Artificial movements inspired for global optimization

Hindriyanto Dwi Purnomo\textsuperscript{a, b}, Hui-Ming Wee\textsuperscript{a*, b}, Yugowati Praharsi\textsuperscript{a, b}

\textsuperscript{a}Department of Industrial and System Engineering, Chung Yuan Christian University, 32023, Taiwan
\textsuperscript{b}Department of Information Technology, Satya Wacana Christian University, 50711, Indonesia

Abstract

A new algorithm is proposed in this study for continuous problem optimization. The algorithm implements artificial movements move off and move forward, which mimics the soccer player’s movement during a game. Both movements incorporate the social learning as well as cognitive learning. The performance of the proposed method was compared to those of the genetic algorithm and the particle swarm optimization. The proposed method outperforms the PSO on multi-modal functions and unimodal functions with high dimensionality. The method also shows better performance than the genetic algorithm on most of the problems used in the experiment. The experiment results reveal that the proposed method is potentially a powerful optimization technique.

© 2012 The Authors. Published by Elsevier Ltd. Selection and/or peer-review under responsibility of the Centre of Humanoid Robots and Bio-Sensor (HuRoBs), Faculty of Mechanical Engineering, Universiti Teknologi MARA.

Keywords: artificial movements, move off, move forward, evolution operators, social learning, cognitive learning

Nomenclature

\begin{itemize}
\item $P$: a set of simultaneous vector solution
\item $X_i^t$: a vector solution $i$ at time $t$
\item $x_i$: decision variables $i$
\item $B$: the best solution so far
\item $n$: the number of decision variable
\item $s$: the number of players
\end{itemize}

1. Introduction

Optimization is applied in various fields and its applications are countless. Many real problems are difficult to solve using exact methods; therefore, metaheuristics methods are necessary in practices [1]. The algorithms provide good solutions in a reasonable time; therefore, it receives significant interest both in research and industrial practices. The algorithms overcome the drawback of conventional, computational-based numerical linear and nonlinear programming methods in which gradient information is considered necessary [2]. For many years, natural processes have been used as model and metaphor to solve many optimization problems. Simulated Annealing [3], Genetic Algorithm (GA) [5], Ant Colony Optimization (ACO) [6], Particle Swarm Optimization (PSO) [7] and Evolution Strategies [8] are among the most popular meta-heuristics.

* Correspondence to: Dr. H.M. Wee, Department of Industrial Engineering Chung Yuan Christian University, No.200, Chung Pei Rd., Chungli, 32023, Taiwan, ROC. Tel.: +886-3-2654409; Fax:+886-3-2654499; E-mail : wee@cyu.edu.tw (Dr. H.M. Wee), hindriyanto.purnomo@staff.uksw.edu (H.D. Purnomo), yugowati.praharsi@staff.uksw.edu (Y. Praharsi)
In this paper, we propose a new method that is inspired by the soccer player movements, to solve the global optimization of multivariate systems. The rest of the paper is organized as follows. Section 2 describes the background and the structure of the proposed method. Section 3 presents the selected benchmark problems. Section 4 discusses the performance of the algorithm. Conclusion and future directions are presented in Section 5.

2. The proposed method

We adopt the basic player’s movement and introduce it as artificial movements, the move forward and the move off. Move forward is a movement in which a player move approaching the ball dribbler. The move forward represents local search between the player and the ball dribbler. On the other hand, move off is a movement in which a player exploring the solution space in order to obtain new promising area and can be assumed as global search. The proposed method is described as follow:

2.1. Representation of solution

The optimization problem can be expressed as:

\[
\text{min } f(X)
\]  

where \(f(.)\) is the objective function. A team is defined as the population of the solutions \(P = \{X^1, X^2, ..., X^i, ..., X^n\}\). For each player \(i\), his position is denoted as a vector \(X^i = \{x^i_1, x^i_2, ..., x^i_n\}\) where \(n\) is the number of decision variables.

2.2. Initializing players’ position and the ball dribbler

A initial team \(P_0\) consists of \(s\) players \(P_0 = \{X_0^1, X_0^2, ..., X_0^s\}\). Initial player \(i\) position \(X_0^i\) is generated randomly. The player encodes the potential solution \(X_0^i = \{x_{1.0}^i, x_{2.0}^i, ..., x_{n.0}^i\}\). Objective function(s) is used to evaluate the player’s position and the best player is selected as the initial ball dribbler \(B_0\). The initial players’ position is also considered as the initial players’ best position \(Pb = P_0\).

