An automated hybrid genetic-conjugate gradient algorithm for multimodal optimization problems

Pradeep Kumar Gudla, Ranjan Ganguli *

Department of Aerospace Engineering, Indian Institute of Science, Bangalore 560012, India

Abstract

The genetic algorithm (GA) have good global search characteristics and local optimizing algorithm (LOA) have good local search characteristics. In the present work, best characteristics of GA and LOA are combined to develop a hybrid genetic algorithm (HGA). A bank of GA’s are used to get a good starting solution for a conjugate gradient algorithm. The number of GA banks is selected using an automated procedure based on Fibonacci numbers. This automated hybrid genetic algorithm (AHGA) is used for solving general multimodal optimization problems while assuring global optimality to a significant degree. The designed algorithm is also tested against a variety of standard test functions. Besides assuring global optimality to a significant extent AHGA is also found to be an efficient algorithm requiring only one tuning error parameter saving considerable time on the part of the user. The method also addresses the problem of selecting a good starting design for gradient based algorithm. Further in the few cases where the algorithm does not converge to a global minima, a local minima is assured because of the use of the gradient based local search in the final stage of the algorithm. Further, the algorithm assures one final solution to the optimization problem and addresses the
problem of providing a deterministic output which inhibits the use of GA in engineering optimization software and engineering applications.
© 2004 Elsevier Inc. All rights reserved.

Keywords: Genetic algorithm; Conjugate gradient; Hybrid genetic algorithm; Multimodal optimization

1. Introduction

Genetic algorithm (GA) is a well known global optimization technique. GA is a heuristic based optimization technique inspired by natural selection process and population genetics theory. GA’s provide a general architecture for solving complex optimization problems. It was first proposed by Holland [1] and since then it has been well studied and has wide range of applications including optimization [2].

A simple basic binary GA starts with a set of randomly generated population of strings forming the initial generation. Each string of the population which are coded in a binary format, represents a point in the solution space. Once the initial generation is generated, members of the mating pool are selected using reproduction operator, by which a member is selected with a probability proportional to its fitness value. Hence, members with a better fitness value have many copies in the mating pool. Members of the mating pool are selected in pairs, randomly, and are crossed using the crossover operator to form the offsprings. The probability of a member of the mating pool getting crossed equals the crossover probability. Later, bit of each member from the mating pool is mutated with the probability equal to mutation probability. The resulting offsprings form a new generation and the process is repeated for several generations. The performance of GA can be improved by employing an ‘elitist strategy’. Elitist strategy involves replacing the weakest member of the present generation with the elite member, member with highest fitness value, of the previous generation. Such a strategy ensures monotonic improvement in the best fitness value of each generation and helps in improving the performance of GA. The parameters involved which include population size, crossover probability, mutation probability and the number of generations together form the parameters of the GA and determine the performance of the GA.

GA is considered to have better global optimizing properties than other heuristic optimization techniques, especially in the case of problems having large search spaces with many local extrema. However, the use of GA does not guarantee convergence to global optima because of its poor exploitation capabilities. GA also has other drawbacks such as premature convergence [2,6,7] which occurs because of the loss of diversity in the population and it is a com-
monly encountered problem when the search goes on for several generations. Another problem with GA is the need to properly select the starting population size, crossover probability, mutation probability and the maximum number of generations. Considerable human time besides computational effort is needed to select these parameters using numerical experiments. One more shortcoming of GA is the lack of a good convergence criteria. For gradient based methods, the norm of the gradient can be used as a descent function and should decrease as the iterations progress and become very small at the local optimal point. However, for GA, it is not possible to tell when the optimal point is reached particularly for engineering problems where the point is not known in advance. Because of this ambiguity in the definition of convergence criteria, engineering software mostly use gradient based methods which offer a clear (though often local) optimal point.

