Taboo evolutionary programming: a new method of global optimization

BY MINGJUN JI AND JACEK KLINOWSKI *

Department of Chemistry, University of Cambridge, Lensfield Road, Cambridge CB2 1EW, UK

We introduce taboo evolutionary programming, a very efficient global optimization method which combines features of single-point mutation evolutionary programming (SPMEP) and taboo search. As demonstrated by solving 18 benchmark problems, the algorithm is not trapped in local minima and quickly approaches the global minimum. The results are superior to those from SPMEP, fast evolutionary programming and generalized evolutionary programming. The method is easily applicable to real-world problems, and the central idea may be introduced into other algorithms.

Keywords: single-point mutation evolutionary programming; fast evolutionary programming; generalized evolutionary programming; taboo search

1. Introduction

Many problems in virtually all fields of physical science as well as in chemistry, technology, economics, logistics, travel scheduling and in the design of microprocessor circuitry involve global optimization: the determination of the global minimum of a function of an arbitrary number of independent variables. In most cases of practical interest, global optimization is very difficult because of the omnipresence of local minima, the number of which tends to increase exponentially with the size of the problem. Conventional minimization techniques are time consuming and converge to whichever local minimum they first encounter.

Evolutionary programming is one of the number of general metaheuristics known as evolutionary algorithms (Fogel et al. 1966), which include genetic algorithms, evolutionary strategies, evolutionary programming and genetic programming (Eiben & Schoenauer 2002; Eiben & Smith 2003). Although evolutionary algorithms do not rely on gradient information, their global convergence is guaranteed (Fogel 1994).

Classical evolutionary programming relies solely on mutation (Fogel et al. 1966). Fogel (1992) and Bäck & Schwefel (1993) have shown that evolutionary programming with self-adaptive mutation usually performs well. Yao et al. (1999) put forward fast evolutionary programming (FEP) which replaces Gaussian mutation by Cauchy mutation. Lee & Yao (2001, 2004) used evolutionary algorithms with adaptive Lévy mutations and studied the relationship between the parameter \( \alpha \) in Lévy distribution and evolutionary programming, while

* Author for correspondence (jk18@cam.ac.uk).
Iwamatsu’s generalized evolutionary programming (GEP) is based on a Lévy-type distribution (Iwamatsu 2002). Tu & Lu (2004) significantly put forward a stochastic genetic algorithm, explored the search space region-by-region and obtained very good results. Ji et al. (2004) described single-point mutation evolutionary programming (SPMEP), in which only one component of each solution is mutated in each iteration.

While the results from these methods are satisfactory, the principal requirement from any global optimization algorithm is that it must be able to avoid entrapment in local minima and continue the search for the optimal solution whatever the initial conditions. It is not easy for evolutionary programming to fulfil this condition.

Taboo search (TS), introduced by Cvijović & Klinowski (1995) for continuous functions, avoids entrapment in local minima, gives the optimal final solution, and was superior to other methods of global optimization available at the time. It has solved optimization problems such as the structure of clusters (Wales & Doye 1997), molecular docking (Westhead et al. 1997) and conformational energy optimization of oligopeptides (Morales et al. 2000). In theory, under the specified conditions, TS converges to the global minima with probability one (Ji & Klinowski 2006). We describe taboo evolutionary programming (TEP), a new evolutionary algorithm, using improved search paths and taboo conditions, and compare its performance with that of SPMEP, FEP and GEP.

2. Taboo evolutionary programming

Consider the following continuous global optimization problem,

\[
(P) \begin{cases} 
\min f(x), \\
\text{s.t. } x \in \Omega,
\end{cases}
\]

where \( \Omega := \{x \in \mathbb{R}^n | a \leq x(j) \leq b, a, b \in \mathbb{R}, j = 1, 2, \ldots, n \} \). \( f \) is a real-valued continuous function defined on \( \Omega \).