2.3. Player’s movement

At each step, by a probability \(m\), a player \(i\) move into new position using move off, otherwise, the player move into new position using move forward by a probability \(1-m\). When a player selects the move off, the player explores the solution space, regardless of the ball dribbler’s position, but he can still consider his knowledge by a probability. On the other hand, when a player selects the move forward, he will move approaching the ball dribbler. But a player still has a chance to do the move off after approaching the ball.

2.4. The ball dribbler’s and players’ knowledge update

The players will be evaluated in order to select a candidate of ball dribbler. If the candidate of ball dribbler is better than the current ball dribbler, then the ball dribbler is replaced. Each player will also update their best position \(X_b^i\). The players’ movements and the ball dribbler’s and players’ knowledge update are repeated until the stopping criteria are met. The move off incorporates random movements while the move forward is conducted by considering the player’s current position, the player’s best position and the ball dribbler’s position. The ball dribbler’s and players’ knowledge update adopts the elitism strategy in which replacement is done when the new solutions are better than the current solutions.

Pseudocode of the proposed method

\[
P(X) = P_0; \quad /* Initialize players \\
B = \text{best}(P(X)) \quad /* Initialize ball dribbler \\
P_b(X) = P(X) \quad /* initialize player’s best position \\
While \text{termination criteria not met} \\
\text{For } i = 1 : s \quad /*s = number of players \\
\quad \text{If random } \leq m \\
\quad \quad \text{If random } \leq c \quad /*X_i^* = \text{move off}(X_b^i) \\
\quad \quad \text{Else} \quad X_i^* = \text{move off}(X_{i-1}^i) \\
\quad \quad \text{Endif} \\
\quad \text{Else} \quad X_i^* = \text{move forward}(X_{i-1}^i, X_b^i, B) \\
\text{Endfor}
\]
If $\text{random} < l$

$X_t^i = \text{move\_off}(X')$

Else

$X_t^i = X'$

Endif

Endfor

Update\_ball\_dribbler()

Update\_player\_best\_position()

Endwhile

3. Numerical Example

The proposed algorithm is evaluated based on a set of continuous benchmark problems listed in Table 1. The functions consist of a set of unimodal functions ($f_1 \sim f_7$), multi modal functions with many local minima ($f_8 \sim f_{13}$), and multi modal functions with a few local minima ($f_{14} \sim f_{20}$). The benchmark functions were studied and tested in [9-18]. We tested the proposed method on ($f_1 \sim f_{13}$), in 10 and 50 dimensions. The maximum number of evaluations was used as the termination criteria. Each experiment was repeated 30 times and the average fitness of the best players was recorded.