In summary, it can be said that GA employs a probabilistic approach and has better global optimizing properties but shows poor convergence to optimality. Therefore, often GA is hybridized using a local optimization algorithm (LOA) to improve its performance as a global optimization technique while overcoming the limitations of poor convergence and weak exploitation capabilities. In the literature, many algorithms [6–21] have been designed combining the best characteristics of the GA and LOA. However, these studies are limited to certain specific problems and have been found to give only a marginally better solution than the solutions obtained via other existing algorithms. Recently, few works [22,23] have also addressed general optimization problems. The various kind of hybridizations using LOA can be classified into three types:

1. Pre-hybridization [18,19], in which the initial population of GA is generated using an LOA. Lee and Lee [18] and Chen et al. [19] used this approach for reducing the solution space for GA and improving the efficiency. Such an approach seems to be well suited to the specific problem they addressed and does not seem to be suitable for general optimization.

2. Organic-hybridization [11–17], in which a LOA is used as one of the operators of GA for improving each member of the population in each generation. Many of these works addressed specific problems and used LOA which are known to perform well on the respective problems. Konak and Smith [11] developed a HGA for solving the backbone design of communication networks to obtain an improved solution. Alfonsas [12] developed a similar HGA for solving QAP (quadratic assignment problem) which was found to give a better performance as far as efficiency is concerned. These works were developed for solving certain specific problems, where only marginally improved solutions were achieved. However, for the general optimization problem, such hybridization results in reducing the global optimizing properties of HGA while improving its convergence characteristics when compared to a GA. Wang and Wu [14], Kim et al. [15] and Park and Gilbert [16] have shown in their works that such an HGA looses its dependency on the GA parameters. This type of HGA
is also quite sensitive to the initial seed number reflecting the greater say LOA enjoys on the convergence characteristics of this HGA. While HGA converged to better solution in [14–16], a global optima was not assured. In summary, though the organic-hybridization is computationally more efficient than a GA it offers little assurance of global minima and also lacks a proper convergence criteria.

(3) Post-hybridization [20–22], in which GA is used to provide an initial design for LOA. This kind of hybridization seems to be the best way of combining the best characteristics of the two approaches as no compromise is made on the global and local optimizing characteristics unlike in the case of organic-hybridization. Kasprzyk and Jaskula [20] used a post-hybridized GA using simplex method for deconvolution of electrochemical responses of SDLSV (semidifferential linear sweep voltametry) methods. Here GA was run till the population diversity is lost, upon which a simplex method is initiated to improve the solution. Zhou et al. [21] used both post- and organic-hybridized GA for solving the job shop scheduling problem. The HGAs [20,21] are developed in an attempt to obtain an improved solution of certain specific problems and their designs were developed to exploit problem specific features. However, few attempts are also made to develop HGA for general global optimization problems. Rachid and Patrick [22] proposed a continuous hybrid algorithm (CHA) combining GA and Nelder–Mead simplex algorithm for continuous multimodal optimization problems. CHA showed better efficiency than few other algorithms when tested against a set of test functions. However, CHA does not seem to have a proper criteria for transferring the search from a GA to LOA and at the same time its global optimizing properties do not seem to be any better than that of a GA. However, CHA exhibits better convergence characteristics than GA which is also the case with the other post-hybridized algorithms.

In the present work, we develop a unique design of hybrid genetic algorithm (HGA), where GA is post-hybridized with a LOA. This algorithm does not compromise on the global and local search characteristics of the individual methods used and also seems to exhibit improved global optimizing properties upon GA. An automated hybrid genetic algorithm (AHGA) is developed which addresses the criteria of transferring the search from GA to LOA which is a key weakness of post-hybridized techniques. The AHGA is tested on several test functions and appears to assure global optimality to a significant degree.