In classical evolutionary programming (Fogel et al. 1966), each parent \((x_i, \eta_i), i = 1, \ldots, \mu\), creates a single offspring \((x'_i, \eta'_i)\), as follows:

\[
x'_i(j) = x_i(j) + \eta_i(j) N_j(0, 1), \quad \eta'_i(j) = \eta_i(j) \exp(\tau' N(0, 1) + \tau N_j(0, 1)),
\]

for \( j = 1, \ldots, n \),

where \( x_i(j), x'_i(j), \eta_i(j) \) and \( \eta'_i(j) \) denote the \( j \)th component of the vectors \( x_i, x'_i, \eta_i \) and \( \eta'_i \), respectively. \( N(0, 1) \) is a normally distributed one-dimensional random number with a mean of zero and a standard deviation of one. \( N_j(0, 1) \) indicates that the random number is generated anew for each value of \( j \). The factors \( \tau \) and \( \tau' \) are commonly set to \( (\sqrt{2/\pi})^{-1} \) and \( (\sqrt{2n})^{-1} \) (Bäck & Schwefel 1993; Yao et al. 1999). FEP replaces the Gaussian random number \( N_j(0, 1) \) by the Cauchy random \( \delta_j \). In GEP, \( N_j(0, 1) \) is replaced by the Lévy-type random \( \delta_j \). In SPMEP, each parent \((x_i, \eta_i), i = 1, \ldots, \mu\), creates a single offspring \((x'_i, \eta'_i)\) by \( x'_i(j) = x_i(j) + \eta_i N_j(0, 1), \eta'_i = \eta_i \exp(-\alpha) \), where \( j_i \) is randomly chosen from the set \( \{1, \ldots, n\} \). Despite improving the probability of convergence to global minima, these algorithms are still easily trapped when solving problems involving multimodal functions with many local minima.

\( \text{Proc. R. Soc. A} \) (2006)
(a) Taboo evolutionary programming for solving global optimization problems

The TEP algorithm combines the characteristics of SPMEP and TS, and proceeds as follows:

(1) Generate the initial population of $\mu$ individuals based on a uniform distribution, and set $k=1$. Each individual is taken as a vector $x_i, \forall i \in \{1, \ldots, \mu\}$.

(2) Evaluate the fitness score for each individual, $x_i, \forall i \in \{1, \ldots, \mu\}$, of the population on the objective function, $f(x_i)$.

(3) Each parent $x_i, i = 1, \ldots, \mu$, creates a single offspring $x'_i$ by

$$x'_i(j_i) = x_i(j_i) + \eta N_i(0, 1), \quad \eta := \eta \exp(-\alpha),$$

where $j_i$ is randomly chosen from the set $\{1, \ldots, n\}$. Other components of $x'_i$ are equal to the corresponding $x_i$'s. $N_i(0, 1)$ denotes a normally distributed one-dimensional random number with a mean of zero and a standard deviation of one. Here, the parameter $\alpha = 1.01$. The initial value of $\eta$ is $(b-a)/2$. If $\eta < 10^{-4}$, then $\eta = (b-a)/2$.

(4) Calculate the fitness of each offspring $x'_i, \forall i \in \{1, \ldots, \mu\}$.

(5) Perform search using the following improved paths:

(5.1) Choose an improved path as follows. For each $x'_i$, if $f(x'_i) \leq f(x_i)$, then $y_i := x'_i, d_i := x'_i - x_i$ is an improved path. Put a pair of vectors $(y_i, d_i)$ into the set $A$.

(5.2) Choose $r$ best fitness individuals from the set $A$ as the parents with improved paths. Note that $(y_m, d_m), m = 1, \ldots, r$, where $y_m$ is an objective variable and $d_m$ is the corresponding improved path. Set $A := \emptyset$.

(5.3) Calculate fitness: for each $m = 1, \ldots, r$, $y'_m = y_m + \rho d_m, \rho := \rho \exp(-\alpha)$. The initial value of $\rho$ is 1. If $\rho < 10^{-6}$, then $\rho = 1$.

(5.4) For each $m = 1, \ldots, r$, if $f(y'_m) \leq f(y_m)$, then set $(y'_m, d_m)$ as a parent of the next generation with improved search paths, and put into the set $A$.

(5.5) Record the number, $\tau$, of members in set $A$.

(5.6) Choose the parents for the next generation.

(6.1) Perform a comparison over the union of parents $x_i$ and offspring $x'_i, \forall i \in \{1, \ldots, \mu\}$. For each individual, $q$ opponents are chosen uniformly at random from all the parents and offspring. For each comparison, if the individual’s fitness is equal to or greater than the opponent’s, it scores a ‘win’. Select the $\mu - \tau$ individuals out of $x_i$ and $x'_i, \forall i \in \{1, \ldots, \mu\}$, which have the most wins to be put into the set $B$.

