Abstract: This paper describes an object-oriented software system for continuous optimization by a new metaheuristic method, the Bat Algorithm, based on the echolocation behavior of bats. Bat algorithm was successfully used for many optimization problems and there is also a corresponding program in MATLAB. We implemented a modified version in C# which is easier for maintenance since it is object-oriented and which uses threads and significantly increases execution speed on multicore processors. The application includes flexible GUI (graphical user interface) and it was successfully tested on standard benchmark problems.

Key-Words: Bat algorithm, Optimization metaheuristics, Software system, Swarm intelligence, Nature-inspired algorithms

1 Introduction

In decision theory, mathematics and information science, optimization is based on the selection of the best solution within certain given domain, which can minimize or maximize some function. A lot of problems in business and industry belong in the group of intractable continuous or discrete optimization problems. Different approaches have been developed for solving such optimization problems [1]. These approaches can be classified into following groups: classical methods, stochastic algorithms, population based algorithms and other approaches. Classical methods for the same origin values follow the same path and always give the same final solutions. Stochastic algorithms are based on the randomization, and they give different final solutions each time, even starting from the identical initial values. Population based algorithms deal with a set of solutions trying to improve them each iteration. Although the difference exists between them, they will find the similar optimal solutions each time. By the nature of phenomenon simulated, population based algorithms can be classified into: evolutionary algorithms (EA) and swarm intelligence based algorithms.

The Evolutionary algorithms are optimization techniques [2] which are based on the Darwin’s principle of survival of the fittest. The Evolutionary algorithms consist of the following disciplines: genetic algorithms (GA), evolution strategies, genetic programming, evolutionary programming and differential evolution. Although all these algorithms or methods have been developed independently, when solving problems all of them share characteristics like: variation operators and selection operators. EA are distinguished by their representation of solutions. GA supports the binary representation of solutions. Evolution strategies and differential evolution work on real-valued solutions. EA can be applied on a lot of area of optimization problems as modelling and simulation. Essentially, the differential evolution (DE) has successfully been employed in the following areas of optimization: function optimization [3], large-scale global optimization [4], and many others.

The Swarm intelligence is a research branch that models the population of interacting agents. Although centralized component that controls the behaviour of individuals does not exist in swarm intelligence systems, local interactions between all individuals often gives intelligent global behaviour. Ant colonies, schooling of fish, flocking of birds, bee’s behaviour and many others are examples of these systems. These behaviours have inspired researchers to implement them in computer software for optimization problems.

The best well-known classes of the swarm-intelligence algorithms are: ACO (ant colony optimization) [5], [6], [7], [8], PSO (particle swarm optimization) [9], ABC (artificial bee colony) [10], [11], [12], CS (cuckoo search) [13], FA (firefly algorithm) [14], HSO (human seeker optimization) [15] for image thresholding [16] with entropy function [17] and BA (bat algorithm).
Echolocation is an important feature of bat behaviour. It means that bats emit a very loud sound pulse and listen for the echo that bounces back from the surrounding objects whilst flying. Their pulses vary in properties and can be correlated with their hunting strategies, depending on the species. Most bats use short, frequency-modulated signals to sweep through about an octave, while others more often use constant-frequency signals for echolocation. Their signal bandwidth varies depending on the species, and often increased by using more harmonics. This phenomenon has been inspired Yang [18] to develop the Bat Algorithm (BA). This algorithm gives good results when dealing with lower-dimensional optimization problems, but may become problematic for higher-dimensional problems because of its tending to converge very fast initially.

In this paper, we present a framework for the Bat Algorithm (BA) proposed by Yang [19]. We developed our BA software for solving combinatorial and numeric optimization problems in C# programming language.