| Test function | $n$ | $S$ | $f_{\text{min}}$ |
|---------------|----|-----|-----------------|
| $f_1 = \sum_{i=1}^{n} x_i^2$ | 10/50 | [-100, 100]$^a$ | 0 |
| $f_2 = \sum_{i=1}^{n} x_i^2 + \prod_{i=1}^{n} x_i$ | 10/50 | [-10, 10]$^a$ | 0 |
| $f_3 = \sum_{i=1}^{n} (\sum_{j=1}^{i} x_j)^2$ | 10/50 | [-100,100]$^a$ | 0 |
| $f_4 = \max\{x_i, 1 \leq i \leq n\}$ | 10/50 | [-100,100]$^a$ | 0 |
| $f_5 = \sum_{i=1}^{n} (100(x_{i+1} - x_i^2 + (x_i - 1))^2)$ | 10/50 | [-30, 30]$^a$ | 0 |
| $f_6 = \sum_{i=1}^{n} (x_i + 0.5)^2$ | 10/50 | [-100, 100]$^a$ | 0 |
| $f_7 = \sum_{i=1}^{n} x_i^4 + \text{rand}[0,1]$ | 10/50 | [-1.28, 1.28]$^a$ | 0 |
| $f_8 = \sum_{i=1}^{n} x_i \sin(\sqrt{|x_i|})$ | 10/50 | [-500, 500]$^a$ | 4189.83/20949.14 |
| $f_9 = \sum_{i=1}^{n} x_i^2 - 10 \cos(2\pi x_i) + 10]$ | 10/50 | [-5.12, 5.12]$^a$ | 0 |
| $f_{10} = -20 \exp\left(-0.2 \frac{1}{n} \sum_{i=1}^{n} x_i^2\right)$ | 10/50 | [-32, 32]$^a$ | 0 |
| $f_{11} = \frac{1}{4000} \sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos(\frac{x_i}{\sqrt{n}}) + 1$ | 10/50 | [-600, 600]$^a$ | 0 |
| $f_{12} = \frac{1}{n} (10 \sin^2(\pi y_i) + \sum_{i=1}^{n} (y_i - 1)^2 [1 + 10\sin^2(\pi y_{i+1})] + (y_n - 1)^2) + \sum_{i=1}^{n} u(x_i, 100,100,4).$ | 10/50 | [-50, 50]$^a$ | 0 |
| $y_i = 1 + \frac{1}{4} (x_i + 1)$ | 10/50 | [-50, 50]$^a$ | 0 |
| $u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m & x_i > a \\ 0 & -a \leq x_i \leq a \\ k(-x_i - a)^m & x_i < -a \end{cases}$ | 10/50 | [-50, 50]$^a$ | -1.15 |
| $f_{13} = 0.1 \sin^2(3\pi x_1) + \sum_{i=1}^{n} (x_i - 1)^2 [1 + \sin^2(3\pi x_{i+1})] + (x_n - 1)^2 [1 + \sin^2(2\pi x_n)] + \sum_{i=1}^{n} u(x_i, 5,100,4)$ | 10/50 | [-50, 50]$^a$ | 0.998 |
| $f_{14} = \left[\frac{1}{500} + \sum_{i=1}^{n} \frac{1}{\sin^2(\pi x_i)}\right]^{-1}$ | 2 | [-65.54, 65.54]$^a$ | 0.0003075 |
| $f_{15} = \sum_{i=1}^{n} [u_i - \frac{1}{b_i + b_i x_i}]^2$ | 4 | [-5, 5]$^a$ | -1.0316 |
| $f_{16} = 4x_1^2 - 2.1x_1^4 + \frac{1}{3} x_1^6 + x_1 x_2 - 4x_2^2 + 4x_2^4$ | 2 | [-5, 5]$^a$ | 0.398 |
| $f_{17} = \left[\frac{x_2}{4\pi^2} x_2^2 + \frac{5}{\pi} x_1 - 6\right]^2 + 10 (1 - \frac{1}{8\pi}) \cos x_1 + 10$ | 2 | [-2, 2]$^a$ | 3 |
| $f_{18} = [1 + (x_1 + x_2 + 1)^2(19 - 14x_1 + 3x_2^2 - 14x_2 + 6x_1 x_2 + 3x_2^2)] [30 + (2x_1 - 3x_2)^2]$ | 2 | [-2, 2]$^a$ | 3 |
In this study, the move off operator adopts the dynamical non uniform mutation that has been widely used in Genetic Algorithm, e.g. Coello [19], Zhao et al. [20], Michalewicz [21]. The non uniform mutation improves the fine tuning capabilities by providing search in different ways as needed (e.g. exploring wider or narrower regions) over time [21]. For the move forward, we consider the player’s current position $X_{t}^{i}$, the player’s best position $X_{b}^{i}$, and the ball dribbler’s position $B_{b}$. Center of mass principle is implemented to calculate the player’s new position and it is given by:

$$X_{t+1}^{i} = \frac{\alpha X_{t}^{i} + \beta X_{b}^{i} + \gamma B_{b}}{\alpha + \beta + \gamma}$$

where $\alpha$, $\beta$ and $\gamma$ are the weight for the player’s current position, the player’s best position, and the ball dribbler’s position respectively. The parameters of the proposed method was set as follows: team size $s = 10$, probability of move off the ball $m = 0.05 - 0.1$, $l = 0.05 - 0.3$, $c = 0.5$, $\alpha = (1- \omega)$, $\beta = (1- \omega)$ and $\omega = 0.618$ is a golden ratio. Golden ratio is used because it has good properties in narrowing the search spaces as demonstrated in the golden section search.