(6.2) Make the individuals $x, x'$ and $y$ from sets $B$ and $A$ the parents of the next generation. Set $B := \emptyset$.

(7) Check the taboo status.

(7.1) Record the current optimal fitness, $f^*_k$, and the current optimal solution, $x^*_k$.

(7.2) When $k > L$, compare the optimal fitness of the current generation with the optimal fitness of the previous $L$ generations. Thus, if $|f^*_k - f^*_{k-L}| \leq \sigma_1$, then $f^* := f^*_k, x^* := x^*_k$. Put the pair of the vectors $(f^*, x^*)$ into the taboo table $\Psi$. The length of taboo table $\Psi$ is $l$.

(7.3) For any $(f^*, x^*)$ in $\Psi$, if the current optimal fitness $f^*_k$ and the optimal solution $x^*_k$ satisfies the taboo conditions, $|f^*_k - f^*| \leq \sigma_1$ and $\|x^*_k - x^*\| \leq \sigma_2$, then generate the initial population of $\mu$ individuals and set new individuals as the $k$th generation.
(8) Terminate if the halting criterion is satisfied. Otherwise, set $k = k + 1$ and go to Step (3).

(b) The features of taboo evolutionary programming

(i) TEP introduces improved search paths in Step (5), and more quickly approaches the local or global minima. The taboo condition introduced in Step (7) increases the probability that the algorithm avoids entrapment in local minima.

(ii) TEP uses the same number of calculations as FEP, GEP and SPMEP. But, as a result of the No-Free-Lunch theorem (Wolpert & Macready 1997), TEP requires more CPU time because it generates $r$ solutions with improved paths in Step (5).

3. Computational results

We examined the performance of TEP by applying it to 18 benchmark functions used by Schwefel (1995), Yao et al. (1999) and Iwamatsu (2002) (see appendix A), a larger set of test functions than that considered by other empirical studies. The set includes unimodal, multimodal, low- and high-dimensional functions.

Functions $f_1 - f_{11}$ are high-dimensional and functions $f_1 - f_3$ are unimodal. Function $f_4$ is a discontinuous step function with a single minimum. Function $f_5$ is a noisy quartic function, where random$[0, 1)$ is a uniformly distributed random variable in $[0, 1)$. Multimodal functions $f_6 - f_{11}$ with the number of local minima increasing exponentially with the dimension of the problem appear to be the most difficult test for optimization algorithms. Functions $f_{12} - f_{18}$ are low-dimensional with only a few local minima. The final results for multimodal functions $f_6 - f_{11}$ are important since they reflect the ability of the algorithm to escape from poor local optima.

Table 1 compares our results with those given by Yao et al. (1999) and Ji et al. (2004), where the initial population $\mu = 90$ and the number of opponents $q = 10$. The halting criterion is that the maximum number of generations, $N$, is reached. The initial populations are generated uniformly at random in the domain. The number of individuals with improved search paths, $r$, is 10, which gives the same number of function evaluations as that used by Yao et al. (1999) and Ji et al. (2004). The error $\sigma_1$ varies between $10^{-6}$ and $10^{-9}$. $\sigma_2 = 1$. $L = 100$ and 50 for high- and low-dimensional functions, respectively. The length of the taboo list $l$ is chosen to be between 1 and 6.

Table 2 compares our results with those of Iwamatsu (2002) and Ji et al. (2004), where the dimensions $n$ of functions $f_1, f_7, f_8$ and $f_9$ are fixed at 10 and the dimension $n$ of function $f_{10}$ is 4. The initial population $\mu = 40$. Iwamatsu (2002) and Ji et al. (2004) used the same number of function evaluations. Other conditions are the same as those given in table 1.

Table 1 shows that the performance of TEP is better than that of SPMEP and FEP for functions $f_1 - f_3$, and is similar for function $f_4$. Function $f_5$ is perturbed by the random variable random$[0, 1)$. When random$[0, 1) = 0$, the global optimum of $f_5$ is zero, and thus the algorithms are affected by the random variable. For function $f_5$, the performance of FEP is better than that of TEP and
Table 1. Comparison of the performance of TEP, SPMEP and FEP with functions \( f_1 \) – \( f_{18} \). All results have been averaged over 50 runs. ‘Mean best’ indicates mean best objective values found in the last generation, and ‘s.d.’ stands for standard deviation. The value of \( t \) for a two-tailed \( t \)-test measures the level of significance of the comparisons.

