert x_i\rvert}\right)$ | $-500 \leqslant x_i \leqslant 500$ | $n$ |
| Rosenbrock | $\sum_{i=1}^{n/2} 100(x_{2i} - x_{2i-1}^2)^2 + (1 - x_{2i-1})^2$ | $-10 \leqslant x_i \leqslant 10$ | $n$ |
| Zakharov($n$) | $\sum_{j=1}^{n} x_j^2 + \left(\sum_{j=1}^{n} 0.5jx_j\right)^2 + \left(\sum_{j=1}^{n} 0.5jx_j\right)$ | $-5 \leqslant x_i \leqslant 10$ | $n$ |
| DeJoung | $x_1^2 + x_2^2 + x_3^2$ | $-2.56 \leqslant x_i \leqslant 5.12$ | 3 |
| Hartmann(3,4) | $-\sum_{i=1}^{4} c_i \exp\left(\sum_{j=1}^{3} a_{ij}(x_j - p_{ij})^2\right)$ (see Table 15) | $0 \leqslant x_i \leqslant 1$ | 3 |
| Colville | $100(x_2 - x_1^2)^2 + (1 - x_1)^2 + 90(x_4 - x_3^2)^2 + (1 - x_3)^2$ $+10.1((x_2 - 1)^2 + (x_4 - 1)^2) + 19.8(x_2 - 1)(x_4 - 1)$ | $-10 \leqslant x_i \leqslant 10$ | 4 |
| Shekel($n$) | $-\sum_{i=1}^{n}((x - a_i)^{\mathrm{T}}(x - a_i) + c_i)$; $x = (x_1, x_2, x_3, x_4)^{\mathrm{t}}$; $a_i = (a_i^1, a_i^2, a_i^3, a_i^4)^{\mathrm{T}}$ (see Table 16) | $0 \leqslant x_i \leqslant 10$ | 4 |

Table 15
Parameters used in *Hartmann(3,4)*

| i | $a_{ij}$ | | | $c_i$ | $p_{ij}$ | | |
|---|---|---|---|---|---|---|---|
| 1 | 3.0 | 10.0 | 30.0 | 1.0 | 0.3689 | 0.1170 | 0.2673 |
| 2 | 0.1 | 10.0 | 35.0 | 1.2 | 0.4699 | 0.4387 | 0.7470 |
| 3 | 3.0 | 10.0 | 30.0 | 3.0 | 0.1091 | 0.8732 | 0.5547 |
| 4 | 0.1 | 10.0 | 35.0 | 3.2 | 0.0381 | 0.5743 | 0.8828 |

Table 16
Parameters used in *Shekel(n)*

| i | $a_j^{\mathrm{T}}$ | | | | $c_i$ |
|---|---|---|---|---|---|
| 1 | 4.0 | 4.0 | 4.0 | 4.0 | 0.1 |
| 2 | 1.0 | 1.0 | 1.0 | 1.0 | 0.2 |
| 3 | 8.0 | 8.0 | 8.0 | 8.0 | 0.2 |
| 4 | 6.0 | 6.0 | 6.0 | 6.0 | 0.4 |
| 5 | 3.0 | 7.0 | 3.0 | 7.0 | 0.4 |
| 6 | 2.0 | 9.0 | 2.0 | 9.0 | 0.6 |
| 7 | 5.0 | 5.0 | 3.0 | 3.0 | 0.3 |
| 8 | 8.0 | 1.0 | 8.0 | 1.0 | 0.7 |
| 9 | 6.0 | 2.0 | 6.0 | 2.0 | 0.5 |
| 10 | 7.0 | 3.6 | 7.0 | 3.6 | 0.5 |

*Linear Equations* [12], *Frequency Modulation Sounds Parameter Identification Problem* [44], and *Polynomial Fitting Problem* [40]. They are described below.

*Systems of Linear Equations*. In this problem, given the matrix $A$ and vector $B$, the elements of a vector, $X$, has to be obtain such that: $A \cdot X = B$. The evaluation function used for these experiments is

$$P_{\mathrm{sle}}(x_1, \ldots, x_n) = \sum_{i=1}^{n} \sum_{j=1}^{n} (a_{ij} \cdot x_j) - b_j.$$

Clearly, the best value for this objective function is $P_{\mathrm{sle}}(x^*) = 0$. Inter-parameter linkage (i.e. nonlinearity) is easily controlled in systems of linear equations, their nonlinearity does not deteriorate when the numbers of parameters used increases, and they have proven to be quite difficult.

We have considered a 10-parameter problem instance. Its matrices are shown in Table 19.

Table 17
Functions: Part 2

| Name | $f(x)$ | Range | Dimension |
|---|---|---|---|
| *Perm*($n$, $\beta$) | $\sum_{k=1}^{n}\left(\sum_{i=1}^{n}(i^k+\beta)\left(\left(\frac{x_i}{i}\right)^k-1\right)\right)$ | $-n \leqslant x_i \leqslant n$ | $n$ |
| Perm0($n$, $\beta$) | $\sum_{k=1}n\left(\sum_{i=1}^{n}(i+\beta)\left(x_i^k-\left(\frac{1}{i}\right)^k\right)\right)^2$ | $-n \leqslant x_i \leqslant n$ | $n$ |
| PowerSum($b_1 \cdots b_n$) | $\sum_{k=1}^{n}\left(\left(\sum_{i=1}^{n}x_i^k\right)-b_k\right)^2$ | $0 \leqslant x_i \leqslant n$ | 2 |
| Hartman(6,4) | $-\sum_{i=1}^{4}c_i\exp\left(-\sum_{j=1}^{6}a_{ij}\left(x_j-p_{ij}\right)^2\right)$ (see Table 18) | $0 \leqslant x_i \leqslant n$ | 6 |
| Trid($n$) | $\left(\sum_{i=2}^{n}(x_i-1)^2\right)-\sum_{i=2}^{n}x_ix_{i-1}$ | $-n^2 \leqslant x_i \leqslant n^2$ | $n$ |
| Rastrigin($n$) | $10n+\sum_{i=1}n\left(x_i^2-10\cos(2\pi x_i)\right)$ | $-2.56 \leqslant x_i \leqslant 5.12$ | $n$ |
| Griewank($n$) | $\sum_{i=1}^{n}\frac{x_i^2}{4000}-\prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right)+1$ | $-300 \leqslant x_i \leqslant 600$ | $n$ |
| Sum Squares($n$) | $\sum_{i=1}^{n}ix_i^2$ | $-5 \leqslant x_i \leqslant 10$ | $n$ |
| Powell($n$) | $\sum_{j=1}^{n/4}(x_{4j-3}+10x_{4j-2})^2+5(x_{4j-1}-x_{4j})^2+(x_{4j-2}-2x_{4j-1})^4$ $+10(x_{4j-3}-x_{4j})^4$ | $-4 \leqslant x_i \leqslant 5$ | $n$ |
| Dixon & Price($n$) | $\sum_{i=1}^{n}i(2x_i^2-x_{i-1})^2+(x_1-1)^2$ | $-10 \leqslant x_i \leqslant 10$ | $n$ |
| Levy($n$) | $\sin^2(\pi y_1)+\sum_{i=1}^{k-1}(y_i-1)^2(1+10\sin^2(\pi y_i+1))+(y_k-1)^2(1+\sin^2(2\pi x_k))$ where $y_i=1+\frac{x_i-1}{4}$ for $i=1,\cdots,n$ | $-10 \leqslant x_i \leqslant 10$ | $n$ |
| Sphere($n$) | $\sum_{i=1}^{n}x_i^2$ | $-2.56 \leqslant x_i \leqslant 5.12$ | $n$ |
| Ackley($n$) | $20+e-20e^{-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n}x_i^2}}-e^{\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)}$ | $-15 \leqslant x_i \leqslant 30$ | $n$ |

Table 18
Parameters used in *Hartman(6,4)*

| $i$ | $a_{ij}$ | | | | | | $c_i$ | $p_{ij}$ | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 10.0 | 3.0 | 17.0 | 3.5 | 1.7 | 8.0 | 1.0 | 0.1312 | 0.1696 | 0.5569 | 0.0124 | 0.8283 | 0.5886 |
| 2 | 0.05 | 10.0 | 17.0 | 0.10 | 8.0 | 14.0 | 1.2 | 0.2329 | 0.4135 | 0.8307 | 0.3736 | 0.1004 | 0.9991 |
| 3 | 3.0 | 3.5 | 1.7 | 10.0 | 17.0 | 8.0 | 3.0 | 0.2348 | 0.1451 | 0.3522 | 0.2883 | 0.3047 | 0.6650 |
| 4 | 17.0 | 8.0 | 0.05 | 10.0 | 0.1 | 14.0 | 3.2 | 0.4047 | 0.8828 | 0.8732 | 0.5743 | 0.1091 | 0.0381 |

Table 19
Matrices for the system of linear equations

$$\begin{vmatrix} 5 & 4 & 5 & 2 & 9 & 5 & 4 & 2 & 3 & 1 \\ 9 & 7 & 1 & 1 & 7 & 2 & 2 & 6 & 6 & 9 \\ 3 & 1 & 8 & 6 & 9 & 7 & 4 & 2 & 1 & 6 \\ 8 & 3 & 7 & 3 & 7 & 5 & 3 & 9 & 9 & 5 \\ 9 & 5 & 1 & 6 & 3 & 4 & 2 & 3 & 3 & 9 \\ 1 & 2 & 3 & 1 & 7 & 6 & 6 & 3 & 3 & 3 \\ 1 & 5 & 7 & 8 & 1 & 4 & 7 & 8 & 4 & 8 \\ 9 & 3 & 8 & 6 & 3 & 4 & 7 & 1 & 8 & 1 \\ 8 & 2 & 8 & 5 & 3 & 8 & 7 & 2 & 7 & 5 \\ 2 & 1 & 2 & 2 & 9 & 8 & 7 & 4 & 4 & 1 \end{vmatrix} \begin{vmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{vmatrix} = \begin{vmatrix} 40 \\ 50 \\ 47 \\ 59 \\ 45 \\ 35 \\ 53 \\ 50 \\ 55 \\ 40 \end{vmatrix}$$

*Frequency Modulation Sounds Parameter Identification Problem.* The problem is to specify six parameters $a_1, w_1, a_2, w_2, a_3, w_3$ of the frequency modulation sound model represented by

$$y(t) = a_1 \cdot \sin(w_1 \cdot t \cdot \theta + a_2 \cdot \sin(w_2 \cdot t \cdot \theta + a_3 \cdot \sin(w_3 \cdot t \cdot \theta))),$$

with $\theta = \frac{2\pi}{100}$. The objective function is defined as the summation of square errors between the evolved data and the model data as follows:

$$P_{\text{fms}}(a_1, w_1, a_2, w_2, a_3, w_3) = \sum_{t=0}^{100} (y(t) - y_0(t))^2,$$

where the model data are given by the following equation:

$$y_0(t) = 1.0 \cdot \sin(5.0 \cdot t \cdot \theta - 1.5 \cdot \sin(4.8 \cdot t \cdot \theta + 2.0 \cdot \sin(4.9 \cdot t \cdot \theta))).$$

Each parameter is in the range $-6.4$ to 6.35. This problem is a highly complex multimodal one having strong epistasis, with minimum value $P_{\text{fms}}(x^*) = 0$.

*Polynomial Fitting Problem.* In this problem the coefficients of the following polynomial in $z$:

$$P(z) = \sum_{j=0}^{2k} c_j \times z^j, \quad k > 0 \text{ is integer},$$

have to be found, such that

$$P(z) \in [-1, 1] \quad \text{for } z \in [-1, 1], \quad \text{and}$$

$$P(1.2) \geqslant T_{2k}(1.2) \quad \text{and} \quad P(-1.2) \geqslant T_{2k}(-1.2),$$

where $T_{2k}(z)$ is a Chebychev polynomial of degree $2k$.