2 BAT algorithm

The BAT algorithm is new population based meta-heuristic approach proposed by Xin-She Yang [19]. The algorithm exploits the so-called echolocation of the bats. The bats use sonar echoes to detect and avoid obstacles. It’s generally known that sound pulses are transformed into a frequency which reflects from obstacles. The bats navigate by using the time delay from emission to reflection. The pulse rate is usually defined as 10 to 20 times per second, and it only lasts up about 8 to 10 ms. After hitting and reflecting, the bats transform their own pulse into useful information to explore how far away the prey is. The bats are using wavelength \( \lambda \) that vary in the range from 0.7 to 17 mm or inbound frequencies \( f \) of 20-500 kHz. Hence, we can also vary \( f \) while fixing \( \lambda \), because \( \lambda \) and \( f \) are related due to the fact \( \lambda f \) is constant. The pulse rate can be simply determined in the range from 0 to 1, where 0 means that there is no emission and 1 means that the bat’s emitting is their maximum [20], [21]. The bat behaviour can be used to formulate a new BA. Yang used three generalized rules when implementing the bat algorithms:

1. All bats use echolocation to sense distance, and they also ‘know’ the difference between food/prey and background barriers in some magical way;

2. Bats fly randomly with velocity \( v_i \) at position \( x_i \), with a fixed frequency \( f_{\text{min}} \), varying wavelength \( \lambda \) and loudness \( A_0 \) to search for prey. They can automatically adjust the wavelength of their emitted pulses and adjust the rate of pulse emission \( r \in [0,1] \), depending on the proximity of their target;

3. Although the loudness can vary in many ways, we assume that the loudness varies from a large (positive) \( A_0 \) to a minimum constant value \( A_{\text{min}} \).

Main steps of the algorithm are given below:

Initialization;
Repeat
Generation of new solutions;
Local searching;
Generation of a new solution by flying randomly;
Finding the current best solution;
Until (requirements are met).

In BA algorithm, initialization of the bat population is performed randomly. In our simulations, we use virtual bats naturally. Namely, generating new solutions is performed by moving virtual bats according to the following equations:

\[
\begin{align*}
\lambda_i &= \lambda_i^{t-1} + (\lambda_i^{t-1} - \lambda^*) f_i \\
\lambda_i &= \lambda_i^{t-1} + (\lambda_i^{t-1} - \lambda^*) f_i \\
x_i &= x_i^{t-1} + v_i^{t-1} + v_i
\end{align*}
\]

where \( \beta \in [0,1] \) is a random vector drawn from a uniform distribution. Here \( x^* \) is the current global best location (solution) which is located after comparing all the solutions among all the bats. In our implementation, we will use \( f_{\text{min}}=0 \) and \( f_{\text{max}}=100 \), depending the domain size of the problem of interest. Initially, each bat is randomly assigned a frequency which is drawn uniformly from \([f_{\text{min}}, f_{\text{max}}]\). A random walk with direct exploitation is used for the local search that modifies the current best solution according the equation:

\[
x_{\text{new}} = x_{\text{old}} + \varepsilon A'
\]

where \( \varepsilon \in [-1,1] \) is a random number, while \( A' \) is the average loudness of all the best at this time step. The local search is launched with the proximity depending on the rate \( r_i \) of pulse emission. As the loudness usually decreases once a bat has found its prey, while the rate of pulse emission increases, the loudness can be chosen as any value of
We’ve developed our software for BA algorithm called BATapp. We could use existing Xin-She Yang’s MATLAB software, but we chose to develop a new version because we wanted to implement few improvements. Firstly, in order to make algorithm execute faster, we used multiple threads. Each algorithm’s run executes within a different thread, so it runs much faster. Each thread puts best result in an array length of number of runs. Then, we calculate mean result according to the values stored in this array. Threads do not make conflicts with each other, they execute independently. We noticed great performance increase when we run our software on multiple core processors because each thread execute on different core in parallel way. Speed test will be presented in section 4.1.

Secondly, our software is object-oriented. With object-oriented concept, software scalability and maintenance is much easier. So, if we want to implement new logic for different optimization problems, it will take substantially less time.