<table>
<thead>
<tr>
<th>( F )</th>
<th>( N )</th>
<th>TEP Mean best</th>
<th>s.d.</th>
<th>SPMEP Mean best</th>
<th>s.d.</th>
<th>FEP Mean best</th>
<th>s.d.</th>
<th>TEP–SPMEP ( t )-test</th>
<th>TEP–FEP ( t )-test</th>
</tr>
</thead>
<tbody>
<tr>
<td>( f_1 )</td>
<td>1500</td>
<td>5.6 \times 10^{-7}</td>
<td>2.4 \times 10^{-7}</td>
<td>1.3 \times 10^{-4}</td>
<td>1.0 \times 10^{-4}</td>
<td>5.7 \times 10^{-4}</td>
<td>1.3 \times 10^{-4}</td>
<td>-9.153</td>
<td>-30.97</td>
</tr>
<tr>
<td>( f_2 )</td>
<td>2000</td>
<td>9.9 \times 10^{-5}</td>
<td>2.7 \times 10^{-5}</td>
<td>5.1 \times 10^{-4}</td>
<td>1.5 \times 10^{-5}</td>
<td>8.1 \times 10^{-3}</td>
<td>7.7 \times 10^{-4}</td>
<td>-94.09</td>
<td>-73.43</td>
</tr>
<tr>
<td>( f_3 )</td>
<td>5000</td>
<td>1.7 \times 10^{-4}</td>
<td>4.9 \times 10^{-5}</td>
<td>6.5 \times 10^{-3}</td>
<td>2.0 \times 10^{-3}</td>
<td>0.3</td>
<td>0.5</td>
<td>-22.37</td>
<td>-4.240</td>
</tr>
<tr>
<td>( f_4 )</td>
<td>1500</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>( f_5 )</td>
<td>3000</td>
<td>2.8 \times 10^{-2}</td>
<td>6.9 \times 10^{-3}</td>
<td>1.0 \times 10^{-2}</td>
<td>1.5 \times 10^{-3}</td>
<td>7.6 \times 10^{-3}</td>
<td>2.6 \times 10^{-3}</td>
<td>18.03</td>
<td>19.56</td>
</tr>
<tr>
<td>( f_6 )</td>
<td>9000</td>
<td>-12 569.5</td>
<td>3.5 \times 10^{-8}</td>
<td>-12 569.5</td>
<td>9.1 \times 10^{-12}</td>
<td>-12 554.5</td>
<td>52.6</td>
<td>0</td>
<td>-2.017</td>
</tr>
<tr>
<td>( f_7 )</td>
<td>5000</td>
<td>7.7 \times 10^{-8}</td>
<td>3.0 \times 10^{-8}</td>
<td>2.9 \times 10^{-7}</td>
<td>6.0 \times 10^{-8}</td>
<td>4.6 \times 10^{-2}</td>
<td>1.2 \times 10^{-2}</td>
<td>-22.45</td>
<td>-27.11</td>
</tr>
<tr>
<td>( f_8 )</td>
<td>1500</td>
<td>2.2 \times 10^{-4}</td>
<td>7.1 \times 10^{-5}</td>
<td>1.9 \times 10^{-3}</td>
<td>4.4 \times 10^{-4}</td>
<td>1.8 \times 10^{-2}</td>
<td>2.1 \times 10^{-3}</td>
<td>-26.65</td>
<td>-59.83</td>
</tr>
<tr>
<td>( f_9 )</td>
<td>2000</td>
<td>6.2 \times 10^{-3}</td>
<td>1.1 \times 10^{-2}</td>
<td>5.6 \times 10^{-3}</td>
<td>1.7 \times 10^{-3}</td>
<td>1.6 \times 10^{-2}</td>
<td>2.2 \times 10^{-2}</td>
<td>0.381</td>
<td>-2.817</td>
</tr>
<tr>
<td>( f_{10} )</td>
<td>1500</td>
<td>5.4 \times 10^{-9}</td>