The solution to the polynomial fitting problem depends on the coefficients of $T_{2k}(z)$. This polynomial oscillates between $-1$ and 1 when its argument $z$ is between $-1$ and 1. Outside this region the polynomial rises steeply in direction of high positive ordinate values. This problem has its roots in electronic filter design and challenges an optimization procedure by forcing it to find parameter values with grossly different magnitudes, something very common in technical systems. The Chebychev polynomial employed here is

$$T_8(z) = 1 - 32 \cdot z^2 + 160 \cdot z^4 - 256 \cdot z^6 + 128 \cdot z^8.$$

So, it is a nine-parameter problem. The pseudocode algorithm shown below was used in order to transform the constraints of this problem into an objective function to be minimized, called $P_{\text{Chev}}$. We consider that $C = (c_0, \ldots, c_8)$ is the solution to be evaluated and $P_C(z) = \sum_{j=0}^{8} c_j \times z^j$.

```
**Choose** $p_0, p_2, \ldots, p_{100}$ from $[-1, 1]$;
$R = 0$;
**For** $i = 0, \ldots, 100$ **do**
    **If** $(-1 > P_C(p_i)$ **or** $P_C(p_i) > 1)$ **then** $R \leftarrow R + (1 - P_C(p_i))^2$;
**If** $(P_C(1.2) - T_8(1.2) < 0)$ **then** $R \leftarrow R + (P_C(1.2) - T_8(1.2))^2$;
**If** $(P_C(-1.2) - T_8(-1.2) < 0)$ **then** $R \leftarrow R + (P_C(-1.2) - T_8(-1.2))^2$;
**Return** $R$;
```

Table 20
Results of CSS instances with average combination and $n_{ev} = 25{,}000$

| Functions | A | A-S 10 | A-S 50 | A-S 100 | A-SW 10 | A-SW 50 | A-SW 100 |
|---|---|---|---|---|---|---|---|
| $f_1$ | 4495(62%) | 3251(40%) | 59(100%) | 65(100%) | 3032(100%) | 232(100%) | 106(100%) |
| $f_2$ | 12948(6%) | 8045(14%) | 8392(94%) | 108(100%) | 5925(100%) | 8003(100%) | 872(100%) |
| $f_3$ | 5730(74%) | 4764(82%) | 4028(100%) | 3615(100%) | 4546(100%) | 11945(100%) | 14297(88%) |
| $f_4$ | 11392(32%) | 7253(22%) | 7885(100%) | 190(100%) | 5350(100%) | 8555(100%) | 4583(100%) |
| $f_5$ | 10438(14%) | 3872(20%) | 485(100%) | 762(100%) | 4377(100%) | 2051(100%) | 2748(100%) |
| $f_6$ | 1854(28%) | 3731(62%) | 391(100%) | 132(100%) | 4093(100%) | 6198(100%) | 3975(100%) |
| $f_7$ | 5461(68%) | 4107(76%) | 325(100%) | 62(100%) | 4075(98%) | 3926(100%) | 138(100%) |
| $f_8$ | 5182(82%) | 2866(82%) | 94(100%) | 54(100%) | 2921(100%) | 960(100%) | 165(100%) |
| $f_9$ | 4623(78%) | 4435(88%) | 57(100%) | 54(100%) | 2650(100%) | 224(100%) | 147(100%) |
| $f_{10}$ | 4.873186e+01 | 1.764338e+01 | 1.421264e+01 | 1.421263e+01 | 7.521872e+01 | 1029(100%) | 1963(100%) |
| $f_{11}$ | 6078(18%) | 5370(14%) | 5420(100%) | 292(100%) | 6144(96%) | 14284(98%) | 17452(36%) |
| $f_{12}$ | 8042(54%) | 5257(42%) | 176(100%) | 59(100%) | 3709(100%) | 1068(100%) | 118(100%) |
| $f_{13}$ | 2.477039e+03 | 6388(12%) | 12972(80%) | 8219(100%) | 12873(94%) | 9039(100%) | 13912(100%) |
| $f_{14}$ | 1.302887e−03 | 3.772719e−05 | 3.772719e−05 | 3.772719e−05 | 5(100%) | 5(100%) | 6(100%) |
| $f_{15}$ | 4.805646e+00 | 1.317702e+00 | 23074(6%) | 23200(100%) | 10950(4%) | 23662(46%) | 8.099810e−03 |
| $f_{16}$ | −4.521676e+00 | 8053(16%) | 11114(90%) | 1197(100%) | 5926(84%) | 8316(100%) | 568(100%) |
| $f_{17}$ | −4.926995e+00 | 5897(14%) | 11248(84%) | 1837(100%) | 6220(84%) | 8384(100%) | 637(100%) |
| $f_{18}$ | −4.981920e+00 | 8138(16%) | 12050(86%) | 2264(100%) | 6058(92%) | 8693(100%) | 762(100%) |
| $f_{19}$ | 4.539528e+01 | 3.142514e−01 | 2.457277e−03 | 24996(2%) | 3.786004e−02 | 3.916290e−02 | 4.787346e−01 |
| $f_{20}$ | 9.548831e+00 | 2.048213e−01 | 9.753939e−04 | 24593(32%) | 3.563897e−03 | 23367(4%) | 5.915593e−04 |
| $f_{21}$ | 1.901987e−01 | 1.344705e−01 | 22164(4%) | 21620(30%) | 1.836404e−03 | 5.445992e−04 | 6.056357e−03 |
| $f_{22}$ | 1.364826e−04 | 1479(100%) | 401(100%) | 394(100%) | 5(100%) | 5(100%) | 5(100%) |
| $f_{23}$ | 8.691992e+02 | 6.932100e+02 | 5.026535e+02 | 4.615707e+02 | 8.555608e+02 | 14392(72%) | 10179(72%) |
| $f_{24}$ | −3.285721e+01 | −4.929780e+01 | 13928(80%) | 10951(100%) | 7642(100%) | 10073(100%) | 14903(100%) |
| $f_{25}$ | −5.315097e+01 | −1.741643e+02 | −2.092337e+02 | 23037(10%) | −2.074414e+02 | 19731(24%) | 23172(28%) |
| $f_{26}$ | 3.671281e+01 | 1.694190e+01 | 9.691185e+00 | 8.796759e+00 | 7.190457e+00 | 2.707225e+00 | 6.619372e+00 |
| $f_{27}$ | 7.291894e+00 | 2.230309e+00 | 8.015551e−01 | 5.745147e−01 | 4.181902e+00 | 6.171703e−02 | 2.921184e−02 |
| $f_{28}$ | 3.964451e+01 | 8.354419e+00 | 1.897373e−01 | 7.478959e−03 | 5.201159e−03 | 24744(72%) | 24847(2%) |
| $f_{29}$ | 1.550863e+01 | 3.814218e+01 | 1.699491e+01 | 8.105333e+00 | 7.897178e+00 | 6.167869e+00 | 5.502973e+00 |
| $f_{30}$ | 7.166821e+02 | 7.278385e+01 | 1.589024e+00 | 2.819667e−02 | 2.446548e−02 | 24913(2%) | 2.727571e−03 |
| $f_{31}$ | 1.156995e+02 | 5.886076e+01 | 5.707052e+01 | 5.986409e+01 | 2.185188e+01 | 3.136370e+01 | 3.364579e+01 |
| $f_{32}$ | 3.677444e+01 | 2.788269e+01 | 9.167075e+00 | 5.713997e+00 | 1.677493e+01 | 1.142510e−01 | 3.641588e−01 |
| $f_{33}$ | 3.482395e+02 | 2.078542e+02 | 8.547853e+01 | 5.394773e+01 | 7.060854e−01 | 5.430729e−02 | 3.030565e−01 |
| $f_{34}$ | 6.066603e+01 | 2.366067e+02 | 5.629692e+01 | 4.103627e+01 | 2.437798e+01 | 1.807899e+01 | 2.564524e+01 |
| $f_{35}$ | 1.061222e+05 | 1.912938e+04 | 4.841746e+03 | 1.222154e+03 | 1.203542e+01 | 3.512694e+00 | 8.872391e+00 |
| $f_{36}$ | 1.568707e+01 | 2.663586e+01 | 6.603822e+00 | 4.575637e+00 | 2.537351e+00 | 4.651300e−01 | 1.684909e+00 |
| $f_{37}$ | 3.291454e+01 | 2.607162e+02 | 4.764015e+01 | 2.837178e+01 | 1.035493e+01 | 3.915858e+00 | 1.976865e+01 |
| $f_{38}$ | 20399(10%) | 20157(18%) | 19490(100%) | 20827(6%) | 7539(100%) | 22480(44%) | 3.137262e+03 |
| $f_{39}$ | 2.100659e+01 | 2.055288e+01 | 1.230474e+01 | 1.014949e+01 | 1.847689e−02 | 1.452993e−03 | 1.681603e−02 |
| $f_{40}$ | 1.398503e+01 | 1.403284e+01 | 1.306967e+01 | 1.339294e+01 | 1.358756e+00 | 1.224861e+01 | 1.538767e+01 |
| $f_{41}$ | 1.092729e+00 | 3.537832e+00 | 4.638581e−01 | 1.455110e−01 | 1.155768e−02 | 2.814789e−04 | 2.792748e−03 |
| $f_{42}$ | 3.864811e+01 | 1.564716e+02 | 7.757815e+01 | 5.012898e+01 | 2.564599e+01 | 2.313399e+01 | 3.395975e+01 |
| $f_{43}$ | 9.883974e+03 | 9.902742e+03 | 9.872872e+03 | 9.897276e+03 | 9.886531e+03 | 8.575071e+03 | 8.464005e+03 |
| $f_{44}$ | 3.225228e+02 | 1.758841e+02 | 1.943536e+02 | 2.256733e+02 | 1.356565e+02 | 1.539154e+02 | 1.925844e+02 |
| $f_{45}$ | 5.032617e+02 | 4.978353e+02 | 3.838256e+02 | 3.348829e+02 | 4.135740e+02 | 9.536303e−01 | 1.098985e+00 |
| $f_{46}$ | 9.824179e+01 | 2.345598e+02 | 6.889354e+01 | 2.950849e+01 | 1.799457e+02 | 4.577628e+01 | 1.170855e+02 |
| $f_{47}$ | 2.526427e+01 | 2.154433e+01 | 1.512113e+01 | 1.465682e+01 | 1.944825e+01 | 1.644269e+01 | 1.739829e+01 |
| $f_{48}$ | 3.503341e+03 | 1.410794e+04 | 4.160209e+03 | 1.737367e+03 | 9.267834e+03 | 8.067575e+02 | 6.944975e+02 |