The performance of the proposed method is compared with the PSO and GA. The GA has been implemented for two type of crossover operators, the Heuristic crossover (GA-HX) and Laplace crossover (GA-LX). GA-HX was introduced by Wright [18] while GA-LX was introduced by Deep and Thakur [16]. For both GA-HX and GA-LX, dynamical non uniform mutation is applied. Both algorithms use roulette wheel selection. The crossover rate and mutation rate were set as $cr = 0.9$ and $pm = 0.01$ respectively. The value of location parameter for the Laplacian crossover was set to $\theta$ as mentioned in Deep and Thakur [16]. The PSO parameters were set as follows: population size $= 100$, $v_{\text{max}} = 1$, $v_{\text{min}} = -1$, $\phi_{1} = 2$, $\phi_{2} = 2$ and decrement $\omega$ from 0.9 to 0.4. The parameters were set following the suggestion of Clerc and Kennedy [22].

4. Result and discussion

4.1. Problem of low dimensionality

For the unimodal functions, the PSO is superior on $(f_1 - f_4)$. In these problems, the PSO converge slowly in the beginning and then turn to converge quickly. On $f_5$ and $f_7$, the proposed method, the GA-LX and GA-HX have similar convergence pattern while the PSO much slower in the beginning. For $f_6$, all algorithms reached optimal value, however, the proposed method converge faster than the other algorithms.

| $f(x)$ | # eval | PSO mean | PSO stdev | GA-LX mean | GA-LX stdev | GA-HX mean | GA-HX stdev | The proposed method mean | The proposed method stdev |
|-------|--------|----------|-----------|-------------|-------------|-------------|-------------|--------------------------|--------------------------|
| 1     | 50K    | 1.89E-54 | 6.08E-54  | 2.09E-03    | 1.05E-03    | 2.30E-04    | 5.86E-05    | 5.27E-26                 | 8.04E-26                 |
| 2     | 50K    | 6.75E-29 | 2.98E-28  | 9.06E-03    | 2.78E-03    | 3.57E-03    | 1.14E-03    | 1.12E-13                 | 2.15E-13                 |
| 3     | 50K    | 8.52E-04 | 6.79E-04  | 2.66E+01    | 4.41E+01    | 8.22E+00    | 3.85E+00    | 2.80E-03                 | 3.27E-03                 |
| 4     | 50K    | 2.65E-13 | 3.96E-13  | 1.13E-01    | 2.98E-02    | 4.01E-02    | 1.46E-02    | 1.26E-06                 | 1.04E-06                 |
| 5     | 50K    | 4.72E+00 | 8.81E-01  | 2.16E+00    | 2.51E+00    | 3.60E+00    | 1.99E+00    | 1.81E+00                 | 2.01E+00                 |
| 6     | 20K    | 0.00E+00 | 0.00E+00  | 0.00E+00    | 0.00E+00    | 0.00E+00    | 0.00E+00    | 0.00E+00                 | 0.00E+00                 |
| 7     | 50K    | 4.22E-03 | 2.11E-03  | 3.43E-03    | 1.55E-03    | 3.14E-03    | 1.24E-03    | 1.92E-03                 | 1.09E-03                 |
| 8     | 100K   | -2.99E+03| 2.57E+02  | -4.19E+03   | 5.09E-04    | -4.19E+03   | 8.55E-05    | -4.19E+03               | 1.27E-18                 |
| 9     | 100K   | 1.29E+00 | 9.48E-01  | 1.17E-04    | 5.00E-05    | 2.35E-05    | 1.80E-05    | 5.20E-07                 | 2.65E-06                 |
| 10    | 100K   | 6.75E-15 | 3.14E-15  | 5.79E-03    | 1.71E-03    | 2.64E-03    | 7.75E-04    | 5.70E-14                 | 5.29E-24                 |
| 11    | 100K   | 7.92E-02 | 3.49E-02  | 4.20E-02    | 2.83E-02    | 4.48E-02    | 2.66E-02    | 5.28E-02                 | 2.67E-02                 |
| 12    | 100K   | 7.72E-26 | 2.90E-25  | 1.28E-05    | 3.85E-04    | 3.18E-05    | 6.51E-05    | 1.94E-27                 | 7.35E-27                 |
GA HX is the slowest.

For the multi modal functions with many local optima \((f_8-f_{13})\), the proposed method produced the best value in 4 out of 6 problems. In \(f_8\) the proposed method, the GA-LX and the GA-HX reached very close to the optimal value. However, the proposed method produced much higher precision than the other two. For \(f_6\), \(f_7\) and \(f_{13}\) the proposed method outperforms the other algorithms. At \(f_{13}\), all algorithms have the same convergence pattern; however, the GA-LX performs slightly better than the other algorithms.

On the low multi modal function \((f_4-f_{10})\), all algorithms approximately have the same pattern. They able to reach close to the optimal solutions at \(f_4, f_5, f_6\) and \(f_8\). The proposed method and PSO have relatively the same precisions which are better than the precision of the GA-LX and the GA-HX. In term of CPU time, the PSO is the fastest while the GA HX is the slowest.

### Table 3. The CPU time for benchmark problems with 10 or less dimension (mean and standard deviation of 30 runs). The best mean CPU time is emphasized in boldface.

| f(x) | PSO | GA-LX | GA-HX | The proposed method |
|------|-----|-------|-------|---------------------|
| \(\mu_{\text{PSO}}\) | \(\sigma_{\text{PSO}}\) | \(\mu_{\text{LX}}\) | \(\sigma_{\text{LX}}\) | \(\mu_{\text{HX}}\) | \(\sigma_{\text{HX}}\) | \(\mu\) | \(\sigma\) |
| 1    | 0.31 | 0.03  | 3.70  | 0.40               | 6.2    | 0.20          | 2.23   | 0.09          |
| 2    | 0.38 | 0.04  | 4.08  | 0.07               | 7.40   | 0.20          | 2.58   | 0.10          |
| 3    | 0.46 | 0.03  | 5.38  | 0.20               | 9.71   | 0.35          | 2.66   | 0.15          |
| 4    | 0.46 | 0.03  | 5.23  | 0.22               | 9.46   | 0.32          | 2.66   | 0.11          |
| 5    | 0.32 | 0.02  | 3.97  | 0.15               | 6.68   | 0.17          | 2.56   | 0.11          |
| 6    | 0.13 | 0.02  | 1.61  | 0.10               | 2.70   | 0.13          | 1.02   | 0.07          |
| 7    | 0.43 | 0.03  | 5.52  | 0.25               | 9.28   | 0.29          | 2.75   | 0.13          |
| 8    | 0.97 | 0.04  | 10.43 | 0.35               | 18.81  | 0.56          | 5.41   | 0.16          |
| 9    | 0.77 | 0.04  | 9.00  | 0.69               | 14.86  | 0.23          | 5.23   | 0.28          |
| 10   | 0.84 | 0.13  | 9.05  | 0.28               | 15.94  | 0.27          | 5.27   | 0.2           |

### 4.2. Problem of high dimensionality

For the unimodal functions \((f_1-f_3)\), the proposed method performs the best in 6 out of 7 problems. On \(f_1\), all algorithms converge similar to the result of 10 dimensions. For \((f_1-f_3)\), the proposed method clearly produces better results than the other algorithms and it steadily converges faster than the other algorithms. On \(f_3\) the proposed method reached the closest to the optimum solution while the GA-LX and GA-HX almost as well. Function \(f_6\), only the proposed method can hit the optimum value. On \(f_5\), the GA-LX perform the best and the GA-HX is slightly affected. On highly multi modal function \((f_8-f_{13})\), there are some differences from those in 10 dimensions. At \(f_6\), only the GA-LX and GA-HX can reach the optimum value. In functions \(f_6, f_{10}, f_{13}\), the proposed method outperforms the other algorithms. In term of CPU time, the PSO still the fastest while the GA-HX is the slowest. However, the time ratios between the PSO and the other algorithms in the high dimensionality problems are lower than the time ratio in the low dimensionality problems.

Based on the experiments, the proposed method is good to be implemented in multi modal functions and functions with high dimensionality. Although its the CPU time is slower than PSO, it significantly produce better results in most
problems. Regarding the convergence speed, the proposed method has fast convergence in the early steps and followed by slow convergence rate. The convergence speed of the algorithms, based on the average best solution over 30 runs, is shown in figure 1.

Table 4. Result for benchmark problems with 50 dimensions (mean and standard deviation of 30 runs). The best performing algorithm(s) is emphasized in boldface

| f(x) | # eval | PSO   | GA-LX   | GA-HX   | The proposed method |
|------|--------|-------|---------|---------|---------------------|
|      |        | mean  | stdev   | mean    | stdev   | mean    | stdev   |
| 1    | 500K   | 1.40E-17 | 2.64E-17 | 7.15E-01 | 1.09E-01 | 3.85E+00 | 6.60E-01 | **1.85E-30** | 3.54E-30 |
| 2    | 500K   | 3.41E+02 | 1.63E+02 | 2.87E+00 | 1.96E-01 | 1.19E+01 | 8.47E-01 | **5.08E-16** | 3.23E-16 |
| 3    | 500K   | 3.83E+04 | 1.03E+04 | 2.97E+04 | 6.20E+03 | 3.12E+03 | 1.20E+03 | **2.93E+01** | 1.08E+01 |
| 4    | 500K   | 1.50E+00 | 2.97E+00 | 7.68E-01 | 9.88E-02 | 1.03E+00 | 1.47E-01 | **1.22E-03** | 4.08E-04 |
| 5    | 500K   | 4.51E+03 | 4.84E+03 | 3.89E+01 | 3.80E+01 | 6.17E+01 | 2.04E+01 | **2.95E+01** | 3.89E+01 |
| 6    | 500K   | 3.35E+01 | 1.80E+01 | 2.83E+01 | 3.34E+00 | 1.10E+02 | 1.52E+01 | **0.00E+00** | 0.00E+00 |
| 7    | 500K   | 2.98E+01 | 2.16E+01 | 1.07E-02 | 2.72E-03 | 1.09E+02 | 2.02E-02 | **1.38E-02** | 3.24E-03 |
| 8    | 500K   | -1.17E+04 | 8.36E+02 | **-2.09E+04** | 6.23E-03 | -2.09E+04 | 4.26E-02 | **-2.04E+04** | 2.62E+02 |
| 9    | 500K   | 2.97E+02 | 6.19E+01 | 2.83E+01 | 4.54E+00 | 6.86E+01 | 8.78E+00 | **2.23E+00** | 1.64E+00 |
| 10   | 500K   | 1.73E+01 | 4.10E+00 | 1.75E-01 | 1.67E-02 | 7.91E-01 | 1.98E-01 | **6.89E-14** | 1.14E-14 |
| 11   | 500K   | 9.43E-03 | 1.33E-02 | 5.17E-01 | 7.38E-02 | 1.02E+00 | 1.65E-02 | **4.82E-02** | 4.35E-02 |
| 12   | 500K   | 9.45E+00 | 1.83E+00 | 2.00E-03 | 1.21E-03 | 1.49E-02 | 8.23E-03 | **2.40E-31** | 3.67E-31 |
| 13   | 500K   | 2.78E-01 | 9.61E-01 | -3.95E-01 | 2.75E-01 | -6.92E-01 | 3.07E-01 | **-1.15E+00** | 2.41E-08 |

Table 5. The CPU time for benchmark problems with 50 dimensions (mean and standard deviation of 30 runs). The best CPU time is emphasized in boldface

| f(x) | PSO   | GA-LX   | GA-HX   | The proposed method |
|------|-------|---------|---------|---------------------|
|      | mean  | stdev   | mean    | stdev   | mean    | stdev   |
| 1    | 6.3   | 0.3     | 49.0    | 5.0     | 75.0    | 3.0     | 23.7    | 1.1     |
| 2    | 11.0  | 0.6     | 38.8    | 2.2     | 65.8    | 3.1     | 26.2    | 0.4     |
| 3    | 24.7  | 1.1     | 44.0    | 2.0     | 78.8    | 3.7     | 43.6    | 1.5     |
| 4    | 8.20  | 0.3     | 73.8    | 3.9     | 132.3   | 4.0     | 29.0    | 0.5     |
| 5    | 6.10  | 0.2     | 52.3    | 1.9     | 88.4    | 3.3     | 25.9    | 0.9     |
| 6    | 13.7  | 0.8     | 50.5    | 0.9     | 89.5    | 3.9     | 24.8    | 1.4     |
| 7    | 12.1  | 0.8     | 92.7    | 0.3     | 196.8   | 34.4    | 19.7    | 1.8     |

![Fitness values over number of evaluations](image1)

![Fitness values over number of evaluations](image2)

![Fitness values over number of evaluations](image3)

a. f1, 10 dimensions
b. f6, 10 dimensions
c. f14, 2 dimensions
5. Conclusion

This paper describes a new algorithm that is based on two artificial movements, move off and move forward. The artificial movements mimic the basic soccer players’ movements. Unconstraint continuous benchmark problems are used to evaluate the performance of the proposed method. A center of mass principle is implemented in the local search. The performance of the proposed algorithm shows improvements over the PSO, the GA-LX and the GA-HX. The experiment results show that the proposed method outperforms the PSO on multi-modal functions and unimodal functions with high dimensionality. The proposed method also shows better performance than the GA-LX and GA-HX on most of the problems. This study confirms that the proposed method has the potential to be a powerful optimization technique. Future research can be done to improve the method for applications in constraint problems, dynamic optimization problems and combinatorial problems.

References

[1] E.-G. Talbi, Meta-heuristics, from design to implementation. Lille: John Wiley & Sons, 2009.
[2] K. S. Lee and Z. W. Geem, “A new meta-heuristic algorithm for continuous engineering optimization: harmony search theory and practice,” Computer Methods in Applied Mechanics and Engineering, vol. 194, pp. 3902–3933, 2005.
[3] S. Kirkpatrick, C. D. Gelatt, Jr., and M. P. Vecchi, “Optimization by simulated annealing,” Science, vol. 220, pp. 671-680 1983.
[4] F. Glover, “Tabu search - Part II,” ORSA Journal on Computing, vol. 2, pp. 4-32, 1990.
[5] J. H. Holland, Adaptation in Natural and Artificial Systems. Ann Arbor: University of Michigan Press, 1975.
[6] M. Dorigo and G. D. Caro, “Ant colony optimization: a new meta-heuristic,” Proceedings of the 1999 Congress on Evolutionary Computation, pp. 1470-1477, 1999.
[7] J. Kennedy and R. Eberhart, “Particle swarm optimization,” Proceedings of IEEE International Conference on Neural Networks, vol. IV, pp. 1942-1948, 1995.
[8] H.-P. Schwefel and H.-G. Beyer, “Evolution strategies – A comprehensive introduction,” Journal of Natural Computing, vol. 1, pp. 3-52, 2002.
[9] X. Yao, Y. Liu, and G. Lin, “Evolutionary programming made faster,” IEEE Transactions on Evolutionary Computation, vol. 3, pp. 82-102, 1999.
[10] J. Vesterstrom and R. Thomsen, “A comparative study of differential evolution, particle swarm optimization, and evolutionary algorithms on numerical benchmark problems,” Congress on Evolutionary Computation, vol. 2, pp. 1980 - 1987, 2004.
[11] S. He, Q. H. Wu, J. Y. Wen, J. R. Saunders, and R. C. Patonc, “A particle swarm optimizer with passive congregation * BioSystems, vol. 78, pp. 135-147, 2004.
[12] J.-T. Tsai, T.-K. Liu, and J.-H. Chou, “Hybrid Taguchi-genetic algorithm for global numerical optimization,” IEEE Transactions on Evolutionary Computation, vol. 8, pp. 365 - 377 2004.
[13] K. Challapilla, “Combining mutation operators in evolutionary programming,” IEEE Transactions on Evolutionary Computation, vol. 2, pp. 91-96, 1998.
[14] V. L. H. A. K. Qin, and P. N. Suganthan, “Differential evolution algorithm with strategy adaptation for global numerical optimization,” IEEE Transactions on Evolutionary Computation, vol. 13, pp. 398-417, 2009.
[15] J. Brest, S. s. Greiner, B. Bo’skovic’, M. Mernik, and V. Zumer, “Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark problems,” IEEE Transactions on Evolutionary Computation, vol. 10, pp. 646-657, 2006.
[16] K. Deep and M. Thakur, “A new crossover operator for real coded genetic algorithms,” Applied Mathematics and Computation, vol. 188, pp. 859-911, 2007.
[17] K. Deep and M. Thakur, “A new mutation operators for real coded genetic algorithms,” Applied Mathematics and Computation, vol. 193, pp. 211-230, 2007.
[18] A. H. Wright, “Genetic algorithms for real parameter optimization, in G.J.E. Rawlins (Ed), Foundations of Genetic Algorithms I,” Morgan Kaufmann, San Mateo, pp. 205-218, 1991.
[19] C. A. C. Coello, “Use of a self-adaptive penalty approach for engineering optimization problems”, Computers in Industry, 41, 113-127, 2000.
[20] Zhao, X., Gao, X-S. & Hu, Z-C, Evolutionary programming based on non-uniform mutation. Applied Mathematics and computation, 192, 1-11, 2007.
[21] Michalewicz, Z. Genetic algorithm + Data structures = evolution programs. Springer-Verlag, 1992.
[22] Clerc, M. & Kennedy, J. The particle swarm—explosion, stability, and convergence in a multidimensional complex space. IEEE Transactions on Evolutionary Computation 6, 58-73, 2002.