We chose to develop BATapp in C# because of its many advantages over C, C++ and Java. We prefer C# over C even though C is faster. With C# we could gain more control over BA algorithm execution. Some of C# advantages which made us chose this programming language are:

- Usually it is much more efficient than Java and runs faster.
- CIL (Common (.NET) Intermediate Language) is a standard language, while java byte codes are not.
- It has more primitive types (value types), including unsigned numeric types.
- Indexers let you access objects as if they were arrays.
- Conditional compilation.
- Simplified multithreading.
- Operator overloading. It can make development a bit trickier but they are optional and sometimes very useful.
- Limited use of pointers if you really need them, as when calling unmanaged (native) libraries which does not run on top of the virtual machine (CLR).
- More clean events management using delegates.

We developed our software using Visual Studio 2010 Ultimate environment and .NET Framework 4. A framework is a special kind of software library that is similar to an application program interface (API) in the class of packages that make possible faster development of applications. Two main components of .NET Framework are Common Language Runtime (CLR) and Class Library. CLR is the .NET runtime environment responsible for program execution management and for providing container services—debugging, exception management, memory management, profiling, and security. The CLR provides a managed environment for code execution, which makes code more secure by protecting the code from doing things such as illegal memory access operations, manages memory for the program and adds additional runtime support not available in native programs, like garbage collection. The .NET class libraries are pre-written classes that provide a rich assortment of pre-defined code.

So, using previously described environment makes our code more robust, errorless and performance is much better.

There are a large number of connections between classes in our program. BA algorithm cannot be used in its basic form for all function optimization problems. So, we created abstract class BATAbstract which is inherited by problem specific classes. BATAbstract has the following methods: init, initial, memorizeBestSolution, calculateFitness, calculateFrequencies, calculateVelocities, findBestSolution, run. These methods which will be briefly described, form the basis of BAT metaheuristics. Method calculateFitness calculates the fitness of a solution. Method memorizeBestSolution memorizes best solution and value of global min found. Method init initializes variables and counters. Variables are initialized within ranged defined by the user. Method initial initializes bat populations (solutions) at the beginning of the process. Methods calculateFrequencies and calculateVelocities calculate frequencies and velocities according to the equations (1) and (2) respectively. Method findBestSolution calculates the current best solution according the equation (4). Method run is used for implementing multiple thread functionality into our software. In the Run method, previously described functions are being executed. Pseudo-code for Run method is given in Figure 1.
initial
memorizeBestSolution
Repeat
calculateFrequencies
calculateVelocities
findBestSolution
memorizeBestSolution
Until max iterations are met

Fig. 1: Pseudocode for Run method

Screenshot of basic *Graphical user interface* (GUI) of BATapp can be seen in Figure 2. As we can see from the Fig.2, user can adjust multiple parameters for BAT algorithm. Parameters are divided into two groups: Bat control parameters and problem specific parameters.

**Control parameters are:**
- **Bat Num NP** is number of bats in the echolocation;
- **Loudness** is the average loudness of all the bats;
- **Pulse rate** is bat’s emitting short and loud sound impulses;
- **Frequency min** is the minimum wavelength of the emitted pulses;
- **Frequency max** is the maximum wavelength of the emitted pulses;
- **Max Cycle** defines the number of cycles for foraging. This is a stopping criterion.

**Problem specific parameters are:**
- **Param Num D** is the number of parameters of the problem to be optimized;
- **Runtime** defines the number of times to run the algorithm;
- **Lower bound** is lower bound of problem parameters;
- **Upper bound** is upper bound of problem parameters.

Fig. 2: Screenshot of BAapp GUI

In the *results* text area, we can see results for each algorithm’s run, and below, mean results of all runs is shown. Button *details* give us additional information about the function to be optimized (Fig.3 and Fig.4).