Table 21
Results of CSS instances with average combination and $n_{ev} = 50{,}000$

| Functions | A | A-S 10 | A-S 50 | A-S 100 | A-SW 10 | A-SW 50 | A-SW 100 |
|---|---|---|---|---|---|---|---|
| $f_1$ | 11074(68%) | 15558(66%) | 59(100%) | 65(100%) | 3032(100%) | 232(100%) | 106(100%) |
| $f_2$ | 27678(26%) | 9910(26%) | 8358(94%) | 108(100%) | 5925(100%) | 8003(100%) | 872(100%) |
| $f_3$ | 17411(82%) | 5127(86%) | 4028(100%) | 3615(100%) | 4546(100%) | 11945(100%) | 18007(100%) |
| $f_4$ | 25074(84%) | 5323(40%) | 7885(100%) | 190(100%) | 5350(100%) | 8555(100%) | 4583(100%) |
| $f_5$ | 18704(38%) | 6725(32%) | 485(100%) | 762(100%) | 4377(100%) | 2051(100%) | 2748(100%) |
| $f_6$ | 396(22%) | 7398(72%) | 391(100%) | 132(100%) | 4093(100%) | 6198(100%) | 3975(100%) |
| $f_7$ | 11013(94%) | 5731(78%) | 325(100%) | 62(100%) | 4789(100%) | 3926(100%) | 138(100%) |
| $f_8$ | 10592(100%) | 7773(90%) | 94(100%) | 54(100%) | 2921(100%) | 960(100%) | 165(100%) |
| $f_9$ | 12581(94%) | 7816(96%) | 57(100%) | 54(100%) | 2650(100%) | 224(100%) | 147(100%) |
| $f_{10}$ | 4.568131e+01 | 8.278274e+00 | 7.106348e+00 | 4.737560e+00 | 9.972440e+01 | 1029(100%) | 1963(100%) |
| $f_{11}$ | 28250(4%) | 5545(10%) | 5420(100%) | 292(100%) | 5746(94%) | 13948(100%) | 23847(100%) |
| $f_{12}$ | 18264(70%) | 14488(64%) | 176(100%) | 59(100%) | 3709(100%) | 1068(100%) | 118(100%) |
| $f_{13}$ | 3.211981e+03 | 20008(18%) | 14698(84%) | 8219(100%) | 13580(94%) | 9039(100%) | 13912(100%) |
| $f_{14}$ | 1.566265e−03 | 3.772719e−05 | 3.772719e−05 | 3.772719e−05 | 5(100%) | 5(100%) | 6(100%) |
| $f_{15}$ | 3.896128e+00 | 1.091291e+00 | 24235(10%) | 23421(100%) | 10610(4%) | 24729(86%) | 39267(82%) |
| $f_{16}$ | −5.676348e+00 | 17018(14%) | 11114(90%) | 1197(100%) | 9452(96%) | 8316(100%) | 568(100%) |
| $f_{17}$ | −5.766817e+00 | 5672(12%) | 11460(84%) | 1837(100%) | 8856(98%) | 8384(100%) | 637(100%) |
| $f_{18}$ | −5.516780e+00 | 9405(10%) | 12050(86%) | 2264(100%) | 8998(98%) | 8693(100%) | 762(100%) |
| $f_{19}$ | 3.130638e+01 | 2.652995e−01 | 2.417995e−03 | 35618(30%) | 1.477766e−02 | 45827(2%) | 2.971024e−02 |
| $f_{20}$ | 3.639802e+00 | 1.280329e−01 | 1.260188e−03 | 29048(64%) | 10667(2%) | 32129(22%) | 43945(10%) |
| $f_{21}$ | 1.139009e−01 | 1.013293e−01 | 22164(4%) | 25468(56%) | 8.562680e−04 | 35474(16%) | 43570(4%) |
| $f_{22}$ | 35789(4%) | 1479(100%) | 401(100%) | 394(100%) | 5(100%) | 5(100%) | 5(100%) |
| $f_{23}$ | 7.557223e+02 | 6.749472e+02 | 5.111435e+02 | 4.592233e+02 | 8.057628e+02 | 14743(88%) | 20022(96%) |
| $f_{24}$ | −3.525688e+01 | 26514(4%) | 14790(82%) | 10951(100%) | 7642(100%) | 10073(100%) | 14903(100%) |
| $f_{25}$ | −5.845191e+01 | −1.762982e+02 | −2.095566e+02 | 27712(96%) | 48662(12%) | 28915(50%) | 28502(74%) |
| $f_{26}$ | 3.196390e+01 | 1.751431e+01 | 9.671264e+00 | 7.820723e+00 | 6.639897e+00 | 46451(20%) | 9.026495e−01 |
| $f_{27}$ | 7.847097e+00 | 2.782024e+00 | 7.564416e−01 | 3.002372e−01 | 3.303316e+00 | 42081(8%) | 43180(80%) |
| $f_{28}$ | 3.708076e+01 | 9.596960e+00 | 1.770890e−01 | 2.362650e−03 | 3.993481e−03 | 25327(100%) | 30638(100%) |
| $f_{29}$ | 9.988137e+00 | 3.044578e+01 | 1.752342e+01 | 6.801684e+00 | 7.748381e+00 | 5.007183e+00 | 4.329500e+00 |
| $f_{30}$ | 5.327165e+02 | 9.945566e+01 | 1.523272e+00 | 4.872865e−03 | 1.141444e−02 | 32962(96%) | 39359(92%) |
| $f_{31}$ | 1.086446e+02 | 5.410379e+01 | 3.867251e+01 | 3.923264e+01 | 2.084147e+01 | 1.558646e+01 | 1.336928e+01 |
| $f_{32}$ | 3.297391e+01 | 2.797297e+01 | 5.685273e+00 | 2.252134e+00 | 1.994313e+00 | 49800(2%) | 7.518748e−04 |
| $f_{33}$ | 3.380801e+02 | 2.069725e+02 | 8.038028e+01 | 3.585555e+01 | 2.462571e−01 | 48482(6%) | 2.409492e−03 |
| $f_{34}$ | 2.831090e+01 | 2.330962e+02 | 5.982933e+01 | 3.624471e+01 | 2.258575e+01 | 1.679650e+01 | 1.656665e+01 |
| $f_{35}$ | 9.277589e+04 | 2.067851e+04 | 4.896130e+03 | 7.115791e+02 | 4.211476e+00 | 8.521118e−01 | 8.369693e−02 |
| $f_{36}$ | 7.004126e+00 | 2.966574e+01 | 8.365118e+00 | 2.673407e+00 | 1.436270e+00 | 2.124654e−01 | 1.371583e−01 |
| $f_{37}$ | 9.326137e+00 | 2.007701e+02 | 3.765279e+01 | 8.572660e+00 | 5.212566e+00 | 1.091511e+00 | 3.267276e−01 |
| $f_{38}$ | 30305(20%) | 22372(24%) | 19490(100%) | 26091(100%) | 7539(100%) | 26031(98%) | 44803(80%) |
| $f_{39}$ | 1.994750e+01 | 2.030755e+01 | 1.115737e+01 | 7.118457e+00 | 1.383008e−02 | 47155(100%) | 49325(26%) |
| $f_{40}$ | 1.380028e+01 | 1.400962e+01 | 1.270217e+01 | 1.163112e+01 | 4.079616e−01 | 4.995921e+00 | 1.149521e+01 |
| $f_{41}$ | 7.205580e−01 | 3.367512e+00 | 6.604856e−01 | 1.265742e−01 | 8.525758e−03 | 41006(100%) | 47063(100%) |
| $f_{42}$ | 2.866305e+01 | 1.442163e+02 | 7.487326e+01 | 3.912127e+01 | 2.558928e+01 | 2.228399e+01 | 2.178021e+01 |
| $f_{43}$ | 9.877439e+03 | 9.902106e+03 | 9.870922e+03 | 9.861414e+03 | 9.884616e+03 | 8.309106e+03 | 8.025182e+03 |
| $f_{44}$ | 3.134398e+02 | 1.268564e+02 | 1.102564e+02 | 1.247502e+02 | 1.295620e+02 | 8.582814e+01 | 1.105471e+02 |
| $f_{45}$ | 5.073042e+02 | 4.886622e+02 | 3.838905e+02 | 2.838981e+02 | 2.968400e+02 | 2.071416e−03 | 5.794679e−03 |
| $f_{46}$ | 4.828107e+01 | 2.202468e+02 | 6.813753e+01 | 1.656185e+01 | 1.531308e+02 | 3.673621e+01 | 4.378755e+01 |
| $f_{47}$ | 2.419638e+01 | 2.139149e+01 | 1.453266e+01 | 1.213491e+01 | 1.812524e+01 | 1.246198e+01 | 1.323554e+01 |
| $f_{48}$ | 1.284203e+03 | 1.292494e+04 | 3.071011e+03 | 8.747302e+02 | 4.123514e+03 | 5.439471e+02 | 3.923095e+02 |

Table 22
Results of CSS instances with average combination and $n_{\text{ev}} = 100{,}000$