2. Hybrid genetic algorithm

The hybrid genetic algorithm (HGA) developed in this study uses a typical basic GA with elitist strategy to reach near the global optima and then uses a
gradient based search method for faster convergence to global optima. However, it is a known fact that GA show better global optimizing properties when run several times for fewer number of generations rather than run fewer times for many generations. Such a behavior of GA is due its well known weaknesses called premature convergence and weak exploitation capabilities. Because of which, GA’s, when run for several generations either tend to prematurely converge to a local minima owing to loss in the diversity in the population or hang around the global optima without converging. Therefore, instead of using the GA only once a bank of \( n \) GA’s are used with the same population size, crossover and mutation probability, and number of generations. Such a design also seems to improve upon the global optimizing characteristics of GA. The best point obtained from these \( n \) GA’s is then used as the starting design for the conjugate gradient method (CG), a gradient search method. In the computer implementation, a GA subroutine is called \( n \) times with a different random initial population and the best solution point obtained is used as an initial condition for the conjugate gradient subroutine. The number of generations is kept constant for all the \( n \) runs. The architecture of the algorithm is shown in Fig. 1.

In the case of difficult problems HGA reaches near global optima at a larger value of \( n \). Hence the HGA hybridizes an exhaustive search along with GA and CG. The HGAs performance is better and efficient when each GA is able to carry out a reasonably good search so that it reaches near the global optima in one of these runs. Such a performance of GA depends upon the choice of the GA parameters which vary from problem to problem and there exists no definitive set of parameter values for which GA can do an optimum search. However, it is still feasible to obtain a set of rules determining the parameter values so that GA can do, if not optimum, a reasonably good search. Among the parameters, population size \([4,5]\) (\( N \)), number of generations and mutation probability (\( P_m \)) seem to play a key role in the performance of GA. It is especially important to scale these parameter values appropriately as the dimension of the optimizing function increases for good performance of the GA.

![Fig. 1. A schematic representation of the hybrid genetic algorithm.](image-url)
As far as scaling the number of generations with the dimension or size of the solution space is concerned not much understanding has been achieved as seen from the literature and therefore a linear variation with the dimension is adopted, i.e.,

$$\text{Number of Generations} = N_a \times d,$$

(1)

where $d$ is the number of design variables and $N_a$ is an integer. However, in the case of population size $[5,4]$ many criteria have been designed on the basis of schema ideology. But most of these are practically impossible to implement for higher dimensional problems. Holland [1] mentions that the effective number of schemata that are processed in a population size of $N$ are of the order of $O(N^3)$. The number of lowest order building blocks (BBs) in a member of string length $L$ and $d$ variables is equal to $(2L)^d$. Therefore, minimum size of the population such that all the above BBs are being processed is of the order of $(2L)^{d/3}$, i.e.,

$$N \approx O(2^{d/3}L^{d/3}) \quad \text{or} \quad \approx O(K(L,d)2^{d/3}),$$

where $K$ is a function of $L$ and $d$. However, such an estimate for higher dimension can still be practically impossible to be implemented because of high computation time. Therefore, we assume $K$ to be a constant and an appropriate value is chosen i.e.,

$$N = N_b \times 2^{d/3} \quad \text{(rounded to the nearest integer)},$$

(2)

where $N_b$ is an integer. Now looking at the mutation probability, in the case of a constant value the number of mutations grows linearly with the dimension of the problem and converts the search process into a completely random search in the case of higher dimensional problems. Therefore, $P_m$ is chosen such that in a generation, on an average, a certain percent of the population is changed by the mutation operator and can be given by,

$$P_m = \frac{N_c}{d \times L \times P_c \times 100},$$

(3)

where $P_c$ is the crossover probability and $N_c$ is an integer. A set of test functions are chosen to study the performance of the algorithm. The test functions are given in Appendix A.

2.1. Numerical studies

Numerical studies are carried out studying the performance of the algorithm by executing it on each of the test functions several times and noting the success rate of the algorithm for different values of $n$. Here $n$ is the number of GA banks as shown in Fig. 1. The other parameter values chosen were typical of a basic binary GA besides population size ($N$), mutation probability ($P_m$)
and the number of generations, which are given by (2), (3), and (1) respectively. They are also shown in Table 1. Appropriate values for \(N_a\), \(N_b\), and \(N_c\) are also given in Table 1 and are used for the numerical experiments. These values appear to give good results for a wide variety of problems.