Fig. 3: Results for Sphere function

In our algorithm, the dimension of the problem significantly affects the result optimization. To test the impact of the problem dimension on the results, three different sets of dimension were taken into account: \(D=10\), \(D=20\) and \(D=30\). The functions with dimension \(D=10\) were limited to maximally 1000 evaluations, the functions with dimension \(D=20\) were limited to 2000, while the functions with dimension \(D = 30\) to 3000 evaluations. The initial loudness was set to \(A_0 = 0.5\) and so was the initial pulse rate \(r_0 = 0.5\). The frequency was taken from interval \(f_i \in [0.0, 2.0]\). Furthermore, we have tried to use different bat population size from \(NP=10\) to 250, and we have found that for most problems, \(NP=15\) to 50 is sufficient. Therefore, we use a fixed population \(NP=40\) for all our simulations. The algorithm optimized each function 30 times and results were measured according to the best, worst and mean values in these runs.

Fig. 4: Additional information about selected function
Recent Researches in Medicine, Biology and Bioscience

4 Tests and Results
For test purposes, we created test application in C# without multiple threads, like Yang`s software in MATLAB programming language. Our test suite consists of five standard benchmark functions:

The Sphere's function has a value 0 at its global minimum is (0,0,...,0). Definition:

\[ f_1(x) = \sum_{i=1}^{D} x_i^2 \]

where \( x_i \in [-5.12, 5.12] \).

The Rosenbrock's function has a value 0 at its global minimum is (1,1,...,1). Definition:

\[ f_2(x) = \sum_{i=1}^{D-1} [100(x_i^2 - x_{i+1}^2) + (x_i - 1)^2] \]

where \( x_i \in [-2.048, 2.048] \).

The Griewank's value is 0, and its global minimum is (0,0,...,0). Definition:

\[ f_3(x) = \sum_{i=1}^{D} \frac{x_i^2}{4000} - \prod_{i=1}^{D} \cos(x_i / \sqrt{i}) + 1 \]

where \( x_i \in [-50, 50] \).

The Rastrigin's function has value 0, and global minimum (0,0,...,0). Definition:

\[ f_4(x) = 10D + \sum_{i=1}^{D} (x_i^2 - 10 \cos(2\pi x_i)) \]

where \( x_i \in [-5.12, 5.12] \).

The fifth function is Ackley's function whose value is 0 at its global minimum (0,0,...,0). Definition:

\[ f_5(x) = 200 + e^{-0.2 \sqrt{\frac{1}{D} \sum_{i=1}^{D} x_i^2}} - e^{\frac{1}{D} \sum_{i=1}^{D} \cos(2\pi x_i)} \]

where \( x_i \in [-15, 30] \).

We ran two types of tests. First, we ran speed test, where we compare single thread application (STA) to multiple threaded application (MTA) BATapp (as described in section 3). Second, we ran optimization tests. Tests were done on AMD Core2Duo TK-55 (1.8 GHz, 2 x 256 KB L2 cache) with 2GB of RAM on Windows 7 Operating System in Visual Studio 2010. In Table 1, we show results of speed tests (in seconds) between single thread and multiple threads BAT software.

<table>
<thead>
<tr>
<th>Single thread app</th>
<th>D</th>
<th>f_1</th>
<th>f_2</th>
<th>f_3</th>
<th>f_4</th>
<th>f_5</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>3.931</td>
<td>7.482</td>
<td>5.416</td>
<td>4.994</td>
<td>3.125</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>15.05</td>
<td>30.375</td>
<td>20.42</td>
<td>18.732</td>
<td>9.009</td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>33.306</td>
<td>68.828</td>
<td>45.736</td>
<td>41.467</td>
<td>17.501</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>One run one thread</th>
<th>D</th>
<th>f_1</th>
<th>f_2</th>
<th>f_3</th>
<th>f_4</th>
<th>f_5</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>1.977</td>
<td>3.615</td>
<td>2.731</td>
<td>2.315</td>
<td>1.463</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>7.507</td>
<td>15.096</td>
<td>10.584</td>
<td>8.975</td>
<td>4.874</td>
<td></td>
</tr>
</tbody>
</table>

Table 1: Speed test results
From Table 1, we can see that MTA is substantially faster than STA. So, when each run executes within different thread, great performance gain is achieved.