| Functions | A | A-S 10 | A-S 50 | A-S 100 | A-SW 10 | A-SW 50 | A-SW 100 |
|---|---|---|---|---|---|---|---|
| $f_1$ | 21435(90%) | 22313(76%) | 59(100%) | 65(100%) | 3032(100%) | 232(100%) | 106(100%) |
| $f_2$ | 55228(52%) | 27893(50%) | 8375(96%) | 108(100%) | 5925(100%) | 8003(100%) | 872(100%) |
| $f_3$ | 25226(100%) | 9482(84%) | 4028(100%) | 3615(100%) | 4546(100%) | 11945(100%) | 18007(100%) |
| $f_4$ | 32103(100%) | 12772(22%) | 7885(100%) | 190(100%) | 5350(100%) | 8555(100%) | 4583(100%) |
| $f_5$ | 42607(56%) | 26982(30%) | 485(100%) | 762(100%) | 4377(100%) | 2051(100%) | 2748(100%) |
| $f_6$ | 9091(18%) | 10451(84%) | 391(100%) | 132(100%) | 4093(100%) | 6198(100%) | 3975(100%) |
| $f_7$ | 13223(100%) | 11148(86%) | 325(100%) | 62(100%) | 4789(100%) | 3926(100%) | 138(100%) |
| $f_8$ | 10592(100%) | 13303(98%) | 94(100%) | 54(100%) | 2921(100%) | 960(100%) | 165(100%) |
| $f_9$ | 16867(100%) | 10146(100%) | 57(100%) | 54(100%) | 2650(100%) | 224(100%) | 147(100%) |
| $f_{10}$ | 3.552374e+01 | 7.994367e+00 | 6.765843e−05 | 2.545514e−05 | 8.658698e+01 | 1029(100%) | 1963(100%) |
| $f_{11}$ | 42323(12%) | 8403(16%) | 5420(100%) | 292(100%) | 6575(100%) | 13948(100%) | 23847(100%) |
| $f_{12}$ | 28869(100%) | 21951(82%) | 176(100%) | 59(100%) | 3709(100%) | 1068(100%) | 118(100%) |
| $f_{13}$ | 2.878806e+03 | 27479(28%) | 13868(86%) | 8219(100%) | 18574(98%) | 9039(100%) | 13912(100%) |
| $f_{14}$ | 5.548391e−04 | 3.772719e−05 | 3.772719e−05 | 3.772719e−05 | 5(100%) | 5(100%) | 6(100%) |
| $f_{15}$ | 2.561524e+00 | 8.165845e−01 | 24285(10%) | 23421(100%) | 10592(2%) | 25922(94%) | 37896(94%) |
| $f_{16}$ | −6.165482e+00 | 17526(16%) | 11181(88%) | 1197(100%) | 10808(100%) | 8316(100%) | 568(100%) |
| $f_{17}$ | −5.497251e+00 | 13771(12%) | 11608(86%) | 1837(100%) | 8823(98%) | 8384(100%) | 637(100%) |
| $f_{18}$ | −5.750058e+00 | 20133(8%) | 17360(90%) | 2264(100%) | 11180(98%) | 8693(100%) | 762(100%) |
| $f_{19}$ | 1.889669e+01 | 1.450726e−01 | 2.567199e−03 | 35016(28%) | 6.458802e−03 | 3.638562e−03 | 88249(4%) |
| $f_{20}$ | 2.183872e+00 | 7.816190e−02 | 1.332706e−03 | 29063(66%) | 1.563970e−03 | 35146(20%) | 51920(32%) |
| $f_{21}$ | 5.863955e−02 | 6.234422e−02 | 22667(6%) | 25537(56%) | 3.805371e−04 | 38283(12%) | 75774(16%) |
| $f_{22}$ | 44382(6%) | 1479(100%) | 401(100%) | 394(100%) | 5(100%) | 5(100%) | 5(100%) |
| $f_{23}$ | 6.837380e+02 | 6.305916e+02 | 4.641912e+02 | 4.505078e+02 | 7.944516e+02 | 20663(98%) | 16891(98%) |
| $f_{24}$ | −3.789352e+01 | 15571(4%) | 18635(90%) | 10951(100%) | 7642(100%) | 10073(100%) | 14903(100%) |
| $f_{25}$ | −6.865895e+01 | −1.771048e+02 | −2.095764e+02 | 27796(96%) | 58316(100%) | 49406(100%) | 37957(100%) |
| $f_{26}$ | 3.040757e+01 | 1.685428e+01 | 9.844015e+00 | 7.355569e+00 | 5.875442e+00 | 53085(46%) | 68289(92%) |
| $f_{27}$ | 7.415561e+00 | 2.486820e+00 | 7.322774e−01 | 2.331749e−01 | 2.209664e+00 | 41519(12%) | 45209(94%) |
| $f_{28}$ | 3.321437e+01 | 9.745727e+00 | 1.454554e−01 | 2.021885e−03 | 1.927107e−03 | 25327(100%) | 30638(100%) |
| $f_{29}$ | 9.012775e+00 | 2.892736e+01 | 1.609748e+01 | 6.350123e+00 | 6.793416e+00 | 4.044014e+00 | 65582(6%) |
| $f_{30}$ | 5.284858e+02 | 8.626617e+01 | 1.627879e+00 | 4.572595e−03 | 5.901610e−03 | 34260(100%) | 41894(100%) |
| $f_{31}$ | 1.049695e+02 | 5.604364e+01 | 3.817690e+01 | 3.270031e+01 | 1.953302e+01 | 1.332921e+01 | 5.981080e+00 |
| $f_{32}$ | 3.354334e+01 | 2.590402e+01 | 5.076167e+00 | 1.550312e+00 | 2.268281e−02 | 55546(76%) | 62782(94%) |
| $f_{33}$ | 3.059898e+02 | 2.153750e+02 | 8.799361e+01 | 2.658387e+01 | 1.272582e−01 | 71408(42%) | 67979(80%) |
| $f_{34}$ | 2.067580e+01 | 2.040045e+02 | 5.555925e+01 | 3.152109e+01 | 1.980073e+01 | 1.645973e+01 | 1.624744e+01 |
| $f_{35}$ | 7.972229e+04 | 1.934855e+04 | 4.965156e+03 | 7.665297e+02 | 1.673975e+00 | 9.801505e−02 | 96431(2%) |
| $f_{36}$ | 3.772769e+00 | 1.952026e+01 | 6.198155e+00 | 2.603502e+00 | 7.761122e−01 | 6.393214e−02 | 4.751352e−02 |
| $f_{37}$ | 2.748572e+00 | 1.245165e+02 | 4.047646e+01 | 7.000093e+00 | 3.488934e+00 | 3.435861e−01 | 5.681354e−02 |
| $f_{38}$ | 51011(76%) | 44821(26%) | 19490(100%) | 26091(100%) | 7539(100%) | 28052(100%) | 45964(100%) |
| $f_{39}$ | 1.894339e+01 | 1.945148e+01 | 1.148252e+01 | 3.782681e+00 | 9.492146e−03 | 46784(100%) | 52915(100%) |
| $f_{40}$ | 1.355881e+01 | 1.365634e+01 | 1.218741e+01 | 1.108025e+01 | 4.591285e−01 | 1.472246e−03 | 3.365022e+00 |
| $f_{41}$ | 3.619886e−01 | 2.972191e+00 | 5.608871e−01 | 1.362783e−01 | 4.873395e−03 | 41006(100%) | 47381(100%) |
| $f_{42}$ | 2.444540e+01 | 1.399445e+02 | 6.276827e+01 | 4.281121e+01 | 2.389321e+01 | 2.176554e+01 | 2.123099e+01 |
| $f_{43}$ | 9.873876e+03 | 9.894939e+03 | 9.869283e+03 | 9.860955e+03 | 9.890213e+03 | 8.226050e+03 | 7.719751e+03 |
| $f_{44}$ | 3.207480e+02 | 7.687706e+01 | 6.755800e+01 | 7.753551e+01 | 1.269375e+02 | 7.949769e+01 | 5.348926e+01 |
| $f_{45}$ | 5.145244e+02 | 4.942640e+02 | 3.922601e+02 | 1.887260e+02 | 1.266021e+02 | 73122(84%) | 79261(90%) |
| $f_{46}$ | 2.611656e+01 | 1.691584e+02 | 6.662800e+01 | 1.681278e+01 | 1.121659e+02 | 3.563965e+01 | 2.559633e+01 |
| $f_{47}$ | 2.328121e+01 | 2.082928e+01 | 1.269110e+01 | 66474(2%) | 1.747625e+01 | 83401(8%) | 97483(2%) |
| $f_{48}$ | 5.369802e+02 | 9.190404e+03 | 2.279762e+03 | 6.669642e+02 | 1.167642e+03 | 4.737414e+02 | 3.627938e+02 |

Table 23
Results of CSS instances with average combination and $n_{\mathrm{ev}} = 500{,}000$

| Functions | A | A-S 10 | A-S 50 | A-S 100 | A-SW 10 | A-SW 50 | A-SW 100 |
|---|---|---|---|---|---|---|---|
| $f_1$ | 39020(100%) | 57963(100%) | 59(100%) | 65(100%) | 3032(100%) | 232(100%) | 106(100%) |
| $f_2$ | 98519(100%) | 141040(94%) | 12389(100%) | 108(100%) | 5925(100%) | 8003(100%) | 872(100%) |
| $f_3$ | 25226(100%) | 37351(100%) | 4028(100%) | 3615(100%) | 4546(100%) | 11945(100%) | 18007(100%) |
| $f_4$ | 32103(100%) | 129590(42%) | 7885(100%) | 190(100%) | 5350(100%) | 8555(100%) | 4583(100%) |
| $f_5$ | 93711(100%) | 203471(72%) | 485(100%) | 762(100%) | 4377(100%) | 2051(100%) | 2748(100%) |
| $f_6$ | 38027(22%) | 94811(100%) | 391(100%) | 132(100%) | 4093(100%) | 6198(100%) | 3975(100%) |
| $f_7$ | 13223(100%) | 63854(100%) | 325(100%) | 62(100%) | 4789(100%) | 3926(100%) | 138(100%) |
| $f_8$ | 10592(100%) | 15492(100%) | 94(100%) | 54(100%) | 2921(100%) | 960(100%) | 165(100%) |
| $f_9$ | 16867(100%) | 10146(100%) | 57(100%) | 54(100%) | 2650(100%) | 224(100%) | 147(100%) |
| $f_{10}$ | 2.026702e+01 | 7.452133e+00 | 8.045320e−05 | 2.545513e−05 | 3.445431e+01 | 1029(100%) | 1963(100%) |
| $f_{11}$ | 172442(44%) | 215241(42%) | 5420(100%) | 292(100%) | 6575(100%) | 13948(100%) | 23847(100%) |
| $f_{12}$ | 28869(100%) | 38814(100%) | 176(100%) | 59(100%) | 3709(100%) | 1068(100%) | 118(100%) |
| $f_{13}$ | 3.105499e+03 | 204405(88%) | 37396(98%) | 8240(100%) | 24053(100%) | 9039(100%) | 13912(100%) |
| $f_{14}$ | 4.120241e−04 | 3.772719e−05 | 3.772719e−05 | 3.772719e−05 | 5(100%) | 5(100%) | 6(100%) |
| $f_{15}$ | 1.015280e+00 | 2.069866e−01 | 64765(16%) | 23421(100%) | 243034(12%) | 54838(100%) | 81442(98%) |
| $f_{16}$ | −7.117034e+00 | 266591(52%) | 36539(100%) | 1197(100%) | 10808(100%) | 8316(100%) | 568(100%) |
| $f_{17}$ | 21395(4%) | 237569(42%) | 35922(100%) | 1837(100%) | 11027(100%) | 8384(100%) | 637(100%) |
| $f_{18}$ | −7.366983e+00 | 194067(32%) | 34130(98%) | 2264(100%) | 11490(100%) | 8693(100%) | 762(100%) |
| $f_{19}$ | 8.644641e+00 | 3.632883e−02 | 21106(2%) | 125481(54%) | 4.000520e−03 | 3.439866e−03 | 92017(4%) |
| $f_{20}$ | 3.367218e−01 | 2.030285e−02 | 23267(2%) | 60758(84%) | 5.632660e−04 | 41513(26%) | 76485(26%) |
| $f_{21}$ | 5.371200e−03 | 1.474589e−02 | 23321(2%) | 160890(94%) | 161355(4%) | 62426(6%) | 89931(16%) |
| $f_{22}$ | 214626(18%) | 1479(100%) | 401(100%) | 462(100%) | 5(100%) | 5(100%) | 5(100%) |
| $f_{23}$ | 5.240975e+02 | 4.820599e+02 | 3.728691e+02 | 3.838424e+02 | 6.069172e+02 | 28846(100%) | 18858(100%) |
| $f_{24}$ | −3.974014e+01 | 105151(10%) | 38258(100%) | 10951(100%) | 7642(100%) | 10073(100%) | 14903(100%) |
| $f_{25}$ | −8.672579e+01 | −1.727134e+02 | −2.096286e+02 | 34730(100%) | 58316(100%) | 49406(100%) | 37957(100%) |
| $f_{26}$ | 2.193829e+01 | 1.477353e+01 | 8.434255e+00 | 6.441310e+00 | 3.983499e+00 | 138259(92%) | 95044(100%) |
| $f_{27}$ | 6.494275e+00 | 2.574371e+00 | 6.524438e−01 | 1.398736e−01 | 8.563326e−01 | 291103(66%) | 68394(100%) |
| $f_{28}$ | 2.787670e+01 | 9.041870e+00 | 1.166383e−01 | 7.479607e−04 | 1.404143e−04 | 25327(100%) | 30638(100%) |
| $f_{29}$ | 7.892050e+00 | 1.987869e+01 | 1.098607e+01 | 4.367944e+00 | 5.073874e+00 | 5.710367e−01 | 67637(4%) |
| $f_{30}$ | 3.640012e+02 | 7.396698e+01 | 1.233121e+00 | 2.601381e−03 | 1.089015e−03 | 34260(100%) | 41894(100%) |
| $f_{31}$ | 8.433245e+01 | 5.327865e+01 | 3.399461e+01 | 2.913180e+01 | 1.514659e+01 | 1.054794e+01 | 305340(4%) |
| $f_{32}$ | 2.837300e+01 | 3.118862e+01 | 4.799978e+00 | 1.483711e+00 | 2.260587e−02 | 89477(82%) | 73504(100%) |
| $f_{33}$ | 2.632084e+02 | 1.877932e+02 | 8.889259e+01 | 2.335212e+01 | 3.560631e−02 | 135287(100%) | 78992(100%) |
| $f_{34}$ | 1.831513e+01 | 1.435899e+02 | 4.245554e+01 | 2.703436e+01 | 1.802527e+01 | 1.257766e+01 | 1.190176e+01 |
| $f_{35}$ | 4.779434e+04 | 1.790616e+04 | 3.833737e+03 | 6.915808e+02 | 4.780645e−01 | 432944(12%) | 273035(100%) |
| $f_{36}$ | 1.112771e+00 | 1.261270e+01 | 4.420976e+00 | 1.468090e+00 | 1.271519e−01 | 4.356530e−03 | 2.682036e−03 |
| $f_{37}$ | 5.852255e−01 | 2.841096e+01 | 1.090247e+01 | 2.980476e+00 | 8.092004e−01 | 3.199151e−02 | 150005(10%) |
| $f_{38}$ | 109962(100%) | 132754(72%) | 19490(100%) | 26012(100%) | 7539(100%) | 28052(100%) | 47761(100%) |
| $f_{39}$ | 1.588427e+01 | 2.023542e+01 | 1.133478e+01 | 2.087626e+00 | 3.740352e−03 | 46784(100%) | 52915(100%) |
| $f_{40}$ | 1.338863e+01 | 1.345532e+01 | 1.197801e+01 | 1.114582e+01 | 1.819238e−01 | 8.530485e−06 | 200147(100%) |
| $f_{41}$ | 6.187255e−02 | 1.976001e+00 | 4.727145e−01 | 2.163397e−01 | 8.060558e−04 | 41006(100%) | 47381(100%) |
| $f_{42}$ | 2.343620e+01 | 9.885612e+01 | 5.489042e+01 | 3.571926e+01 | 2.365104e+01 | 1.848068e+01 | 1.760120e+01 |
| $f_{43}$ | 9.872505e+03 | 9.885524e+03 | 9.862976e+03 | 9.850451e+03 | 9.885467e+03 | 8.001402e+03 | 7.619732e+03 |
| $f_{44}$ | 2.951759e+02 | 5.171998e+00 | 5.570075e+00 | 2.785676e+00 | 1.158147e+02 | 6.385549e+01 | 2.623548e+01 |
| $f_{45}$ | 4.944554e+02 | 5.070043e+02 | 3.926773e+02 | 8.654872e+00 | 7.932752e−03 | 108138(94%) | 91500(96%) |
| $f_{46}$ | 1.006135e+01 | 5.484565e+01 | 4.641350e+01 | 1.302818e+01 | 3.356234e+01 | 1.861917e+01 | 1.626374e+01 |
| $f_{47}$ | 2.171988e+01 | 1.889923e+01 | 1.559996e+01 | 1.414010e+01 | 1.192708e+01 | 198470(8%) | 206512(2%) |
| $f_{48}$ | 1.975890e+02 | 2.683699e+03 | 1.130521e+03 | 4.313629e+02 | 3.174916e+02 | 3.544777e+02 | 2.462716e+02 |

Table 24
Results of CSS instances with BLX-α and $n_{ev} = 25{,}000$

| Functions | B | B-S 10 | B-S 50 | B-S 100 | B-SW 10 | B-SW 50 | B-SW 100 |
|---|---|---|---|---|---|---|---|
| $f_1$ | 2293(98%) | 4473(82%) | 59(100%) | 65(100%) | 3097(100%) | 232(100%) | 106(100%) |
| $f_2$ | 1445(100%) | 7085(90%) | 10317(100%) | 108(100%) | 7013(98%) | 8017(100%) | 872(100%) |
| $f_3$ | 1915(100%) | 5637(96%) | 4062(100%) | 2688(100%) | 4928(100%) | 12455(100%) | 15826(94%) |
| $f_4$ | 4597(94%) | 8018(58%) | 9217(100%) | 190(100%) | 6403(98%) | 8208(100%) | 4583(100%) |
| $f_5$ | 1820(100%) | 5448(70%) | 485(100%) | 762(100%) | 4918(100%) | 2051(100%) | 2748(100%) |
| $f_6$ | 5491(80%) | 5782(68%) | 391(100%) | 132(100%) | 6295(100%) | 7060(100%) | 3975(100%) |
| $f_7$ | 3644(94%) | 5653(74%) | 325(100%) | 62(100%) | 5063(98%) | 4205(100%) | 138(100%) |
| $f_8$ | 5024(90%) | 4535(80%) | 94(100%) | 54(100%) | 3427(100%) | 960(100%) | 165(100%) |
| $f_9$ | 718(100%) | 3437(98%) | 57(100%) | 54(100%) | 2777(100%) | 224(100%) | 147(100%) |
| $f_{10}$ | 2.545513e−05 | 4.777974e+00 | 9.475095e+00 | 4.737560e+00 | 4.737561e+00 | 1029(100%) | 1963(100%) |
| $f_{11}$ | 17101(16%) | 1.622358e−02 | 5008(100%) | 292(100%) | 12465(60%) | 16511(100%) | 18742(16%) |
| $f_{12}$ | 1285(100%) | 4800(98%) | 176(100%) | 59(100%) | 4184(100%) | 1068(100%) | 118(100%) |
| $f_{13}$ | 6685(96%) | 9802(80%) | 17738(96%) | 8402(100%) | 10014(96%) | 9430(100%) | 14874(100%) |
| $f_{14}$ | 3.786791e−05 | 3.772719e−05 | 3.772719e−05 | 3.772719e−05 | 5(100%) | 5(100%) | 6(100%) |
| $f_{15}$ | 2.457089e+00 | 6.086349e−01 | 3.280560e−03 | 24802(4%) | 2.187521e−02 | 2.233325e−03 | 17360(2%) |
| $f_{16}$ | 6680(34%) | 8432(80%) | 15379(100%) | 1197(100%) | 7677(82%) | 8006(100%) | 568(100%) |
| $f_{17}$ | 8278(52%) | 8660(76%) | 15278(100%) | 1837(100%) | 8332(96%) | 8557(100%) | 637(100%) |
| $f_{18}$ | 8606(56%) | 8453(78%) | 15035(100%) | 2264(100%) | 7936(98%) | 8561(100%) | 762(100%) |
| $f_{19}$ | 3.342710e+00 | 1.046414e−01 | 3.672657e−03 | 9.296700e−04 | 4.336613e−02 | 6.874222e−02 | 9.100675e−01 |
| $f_{20}$ | 1.469844e−01 | 5.421720e−02 | 2.450285e−03 | 1.371945e−05 | 3.484702e−03 | 24330(2%) | 19428(2%) |
| $f_{21}$ | 7.300102e−02 | 3.258660e−02 | 9.595756e−04 | 3.812891e−05 | 9.216793e−03 | 2.738429e−03 | 6.343321e−03 |
| $f_{22}$ | 1595(92%) | 1565(100%) | 401(100%) | 394(100%) | 5(100%) | 5(100%) | 5(100%) |
| $f_{23}$ | 2.346718e+02 | 2.729822e+02 | 1.624436e+02 | 2.165763e+02 | 3.983645e+02 | 11977(98%) | 13312(80%) |
| $f_{24}$ | 1576(4%) | 14642(16%) | 20138(82%) | 11183(100%) | 8641(100%) | 11031(100%) | 14038(100%) |
| $f_{25}$ | −1.225651e+02 | −1.985907e+02 | −2.094358e+02 | −2.097701e+02 | 18919(6%) | 21608(32%) | 24619(8%) |
| $f_{26}$ | 1.161684e+01 | 8.662292e+00 | 5.849073e+00 | 6.666818e+00 | 8.942635e+00 | 4.535482e+00 | 1.264309e+01 |
| $f_{27}$ | 2.456546e−01 | 3.385878e−01 | 7.080620e−01 | 5.284450e−01 | 2.891846e−01 | 8.481270e−02 | 6.673060e−02 |
| $f_{28}$ | 18059(8%) | 3.565672e−02 | 3.663151e−02 | 1.272141e−02 | 1.059867e−04 | 7.645537e−06 | 7.910136e−05 |
| $f_{29}$ | 5.963454e+01 | 1.645030e+01 | 1.175532e+01 | 7.453159e+00 | 8.413052e+00 | 5.064371e+00 | 5.078268e+00 |
| $f_{30}$ | 16887(4%) | 1.477080e−01 | 6.581604e−02 | 2.951610e−02 | 2.724241e−04 | 3.355618e−04 | 9.882236e−03 |
| $f_{31}$ | 3.246125e+01 | 6.220850e+01 | 5.863913e+01 | 4.664674e+01 | 2.743526e+01 | 3.819670e+01 | 4.887434e+01 |
| $f_{32}$ | 1.193513e+00 | 1.995909e+00 | 2.356734e+00 | 3.637511e+00 | 4.608484e−01 | 5.756052e−01 | 8.382299e−01 |
| $f_{33}$ | 5.504380e+00 | 7.054253e+00 | 9.862809e+00 | 1.527274e+01 | 2.202784e−02 | 9.490364e−02 | 6.621656e−01 |
| $f_{34}$ | 5.340634e+02 | 2.067080e+02 | 7.385312e+01 | 6.501161e+01 | 2.449958e+01 | 2.181538e+01 | 4.577701e+01 |
| $f_{35}$ | 3.723582e+03 | 3.643960e+02 | 1.356075e+02 | 3.359131e+02 | 1.346948e+00 | 4.400102e+00 | 4.148140e+01 |
| $f_{36}$ | 3.846496e+01 | 2.207645e+01 | 1.194325e+01 | 1.702330e+01 | 1.999348e+00 | 3.540454e+00 | 1.168585e+01 |
| $f_{37}$ | 2.962667e+02 | 2.493520e+02 | 1.454640e+02 | 1.648329e+02 | 8.267463e+00 | 2.737355e+01 | 7.555184e+01 |
| $f_{38}$ | 21344(40%) | 6.486400e+01 | 7.334841e+02 | 2.860002e+03 | 13918(100%) | 1.888061e+03 | 1.208994e+04 |
| $f_{39}$ | 6.693894e−01 | 3.839847e+00 | 3.764758e+00 | 8.590582e+00 | 1.300332e−02 | 2.770096e−02 | 6.884185e−02 |
| $f_{40}$ | 4.623773e+00 | 8.336942e+00 | 1.020364e+01 | 1.511696e+01 | 1.489722e+00 | 1.435313e+01 | 1.723461e+01 |
| $f_{41}$ | 2.155491e−01 | 9.835393e−01 | 2.620984e−01 | 1.354927e−01 | 4.283268e−03 | 6.109269e−03 | 1.320093e−02 |
| $f_{42}$ | 2.444142e+02 | 2.022785e+02 | 1.124055e+02 | 1.070844e+02 | 2.573467e+01 | 3.245817e+01 | 5.922239e+01 |
| $f_{43}$ | 9.889227e+03 | 9.882613e+03 | 9.875219e+03 | 9.908319e+03 | 9.884380e+03 | 8.927113e+03 | 8.700666e+03 |
| $f_{44}$ | 7.613853e+01 | 1.374972e+02 | 2.097282e+02 | 2.430092e+02 | 5.891044e+01 | 1.316606e+02 | 2.188517e+02 |
| $f_{45}$ | 7.862178e+00 | 5.797251e+01 | 1.544389e+02 | 2.110298e+02 | 9.784449e+00 | 1.588714e+00 | 1.466347e+00 |
| $f_{46}$ | 4.975716e+02 | 1.288176e+02 | 8.816791e+01 | 4.740309e+01 | 1.770189e+02 | 9.253211e+01 | 2.045960e+02 |
| $f_{47}$ | 2.072306e+01 | 1.573367e+01 | 8.779191e+00 | 1.175357e+01 | 1.151704e+01 | 1.174643e+01 | 1.681185e+01 |
| $f_{48}$ | 2.948807e+04 | 7.217604e+03 | 6.928449e+03 | 9.044918e+03 | 1.480564e+04 | 1.517147e+03 | 1.742993e+03 |

