A mini-benchmark

You have designed a nice brand new algorithm. Before to run it on a large (unbiased) benchmark, you may try it on this mini-benchmark of four problems. It has been carefully chosen to be deceptive, in average, if the algorithm is not well “balanced” or is biased in favour of a diagonal of the search space.

Tripod

The function to be minimised is ( [2])

\[
\begin{align*}
f &= \frac{1-\text{sign}(x_2)}{2} (|x_1| + |x_2 + 50|) \\
    &+ \frac{1+\text{sign}(x_2)\ 1-\text{sign}(x_1)}{2} (1 + |x_1 + 50| + |x_2 - 50|) \\
    &+ \frac{1+\text{sign}(x_1)}{2} (2 + |x_1 - 50| + |x_2 - 50|)
\end{align*}
\]

(1)

with

\[
\text{sign}(x) = \begin{cases} 
-1 & \text{if } x \leq 0 \\
1 & \text{else}
\end{cases}
\]

The search space is \([-100, 100]^2\). The solution point is \((0, -50)\), where \(f = 0\). Here, we allow \(10^4\) fitness evaluations and a run is said to be successful if it finds a fitness less than 0.0001.

Figure 1: Tripod function. Not that difficult, but may be deceptive for algorithms that are easily trapped into a local minimum
Rosenbrock F6

\[ f = 390 + \sum_{d=2}^{10} \left( 100 \left( z_{d-1}^2 - z_d \right)^2 + (z_{d-1} - 1)^2 \right) \]  

(2)

with \( z_d = x_d - o_d \). The search space is \([-100,100]^{10}\). The offset vector \( O = (o_1, \ldots, o_{10}) \) is defined by its C code below. The solution point is \( O + (1, \ldots, 1) \) where \( f = 390 \). There is also a local minimum at \((o_1 - 2, \ldots, o_{10})\), where \( f = 394 \). Here, we allow \( 10^5 \) fitness evaluations and a run is said to be successful if it finds a fitness less than 0.01. This problem is coming from the CEC 2005 benchmark and is difficult for algorithms that reduce too quickly the searched area space, and particularly discriminant in terms of success rate.

Offset (C code)

static double offset_2[10] = { 81.0232, -48.395, 19.2316, -2.5231, 70.4338, 47.1774, -7.8358, -86.6693, 57.8532, -9.9533}

Compression spring

This is a simplified version of a more difficult problem (see[4, 1, 3]). There are three variables

\[
\begin{align*}
  x_1 &\in \{1, \ldots, 70\} \quad \text{granularity} \ 1 \\
  x_2 &\in \ [0.6, 3] \\
  x_3 &\in \ [0.207, 0.5] \quad \text{granularity} \ 0.001
\end{align*}
\]

and four constraints

\[
\begin{align*}
  g_1 &:= \frac{8CF_{max}x_2}{\pi x_3^3} - S \leq 0 \\
  g_2 &:= l_f - l_{max} \leq 0 \\
  g_3 &:= \sigma_p - \sigma_{pm} \leq 0 \\
  g_4 &:= \sigma_w - \frac{F_{max} - F_p}{K} \leq 0
\end{align*}
\]

with

\[^1\text{The Rosenbrock function is indeed multimodal as soon the dimension is greater than three [5].}\]
\[ C_f = 1 + 0.75 \frac{x_3}{x_2 - x_3} + 0.615 \frac{x_1}{x_2} \]
\[ F_{\text{max}} = 1000 \]
\[ S = 189000 \]
\[ l_f = \frac{F_{\text{max}}}{K} + 1.05 (x_1 + 2) x_3 \]
\[ l_{\text{max}} = 14 \]
\[ \sigma_p = \frac{F_p}{K} \]
\[ \sigma_{pm} = 6 \]
\[ F_p = 300 \]
\[ K = 11.5 \times 10^6 \frac{x_3^4}{8x_1 x_2} \]
\[ \sigma_w = 1.25 \]

and the function to be minimised is

\[ f = \pi^2 x_2 x_3^2 (x_1 + 2) \frac{(x_1 + 2)}{4} \] (3)

The best known solution is \( (7, 1.386599591, 0.292) \), which gives the fitness value \( f^* = 2.6254214578 \). For the results given on the table 1, a penalty method has been used to take the constraints into account, but any method is of course acceptable. Here, we allow \( 2 \times 10^4 \) fitness evaluations and a run is said to be successful if it finds a fitness \( f \) so that \( |f - f^*| \leq 10^{-10} \). Because of the granularities this problem may be deceptive for some algorithms.

**Gear train**

For more details, see [4, 3]. The function to be minimised is

\[ f(x) = \left( \frac{1}{6.931} - \frac{x_1 x_2}{x_3 x_4} \right)^2 \] (4)

The search space is \( \{12, 13, \ldots, 60\}^4 \). There are several solutions, depending on the required precision. Here, we used \( 10^{-13} \). So, a possible solution is \( f^* = f(19, 16, 43, 49) = 2.7 \times 10^{-12} \). We allow \( 2 \times 10^4 \) fitness evaluations, and a run is then said successful if it finds a fitness \( f \) so that \( |f - f^*| \leq 10^{-13} \). In this problem, only “integer” positions are acceptable. A lot of algorithms are not comfortable with such constraints.

**Reasonable results**

Some results with two PSO variants are given in the table 1. We say that an algorithm A “beats” an algorithm B if the mean success rate of A is greater than the one of B. Two cases:

- your algorithm does not beat the algorithm 1, i.e. SPSO-2007 (Standard PSO 2007). Forget it.
Table 1: Success rates over 100 runs

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Function</th>
<th>1</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>SPSO-2007</td>
<td>50%</td>
<td>94%</td>
</tr>
<tr>
<td></td>
<td>Rosenbrock F6</td>
<td>82%</td>
<td>75%</td>
</tr>
<tr>
<td></td>
<td>Compression spring</td>
<td>35%</td>
<td>72%</td>
</tr>
<tr>
<td></td>
<td>Gear train</td>
<td>6%</td>
<td>22%</td>
</tr>
<tr>
<td></td>
<td>Mean</td>
<td>43.25%</td>
<td>65.5%</td>
</tr>
</tbody>
</table>

- it beats Algorithm 2. It is really promising. It is worth running it on a more complete benchmark. Be sure this benchmark is non biased (the solution point must not be on a diagonal of the search space)

References