<td>5.8 \times 10^{-11}</td>
<td>8.5 \times 10^{-7}</td>
<td>9.7 \times 10^{-9}</td>
<td>9.2 \times 10^{-6}</td>
<td>3.6 \times 10^{-6}</td>
<td>-615.7</td>
<td>-18.06</td>
</tr>
<tr>
<td>( f_{11} )</td>
<td>1500</td>
<td>1.6 \times 10^{-7}</td>
<td>7.4 \times 10^{-8}</td>
<td>1.4 \times 10^{-5}</td>
<td>9.2 \times 10^{-6}</td>
<td>1.6 \times 10^{-4}</td>
<td>7.3 \times 10^{-5}</td>
<td>-10.64</td>
<td>-15.48</td>
</tr>
<tr>
<td>( f_{12} )</td>
<td>100</td>
<td>0.998</td>
<td>1.0 \times 10^{-15}</td>
<td>1.00</td>
<td>1.6 \times 10^{-15}</td>
<td>1.22</td>
<td>0.56</td>
<td>-7.495</td>
<td>-2.803</td>
</tr>
<tr>
<td>( f_{13} )</td>
<td>4000</td>
<td>3.9 \times 10^{-4}</td>
<td>3.4 \times 10^{-5}</td>
<td>4.5 \times 10^{-4}</td>
<td>1.5 \times 10^{-4}</td>
<td>5.0 \times 10^{-4}</td>
<td>3.2 \times 10^{-4}</td>
<td>-2.759</td>
<td>-2.417</td>
</tr>
<tr>
<td>( f_{14} )</td>
<td>100</td>
<td>-1.03</td>
<td>2.5 \times 10^{-10}</td>
<td>-1.03</td>
<td>4.3 \times 10^{-10}</td>
<td>-1.03</td>
<td>4.9 \times 10^{-7}</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>( f_{15} )</td>
<td>100</td>
<td>0.398</td>
<td>1.1 \times 10^{-11}</td>
<td>0.398</td>
<td>5.7 \times 10^{-10}</td>
<td>0.398</td>
<td>1.5 \times 10^{-7}</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>( f_{16} )</td>
<td>100</td>
<td>3</td>
<td>6.1 \times 10^{-9}</td>
<td>3.00</td>
<td>5.6 \times 10^{-9}</td>
<td>3.02</td>
<td>0.11</td>
<td>0</td>
<td>-1.286</td>
</tr>
<tr>
<td>( f_{17} )</td>
<td>100</td>
<td>-3.86</td>
<td>1.2 \times 10^{-9}</td>
<td>-3.86</td>
<td>1.5 \times 10^{-9}</td>
<td>-3.86</td>
<td>1.4 \times 10^{-5}</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>( f_{18} )</td>
<td>200</td>
<td>-3.28</td>
<td>2.1 \times 10^{-2}</td>
<td>-3.25</td>
<td>5.4 \times 10^{-2}</td>
<td>-3.27</td>
<td>5.9 \times 10^{-2}</td>
<td>-3.661</td>
<td>-1.129</td>
</tr>
</tbody>
</table>
SPMEP. For functions $f_{6} - f_{11}$, TEP is the best of the three algorithms. Its precision is higher than that of SPMEP and $10^{-10}$ times better than that of FEP. For low-dimensional functions $f_{12} - f_{18}$, TEP is not obviously superior, and the three algorithms have the same precision. However, the standard deviations of TEP are better, showing that TEP is very robust.