<table>
<thead>
<tr>
<th>D</th>
<th>Val</th>
<th>f_1</th>
<th>f_2</th>
<th>f_3</th>
<th>f_4</th>
<th>f_5</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>Best</td>
<td>2.0992</td>
<td>0.1297</td>
<td>0.2584</td>
<td>20.8992</td>
<td>1.3687</td>
</tr>
<tr>
<td></td>
<td>Worst</td>
<td>9.6040</td>
<td>1.3065</td>
<td>77.6180</td>
<td>12.2353</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Mean</td>
<td>4.5506E-005</td>
<td>1.6826</td>
<td>53.1390</td>
<td>7.8790</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>Best</td>
<td>9.1941</td>
<td>0.0124</td>
<td>91.5762</td>
<td>1.5565</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Worst</td>
<td>19.5247</td>
<td>1.3233</td>
<td>154.2627</td>
<td>1.5569</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Mean</td>
<td>2.0131E-004</td>
<td>1.6651</td>
<td>123.0168</td>
<td>1.5567</td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>Best</td>
<td>15.8279</td>
<td>2.9947E-005</td>
<td>161.2700</td>
<td>1.6132</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Worst</td>
<td>29.3905</td>
<td>0.1525</td>
<td>273.1722</td>
<td>1.6134</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Mean</td>
<td>4.7397E-004</td>
<td>23.6417</td>
<td>203.6941</td>
<td>1.6133</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>D</th>
<th>Val</th>
<th>f_1</th>
<th>f_2</th>
<th>f_3</th>
<th>f_4</th>
<th>f_5</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>Best</td>
<td>1.3566E-005</td>
<td>0.0254</td>
<td>0.2877</td>
<td>30.8522</td>
<td>1.3685</td>
</tr>
<tr>
<td></td>
<td>Worst</td>
<td>5.6175E-005</td>
<td>0.7407</td>
<td>2.3635</td>
<td>0.7827</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Mean</td>
<td>4.3994E-005</td>
<td>1.3789</td>
<td>51.7793</td>
<td>5.2922</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>Best</td>
<td>6.1692E-005</td>
<td>1.6261</td>
<td>84.5816</td>
<td>11.8183</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Worst</td>
<td>23.6417</td>
<td>1.5567</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Mean</td>
<td>2.9848E-004</td>
<td>1.5565</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>Best</td>
<td>6.3146</td>
<td>2.6670</td>
<td>1.6132</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Worst</td>
<td>123.0168</td>
<td>1.6134</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Mean</td>
<td>4.7397E-004</td>
<td>203.6941</td>
<td>1.6133</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 2: Results for function optimization
In Table 2, we show results of optimization tests between STA and MTA BATapp. From Table 2 can be seen that BATapp gives noticeable better results than ordinary BAT software. Notice, if we increase the number of iterations for instance Rosenbrock’s function on maxCycle=2500, and decrease the dimension D from 20 to 5, we shall get that the best solution (min solution) is equal 0.0017. Hence, this solution tends to the global minimum. So, by fine-tuning of parameters, we can be obtained the optimal global solution with of course a large number of iterations.
5 Conclusion
We implemented and tested a software system in C# for optimization problems based on a modification of Yang’s BA algorithm and corresponding software. Object-oriented design and appropriate GUI allow for easy modifications and applications to different optimization problems. Performance was tested and proved to be superior to existing software since use of threads better utilizes multicore processors. Benchmark problems that are used in the literature were tested and system is ready to be applied to new problems.

References: