Applied Computer Science, vol. 12, no. 4, pp. 59–77

Submitted: 2016-11-20
Revised: 2016-12-05
Accepted: 2016-12-09

Swarm intelligence, Bio-inspired, Bat Algorithm,
Multi-swarm optimisation, Non linear optimisation

Evans BAIDOO[*]*, Stephen OPOKU OPPONG*[**]

A COMPARATIVE STUDY ON MULTI-SWARM OPTIMISATION AND BAT ALGORITHM FOR UNCONSTRAINED NON LINEAR OPTIMISATION PROBLEMS

Abstract

A study branch that mocks-up a population of network of swarms or agents with the ability to self-organise is Swarm intelligence. In spite of the huge amount of work that has been done in this area in both theoretically and empirically and the greater success that has been attained in several aspects, it is still ongoing and at its infant stage. An immune system, a cloud of bats, or a flock of birds are distinctive examples of a swarm system. In this study, two types of meta-heuristics algorithms based on population and swarm intelligence - Multi Swarm Optimization (MSO) and Bat algorithms (BA) – are set up to find optimal solutions of continuous non-linear optimisation models. In order to analyze and compare perfect solutions at the expense of performance of both algorithms, a chain of computational experiments on six generally used test functions for assessing the accuracy and the performance of algorithms, in swarm intelligence fields are used. Computational experiments show that MSO algorithm seems much superior to BA.

[*] Department of Computer Science, Kwame Nkrumah University of Science and Technology, Ghana, e-mail: ebaidoo2.cos@st.knust.edu.gh

[**] Department of Information Technology, Academic City College, Ghana, e-mail: geniusstevo@gmail.com

1. INTRODUCTION

Optimization problems are ever-present in our daily life and often come in an array of forms. Ideally, Optimization is the activity of searching for the most favourable solution among a set of solutions to the problem based on some performance criteria. Optimisation problems are traditional and most widely studied problem in Combinatorial Optimization (Yang, 2008). Objective optimization (objective programming) is an area in computing that is concerned with mathematical optimization problems with defined decision making criteria, that involves some objective function to be optimized. In lots of technical fields, non linear objective optimization dealings in uninterrupted domains involve a costly numerical simulation with a varied objective function. Much of these problems can be found in broad applications as finance and logistics, manufacturing and in engineering. Practically, a crude solution to a continuous unconstrained optimisation problem will be meant to carry out a predefined task in some proficient way or the maximum quality or to generate maximum yields making reference to a given limited resource (Yuan, 2016).

Although not much research study has gone into single and multi-objective non linear optimization problems, most researchers have varied viewpoints and as a result, there exist diverse solutions and goals when defining and solving them. Though the paramount solutions can possibly be worked out by hand or through comprehensive search in some plain situations, programmed optimi-zation techniques are needed to respond to most non-trivial problems largely due to their sizes and complex modality of the search space.

Unconstrained non linear optimization is a dynamic and fast rising research area with a greater impact in the real world. Regardless of the existence of a broad variety of such problems that may look relatively different from another, there are complete or approximate algorithms available to tackle like problems in complex situations. The process of fine-tuning optimisation can be likened to finding the highest peak of a landscape with little attention to the actual meaning of the problem as in fig 1.

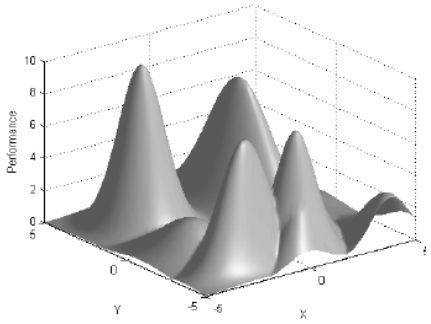

Fig. 1. Visualisation of optimisation problem (Yuan, 2016)

This paper makes use of two approximate algorithms, Multi Swarm Optimisation (MSO) and Bat algorithm (BA) for presenting a solution to the problem. Some authors claim these methods are generally termed meta-heuristic algorithms (Yang, 2008; Blum et al., 2008) although they are based on the intelligence of swarm. These methods dwell on the assurance of obtaining optimal solutions in practically limited time frame. Additionally, no gradient information is needed. This comes in handy in the situation of optimization problems which allows for the objective function to be given completely (that said, when objective function results are attained by simulation), or when there is non-differentiable objective function.

The paper aims to draw comparison between the Multi Swarm algorithm and Bat algorithm for solving continues unconstrained non linear single objective optimisation problems. The remains of the paper are arranged as follows. Section 2 in brief examines the non linear optimization mathematical models for experimenttation; Section 3 details brief explanation of the MSO and BA for testing and solving the problem. Experimental results are revealed in Section 4 and finally in Section 5 the conclusion of the work.

2. NON LINEAR OPTIMISATION PROBLEM

A Non-linear optimization problem is one in which in any case one of the constraints of the decision variable is a flat nonlinear function. Identified and researched in connection to sensitivity analysis, its original mention can be established from a thesis in 1952. In general optimisation problem may be defined mathematically by (Gal & Nedoma, 1972) as

$$P*(\theta) = min_{x\in\mathbb{R}^n} f(x,o) \quad (1)$$
$$g(x,\theta) \leq 0,$$
$$\theta \in \Theta \subset \mathbb{R}^m$$

where: x is the optimisation variable, θ are the parameters, $f(x,o)$is the objective function, and $g(x,\theta)$represent the constraints with $\Theta$the parameter space.