Table 25
Results of CSS instances with BLX-$\alpha$ and $n_{ev} = 50{,}000$

| Functions | B | B-S 10 | B-S 50 | B-S 100 | B-SW 10 | B-SW 50 | B-SW 100 |
|---|---|---|---|---|---|---|---|
| $f_1$ | 2547(98%) | 4434(84%) | 59(100%) | 65(100%) | 3097(100%) | 232(100%) | 106(100%) |
| $f_2$ | 1445(100%) | 7944(98%) | 10317(100%) | 108(100%) | 7057(98%) | 8017(100%) | 872(100%) |
| $f_3$ | 1915(100%) | 6554(94%) | 4062(100%) | 2688(100%) | 4928(100%) | 12455(100%) | 18729(100%) |
| $f_4$ | 6019(100%) | 12336(70%) | 9217(100%) | 190(100%) | 6575(98%) | 8208(100%) | 4583(100%) |
| $f_5$ | 1820(100%) | 11297(74%) | 485(100%) | 762(100%) | 4918(100%) | 2051(100%) | 2748(100%) |
| $f_6$ | 8567(90%) | 7064(58%) | 391(100%) | 132(100%) | 6295(100%) | 7060(100%) | 3975(100%) |
| $f_7$ | 6671(90%) | 12486(80%) | 325(100%) | 62(100%) | 5852(100%) | 4205(100%) | 138(100%) |
| $f_8$ | 9489(92%) | 6989(86%) | 94(100%) | 54(100%) | 3427(100%) | 960(100%) | 165(100%) |
| $f_9$ | 718(100%) | 5330(94%) | 57(100%) | 54(100%) | 2777(100%) | 224(100%) | 147(100%) |
| $f_{10}$ | 2.545513e−05 | 2.430780e+00 | 2.547190e−05 | 7.106328e+00 | 2.545518e−05 | 1029(100%) | 1963(100%) |
| $f_{11}$ | 32029(16%) | 15134(2%) | 5008(100%) | 292(100%) | 17927(88%) | 16438(100%) | 28147(100%) |
| $f_{12}$ | 1285(100%) | 4795(94%) | 176(100%) | 59(100%) | 4184(100%) | 1068(100%) | 118(100%) |
| $f_{13}$ | 8568(100%) | 9575(72%) | 17916(98%) | 8402(100%) | 10441(98%) | 9430(100%) | 14874(100%) |
| $f_{14}$ | 3.772719e−05 | 3.772719e−05 | 3.772719e−05 | 3.772719e−05 | 5(100%) | 5(100%) | 6(100%) |
| $f_{15}$ | 2.212682e+00 | 3.652419e−01 | 2.428830e−03 | 41440(44%) | 21620(2%) | 38440(96%) | 46692(42%) |
| $f_{16}$ | 10240(50%) | 9020(72%) | 15379(100%) | 1197(100%) | 8356(90%) | 8006(100%) | 568(100%) |
| $f_{17}$ | 11916(62%) | 8476(78%) | 15278(100%) | 1837(100%) | 9369(100%) | 8557(100%) | 637(100%) |
| $f_{18}$ | 15142(54%) | 9024(88%) | 15035(100%) | 2264(100%) | 7936(98%) | 8561(100%) | 762(100%) |
| $f_{19}$ | 1.338190e+00 | 1.503776e−01 | 3.176410e−03 | 32673(14%) | 2.883842e−02 | 1.310077e−02 | 8.314696e−02 |
| $f_{20}$ | 8.088661e−02 | 4.765600e−02 | 1.585009e−03 | 46381(22%) | 41584(2%) | 34124(12%) | 43611(4%) |
| $f_{21}$ | 2.357696e−02 | 2.258411e−02 | 5.764134e−04 | 34090(40%) | 3.298201e−03 | 49540(2%) | 1.084150e−03 |
| $f_{22}$ | 2115(98%) | 1565(100%) | 401(100%) | 394(100%) | 5(100%) | 5(100%) | 5(100%) |
| $f_{23}$ | 1.384582e+02 | 2.373684e+02 | 1.417453e+02 | 1.255596e+02 | 4.613096e+02 | 13071(100%) | 17571(100%) |
| $f_{24}$ | 22870(10%) | 14067(24%) | 22185(98%) | 11183(100%) | 8641(100%) | 11031(100%) | 14038(100%) |
| $f_{25}$ | −1.602944e+02 | −2.001218e+02 | 41642(18%) | 39534(82%) | 40809(80%) | 28127(86%) | 31600(84%) |
| $f_{26}$ | 7.997657e+00 | 9.794059e+00 | 3.044512e+00 | 1.477578e+00 | 8.281822e+00 | 8.985103e−01 | 2.278524e+00 |
| $f_{27}$ | 1.278115e−01 | 2.436169e−01 | 4.109635e−01 | 2.883299e−01 | 2.893272e−01 | 48935(4%) | 47798(4%) |
| $f_{28}$ | 36658(42%) | 1.114861e−02 | 2.857995e−03 | 1.031842e−03 | 32634(14%) | 29121(100%) | 32474(100%) |
| $f_{29}$ | 4.235336e+01 | 1.134753e+01 | 8.107626e+00 | 5.792662e+00 | 6.915157e+00 | 4.208158e+00 | 3.223078e+00 |
| $f_{30}$ | 44102(24%) | 50012(2%) | 1.133788e−02 | 2.930693e−03 | 35891(16%) | 38113(100%) | 43376(100%) |
| $f_{31}$ | 3.086553e+01 | 3.822877e+01 | 3.254971e+01 | 2.219017e+01 | 2.571060e+01 | 1.619689e+01 | 1.987548e+01 |
| $f_{32}$ | 5.319417e−01 | 1.258666e+00 | 1.353830e+00 | 1.468641e+00 | 4.694744e−02 | 7.134766e−03 | 6.471753e−03 |
| $f_{33}$ | 1.182004e+00 | 2.617703e+00 | 1.841598e+00 | 4.483436e+00 | 6.473105e−03 | 6.337170e−05 | 4.608779e−04 |
| $f_{34}$ | 2.415699e+02 | 6.733808e+01 | 3.421241e+01 | 3.366011e+01 | 2.905296e+01 | 1.632592e+01 | 1.543936e+01 |
| $f_{35}$ | 2.093212e+02 | 2.458260e+02 | 1.967011e+01 | 2.299215e+01 | 2.055083e−01 | 5.500275e−02 | 1.299461e−01 |
| $f_{36}$ | 1.689293e+01 | 8.168948e+00 | 2.075922e+00 | 1.144241e+00 | 7.721248e−01 | 1.764139e−01 | 3.749085e−01 |
| $f_{37}$ | 1.137223e+02 | 1.081968e+02 | 3.325390e+01 | 9.754416e+00 | 3.813655e+00 | 7.351218e−01 | 8.385001e−01 |
| $f_{38}$ | 32014(68%) | 33756(84%) | 33518(100%) | 41503(100%) | 13918(100%) | 45817(82%) | 1.365349e+03 |
| $f_{39}$ | 1.150533e−01 | 4.762356e−01 | 1.312062e+00 | 1.778036e+00 | 1.994145e−03 | 4.860482e−05 | 3.406971e−05 |
| $f_{40}$ | 3.981742e+00 | 4.221655e+00 | 5.217700e+00 | 4.851977e+00 | 6.015057e−02 | 7.445654e+00 | 1.385961e+01 |
| $f_{41}$ | 7.934347e−02 | 1.469872e−01 | 6.009665e−02 | 1.149928e−02 | 7.888807e−04 | 5.513419e−06 | 48751(6%) |
| $f_{42}$ | 1.589039e+02 | 8.472387e+01 | 5.344123e+01 | 3.876599e+01 | 2.630669e+01 | 2.169178e+01 | 2.134762e+01 |
| $f_{43}$ | 9.879616e+03 | 9.873621e+03 | 9.866847e+03 | 9.861138e+03 | 9.876897e+03 | 8.679081e+03 | 8.560682e+03 |
| $f_{44}$ | 7.212744e+01 | 5.231358e+01 | 8.713166e+01 | 1.091808e+02 | 5.475471e+01 | 5.931018e+01 | 9.749701e+01 |
| $f_{45}$ | 2.902893e+00 | 8.662253e+00 | 8.138304e+01 | 1.163435e+02 | 5.408724e−01 | 2.593443e−02 | 3.637318e−02 |
| $f_{46}$ | 3.657421e+02 | 1.400325e+02 | 5.412154e+01 | 3.136098e+01 | 1.119413e+02 | 2.159920e+01 | 4.131190e+01 |
| $f_{47}$ | 1.896465e+01 | 1.326826e+01 | 7.643055e+00 | 6.588203e+00 | 9.596499e+00 | 49016(2%) | 1.200527e+01 |
| $f_{48}$ | 1.847459e+04 | 4.984550e+03 | 2.068194e+03 | 1.744923e+03 | 5.130401e+03 | 4.373692e+02 | 3.394011e+02 |

Table 26
Results of CSS instances with BLX-$\alpha$ and $n_{ev} = 100{,}000$

| Functions | B | B-S 10 | B-S 50 | B-S 100 | B-SW 10 | B-SW 50 | B-SW 100 |
|---|---|---|---|---|---|---|---|
| $f_1$ | 4952(100%) | 6935(98%) | 59(100%) | 65(100%) | 3097(100%) | 232(100%) | 106(100%) |
| $f_2$ | 1445(100%) | 10881(100%) | 10317(100%) | 108(100%) | 7975(100%) | 8017(100%) | 872(100%) |
| $f_3$ | 1915(100%) | 6774(100%) | 4062(100%) | 2688(100%) | 4928(100%) | 12455(100%) | 18729(100%) |
| $f_4$ | 6019(100%) | 30064(90%) | 9217(100%) | 190(100%) | 8855(100%) | 8208(100%) | 4583(100%) |
| $f_5$ | 1820(100%) | 23758(92%) | 485(100%) | 762(100%) | 4918(100%) | 2051(100%) | 2748(100%) |
| $f_6$ | 16349(92%) | 20366(84%) | 391(100%) | 132(100%) | 6295(100%) | 7060(100%) | 3975(100%) |
| $f_7$ | 6039(100%) | 14006(86%) | 325(100%) | 62(100%) | 5852(100%) | 4205(100%) | 138(100%) |
| $f_8$ | 8718(100%) | 20170(98%) | 94(100%) | 54(100%) | 3427(100%) | 960(100%) | 165(100%) |
| $f_9$ | 718(100%) | 7324(100%) | 57(100%) | 54(100%) | 2777(100%) | 224(100%) | 147(100%) |
| $f_{10}$ | 2.545513e−05 | 2.545513e−05 | 2.549925e−05 | 2.545513e−05 | 2.545513e−05 | 1029(100%) | 1963(100%) |
| $f_{11}$ | 60670(42%) | 21686(8%) | 5008(100%) | 292(100%) | 21170(94%) | 16438(100%) | 28147(100%) |
| $f_{12}$ | 1285(100%) | 11531(98%) | 176(100%) | 59(100%) | 4184(100%) | 1068(100%) | 118(100%) |
| $f_{13}$ | 8568(100%) | 12763(84%) | 18227(98%) | 8402(100%) | 10973(96%) | 9430(100%) | 14874(100%) |
| $f_{14}$ | 3.772719e−05 | 3.772719e−05 | 3.772719e−05 | 3.772719e−05 | 5(100%) | 5(100%) | 6(100%) |
| $f_{15}$ | 1.326089e+00 | 3.837476e−01 | 45954(2%) | 48574(60%) | 2.972855e−03 | 39289(100%) | 52482(100%) |
| $f_{16}$ | 24999(60%) | 14377(88%) | 15379(100%) | 1197(100%) | 13256(86%) | 8006(100%) | 568(100%) |
| $f_{17}$ | 22435(72%) | 10678(96%) | 15278(100%) | 1837(100%) | 9369(100%) | 8557(100%) | 637(100%) |
| $f_{18}$ | 20874(80%) | 14060(80%) | 15035(100%) | 2264(100%) | 7936(98%) | 8561(100%) | 762(100%) |
| $f_{19}$ | 5.561376e−01 | 1.027276e−01 | 5.147861e−03 | 58015(30%) | 9.574073e−03 | 6.714158e−03 | 9.638286e−03 |
| $f_{20}$ | 6.477510e−03 | 5.709562e−02 | 1.182029e−03 | 52767(46%) | 2.530854e−03 | 48668(28%) | 59153(40%) |
| $f_{21}$ | 1.295297e−02 | 3.052629e−02 | 4.410605e−04 | 49273(56%) | 1.149368e−03 | 61731(10%) | 80030(2%) |
| $f_{22}$ | 2132(98%) | 1565(100%) | 401(100%) | 394(100%) | 5(100%) | 5(100%) | 5(100%) |
| $f_{23}$ | 1.970229e+01 | 2.061605e+02 | 1.184589e+02 | 1.255450e+02 | 3.022828e+02 | 13071(100%) | 17571(100%) |
| $f_{24}$ | 71453(34%) | 14660(20%) | 24369(96%) | 11183(100%) | 8641(100%) | 11031(100%) | 14038(100%) |
| $f_{25}$ | −1.714841e+02 | −2.008693e+02 | 48558(24%) | 43379(96%) | 42606(100%) | 34875(100%) | 33729(100%) |
| $f_{26}$ | 5.242147e+00 | 9.972331e+00 | 3.110180e+00 | 1.076668e+00 | 8.392984e+00 | 61270(36%) | 79800(68%) |
| $f_{27}$ | 8.137247e−02 | 2.472601e−01 | 3.066972e−01 | 8.542978e−02 | 2.507939e−01 | 61546(8%) | 67016(86%) |
| $f_{28}$ | 54713(96%) | 47305(6%) | 88161(8%) | 1.739159e−04 | 35379(22%) | 29121(100%) | 32474(100%) |
| $f_{29}$ | 2.934686e+01 | 1.488492e+01 | 7.820399e+00 | 4.702401e+00 | 6.832698e+00 | 2.837759e+00 | 87969(2%) |
| $f_{30}$ | 58271(84%) | 37214(2%) | 2.091396e−03 | 1.950175e−04 | 34530(16%) | 38113(100%) | 43021(100%) |
| $f_{31}$ | 2.245039e+01 | 3.256705e+01 | 2.935356e+01 | 1.178435e+01 | 2.698341e+01 | 1.427011e+01 | 5.959259e+00 |
| $f_{32}$ | 1.477319e−01 | 1.109466e+00 | 1.055155e+00 | 9.876177e−01 | 3.060543e−02 | 69892(62%) | 75704(76%) |
| $f_{33}$ | 4.306086e−02 | 1.524806e+00 | 7.352012e−01 | 1.499948e+00 | 4.759698e−03 | 71546(100%) | 74746(100%) |
| $f_{34}$ | 1.043566e+02 | 4.901324e+01 | 2.626218e+01 | 2.067024e+01 | 2.513066e+01 | 1.559208e+01 | 1.449583e+01 |
| $f_{35}$ | 1.990939e+00 | 2.859664e+02 | 6.051900e+00 | 2.507429e+00 | 1.886134e−01 | 94998(10%) | 1.476643e−03 |
| $f_{36}$ | 7.386955e+00 | 5.459242e+00 | 8.068311e−01 | 4.111370e−01 | 4.483009e−01 | 1.852903e−02 | 1.416934e−02 |
| $f_{37}$ | 5.018072e+01 | 1.260643e+02 | 2.357852e+01 | 4.736835e+00 | 3.410520e+00 | 2.346295e−01 | 9.396018e−02 |
| $f_{38}$ | 39482(96%) | 34678(88%) | 33518(100%) | 41503(100%) | 13918(100%) | 48338(100%) | 85587(92%) |
| $f_{39}$ | 3.827744e−02 | 1.800048e−01 | 3.017486e−01 | 4.721999e−01 | 1.458007e−03 | 73125(100%) | 64445(100%) |
| $f_{40}$ | 2.861456e+00 | 2.711173e+00 | 3.497689e+00 | 3.191682e+00 | 3.254885e−02 | 1.972347e−01 | 7.153613e+00 |
| $f_{41}$ | 5.564298e−03 | 1.085397e−01 | 2.363329e−02 | 4.943316e−03 | 6.753101e−04 | 59636(100%) | 55160(100%) |
| $f_{42}$ | 1.008835e+02 | 6.329263e+01 | 3.628763e+01 | 2.834539e+01 | 2.535653e+01 | 2.103455e+01 | 2.014444e+01 |
| $f_{43}$ | 9.873392e+03 | 9.871759e+03 | 9.866811e+03 | 9.858188e+03 | 9.873655e+03 | 8.653460e+03 | 8.325625e+03 |
| $f_{44}$ | 5.940600e+01 | 2.093625e+01 | 3.026747e+01 | 4.169214e+01 | 5.319985e+01 | 4.529451e+01 | 3.497023e+01 |
| $f_{45}$ | 6.626323e−01 | 3.844955e+00 | 2.321214e+01 | 2.000707e+01 | 3.134460e−02 | 91653(66%) | 92324(74%) |
| $f_{46}$ | 2.852014e+02 | 1.188399e+02 | 4.426497e+01 | 2.397967e+01 | 5.772311e+01 | 8.464340e+00 | 8.680244e+00 |
| $f_{47}$ | 1.543579e+01 | 1.386189e+01 | 6.110089e+00 | 91334(2%) | 1.081318e+01 | 69344(20%) | 90617(14%) |
| $f_{48}$ | 1.128482e+04 | 6.195752e+03 | 1.667574e+03 | 8.631561e+02 | 1.612488e+03 | 3.887660e+02 | 1.358254e+02 |

Table 27
Results of CSS instances with BLX-α and $n_{ev} = 500{,}000$

| Functions | B | B-S 10 | B-S 50 | B-S 100 | B-SW 10 | B-SW 50 | B-SW 100 |
|---|---|---|---|---|---|---|---|
| $f_1$ | 4952(100%) | 22746(100%) | 59(100%) | 65(100%) | 3097(100%) | 232(100%) | 106(100%) |
| $f_2$ | 1445(100%) | 10881(100%) | 10317(100%) | 108(100%) | 7975(100%) | 8017(100%) | 872(100%) |
| $f_3$ | 1915(100%) | 6774(100%) | 4062(100%) | 2688(100%) | 4928(100%) | 12455(100%) | 18729(100%) |
| $f_4$ | 6019(100%) | 49943(98%) | 9217(100%) | 190(100%) | 8855(100%) | 8208(100%) | 4583(100%) |
| $f_5$ | 1820(100%) | 27670(100%) | 485(100%) | 762(100%) | 4918(100%) | 2051(100%) | 2748(100%) |
| $f_6$ | 22932(100%) | 63059(90%) | 391(100%) | 132(100%) | 6295(100%) | 7060(100%) | 3975(100%) |
| $f_7$ | 6039(100%) | 33744(98%) | 325(100%) | 62(100%) | 5852(100%) | 4205(100%) | 138(100%) |
| $f_8$ | 8718(100%) | 19448(100%) | 94(100%) | 54(100%) | 3427(100%) | 960(100%) | 165(100%) |
| $f_9$ | 718(100%) | 7324(100%) | 57(100%) | 54(100%) | 2777(100%) | 224(100%) | 147(100%) |
| $f_{10}$ | 2.545513e−05 | 2.545513e−05 | 2.545513e−05 | 2.545513e−05 | 2.545513e−05 | 1029(100%) | 1963(100%) |
| $f_{11}$ | 137848(100%) | 190447(38%) | 5008(100%) | 292(100%) | 27053(100%) | 16438(100%) | 28147(100%) |
| $f_{12}$ | 1285(100%) | 9960(100%) | 176(100%) | 59(100%) | 4184(100%) | 1068(100%) | 118(100%) |
| $f_{13}$ | 8568(100%) | 37423(100%) | 30266(100%) | 8247(100%) | 14591(100%) | 9430(100%) | 14874(100%) |
| $f_{14}$ | 3.772719e−05 | 3.772719e−05 | 3.772719e−05 | 3.772719e−05 | 5(100%) | 5(100%) | 6(100%) |
| $f_{15}$ | 4.145411e−02 | 177943(2%) | 1.552446e−03 | 50064(66%) | 3.797791e−04 | 39289(100%) | 52482(100%) |
| $f_{16}$ | 73882(76%) | 22949(100%) | 15379(100%) | 1197(100%) | 27557(100%) | 8006(100%) | 568(100%) |
| $f_{17}$ | 52842(98%) | 32222(100%) | 15278(100%) | 1837(100%) | 9369(100%) | 8557(100%) | 637(100%) |
| $f_{18}$ | 41735(100%) | 35124(100%) | 15035(100%) | 2264(100%) | 15184(100%) | 8561(100%) | 762(100%) |
| $f_{19}$ | 4.812189e−02 | 3.669399e−02 | 2.021548e−03 | 142532(50%) | 5.320622e−03 | 3.085972e−03 | 198659(6%) |
| $f_{20}$ | 155431(2%) | 1.237235e−02 | 1.252107e−03 | 81805(56%) | 1.736180e−03 | 138328(44%) | 129415(40%) |
| $f_{21}$ | 283722(8%) | 8.527320e−03 | 3.225175e−04 | 108733(96%) | 2.528219e−04 | 97741(10%) | 137484(10%) |
| $f_{22}$ | 4510(100%) | 1565(100%) | 401(100%) | 462(100%) | 5(100%) | 5(100%) | 5(100%) |
| $f_{23}$ | 7.636540e−05 | 1.211924e+02 | 1.046893e+02 | 1.184465e+02 | 5.921935e+01 | 13071(100%) | 17571(100%) |
| $f_{24}$ | 101485(100%) | 157582(50%) | 32150(98%) | 11183(100%) | 8641(100%) | 11031(100%) | 14038(100%) |
| $f_{25}$ | 380833(12%) | −2.038806e+02 | 47204(26%) | 44869(94%) | 42606(100%) | 34875(100%) | 33729(100%) |
| $f_{26}$ | 313941(100%) | 7.704960e+00 | 2.895993e+00 | 1.015286e+00 | 6.992524e+00 | 181006(70%) | 121146(100%) |
| $f_{27}$ | 212651(2%) | 1.096986e−01 | 2.753277e−01 | 182274(2%) | 1.734481e−01 | 236862(32%) | 96604(100%) |
| $f_{28}$ | 59562(100%) | 381575(16%) | 89899(4%) | 160026(16%) | 241509(64%) | 29121(100%) | 32474(100%) |
| $f_{29}$ | 4.203800e+00 | 1.008406e+01 | 7.332850e+00 | 4.373326e+00 | 5.431421e+00 | 2.739320e−01 | 2.677649e−01 |
| $f_{30}$ | 70416(100%) | 389845(10%) | 102438(6%) | 143603(22%) | 270774(46%) | 38113(100%) | 43021(100%) |
| $f_{31}$ | 8.445748e+00 | 3.144159e+01 | 3.285269e+01 | 1.240396e+01 | 2.331991e+01 | 1.438023e+01 | 390153(6%) |
| $f_{32}$ | 173371(10%) | 1.090352e+00 | 8.884628e−01 | 7.140964e−01 | 389571(8%) | 72411(50%) | 118694(96%) |
| $f_{33}$ | 159645(100%) | 2.126277e+00 | 1.980196e−01 | 4.423011e−01 | 9.868205e−04 | 71546(100%) | 74746(100%) |
| $f_{34}$ | 4.368443e+01 | 5.581233e+01 | 2.216605e+01 | 1.722984e+01 | 2.257543e+01 | 1.151256e+01 | 1.073733e+01 |
| $f_{35}$ | 224617(100%) | 1.459182e+02 | 5.711055e+00 | 1.835935e+00 | 6.038044e−02 | 145442(100%) | 135935(100%) |
| $f_{36}$ | 2.562271e−02 | 7.117902e+00 | 6.160489e−01 | 2.575956e−01 | 2.447435e−01 | 7.693346e−04 | 2.567366e−04 |
| $f_{37}$ | 3.426064e−01 | 1.030321e+02 | 1.295681e+01 | 2.653539e+00 | 2.200404e+00 | 336110(2%) | 244193(36%) |
| $f_{38}$ | 47337(100%) | 37688(90%) | 33518(100%) | 41408(100%) | 13918(100%) | 48338(100%) | 89333(100%) |
| $f_{39}$ | 218831(100%) | 1.659817e−01 | 7.196468e−02 | 7.404395e−02 | 1.056527e−03 | 73125(100%) | 64445(100%) |
| $f_{40}$ | 419236(22%) | 2.565244e+00 | 3.530379e+00 | 2.680752e+00 | 1.778486e−02 | 484087(2%) | 280960(92%) |
| $f_{41}$ | 185191(100%) | 9.641361e−02 | 1.198017e−02 | 8.619657e−04 | 4.584610e−04 | 59636(100%) | 55160(100%) |
| $f_{42}$ | 4.343522e+01 | 7.446302e+01 | 3.012355e+01 | 2.496727e+01 | 2.394210e+01 | 1.782270e+01 | 1.681939e+01 |
| $f_{43}$ | 9.870824e+03 | 9.871804e+03 | 9.866692e+03 | 9.858513e+03 | 9.850991e+03 | 8.255704e+03 | 7.834145e+03 |
| $f_{44}$ | 2.831805e+01 | 1.005868e+01 | 6.654188e+00 | 4.288261e+00 | 5.550178e+01 | 4.099224e+01 | 1.776791e+01 |
| $f_{45}$ | 308009(8%) | 3.088693e+00 | 1.307391e+00 | 1.247956e+00 | 433625(2%) | 91030(70%) | 114901(92%) |
| $f_{46}$ | 1.983071e+02 | 1.311071e+02 | 4.247340e+01 | 1.827674e+01 | 6.645950e+01 | 6.848937e+00 | 5.137987e+00 |
| $f_{47}$ | 330651(38%) | 1.105683e+01 | 110999(2%) | 177150(4%) | 8.800847e+00 | 176325(28%) | 158527(28%) |
| $f_{48}$ | 1.809713e+03 | 4.881057e+03 | 1.551120e+03 | 5.484911e+02 | 3.283202e+02 | 2.093562e+02 | 3.548137e+01 |

Each parameter (coefficient) is in the range $-512$ to 512. The objective function value of the optimum is $P_{\text{Chev}}(C^*) = 0$.

## Appendix B. Results of the experiments

Tables 20–27 show the results of the experiments. The following information appears for each function and algorithm:

- One number only. This means that the algorithm does not achieve a successful solution. Then, the average of the better objective function values is outlined.
- Two numbers. The first number represent the average number of evaluations accomplished to achieve a successful solution. The second number (in parentheses) is the percentage of runs in which the algorithm found a successful solution.

## References

[1] T. Bäck, 1996. Evolutionary Algorithms in Theory and Practice, Oxford.

[2] R. Battiti, G. Tecchiolli, The continuous reactive tabu search: Blending combinational optimization and stochastic search for global optimization, Annals of Operations Research 63 (1996) 53–188.

[3] H.-G. Beyer, K. Deb, On self-adaptive features in real-parameter evolutionary algorithms, IEEE Transactions on Evolutionary Computation 5 (3) (2001) 250–270.

[4] R. Chelouah, P. Siarry, A continuous genetic algorithm designed for the global optimization of multimodal functions, Journal of Heuristics 6 (2000) 191–213.

[5] R. Chelouah, P. Siarry, Tabu search applied to global optimization, European Journal of Operational Research 123/2 (2000) 30–44.

[6] R. Chelouah, P. Siarry, Genetic and Nelder-Mead algorithm algorithms hybridized for a more accurate global optimization of continuous multiminima functions, European Journal of Operational Research 148 (2003) 335–348.

[7] K. Deb, Multi-Objective Optimization Using Evolutionary Algorithms, John Wiley, 2001.

[8] K. Deb, A population-based algorithm-generator for real-parameter optimization, KanGAL Report No. 2003003, Kanpur Genetic Algorithms Laboratory, Department of Mechanical Engineering, Indian Institute of Technology, Kanpur, 2003.

[9] K.A. De Jong, W.M. Spears, A formal analysis of the role of multi-point crossover in genetic algorithms, Annals of Mathematics and Artificial Intelligence 5 (1) (1992) 1–26.

[10] L.J. Eshelman, J.D. Schaffer, Preventing premature convergence in genetic algorithms by preventing incest, in: R. Belew, L.B. Booker (Eds.), Proceedings of the Fourth International Conference on Genetic Algorithms, Morgan Kaufmann, San Mateo, CA, 1991, pp. 115–122.

[11] L.J. Eshelman, J.D. Schaffer, Real-coded genetic algorithms and interval-schemata, in: L.D. Whitley (Ed.), Foundations of Genetic Algorithms 2, Morgan Kaufmann, San Mateo, CA, 1993, pp. 187–202.

[12] L.J. Eshelman, K.E. Mathias, J.D. Schaffer, Convergence controlled variation, in: R. Belew, M. Vose (Eds.), Foundations of Genetic Algorithms 4, Morgan Kaufmann, San Mateo, CA, 1997, pp. 203–224.

[13] C. Fleurent, F. Glover, P. Michelon, Z. Valli, A scatter search approach for unconstrained continuous optimisation, in: Proceedings of the 1996 IEEE International Conference on Evolutionary Computation, IEEE Press, Piscataway, NJ, 1996, pp. 643–648.

[14] D.B. Fogel, Evolutionary Computation: Toward a New Philosophy of Machine Intelligence, IEEE Press, Piscataway, NJ, 1995.

[15] F. Glover, Heuristics for integer programming using surrogate constraints, Decision Sciences 8 (1) (1977) 156–166.

[16] F. Glover, Genetic algorithms and scatter search: Unsuspected potentials, Statistics and Computing 4 (1994) 131–140.

[17] F. Glover, Scatter search and start-paths: Beyond the genetic metaphor, OR Spectrum 17 (1995) 125–137.

[18] F. Glover, M. Laguna, Tabu Search, Kluwer Academic Publishers, 1997.

[19] F. Glover, A template for scatter search and path relinking, in: J.-K. Hao, E. Lutton, E. Ronald, M. Schoenauer, D. Snyers (Eds.), Artificial Evolution, Lecture Notes in Computer Science, vol. 1363, Springer, 1998, pp. 13–54.

[20] F. Glover, Genetic algorithms, evolutionary algorithms and scatter search: Changing tides and untapped potentials, INFORMS Computer Science Technical Section Newsletter 19 (1) (1998) 7–14.

[21] F. Glover, G.A. Kochenberger (Eds.), Handbook of Metaheuristics, Kluwer Academic Publishers, 2003.

[22] D.E. Goldberg, Genetic Algorithms in Search, Optimization, and Machine Learning, Addison-Wesley, New York, 1989.

[23] P. Hansen, N. Mladenović, Variable neighborhood search: Principles and applications, European Journal of Operational Research 130 (2001) 449–467.

[24] W.E. Hart, Adaptive global optimization with local search, Ph.D. Thesis, University of California, San Diego, CA, 1994.

[25] F. Herrera, M. Lozano, J.L. Verdegay, Tackling real-coded genetic algorithms: Operators and tools for the behavioral analysis, Artificial Intelligence Reviews 12 (4) (1998) 265–319.

[26] F. Herrera, M. Lozano, E. Pérez, A.M. Sánchez, P. Villar, Multiple crossover per couple with selection of the two best offspring: An experimental study with the BLX-$\alpha$ crossover operator for real-coded genetic algorithms, in: F.J. Garijo, J.C. Riquelme, M. Toro (Eds.), IBERAMIA 2002, Lecture Notes in Artificial Intelligence, vol. 2527, Springer-Verlag, Berlin, 2002, pp. 392–401.

[27] F. Herrera, M. Lozano, A.M. Sánchez, A taxonomy for the crossover operator for real-coded genetic algorithms: An experimental study, International Journal of Intelligent Systems 18 (3) (2003) 309–338.

[28] J.H. Holland, Adaptation in Natural and Artificial Systems, The MIT Press, London, 1992.

[29] H. Kita, A comparison study of self-adaptation in evolution strategies and real-coded genetic algorithms, Evolutionary Computation Journal 9 (2) (2001) 223–241.

[30] N. Krasnogor, Studies on the theory and design space of memetic algorithms. Ph.D. Thesis, University of the West of England, Bristol, UK, 2002.

[31] M. Laguna, R. Martí, Scatter Search. Methodology and Implementations in C, Kluwer Academic Publishers, 2003.

[32] R. Martí, M. Laguna, F. Glover, Principles of scatter search. Special issue on scatter search and path relinking, European Journal of Operational Research, 169 (2006) 359–372.

[33] Z. Michalewicz, Genetic Algorithms + Data Structures = Evolution Programs, Springer-Verlag, New York, 1992.

[34] P.A. Moscato, On evolution, search, optimization, genetic algorithms and martial arts: Towards memetic algorithms. Technical Report Caltech Concurrent Computation Program Report 826, Caltech, Caltech, Pasadena, CA, 1989.

[35] P.A. Moscato, Memetic algorithms: A short introduction, in: D. Corne, M. Dorigo, F. Glower (Eds.), New Ideas in Optimization, McGraw-Hill, London, 1999, pp. 219–234.

[36] J.A. Nelder, R. Mead, A simplex method for functions minimizations, Computer Journal 7 (1965) 308.

[37] T. Nomura, K. Shimohara, An analysis of two-parent recombinations for real-valued chromosomes in an infinite population, Evolutionary Computation Journal 9 (3) (2001) 283–308.

[38] P. Siarry, G. Berthiau, F. Durbin, J. Haussy, Enhanced simulated annealing for globally minimization functions of many continuous variables, ACM Transactions of Mathematical Software 23 (2) (1997) 209–228.

[39] F.J. Solis, R.J.-B. Wets, Minimization by random search techniques, Mathematical Operations Research 6 (1981) 19–30.

[40] R. Storn, K. Price, Differential evolution—A simple and efficient adaptive scheme for global optimization over continuous spaces, Technical Report TR-95-012, International Computer Science Institute, Berkeley, CA, 1995.

[41] Y.S. Teh, G.P. Rangaiah, Tabu search for global optimization of continuous functions with application to phase equilibrium calculations, Computer and Chemical Engineering 27 (2003) 1665–1679.

[42] T.B. Trafalis, S. Kasap, An affine scaling scatter search approach for continuous global optimization problems, in: C.H. Dagli et al. (Eds.), Intelligent Engineering Systems Through Artificial Neural Networks, ASME Press, 1996, pp. 1027–1032.

[43] T.B. Trafalis, S. Kasap, A novel metaheuristics approach for continuous global optimisation, Journal of Global Optimization 23 (2002) 171–190.

[44] S. Tsutsui, Y. Fujimoto, Forking genetic algorithm with blocking and shrinking modes, in: S. Forrest (Ed.), Proceedings of the Fifth International Conference on Genetic Algorithms, Morgan Kaufmann, San Mateo, CA, 1993, pp. 206–213.