Table 2 shows that with TEP the precision for functions $f_{7}, f_{8}$ and $f_{9}$ is $10^{-9}$-fold better than with GEP and also better than with SPMEP. For high-dimensional and multimodal functions, TEP is clearly superior to GEP and SPMEP. The standard deviations for TEP are smaller than for GEP and SPMEP, which indicates that the best values in all 50 runs are very close to the mean best objective values. TEP is therefore the most robust of the four algorithms.

Figures 1–3 show the status of TEP for functions $f_{1} - f_{18}$. Since we recorded the current best objective value for a large number of iterations (see Table 1), peaks caused by re-initializing the taboo condition are not visible. TEP increases the probability of escape from local minima by using the taboo conditions. Figures 1–3 show that all 50 runs have very high precision and converge to the global minima. TEP comes close to the global minimum within 1000 iterations for high-dimensional functions $f_{1} - f_{11}$, and within 50 iterations for low-dimensional functions $f_{12} - f_{18}$.

### 4. Analysis of taboo evolutionary programming

Consider the reasons why TEP obtains better results than SPMEP and FEP. The three algorithms use different mutation strategies. While TEP adopts $x = x + \eta \times N(0,1)$ and $x = x + \rho \times d$, SPMEP uses $x = x + \eta \times N(0,1)$ and FEP uses $x = x + \eta \times \delta$, where $N(0,1)$ and $\delta$ are the Gaussian and Cauchy random variables, respectively. Figure 4 shows that the spread of points obtained by the Cauchy distribution is wider than the spread of points obtained by the Gaussian distribution. The large variation of the Cauchy points signifies the ease of escape from local minima while, conversely, the small variation of the Gaussian points means that it is more difficult to escape, but benefits the search of local regions.
In addition, because TEP uses improved paths $x = x + \rho \times d$ in Step (5), for unimodal function $f_1 - f_5$ TEP gives better results than SPMEP and FEP.

Although TEP increases the probability of the search becoming locally entrapped (because Gaussian distribution is being used), the taboo condition

Figure 1. Performance of TEP with test functions $f_1 - f_6$. The thick solid line, the thick dotted line and the thin solid line indicate the runs with the best objective value, the worst value and the mean value, respectively.
introduced in Step (7) increases the probability of escape. Step (7.2) places the current best value on the taboo list when the current best value is not improved compared with the best value of the previous $L$ generations. The current best value then becomes entrapped in a local minimum. In order to escape from it, we

Figure 2. Performance of TEP with test functions $f_7 - f_{12}$. The thick solid line, the thick dotted line and the thin solid line indicate the runs with the best objective value, the worst value and the mean value, respectively.
use the taboo condition in Step (7.3). As a result of this condition, the current population is replaced by a new initial population when the current best value and the optimal solution is entrapped in the vicinity of a local minimum. This increases the probability of escaping. In this way, TEP obtains good results for multimodal functions $f_6 - f_{18}$.

Figure 3. Performance of TEP with test functions $f_{13}$ and $f_{18}$. The thick solid line, the thick dotted line and the thin solid line indicate the runs with the best objective value, the worst value and the mean value, respectively.
5. Conclusions

Our algorithm is superior to the other three for solving multimodal and high-dimensional functions. Since, unlike evolutionary programming, TEP only introduces better search paths and taboo conditions, it can easily be applied to real-world problems. The central idea may be introduced into other evolutionary algorithms. The TEP algorithm will be freely available on J.K.’s website (www-klinowski.ch.cam.ac.uk).

We are grateful to the Royal Society of London for an International fellowship to M.-J.J., and to the referees for their constructive comments.

Appendix A

(A) Sphere model

\[ f_1(x) = \sum_{i=1}^{30} x_i^2, \quad -100 \leq x_i \leq 100, \]

\[ \min f_1 = f_1(0, \ldots, 0) = 0. \]

(B) Schwefel’s problem (2.22)

\[ f_2(x) = \sum_{i=1}^{30} |x_i| + \prod_{i=1}^{30} |x_i|, \quad -10 \leq x_i \leq 10, \]

\[ \min f_2 = f_2(0, \ldots, 0) = 0. \]
(C) **Schwefel’s problem (2.21)**
\[ f_3(x) = \max_i \{|x_i|, 1 \leq i \leq 30\}, \quad -100 \leq x_i \leq 100, \]
\[ \min f_3 = f_3(0, \ldots, 0) = 0. \]

(D) **Step function**
\[ f_4(x) = \sum_{i=1}^{30} \left( \left\lfloor x_i + 0.5 \right\rfloor \right)^2, \quad -100 \leq x_i \leq 100, \]
\[ \min f_4 = f_4(0, \ldots, 0) = 0. \]

(E) **Quartic function, i.e. noise**
\[ f_5(x) = \sum_{i=1}^{30} ix_i^4 + \text{random}(0, 1), \quad -1.28 \leq x_i \leq 1.28, \]
\[ \min f_5 = f_5(0, \ldots, 0) = 0. \]

(F) **Generalized Schwefel’s problem (2.26)**
\[ f_6(x) = -\sum_{i=1}^{30} \left( x_i \sin \sqrt{|x_i|} \right), \quad -500 \leq x_i \leq 500, \]
\[ \min f_6 = f_6(420.9687, \ldots, 420.9687) = -12\,569.5. \]

(G) **Generalized Rastrigin’s function**
\[ f_7(x) = \sum_{i=1}^{30} \left[ x_i^2 - 10 \cos (2\pi x_i) + 10 \right], \quad -5.12 \leq x_i \leq 5.12, \]
\[ \min f_7 = f_7(0, \ldots, 0) = 0. \]

(H) **Ackley’s function**
\[ f_8(x) = -20 \exp \left( -0.2 \sqrt{\frac{1}{30} \sum_{i=1}^{30} x_i^2} \right) - \exp \left( \frac{1}{30} \sum_{i=1}^{30} \cos (2 \pi x_i) \right) + 20 + e, \]
\[ -32 \leq x_i \leq 32, \quad \min f_8 = f_8(0, \ldots, 0) = 0. \]

(I) **Generalized Griewank function**
\[ f_9(x) = \frac{1}{4000} \sum_{i=1}^{30} x_i^2 - \prod_{i=1}^{30} \cos \left( \frac{x_i}{\sqrt{i}} \right) + 1, \quad -600 \leq x_i \leq 600, \]
\[ \min f_9 = f_9(0, \ldots, 0) = 0. \]

(J) **Generalized penalized functions**
\[ f_{10}(x) = \frac{\pi}{30} \left\{ 10 \sin^2(\pi y_1) + \sum_{i=1}^{29} (y_i - 1)^2 [1 + 10 \sin^2(\pi y_{i+1})] + (y_n - 1)^2 \right\} \]
\[ + \sum_{i=1}^{30} u(x_i, 10, 100, 4), \]
\[ -50 \leq x_i \leq 50, \quad \min f_{10} = f_{10}(1, \ldots, 1) = 0, \]

**Taboo evolutionary programming**

\[ f_{11}(x) = 0.1 \left\{ \sin^2(3\pi x_1) + \sum_{i=1}^{29} (x_i - 1)^2[1 + \sin^2(3\pi x_{i+1})] + (x_n - 1)^2[1 + \sin^2(2\pi x_{30})] \right\} + \sum_{i=1}^{30} u(x_i,5,100,4), \]
\[ -50 \leq x_i \leq 50, \min f_{11} = f_{11}(1,\ldots,1) = 0, \]

where

\[ u(x_i, a, k, m) = \begin{cases} 
  k(x_i - a)^m & x_i > a, \\
  0 & -a \leq x_i \leq a, \\
  k(-x_i - a)^m & x_i < -a. 
\end{cases} \]

\[ y_i = 1 + \frac{1}{4}(x_i - 1). \]

(K) Shekel’s foxholes function

\[ f_{12}(x) = \left[ \frac{1}{500} + \sum_{j=1}^{25} \frac{1}{j + \sum_{i=1}^{2}(x_i - a_{ij})^6} \right]^{-1}, \quad -65.536 \leq x_i \leq 65.536, \]
\[ \min f_{12} = f_{12}(-32,-32) \approx 1, \]

where

\[ a_{ij} = \begin{pmatrix} 32 & -16 & 0 & 16 & 32 & -32 & \cdots & 0 & 16 & 32 \\ -32 & -32 & -32 & -32 & -32 & -16 & \cdots & 32 & 32 & 32 \end{pmatrix}. \]

(L) Kowalik’s function

\[ f_{13}(x) = \sum_{i=1}^{11} \left[ a_i - \frac{x_1(b_i^2 + b_i x_2)}{b_i^2 + b_i x_3 + x_4} \right]^2, \quad -5 \leq x_i \leq 5, \]
\[ \min f_{13} \approx f_{13}(0.1928, 0.1908, 0.1231, 0.1358) \approx 0.0003075, \]

\[ a = (0.1957, 0.1947, 0.1735, 0.1600, 0.0844, 0.0627, 0.0456, 0.0342, 0.0323, 0.0235, 0.0246), \]
\[ b = (4, 2, 1, 0.5, 0.25, 1/6, 0.125, 0.1, 1/12, 1/14, 0.0625). \]

(M) Six-hump camel-back function

\[ f_{14}(x) = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4, \quad -5 \leq x_i \leq 5, \]
\[ x_{\min} = (0.08983, -0.7126), (-0.08983, 0.7126), \quad \min f_{14} = -1.0316285. \]
(N) Branin function

\[ f_{15}(x) = \left( x_2 - \frac{5.1}{4\pi^2} x_1^2 + \frac{5}{\pi} x_1 - 6 \right)^2 + 10 \left( 1 - \frac{1}{8\pi} \cos x_1 \right) + 10, \]

\[-5 \leq x_1 \leq 10, \quad 0 \leq x_2 \leq 15,\]

\[ x_{\text{min}} = (-3.14212.275), (3.142, 2.275), (9.425, 2.425), \quad \min f_{15} = 0.398. \]

