Solving Non-linear Continuous Mathematical Models using Shuffled Frog Leaping and Memetic Algorithms

Prapol Ittipong¹, Pongchanun Luangpaiboon² and Pupong Pongcharoen³*

Abstract

Metaheuristic may be defined as an iterative search process that intelligently performs the exploration and exploitation in the solution space aiming to efficiently find near optimal solutions. Various natural intelligences and inspirations have been adopted into the iterative process. In this work, two types of metaheuristics called Shuffled Frog Leaping (SFL) and Memetic Algorithm (MA) were adapted to find optimal solutions of eleven non-linear continuous mathematical models. Considering the solution space in a specified region, some models contain global optimum and multiple local optimums. A series of computational experiments using each method were conducted. Experimental results were analysed in terms of best solutions found so far, mean and standard deviation. It was found that the results obtained from Memetic Algorithm were better than those using Shuffled Frog Leaping. However, the average execution time of experimental run using MA was longer than those using SFL.

Keywords: Shuffled Frog Leaping, Memetic Algorithms, Metaheuristics, Optimisation, Non-linear

¹ Student, Department of Computer Science and Information Technology, Faculty of Science, Naresuan University
² Associate Professor, Department of Industrial Engineering, Faculty of Engineering, Thammasat University
³ Assistant Professor, Department of Industrial Engineering, Faculty of Engineering, Naresuan University
* Corresponding author’s email address: pupongp@yahoo.com; pupongp@nu.ac.th
1. Introduction

Optimisation algorithms can be categorised as being either conventional or approximation optimisation algorithms [1]. Conventional optimisation algorithms are usually based upon mathematical models such as Integer Linear Programming, Branch and Bound or Dynamic Programming. These approaches were relatively well developed and attributed to the military services early in World War II. Based on the full enumerative search within these approaches, the optimal solutions are always guaranteed. However, the application of these methods might need exponential computational time in the worst case. This becomes an impractical approach especially for solving a very large size problem. Alternative approaches that can guide the search process to find near optimal solutions in acceptable computational time are therefore more practical and desirable.

Approximation optimisation algorithms (called metaheuristics) have therefore received more attention in the last few decades. Metaheuristics iteratively conduct stochastic search process inspired by natural intelligence. They can be categorised into three groups: physically-based inspiration such as Simulated Annealing [2]; socially-based inspiration for instance Taboo Search [3]; and biologically-based inspiration e.g. Ant Colony Optimisation [4], Artificial Immune System [5], Genetic Algorithm [6], Memetic Algorithm [7], Neural Network [8], Particle Swarm Optimisation [9], and Shuffled Frog Leaping [10]. These alternative approaches have been widely used to solve large-scale combinatorial optimisation problems [11, 12, 13, 14].

Among the metaheuristics, Shuffled Frog Leaping (SFL) has been recently developed. The application of this method was rarely found especially applying to solve three-dimension non-linear continuous mathematical functions. The objective of this paper was to investigate the performance of Shuffled Frog Leaping and Memetic Algorithms to find optimal solutions of eleven non-linear continuous mathematical models.

This paper is organised as follows. Section 2 describes the proposed metaheuristics including Shuffled Frog Leaping, Memetic Algorithm and its pseudo code. Section 3 presents eleven testing functions, all of which are non-linear continuous mathematical functions. For each function, the optimal solution, the equation, considered range and its contour plot are provided. Section 4 presents the design and analysis of computational experiments for comparing the performance of the proposed methods. The conclusions are drawn in section 5 followed by acknowledgement and references.

2. Proposed Methods

Nature has always been a source of inspiration. Various types of nature-inspired algorithms have been developed during the last few decades. These algorithms iteratively conduct stochastic search process adopted from natural intelligence. In this work, two nature-inspired algorithms called Memetic Algorithm (MA) and Shuffled Frog Leaping (SFL) were proposed. The detail of these algorithms and its pseudo code are briefly presented in the subsection.

2.1 Memetic Algorithm (MA)

The name of the Memetic Algorithm (MA) is inspired by Dawkins’ notation of a meme. MA is similar to Genetic Algorithm (GA) but the elements that form a chromosome are called memes, not genes. The main concept of the MA is that each individual and offspring is allowed to gain some experience through a local search before being involved in the evolutionary process [15]. The pseudo code of the Memetic Algorithm is provided in Figure 1.

```
Pseudo code of the Memetic Algorithm (MA)
Begin;
    Generate random population of P solution (chromosomes);
    For i = 1 to number of generation;
        (Crossover)
        Select two parents at random i_a and i_b;
        Generate on offspring i_c = crossover (i_a and i_b);
        (Mutation)
        Select one chromosome i at random;
        Generate an offspring i_c = mutate (i);
        If hybridisation is true then
            For each individual i ∈ P: do local search (SFL);
        End if;
        For each individual i ∈ P: calculate fitness (i);
    Chromosome selection;
End for;
End;
```

Figure 1 Pseudo code of the Memetic Algorithm

From the pseudo code shown in Figure 1, it can be seen that the process of the MA is similar to the GA. The process begins with randomly generating an initial population of chromosomes (candidate solutions). Each chromosome is developed through the evolution process called crossover and mutation operations. The major different between MA and GA is that, in MA, a local search is performed on each population member to improve its
experience and thus obtain a population of local optimum solutions. Afterward, the process of chromosome selection is then applied in order to form a new population. The evolution process is repeated until a termination criterion is satisfied.

2.2 Shuffled Frog Leaping (SFL) algorithm

Shuffled Frog Leaping (SFL) algorithm is one of the biologically-based inspirations. In the SFL algorithm, a group of frogs (candidate solutions) is divided into subgroups (memeplexes), each of which has different cultures by performing a local search. Each frog has their own idea and can be influenced by the ideas of other frogs during the iterative shuffling process of memetic evolution [16]. The pseudo code of the SFL is provided in Figure 2.

![Pseudo code of the Shuffled Frog Leaping (SFL)](image)

The process of SFL begins by initialising a population of frogs, each of which has a fitness value. The frogs are then sorted in a descending order according to their fitness. The entire group can be divided into memeplexes, each of which consisting of frogs. In each memeplex, the position of frog \(i\) \((D_i)\) is adjusted according to the different between the frog with the worst fitness \(X_w\) and the frog with the best fitness \(X_b\) as shown in equation (1), where \(\text{rand}(\cdot)\) is a uniform random number between 0 and 1. The repositioning process shown in equation (2) is used to produce a new frog, where \(D_{\max}\) is the maximum allowed position change.

\[
\text{Position change (} D_i \text{)} = \text{rand}(\cdot) \times (X_b - X_w) \quad (1)
\]

\[
\text{New position} \ X_i = \text{current position} \ X_i + D_i; \quad D_{\max} \leq D_i \leq -D_{\max} \quad (2)
\]

If the repositioning process produces a frog with better fitness, it replaces the worst frog. Otherwise, the process is repeated with respect to the global best frog \(X_g\) with the best fitness across the memeplexes. In case of no improvement, a new frog is randomly generated to replace the worst frog. The evolution process is continued for a specific number of iterations.

3. Tested Functions

In this paper, eleven non-linear continuous mathematical functions were used to test the performance of the proposed methods for searching the optimal solutions. The functions including the equations and its contour plot are illustrated in the following subsections.

3.1 Todd Function

\[
f(x) = 4 \cdot \left(3 - \frac{x}{1.6}\right) \left(0.5 - \frac{x}{1.6}\right) \left(6.2 - \frac{x}{1.6}\right) \left(2 - \frac{x}{1.6}\right)\frac{4 - x}{6}
\]

Range between: 0 < x < 10

\[f(x^*) = 9.515504816, \text{ where } x^* = 8.29584\]

3.2 Branin Function

\[
f(x,y) = 6.0 \cdot \log_{10}(\left[y - 5.1 \cdot \frac{x}{4\pi^2} + 5 \cdot \frac{x}{\pi} - 6\right]^2 + (10.0 - 5.1 \cdot \cos(x) + 10)]
\]

Range between: -20 < x < 20; -20 < y < 20

\[f(x^*,y^*) = 5.150395778, \text{ where } x^* = 0 \text{ and } y^* = 6\]
3.3 Camelback Function

\[ f(x, y) = 10 - \log_{10} \left[ x^2 (4 - 2.1x^2 + \frac{1}{3} x^4) + xy + 4y^2 (y^2 + 1) \right] \]

Range between: -20<x<20; -20<y<20

\[ f(x^*, y^*) = \infty, \text{ where } x^* = 0 \text{ and } y^* = 0 \]

3.6 Parabolic Function

\[ f(x, y) = 12 - \frac{x^2 + y^2}{100} \]

Range between: -20<x<20; -20<y<20

\[ f(x^*, y^*) = 12, \text{ where } x^* = 0 \text{ and } y^* = 0 \]

3.4 Goldstein-price Function

\[ f(x, y) = 10 - \log_{10} \left[ 1 + (1 + x + y)^2 (19 - 14x + 3x^2 - 14y + 6xy + 3y^2) \right] \times \]
\[ (30 + (2x - 3y)^2 (18 - 32x + 12x^2 + 48y - 36xy + 27y^2)) \]

Range between: -20<x<20; -20<y<20

\[ f(x^*, y^*) = 9.52287874528034, \text{ where } x^* = 0 \text{ and } y^* = -1 \]

3.7 Rastrigin Function

\[ f(x, y) = 80 - [20 + x^2 + y^2 - 10 (\cos(2\pi x) + \cos(2\pi y))] \]

Range between: -5<x<5; -5<y<5

\[ f(x^*, y^*) = 80, \text{ where } x^* = 0 \text{ and } y^* = 0 \]

3.5 Montgomery’s Function

\[ f(x, y) = 1217.3 - 31.256x + 86.017y + 0.12917x^2 + 2.8733y^2 + 0.02875xy \]

Range between: 100<x<140; 10<y<20

\[ f(x^*, y^*) = -66.03402, \text{ where } x^* = 119.87494 \text{ and } y^* = 10 \]

3.8 Rosenbrock Function

\[ f(x, y) = 70\left[ 20 - \left( \frac{x^2}{10} + \frac{y^2}{5} + \left( \frac{x}{6} \right)^2 \right) \right] + 150 \]

\[ f(x^*, y^*) = 70\left[ 20 - \left( \frac{x^2}{10} + \frac{y^2}{5} + \left( \frac{x}{6} \right)^2 \right) \right] + 150 \]
3.11 Dejong Function (4 dimensions)

\[ f(x, y, z) = x^2 + y^2 + z^2 \]

Range between: \(-5.12 < x < 5.12; -5.12 < y < 5.12; -5.12 < z < 5.12\)

\(f(x^*, y^*, z^*) = 0\), where \(x^* = 0\) and \(y^* = 0\)

4. Experimental Design and Analysis

In this work, a computer simulation program was developed using Microsoft Visual Studio .NET 2003. A notebook computer with 1.8GHz AMD Turion 64 X2 Dual Core TL-52 processor and 512 MB DDR2 RAM was used for computational experiments. Eleven non-linear continuous mathematical functions were used to test the performance of the proposed methods. For each function, the computational run using each method was repeated 30 times using different random seed numbers. The experimental results obtained from each method including best-so-far (BSF) solutions and its error percentages (as shown in Table 1) were compared to the optimal solutions of all eleven testing functions described in the previous section. It should be noted that the optimal solutions of the Camelback and Dejong functions are infinity and zero, respectively. The error percentage (\%Error) for those functions can not be provided.

From Table 1, it can be seen that both Shuffled Frog Leaping (SFL) and Memetic Algorithm (MA) found the optimal solutions (0.00 %) only the first (Todd) function with single factor. For the function 2-10 (all with two factors), the best-so-far (BSF) solutions obtained from MA were dramatically better than those results obtained from SFL. For example, the BSF solution obtained from MA for the Branin function was deviated from the optimal solution by 0.000001437 %, which was far better than the BSF solution obtained from SFL with the percent error of 0.00094575. Another example on the Rastrigin function that contains multiple peaks in the range considered, the BSF solution obtained from MA for the Rastrigin function was deviated from the optimal solution by 0.0005 %, which was dramatically better than the BSF solution obtained from SFL with the percent error of 0.2831.

For the last (Dejong) function with three factors, the performance of MA was better than the SFL. However, the average execution time of the computational runs for all testing functions using MA was approximately 9.5 second whilst 0.25 second was averagely taken by SFL. Other words, the average execution time taken by SFL was 38 times quicker than the computational time required by the MA.

From the experimental results shown in Table 1, it suggested that SFL algorithm alone can produce an acceptable solution or...
even an optimal solution if the problem was not so complicated (such as Todd function with single factor). The average execution time required by SFL algorithm was also dramatically faster than the MA. When the problem is more complicated especially with three or four dimensions, the SFL algorithm is more suitable to exploit a solution space as a local search by embedding within the MA. The exploitation process can be performed on each population member to improve its experience and thus obtain a population of local optimum solutions. However, Memetic Algorithm (MA) requires longer computational time.

Table 1 Experimental results obtained from the proposed methods on each testing function

<table>
<thead>
<tr>
<th>Function number</th>
<th>Function name Bound</th>
<th>Optimal solutions BSF %Error</th>
<th>Shuffled Frog Leaping (SFL) BSF %Error</th>
<th>Memetic Algorithm (MA) BSF %Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Todd function Upper 9.515504816 0.000000000 9.515504816 0.000000000</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Branin function Upper 5.150395778 0.000000000 5.150395778 0.000001437</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Camelback function Upper Infinity value -14.433180315</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Goldstein-price function Upper 9.522878745 2.037269966 9.517646070 0.05498458</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Montgomery function Lower -66.03402000 -66.033756829</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Parabolic function Upper 12.000000000 0.000054458 11.999993916 0.000011483</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Rastrigin function Upper 80.000000000 0.283059876 79.999073710 0.004907862</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>Rosenbrock function Upper 80.000000000 0.000101738 79.999990888 0.000001140</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>Shekel function Upper 23.445230201 0.007909942 23.445202585 0.000117789</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>Styblinski function Upper 348.332331368 0.000460016 348.332041039 0.000083348</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>Dejong function Lower 0.000000000 -0.001878667</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

5. Conclusions

In this work, two types of metaheuristics called Shuffled Frog Leaping (SFL) and Memetic Algorithm (MA) were adapted to find optimal solutions of eleven non-linear continuous mathematical models. Considering the solution space in a specified region, some models contain global optimum and multiple local optimaums. A series of computational experiments using each method were conducted. Experimental results were analysed in terms of best solutions found so far, mean and standard deviation. It was found that SFL alone can produce an acceptable solution or even an optimal solution if the problem was not so complicated with quicker computational time than the MA. When the problem is more complicated especially with three or four dimensions, the SFL algorithm is more suitable to exploit a search space by embedding within the MA by improving individuals’ experience and obtaining a population of local optimum solutions.

Acknowledgement

The last author would like to thank the Faculty of Engineering, Naresuan University for the financial support.

References