The results in Fig. 2 show that the success rate of the hybrid algorithm increases as \(n\) is increased. At the same time the number of function evaluations

<table>
<thead>
<tr>
<th>Parameter values chosen for test simulations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Population size ((N))</td>
</tr>
<tr>
<td>Crossover probability ((P_c))</td>
</tr>
<tr>
<td>Mutation probability ((P_m))</td>
</tr>
<tr>
<td>String length ((L))</td>
</tr>
<tr>
<td>Maximum number of generations in GA (N_a)</td>
</tr>
<tr>
<td>Maximum number of iterations in CG (N_b)</td>
</tr>
<tr>
<td>(N_c)</td>
</tr>
</tbody>
</table>

Fig. 2. Performance of HGA against various test functions.
also increases, increasing the computational cost and reducing the computational efficiency of the algorithm. Therefore, an optimum value of $n$ that converges to global optima while using minimal computational effort, will be the smallest value of $n$ for which the algorithm reaches a 100% success rate. Therefore, a proper value of $n$ where the search shifts from using a GA methodology to a LOA methodology plays a key role in determining the efficiency and effectiveness of the algorithm. Such an optimum value of $n$ varies widely depending upon the characteristics of the optimization function and is also influenced by the probabilistic nature of GA.

For a typical GA implementation the user needs to perform many numerical experiments with parameters such as population size, crossover probability and mutation probability to obtain optimum parameter values not only for a better performance of GA but also to ensure convergence to a global optima. However in HGA, by using a bank of $n$ GA’s, we are able to transfer the problem of selecting the GA tuning parameters, to that of selecting $n$. Even though HGA takes more computational effort than a GA with optimal parameter values while converging to global optima, it is more efficient in the sense that it saves considerable human time.

3. Automated hybrid genetic algorithm

The key variable in the algorithm discussed above is the number, $n$, of GA runs that are used to obtain the starting point for the CG method. An optimum value of $n$, which varies widely depending upon the characteristics of the optimizing function, is required for an efficient search to be performed while assuring global optimality. To obtain such an optimum value, the user needs to carry out a convergence study spending computational time besides considerable human time. We develop an approach to automate the process of selection of $n$ so that a single run of the algorithm will give a global optima in most cases irrespective of the characteristics of the optimizing function.

In the automated hybrid genetic algorithm (AHGA) the GA subroutine is called separately for $f_{bi}$ number of times, where $fb_i$ is the $i$th term of the Fibonacci series. The Fibonacci series is defined as,

$$f_{bi} = f_{b_{i-1}} + f_{b_{i-2}},$$

where $fb_1 = 1$, $fb_2 = 2$. Thus the resulting series consists of 1,2,3,5,8,13, 21,34,55,89,144,...,$f_i$ is the best function value obtained after $fb_i$ number of independent searches were carried out using GA. The algorithm is said to have converged when

$$\frac{f_i - f_{i-2}}{f_i} \leq \epsilon$$
is satisfied, for \( i \geq 5 \). This criteria is a measure of the fact that the optimal function value predicted by the GA has stopped improving for some time. The schematic representation of the algorithm is shown in Fig. 3 where \( f_{b0} = 0 \). The best search point so far obtained is fed to the conjugate gradient subroutine as its initial estimate. Because of the random nature of GA, using the Fibonacci series ensures that the best point selected by the GA bank has settled down and is therefore close to a global optima. The use of CG from this point should therefore result in rapid convergence to the optimal point, since gradient based methods are locally fast. The key problem of finding a good starting design for the gradient based method is also addressed.

3.1. Numerical studies

Numerical studies are carried out by testing the algorithm on an extended set of test functions than shown in Fig. 2. The algorithm is tested by executing

Fig. 3. Schematic representation of automated hybrid genetic algorithm.
it for 10 number of times on each test function and the success rate of the algorithm is noted. The results are shown in Table 2. It can be observed that AHGA seems to perform well in the case of functions suitable for evolutionary algorithms than functions with steep and shallow valleys which are suitable for a gradient based algorithms. A 100% success rate is obtained in most of the cases, except in the case of Freudenstein and Roth function and Rosenbrock's function because of probabilistic misfortune and also because these are difficult functions [25] for GA. The number of GA calls at each execution varies significantly which is not only due to the probabilistic nature of GA but also because of the criteria of truncating the GA search and calling LOA. Other approaches besides Fibonacci's series were also studied. A geometric progression is also possible but leads to a faster increase in the number of runs needed. The AHGA seems to take more computational time as can be observed from the more average number of runs it takes than the optimum values shown in

Table 2
Performance of the automated hybrid genetic algorithm (AHGA)

<table>
<thead>
<tr>
<th>S. no.</th>
<th>Test function</th>
<th>Success rate</th>
<th>No. of GA calls</th>
<th>Avg. no. of calls</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Freudenstein and Roth</td>
<td>90</td>
<td>8, 8, 8, 8, 8, 13, 13, 21, 34, 34</td>
<td>≈15</td>
</tr>
<tr>
<td>2.</td>
<td>Evolutionary test function</td>
<td>100</td>
<td>8, 8, 8, 8, 8, 8, 8, 8, 8, 8</td>
<td>8</td>
</tr>
<tr>
<td>3.</td>
<td>Himmenblau function</td>
<td>100</td>
<td>8, 8, 8, 8, 8, 8, 13, 21, 34, 34</td>
<td>15</td>
</tr>
<tr>
<td>4.</td>
<td>Rosenbrock's function (R₂)</td>
<td>100</td>
<td>8, 8, 8, 8, 13, 21, 21, 21, 34, 55</td>
<td>≈20</td>
</tr>
<tr>
<td>5.</td>
<td>R₅</td>
<td>80</td>
<td>8, 8, 8, 8, 13, 21, 21, 34, 34</td>
<td>≈19</td>
</tr>
<tr>
<td>6.</td>
<td>Hartmann function (H₃)</td>
<td>100</td>
<td>8, 8, 8, 8, 8, 8, 21, 34, 34</td>
<td>15</td>
</tr>
<tr>
<td>7.</td>
<td>H₁₀</td>
<td>100</td>
<td>8, 8, 13, 21, 21, 21, 21, 34, 55</td>
<td>≈23</td>
</tr>
<tr>
<td>8.</td>
<td>H₁₅</td>
<td>100</td>
<td>8, 8, 8, 13, 13, 21, 21, 34, 55, 89</td>
<td>27</td>
</tr>
<tr>
<td>9.</td>
<td>Shekel function (S₂)</td>
<td>100</td>
<td>8, 8, 8, 8, 8, 8, 8, 8, 13, 21</td>
<td>≈10</td>
</tr>
<tr>
<td>10.</td>
<td>S₅</td>
<td>100</td>
<td>8, 8, 8, 13, 21, 21, 34, 34, 34, 89</td>
<td>27</td>
</tr>
<tr>
<td>12.</td>
<td>Zakharov's function (Z₂)</td>
<td>100</td>
<td>8, 8, 8, 8, 8, 8, 8, 8, 8, 13</td>
<td>≈9</td>
</tr>
<tr>
<td>13.</td>
<td>Z₅</td>
<td>100</td>
<td>8, 8, 8, 8, 21, 21, 21, 55, 89, 377</td>
<td>≈62</td>
</tr>
<tr>
<td>14.</td>
<td>Z₁₀</td>
<td>100</td>
<td>8, 8, 8, 8, 8, 13, 13, 21, 21, 34</td>
<td>≈15</td>
</tr>
<tr>
<td>15.</td>
<td>Z₁₅</td>
<td>100</td>
<td>8, 8, 8, 13, 13, 21, 34, 89, 233, 377</td>
<td>≈81</td>
</tr>
</tbody>
</table>
the previous section (Fig. 2), and a better shifting criteria still remains an open possibility. The use of the Fibonacci series is a conservative choice. The AHGA can be controlled by only one parameter $\epsilon$, for which a value of 2% appears good and was used for all problems in Table 2.