A typical example of a Non-linear function description may take a form as:

$$2x_1^2 + x_2^3 + \log x_3 \quad (2)$$

where: x_1, x_2 and x_3 are decision variables.

Practically, the mathematical modelling of this problem may include variables that exponent to a number or a power, and/or multiplied or divided by some other variables. In most cases they make use of transcendental functions as exp, log, sine and cosine. A function is said to be multimodal if it has beyond one local optima. A variable function is separable if it can be modelled as one variable of a sum of functions. In this case the separability is directly correlated to the concept of epistasis or interrelation along with the variables of the function. Friedman (1994) presented a study on dimensionality problems and its features. Saibal et al (2012) undertake a study on Noisy Non-Linear Optimization Problems and made a comparative analysis of Firefly Algorithm and Particle Swarm Optimization. In the end, they concluded the superior-ability of firefly over particle swarm in solving noisy non linear optimisation problems.

In order to explore the difficulty of same problems behaviour of non linear optimisation problem, the next section of the paper considers six classical unconstrained Non-Linear optimization models taken from (*"Example Functions (single and multi-objective functions)",* 2016; "*Virtual Library of Simulation Experiments: Test Functions and Datasets"*, 2016) with its function definition and a 2D graphical representation.

2.1. Griewank Function

Griewank function has a lot of widespread local minima. Nevertheless, the position of the minima is frequently distributed. It has its global minimum value at 0 with the function initialization range from [−600,600]. Griewank function contain some product term that initiates interdependence with the variables.

$$f_1(x) = \sum_{i=1}^{n} \frac{x_i^2}{4000} - \prod_{i-1}^{n} cos\left(\frac{x_i}{\sqrt{i}}\right) + 1 \qquad (3)$$

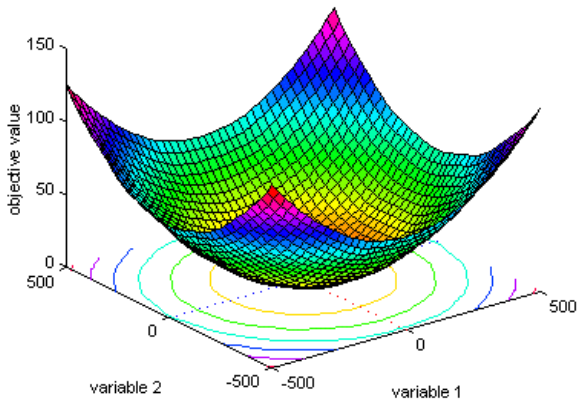

Fig. 2. 2D Plot for Griewangk's function ("Virtual Library of Simulation Experiments: Test Functions and Datasets", 2016)

2.2. Rastrigin function

Rastrigin function was modelled out of the Sphere function with a modulator cosine term to produce many local minima, thereby making the function highly multimodal. Its initialization range is between [−15, 15]. The contour of this function is made up of a great number of local minima where its value enlarges with the distance to the universal minimum.

$$f_2(x) = 10n + \sum_{i=1}^{n}(x_i^2 - 10\cos(2\pi x_i)) \quad (4)$$

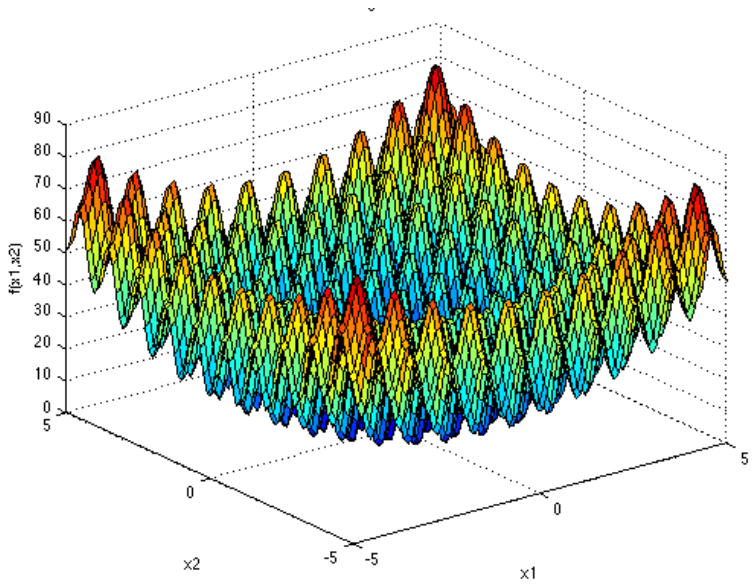

Fig. 3. 2D plot for Rastrigin's function, ("Virtual Library of Simulation Experiments: Test Functions and Datasets", 2016)

2.3. Ackley Function

Ackley function has an exponential term that covers up its surface with many local minima. It is characterized by a nearly flat outer region, and a large hole at the centre. With an initialization range of [−32.768, 32.768], the complexity of this function is moderated. Obtaining very fine results for the Ackley function revolves around applying an effective combination of exploratory and exploitative components in the search strategy. The function definition is stated in (5) with its plot in fig 4.

$$f_3(x) = -a\, exp\left(^{-b}\sqrt{\frac{1}{d}\sum_{i=1}^{d} x_i^2}\right) - exp\left(\frac{1}{d}\sum_{i=1}^{d}\cos(cx_i)\right) + a + \exp(1) \quad (5)$$

where recommended variable values are: a = 20, b = 0.2 and c = 2π.

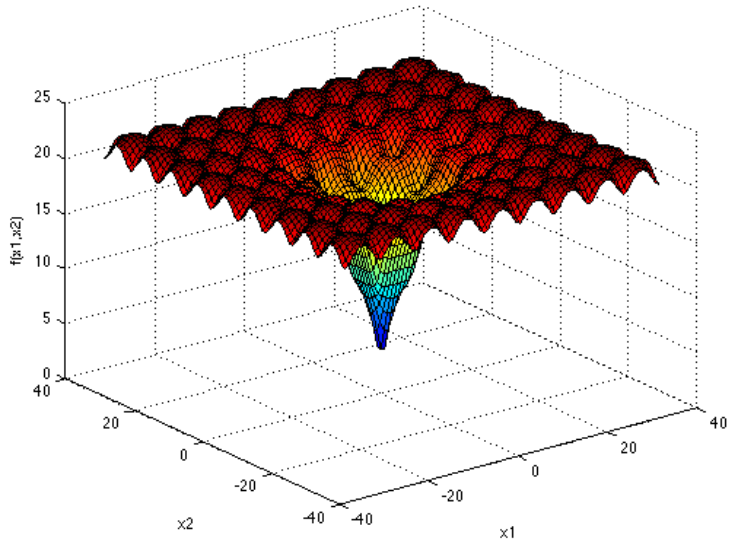

Fig. 4. 2D plot for Ackely function ("Virtual Library of Simulation Experiments: Test Functions and Datasets", 2016)

2.4. Rosenbrock function

Rosenbrock function also known as banana or valley function is a traditional optimization problem with a duo dimensional function illustrating a deep valley having the form of a parabola of the shape $x_1^2 = x_2$ that results to the global minimum. Owing to the non-linearity of the valley, lots of algorithms congregate slowly since they vary the direction of the search constantly and for this reason this problem has been repetitively used in assessing gradient-based optimization algorithms performance. Valley function has an initialization range of [−2.048, 2.048].

$$f_4(x) = \sum_{i=1}^{n-1}[100.(x_{i+1} - x^2{}_i)^2 + (x_i - 1)^2] \quad (6)$$

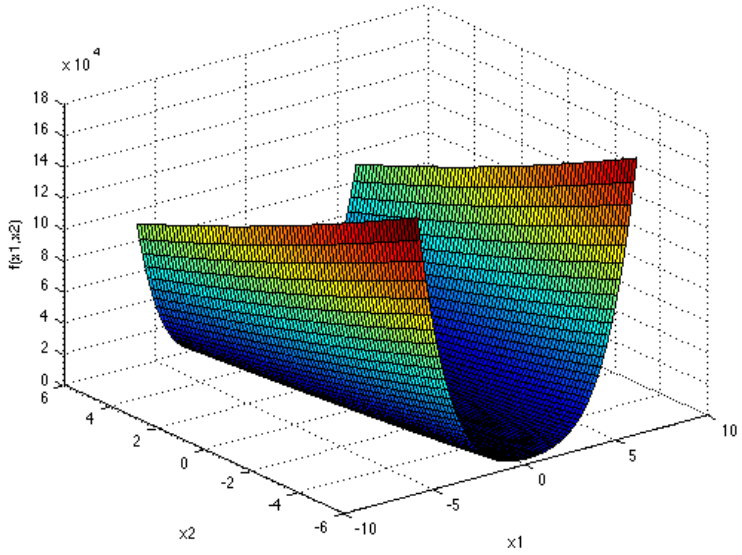

Fig. 5. Plot for Rosenbrock function, ("Virtual Library of Simulation Experiments: Test Functions and Datasets", 2016)

2.5. Schwefel function

The Schwefel function is complex, with many local minima. Initialization range for the function is [−500, 500]. The surface of Schwefel function is made up of a large amount of peaks and valleys. It is a function which possesses two global minimum but the second best minimum stretches away from the global minimum that lots of search algorithms are shuttered in. The function is defined as:

$$f_5(x) = 418.9829d - \sum_{i=1}^{d} x_i \sin(\sqrt{|x_i|}) \quad (7)$$

In addition, the global minimum is close to the limits of the domain. The function has a minimum value of 2 * -418.9829 = -837.9658 which is always hard to find because it is relatively far from the second best solution.

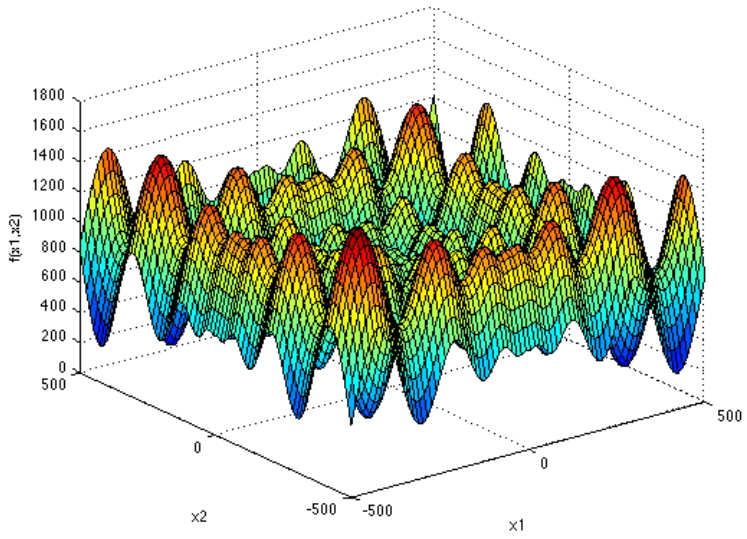

Fig. 6. 2D plot of Schwefel funtction (own study)

2.6. Michalewicz function

This function is multimodal with d! local optima. It has a parameter *m* which describes the "steepness" of the valleys or edges with larger m values leading to more complex search behaving like a needle in a haystack. This function is mostly used to test and check the efficiency of numerical optimization algorithms.

$$f_6(x) = -\sum_{i=1}^{d} \sin(x_i)\, sin^{2m}\left(\frac{ix_i^2}{\pi}\right) \tag{8}$$

Ideally, the suggested value of $m = 10$. The function illustration is defined as (8) with a 2D plot in fig. 7.

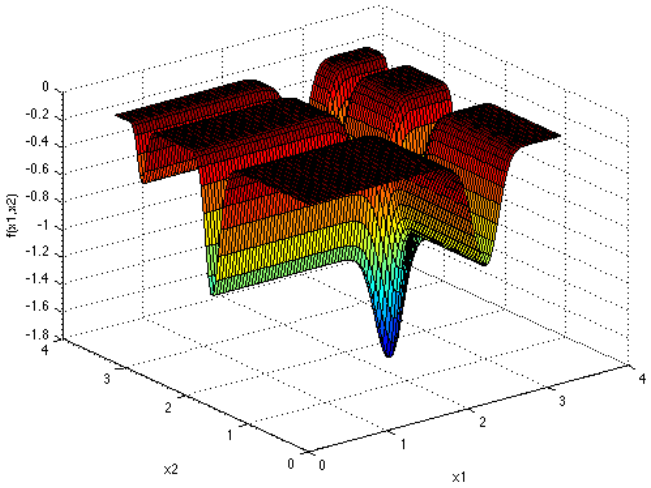

Fig. 7. 2D plot of Michalewicz function, ("Virtual Library of Simulation Experiments: Test Functions and Datasets", 2016).

3. META-HEURISTIC ALGORITHM

Meta-heuristic algorithms are increasingly being useful to solving complicated optimization problems and further to develop solutions to problems with complex nature in countless applications. This is probably due to their ability to handle difficult, ill-behaved, non-linear and multi-dimensional optimization problems as suggested by most researchers and practitioners with the latest of them demonstrated by Pansare and Kavade (2012), Madić et al. (2013), Zain et al. (2010), Ciurana et al. (2009), Rao et al. (2010), Samanta and Chakraborty (2011). In this study, an effort is made to compare the optimization results of two meta-heuristic algorithms applied to solving complex optimization problems, namely, Multi-Swarm optimisation and Bats Algorithm.

3.1.1. Multi Swarm Optimisation

Particle swarm optimization (PSO) has several extensions of which the Multi-swarm optimization is one. PSO mocks-up the activities of flocks such as that of birds and schools of fish. MSO do not rely on one (standard) swarm but rather fall on multiple sub-swarms practice (McCaffrey, 2016). The universal approach in multi-swarm optimization is that whilst a specific diversification process settles on where and when to initiate the sub-swarms, every sub-swarm centres on a specific region. MSO is meta-heuristic, in that the method has a set of design standards and procedures that can be used to create an explicit algorithm to solve a particular optimization problem. Since Multi-swarm optimization is an iterative

process, in its operation, it looks for a better solution, taking into consideration the best solution identified by any of the swarm particle until some stopping criteria is met. A characteristic feature of multi-swarms is that their preliminary positions and preliminary velocities are not arbitrarily selected as in particle swarms. As an alternative, they preserve some information from the earlier paths of the particles. In most cases, the improvement of multi-swarm systems guides to design decisions that on most occasions do not exist throughout the original growth of particle swarm optimization, for instance the number of particles to employ in every sub-swarm, the most favourable value for the check factor and the effects of logical starting positions and starting velocities. Having a clear identified guideline, these design decisions have been carefully revised with clear examples leading to the use of non-random primary positions and primary velocities to develop solutions in multi-swarm systems, which fail for single-swarms (Chen & Montgomery, 2011). Multi swarm optimization has been used to solve many optimization problems. MSO is applicable to solving several machine-learning situations, such as approximating the weights and bias figures of an artificial neural network or approximating the weights of frail learners in ensemble organization and prediction (McCaffrey, 2016).

Zhang and Ding (2011), suggested a multi-swarm self-adaptive and cooperative particle swarm optimization (MSCPSO). Their approach make use of four sub-swarms: with sub-swarms 1 and 2 being basic, sub-swarm 3 manipulated by sub-swarms 1 and 2, whereas sub-swarm 4 is influenced by sub-swarms 1, 2 and 3. In the end all four sub-swarms make use of a cooperative strategy. Although it attained good performances in fine-tuning complex multimodal functions, the approach fail in its application to practical engineering problems.

The activity of MSO which forms a key procedure in its operation is that of calculating for its particle new velocity (9). The velocity of a particle is being swayed by a number of factors such as: the present location of a particle, a particle best recognized location, the best recognized location of whichever particle in the same swarm as the particle and finally, the finest recognised location of whichever particle in any swarm. Equation (10) computes a particle new position after a new velocity has been identified

$$(t+1) = w * v(t) + (c1 * r1) * (p(t) - x(t) + (c2 * r2) * * \big(s(t) - x(t)\big) + (c3 * r3) * (m(t) - (x)) \quad (9)$$

$$x(t+1) = x(t) + v(t+1) \quad (10)$$

where the term $v(t+1)$ represent the new velocity, $v(t)$ is the recent velocity, $x(t)$ is the recent location, $p(t)$ represent particle's best recognized location, $s(t)$ is the finest location of any particle in the particle's swarm and the finest location of whichever particle in any swarm is $m(t)$. In addition to the definition of terms, inertia factor, w and $c1$, $c2$ and $c3$ are all constant with universal names

as cognitive, social, and global weights with *r1*, *r2*, and *r3* being random values between 0 and 1 which present a randomization effect to every velocity update. Reasonable accepted values suggested by a number of particle swarm optimization researches presents 0.729, 1.49445, and 1.49445 for *w*, *c1*, and *c2* respectively. The constant along with the random values and the inertia factor institute a maximum change for every component of the new velocity. Those constants decide to a large extent how each term influences the activity of a particle. Constant *c3* is at its infancy in MSO and not much research has gone into it in obtaining a standard acceptable value.

One characteristic feature of Multi swarm optimisation technique is such that, a particle to be used may die and in such case, it needs to be substituted by a new particle at an arbitrary location, or it may immigrate such that in this case, the swarm is exchange with an arbitrarily chosen particle. The death and immigration instrument attach some element of uncertainty that help prevent the algorithm from returning non-optimal solution but a universal best solution. The next section outlines MSO algorithm.

3.1.2. Multi-Swarm Algorithm Pseudo-Code

Multi-Swarm Algorithm (McCaffrey, 2016) is one type nature-inspired heuristic algorithm which presents strong robustness and the ability to find optimal solution. The main steps of the algorithm are given below:

```
for each swarm iteration
   create particles at arbitrary locations
end for

whileas epoch < maximumEpochs iteration
  for-every swarm iteration
    for-every particle in swarm iteration

was particle dead?
      was particle immigrating?

calculate new velocity with concentration on
       current velocity, best particle location,
       best swarm location, and
       best overall location
     adopt new velocity to renew location
verify if new location is a new particle
       best, or a new swarm best, or
       a new universal best
    end every swarm
  end every particle
end while
return best universal location found
```

```
calculate new velocity with concentration on
      current velocity, best particle location,
      best swarm location, and
      best overall location

adopt new velocity to renew location
verify if new location is a new particle
      best, or a new swarm best, or
      a new universal best
   end every swarm
 end every particle
end while
return best universal location found
```

3.1.3. Bat Algorithm

The Bat algorithm (BA), a meta-heuristic algorithm is stimulated by the activities of bats for global optimization. It principles was inspired and developed in 2010 by Xin-She Yang. This algorithm is a multi-agent approach stimulated by the echolocation conducts of bats, with changing rates of pulse of emission and loudness, where a single pulse can last a little over thousandths of a second (ranging about 8–10 ms) (Altringham, 1996). Yet, the pulse has a continuous frequency which is more often than not in the range of 25–150 kHz which is equivalent to the wavelengths of 2–14 mm.

Yang identified three key features of the micro-bat to illustrate the fundamental structure of BA. These important characteristics as used by Yang are identified as follows (Yang, 2010):

I. Although greater numbers of species of bats make use of echolocation to hunt their prey, only a few fail to adopt this approach but may adopt another form of hunting technique. Conversely, the micro-bat is a renowned example of broadly using the echolocation technique. For this reason, the first characteristic is the behaviour of echolocation.

I. The frequency to which micro-bat transmits a predetermined frequency $f_{min}$with an inconsistent wavelength λ and the loudness A_o to look for prey.

II. Loudness by micro-bat can be regulated in several ways. Ideally, the loudness is believed to progress from an optimistic large value A_o to A_{min}, a minimum constant value.

Yang's method in simulations, make use of virtual bats in nature to identify the updated rules of their location x_i and velocities v_i in a D-dimensional search space. Fresh solutions x_1^t and velocities v_1^t at given time step t are obtained by

$$f_i = f_{min} + (f_{max} - f_{min})\,\beta,$$

$$v_1^t = v_1^{t-1} + (x_1^t - x_*)\,f_i, \tag{11}$$

$$x_1^t = x_1^{t-1} + v_1^t,$$

where *f* is the frequency the bat use in hunting for its prey, with the suffixes, min and max, standing for the minimum and maximum value, and $\beta \in [0, 1]$ represent the random vector obtained from a uniform distribution x_*, designate the present global near best solution which is obtained after evaluating all the results among all the *n* bats. A new solution for each bat is produced locally once a solution is chosen among the current best solutions, using random walk for the local search part. This is illustrated as:

$$x_{new} = x_{old} + \varepsilon A_t \tag{12}$$

where $\varepsilon \in [-1, 1]$ = random number, $A_t = (A_i^t)$ represent the average loudness of every bat at the present time step.

The process is iterative therefore in addition, A_i, the loudness and $r_i$the pulse emission rate are renewed accordingly as the iterations progresses. These formulas are illustrated in equation 13.

$$A_1^{t-1} = \alpha A_i^t,$$
$$r_1^{t+1} = r_i^0[1 - \exp(-\gamma t)], \tag{13}$$

Here α and γ are constants. For simplicity in Yang's experiments, $\alpha = \gamma = 0.9$ (Tsai et al., 2012). Per the idealization and approximations techniques employed, a summary of the basic steps of the bat algorithm is explained in the pseudo-code (Zhou et al., 2014).

Assume objective function f(x), x = [x_1,x_2, …,x_d]^T
Initialize the bat population x_i(I = 1, 2, …, n) and v_i
Identify pulse frequency f_i at x_i
Initialise pulse rates r_i and the loudness A_i
***While** (t < Max number of iterations)*
    Generate new solutions by adjusting frequency,
    and updating velocities and locations/solutions
***If** (rand > r_i)*
    Select a solution among the best solutions
    Generate a local solution around the selected solution
  end if
    Generate a new solution by flying randomly
***If** (rand < A_i& f (x_i) < f (x_*))*
    Accept the new solutions
    Increase r_i and reduce A_i
end if
    grade the bats and find the recent best x_*
end while
    Post-process results and visualisation

4. SIMULATION AND EXPERIMENTAL RESULTS

4.1. Parameters and Setting

The experimental settings is executed in Microsoft Visual C# 2010 version 10.0.3.319.1 RTMRel and carried out on a HP ProBook 4540s Computer with the processor of Intel(R) Core(TM) i3-3110M CPU at 2.40 GHz and 4096 GB memory. The general control parameters for both algorithms are the size of population and the number of maximum generation. The maximum numbers of cycles or generations used for the experiment is 1,000 with 6 dimensions (10, 15, 20, 25 and 30 and 35) of population size 25. The initialisation range [min, max] for all test functions are set to its global specific values.

Other specific control parameters and their values of the algorithms are presented in table 1.

Tab. 1. Parameter settings of algorithms (own study)

Multi Swarm Optimisation		**Bats Algorithm**	
Parameter Setting	Value/Range	Parameter Setting	Value/Range
Number of swarms	5	Initial Pulse, r_i^0	[0, 1]
Randomisation effect (r1, r2, r3)	[0, 1]	Initial loudness, A_i^0	[0.5, 2]
inertia weight w	0.729	$[f_{min}, f_{max})$	[0, 50]
cognitive, c1	1.49445	Random number, $\in$	[0, 1]
social, c2	1.49445		
global weight, c3	0.3645		
Death	0.001		
Immigrate	0.005		

4.2. Results and Findings

Tab. 2. Results obtained by BA and MSO Algorithms on f_1–f_6

Function	Algorithm Dim	BA		MSO	
		Value	Processing Time (ms)	Value	Processing Time (ms)
Griewank f_1	10	36.7378	24.7027	0.06151	411.6582
	15	149.251	27.6363	0.02464	560.5681
	20	159.262	25.5866	0.01232	690.3489
	25	299.961	28.2889	0.00000	858.6691
	30	265.519	30.7918	0.00000	995.7082
	35	427.209	30.3462	0.00000	1155.371
Mean Best		***222.99***	***27.89***	***0.0164***	***778.72***
Rastrigin f_2	10	351.313	22.999	0.00545	348.5491
	15	514.270	27.217	3.97984	487.8739
	20	709.450	29.992	11.9395	628.4436
	25	798.425	29.459	6.96471	764.1113
	30	886.754	30.810	17.9093	899.4347
	35	1429.693	30.708	37.8108	1041.189
Mean Best		***781.65***	***28.53***	***13.102***	***695.06***
Ackley f_3	10	18.9583	16.3993	0.00000	363.1926
	15	17.3149	28.6699	0.00000	505.2514
	20	19.5170	27.4071	0.00000	632.0096
	25	20.2093	28.9098	0.00000	772.8162
	30	19.9750	31.7634	0.00000	900.9023
	35	20.0563	32.1055	0.00000	1109.397
Mean Best		***19.338***	***27.543***	***0***	***713.93***
Rosenbrock f_4	10	26823760.1	21.0828	0.03984	291.2593
	15	23429221.4	26.007	0.44539	401.9415
	20	71148367.85	26.1592	0.28202	513.104
	25	94167522.53	29.5329	7.76763	613.4289
	30	131685159.98	32.0499	19.95645	711.3804
	35	155353591.25	28.117	25.03745	831.7191
Mean Best		***60464937.2***	***27.158***	***11.088***	***560.47***
Schwefel f_5	10	-79.3260	24.1545	-117.88	388.2503
	15	-98.2885	23.1431	-161.75	549.3576
	20	-109.9045	27.9776	-154.61	702.773
	25	-128.1006	23.5357	-171.96	912.7342
	30	-227.9366	30.6533	-354.54	1019.856
	35	-254.4494	34.9313	-398.49	1183.956
Mean Best		***-9.1676***	***27.399***	***-17984.54***	***792.82***
Michalewicz f_6	10	-3.7174	25.4515	-3.57858	536.9318
	15	-4.4228	24.2584	-3.96202	756.5302
	20	-4.9680	26.8576	-5.96101	990.6771
	25	-7.6216	27.1193	-6.46669	1202.339
	30	-7.8105	30.9676	-6.22626	1426.182
	35	-8.9912	35.9179	-8.39956	1659.426
Mean Best		***-6.2553***	***28.429***	***-5.7657***	***1095.3***

Fig 8 and 9 illustrates a graphical representation of the mean best value and time of the performance of the algorithms. From table 1.0 Bats algorithms performance on difficult functions such as Griewank and Ackley is on the downside.Functions with flat outer region with a large hole at its center seem to have no effect on MSO. MSO performance on these functions indicates its ability to moving out of the local minimum in the search space and locating the global minima. That said, MSO can converge to the minimum of both functions as dimensions increases. On Rosenbrock function, f_4, Bats algorithm returned very bad results which was far fetch from the global optima. Conversely MSO produce very good optimum solutions.

With regards to functions with deep valley with parabola shape, although MSO did return much better results in dimensions 10 and 15, later results obtain were a little larger. Bats algorithm on the other hand demonstrates its inability to return a better result returning largely very insignificant values with a flip flop approach (peaks and valleys). BA deteriorates substantially in its performance on this function than any other.

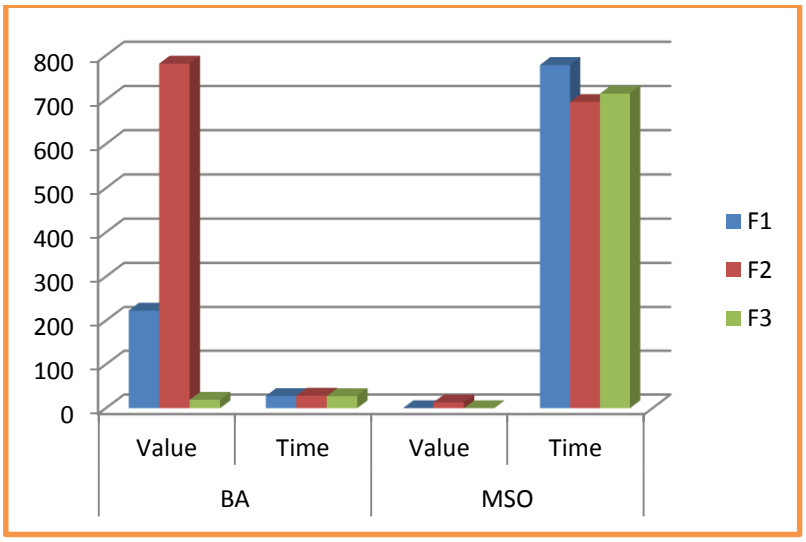

Fig. 8. Mean best value and time for f_1–f_3 by BA and MSO (own study)

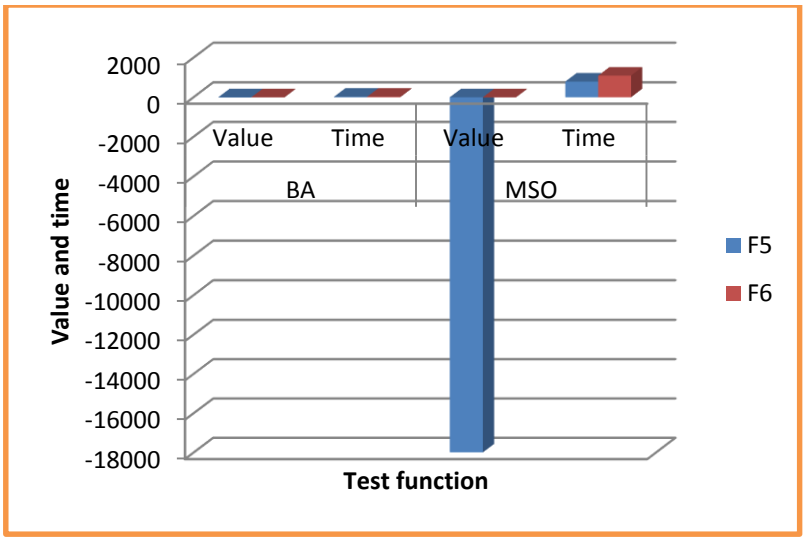

Fig. 9. Mean best value and time for f_5 and f_6 by BA and MSOs (own study)

From the computational experiment, f_5 and f_6 demonstrated better convergence rate on both algorithms as they returned close to optimal solutions. As MSO outperforms BA at f_5, returning best results, BA on the other hand outperforms MSO at f_6. BA shows faster, better convergence rate and a demonstration on its effectiveness in testing optimisation problem than MSO.

On the whole, BA appears to be better in terms of computation process speed rate. This may possibly be due to the outcome from producing completely different arbitrary numbers to be used in the generation procedures of the algorithm. MSO outperforms BA in five non-linear optimisation problems experimented by returning better, optimum and close to optimal values. The searching ability of groups of swarm is very effective for local optimisation thus, the MSO algorithm success in exhibiting better performance on optimising multivariable and multimodal functions. This proof indicates the powerful potential of MSO in solving non-linear optimization problems.

5. CONCLUSION

In this paper, a comparative study of the performance of population based algorithms and swarm intelligence was undertaken. The target is to compare the performance of BA and MSO algorithm in fine-tuning continuous unconstrained non linear optimisation problem. With the intention of demonstrating the performance of both algorithms, they were exposed to six multi dimensional numerical multimodal benchmark functions. From the experimented

simulation results, the conclusion was that, the MSO algorithm out performs BA in returning optimal results although lags behind in processing time. MSO possessed the tendency to escape from the local minimum, so therefore can be used efficiently for multimodal and multivariable optimization. There are several gray areas which remain for future studies such as the exploration into the unique behaviour and characters of the bench mark functions on meta-heuristic optimisation algorithms and the effects of the parameters on the performance of the algorithms

REFERENCES

Altringham, J. D. (1996). *Bats: Biology and Behaviour*. Oxford University Press.

Blum, Ch., Roli, A., & Sampels, M. (2008). *Hybrid Metaheuristics. An Emerging Approach to Optimization.* Springer.

Chen, S., & Montgomery, J. (2011). Selection Strategies for Initial Positions and Initial Velocities in Multi-optima Particle Swarms. *Gecco-2011: Proceedings of the 13th Annual Genetic and Evolutionary Computation Conference*, 53-60.

Ciurana, J., Arias, G., & Ozel, T. (2009). Neural Network Modeling and Particle Swarm Optimization (PSO) of Process Parameters in Pulsed Laser Micromachining of Hardened AISI H13 Steel. *Materials and Manufacturing Processes, 24*(3), 358-368. doi:10.1080/10426910802679568

Example Functions (single and multi-objective functions). Retrieved August, 2016, from http://www.geatbx.com/docu/fcnindex-01.html#P150_6749

Friedman, J. H. (1994). An overview of predictive learning and function approximation. In V. Cherkassky, J. H. Friedman, & H. Wechsler (Eds.), *Statistics to Neural Networks. Theory and Pattern Recognition Applications. NATO ASI Series F* (pp. 1-61). Springer.

Gal, T., & Nedoma, J. (1972). Multiparametric Linear Programming. *Management Science Series a-Theory, 18*(7), 406-422. doi:10.1287/mnsc.18.7.406

Madić, M., Marković, D., & Radovanović, M. (2013). Comparison of meta-heuristic algorithms for solving machining optimization problems. *Mechanical Engineering*, *11*(1), 29-44.

McCaffrey, J. D. (2016, August). Multi-Swarm Optimization with C#. Retrieved from https://jamesmccaffrey.wordpress.com/2013/09/16/multi-swarm-optimization-with-c

Pal, S. K., Rai, C. S., & Singh, P. A. (2012). Comparative Study of Firefly Algorithm and Particle Swarm Optimization for Noisy Non-Linear Optimization Problems. *I.J. Intelligent Systems and Applications, 10*, 50-57. doi: 10.5815/ijisa.2012.10.06

Pansare, V. B., & Kavade, M. V. (2012). Optimization of cutting parameters in multipass turning operation using ant colony algorithm. *International Journal of Engineering Science & Advanced Technology*, 2(4), 955-960.

Rao, R. V., Pawar, P. J., & Davim, J. P. (2010). Optimisation of process parameters of mechanical type advanced machining processes using a simulated annealing algorithm. *International Journal of Materials & Product Technology, 37*(1-2), 83-101.

Samanta, S., & Chakraborty, S. (2011). Parametric optimization of some non-traditional machining processes using artificial bee colony algorithm. *Engineering Applications of Artificial Intelligence, 24*(6), 946-957. doi:10.1016/j.engappai.2011.03.009

Tsai, P. W., Pan, J. S., Liao, B. Y., Tsai, M. J., & Istanda, V. (2012). Bat Algorithm Inspired Algorithm for Solving Numerical Optimization Problems. *Applied Mechanics and Materials, 148-149*, 134-137. doi:10.4028/www.scientific.net/AMM.148-149.134

Virtual Library of Simulation Experiments: Test Functions and Datasets. Retrieved August, 2016, from https://www.sfu.ca/~ssurjano/optimization.html

Yang, X. S. (2008). *Nature-Inspired Metaheuristic Algorithms.* Luniver Press.

Yang, X. S. (2010). A new metaheuristic Bat-inspired algorithm. In J. R. González, D. A. Pelta, C. Cruz, G. Terrazas, & N. Krasnogor (Eds.), *Nature Inspired Cooperative Strategies for Optimization (NICSO 2010)* (pp. 65-74). Springer.

Yuan, B. (2016, August). A Brief Introduction to Global Optimization. Retrieved from http://boyuan.global-optimization.com/optimization.htm

Zain, A. M., Haron, H., & Sharif, S. (2010). Application of GA to optimize cutting conditions for minimizing surface roughness in end milling machining process. *Expert Systems with Applications, 37*(6), 4650-4659. doi:10.1016/j.eswa.2009.12.043

Zhang, J. Z., & Ding, X. M. (2011). A Multi-Swarm Self-Adaptive and Cooperative Particle Swarm Optimization. *Engineering Applications of Artificial Intelligence*, *24*(6), 958-967. doi:10.1016/j.engappai.2011.05.010

Zhou, Y. Q., Xie, J., Li, L. L., & Ma, M. Z. (2014). Cloud Model Bat Algorithm. *The Scientific World Journal, 2014*. doi:10.1155/2014/237102