(O) Goldstein–Price function

\[ f_{16}(x) = [1 + (x_1 + x_2 + 1)^2(19 - 14x_1 + 3x_1^2 + 6x_1x_2 + 3x_2^2)] \times [30 + (2x_1 - 3x_2)^2] \]
\[ \times (18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)], \quad -2 < x_i < 2, \quad i = 1, 2, \]

\[ \min f_{16} = f_{16}(0,-1) = 3. \]

(P) Hartman’s family of functions

\[ f_{17,18}(x) = \sum_{i=1}^{4} c_i \exp \left[ -\sum_{j=1}^{n} \alpha_{ij} (x_j - p_{ij})^2 \right], \]

\[ 0 \leq x_i \leq 1, \quad i = 1, 2, \ldots, n, \]

with \( n = 3, 6 \) for \( f_{17}(x) \) and \( f_{18}(x) \), respectively. For \( n = 3 \), the global minimum is equal to \(-3.86\) and it is reached at the point \((0.114, 0.556, 0.882)\). For \( n = 6 \), the minimum is \(-3.32\) at the point \((0.201, 0.150, 0.477, 0.275, 0.311, 0.657)\).

\[
\begin{array}{cccccc}
\hline
i & \alpha_{i1} & \alpha_{i2} & \alpha_{i3} & c_i & p_{i1} & p_{i2} \\
\hline
1 & 3 & 10 & 30 & 1 & 0.3689 & 0.1170 & 0.2673 \\
2 & 0.1 & 10 & 35 & 1.2 & 0.4699 & 0.4387 & 0.7470 \\
3 & 3 & 10 & 30 & 3 & 0.1091 & 0.8742 & 0.5547 \\
4 & 0.1 & 10 & 35 & 3.2 & 0.03815 & 0.5743 & 0.8828 \\
\hline
\end{array}
\]

\[
\begin{array}{cccccc}
\hline
i & \alpha_{i1} & \alpha_{i2} & \alpha_{i3} & \alpha_{i4} & \alpha_{i5} & \alpha_{i6} & c_i \\
\hline
1 & 10 & 3 & 17 & 3.5 & 1.7 & 8 & 1 \\
2 & 0.05 & 10 & 17 & 0.1 & 8 & 14 & 1.2 \\
3 & 3 & 3.5 & 1.7 & 10 & 17 & 8 & 3 \\
4 & 17 & 8 & 0.05 & 10 & 0.1 & 14 & 3.2 \\
\hline
\end{array}
\]

\[
\begin{array}{cccccc}
\hline
i & p_{i1} & p_{i2} & p_{i3} & p_{i4} & p_{i5} & p_{i6} \\
\hline
1 & 0.1312 & 0.1696 & 0.5569 & 0.0124 & 0.8283 & 0.5886 \\
2 & 0.2329 & 0.4135 & 0.8307 & 0.3736 & 0.1004 & 0.9991 \\
3 & 0.2348 & 0.1451 & 0.3522 & 0.2883 & 0.3047 & 0.6650 \\
4 & 0.4047 & 0.8828 & 0.8732 & 0.5743 & 0.1091 & 0.0381 \\
\hline
\end{array}
\]

(Q) Shekel’s function

\[ f_{19}(x) = -\sum_{i=1}^{m} [(x - a_i)(x - a_i)^T + c_i]^{-1}, \]

with \( m = 5, \ 0 \leq x_j \leq 10 \). This function has five local minima. \( x_{\text{local opt}} \approx a_i, f(x_{\text{local opt}}) \approx 1/c_i \) for \( 1 \leq i \leq m \). The coefficients are defined in Table 3.

Table 3. Shekel function \( f_{19} \).

<table>
<thead>
<tr>
<th>( i )</th>
<th>( a_{ij}, j = 1, \ldots, 4 )</th>
<th>( c_i )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>4</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>5</td>
<td>3</td>
<td>7</td>
</tr>
</tbody>
</table>

References


Taboo evolutionary programming