This algorithm does require more computer time than in the case where GA converges to a global optima with the use of optimal parameter values. However, it saves considerable human time needed to tune the GA parameters and in a way hybridizes the exhaustive search, GA and conjugate gradient to create an approach where a simple GA can be used to get global optima. Since computer power continues to increase at a high rate due to advances in electronics, the tradeoff between computer time and human time tends to value human time more. Hence AHGA can be considered to be more efficient. In addition, the use of banks of GA makes the algorithm suitable for parallel implementation. Also, unlike a typical GA, the use of a gradient based method in the final part assures the selection of one final design and removes the ambiguity suffered by GA due to lack of convergence criteria. In the few cases where the AHGA does not give a global minima, a local minima is assured due to the gradient based method. This lack of a deterministic output has inhibited the use of GA in engineering design optimization in the industry and can be addressed using HGA.

4. Conclusions

A hybrid genetic algorithm (HGA) is developed where a bank of $n$ GA’s are used for providing an initial design for the conjugate gradient method. This design combines the best characteristics of GA and LOA like other post-hybridized algorithms and seems to exhibit better global optimizing properties than a GA. It also transfers the need of tuning the GA parameters to the selection of $n$ and hence saving considerable computational effort and human time on the part of the user.

Later, the selection of the value $n$ is also automated via developing an AHGA. Such a selection of $n$ is possible by a design of a suitable criteria for transferring the search methodology from GA to LOA, which has been one of the weaknesses of the post-hybridized algorithms. The number of GA banks, $n$, is selected using an automated procedure based on Fibonacci numbers. AHGA is tested against a set of test functions and shown to converge to global optimality with near 100% success rate. This method reduces the need to tailor various GA parameters and also the ambiguity in the final optimal point given by GA due to the lack of a convergence criteria. In the few cases where the AHGA does not lead to a global minima, a local minima is assured due to the use of the gradient search to the end. The problem of finding a suitable starting design for the gradient based methods is also addressed. It is also easy to implement using a simple GA with elitist strategy and the CG method.
Appendix A

The test functions that are chosen are:

1. **Freudenstien and Roth function** [9]

   \[ f(x,y) = (-13 + x + ((5 - y)y - 2)y)^2 + (-29 + x + ((y + 1)y - 14)y)^2. \]

   It is a typical test problem for a gradient based search method, with steep walls and a very shallow valley, testing the gradient based convergence characteristics of the algorithm. It has a global minima at \((4,5)\), with \(f_0 = 0\), adjacent to a local minima at \((11.41,-0.89)\), with \(f = 48.98\), in the valley. This function is shown in Fig. 4.

2. **Evolutionary test function**

   \[ f(x,y) = \sum_{i=1}^{m-1} \left( \frac{1}{1 + 500e^{-d_{ix}}} \right)^2 + 3.0 \left( \frac{1}{1 + 500e^{-d_{mx}}} \right)^2. \]

   Here \(d_i\) is the distance of the point \((x,y)\) from the \(i\)th minima, \(d_m\) being the distance from the global minima and ‘\(m\)’ is the number of minima present. The function chosen for test simulation has local minima at \((90,23)\), \((-53,58)\), \((21,84)\), \((40,-70)\), \((-80,60)\) and the global minima at \((-2,27)\). It is a typical

![Fig. 4. Freudenstien and Roth function.](image-url)
test function for a heuristic based algorithm with many minima in steep valleys. It is shown in Fig. 5. It can be observed from the figure that it is a very difficult function for any gradient based method, which are likely to get into one of the local minima depending on the starting design. At the same time it is a good test for the global optimizing properties of any heuristic based algorithm.

3. Modified Himmenblau function [3]

\[ f(x, y) = (x^2 + y - 11)^2 + (x + y^2 - 7)^2 + 0.1((x - 3)^2 + (y - 2)^2) \]

is a two dimensional function having three local minima at \((-2.805, 3.131)\), \((-3.779, -3.283)\), and \((3.584, 1.848)\) with the respective function values being 3.498, 7.386, 1.515, and a global optima at \((3,2)\) where \(f = 0\). This function is shown in Fig. 6. Unlike the above two test functions, it is a more realistic and moderately hard function. Again, gradient based methods are likely to get stuck in the local minima depending on the starting design, but the function is relatively easy for the GA.

4. Hartmann function [8]

\[ H_n(x) = - \sum_{i=1}^{5} C_i \text{exp} \left[ -\sum_{j=1}^{6} a_{ij} (x_{ij} - P_{ij})^2 \right] \]
is an $n$ dimensional function containing four local minima and a global minima. The function value at the $i$th extrema being $f_i = C_i$. Where, $C = [20\ 3\ 9\ 14\ 12]$, $a_{ij} = 0.001$ $\forall i = 1, \ldots, 5$, $j = 1, \ldots, n$ and

$$P = \begin{bmatrix}
\end{bmatrix}.$$  

In contrast with the previously discussed two dimensional functions, it is a multidimensional test function with multiple minima, testing the global optimizing characteristics in a larger search space.

5. Rosenbrock’s function [24]

$$R_n(x_1, x_2, \ldots, x_n) = \sum^{n-1}_{i=1} [100(x_{i+1} - x_i^2)^2 + (1 - x_i)^2], \quad -5 \leq x_i \leq 10.$$  

It is a standard test function having a global minima at $(1, 1, \ldots, 1)$ and several local minima. It is very hard for traditional evolutionary algorithms to obtain the global minimum of Rosenbrock’s function. It was pointed out in [25] that
even the distributed genetic algorithm (DGA) with local populations cannot find the global minimum of Rosenbrock’s function. A two dimensional plot of the function is also shown in Fig. 7. The $R_2$ and $R_5$ functions are used in this study.

6. Shekel function \[22\]

\[
S_n(x_1, x_2, \ldots, x_n) = -\sum_{i=1}^{5} \left[ \left( \sum_{j=1}^{n} (x_j - a_{ij})^2 \right) + c_i \right]^{-1}, \quad 0 \leq x_i \leq 10.
\]

where

\[
[a_{ij}] = \begin{bmatrix}
4 & 4 & 4 & 4 & 4 \\
1 & 1 & 1 & 1 & 1 \\
6 & 6 & 6 & 6 & 6 \\
2 & 9 & 2 & 9 & 4 \\
5 & 5 & 3 & 3 & 9
\end{bmatrix}
\quad \text{and} \quad
[c_i] = \begin{bmatrix}
0.1 \\
0.2 \\
0.4 \\
0.6 \\
0.7
\end{bmatrix}.
\]

Fig. 7. Rosenbrock’s function ($R_2$).
It has one global minima at (4,4,4,4,4) and four local minima at (1,1,1,1,1), (6,6,6,6,6), (2,9,2,9,4) and (5,5,3,3,9) in the five dimensional version, whereas it has a global minima at (4,4) and four local minima at (1,1), (6,6), (2,9) and (5,5) in the two dimensional version. It is typical of an evolutionary test function containing several minima in steep valleys on a flat terrain. A two dimensional plot of the above function is shown in Fig. 8.

7. Zakharov’s function [24]

$$Z_n(x_1, x_2, \ldots, x_n) = \left( \sum_{j=1}^{n} x_j^2 \right) + \left( \sum_{j=1}^{n} 0.5jx_j \right)^2 + \left( \sum_{j=1}^{n} 0.5jx_j \right)^4,$$

$$-5 \leq x_j \leq 10.$$

Zakharov’s function is often used as a benchmark function for testing the performance of optimization algorithms. It has one global minima at (0,0,\ldots,0) and several local minima. The two dimensional version of the above function is plotted in Fig. 9. The $Z_2$, $Z_5$, $Z_{10}$ and $Z_{15}$ functions are used in this study.
References