Near Real-time Performance of Population-based Nature-Inspired Algorithms on Cheaper and Older Smartphones

Iztok Fister Jr.*, Suash Deb†, Iztok Fister*
*University of Maribor,
Faculty of Electrical Engineering and Computer Science
Koroška cesta 46, 2000 Maribor,
Slovenia
iztok.fister1@um.si
†Victoria University,
Decision Sciences and Modeling Program,
Melbourne, Australia
‡IT & educational Consultant, Ranchi, Jharkhand, India

***Abstract*—For solving optimization problems, stochastic population-based nature-inspired algorithms use inspirations from nature. Despite their applicability in real-world environments, their bottleneck is high time complexity. Usually, they are searching for optimal solutions on computing devices of full computational power. However, in some situations, we deal with devices of limited computational power. Some examples of such devices are smartphones, which have been becoming very powerful for running various applications. However, there is still a lack of researches that would study the performance of nature-inspired algorithms on these devices. In this paper, we analyze the performance of one member of the nature-inspired algorithms, the so-called Bat algorithm, on the Android smartphone. Although smartphones nowadays offer a computational power comparable with the personal computer, we focus on the cheaper and older smartphones that are most widespread today.**

***Index Terms*—nature-inspired algorithms, smartphones, optimization**

I. Introduction

Stochastic population-based nature-inspired algorithms are a suitable tool for coping with optimization problems that have arisen in many areas. These kinds of algorithms consist of a population of individuals that undergo variation operators governed by some principles inspired by nature. For example, individuals in Genetic Algorithms (GA) [5] (a member of Evolutionary Algorithms (EAs)) that are governed by Darwinian evolution, undergo operations of mutation, crossover and selection operators, while individuals in the Bat Algorithm (BA) [13] (a member of Swarm Intelligence (SI) based algorithms) that mimics the echolocation observed by microbats, undergo the variation operator guided by this physical phenomenon.

Many variants of the algorithms have been proposed for solving optimization problems. Due to the time complexity, the algorithms are devoted to executing on powerful computing devices that have enough resources, such as, for example, a lot of memory and processor power (i.e., many CPU cores). Moreover, some of these variants were even tailored to specific hardware. Let us mention a few examples:

- Graphics Processing Units (GPU): Were firstly intended for calculations in computer graphics, as well as in video games, but were later transferred also to other fields. GPUs allow the parallelization mechanism easily and fast floating-point operations [14]. So far, researchers have already implemented many variants [14], [9], [6], [2] of nature-inspired algorithms for GPUs‘ platforms and reported high gains over CPUs.
- Field Programmable Gate Array (FPGA): Are intended to decrease computation time by performing as many calculations as possible simultaneously [11]. Similar to GPUs, researchers also implemented many nature-inspired algorithms for FPGA platforms [11], [1].
- Limited hardware: Are special implementations of nature-inspired algorithms for environments where a full power computing device may not be available. It is typical for robotics and control problems [8]. Some works illustrating a solution for such environments can be found in [10], [8].

Smartphones are a new generation of mobile phones that can be considered as personal computers. They run a mobile operating system that is tailored to the limited architecture of mobile devices. At the moment, the most popular mobile operating systems include Android, Blackberry and iOS (formerly iPhone OS)[1]. An interesting aspect of these mobile operating systems is the almost effortless development of applications by users who are not professional programmers. No wonder authors of these operating systems prepared a lot of software development kits that simplify development, testing, and even

[1]Sorted alphabetically.

deployment of various applications that can use all capabilities of this limited hardware.

Android is a mobile operating system based on the Linux operating system. It is an open source software developed originally by Google. In this paper, we explore ways for implementing nature-inspired algorithms running on these devices. As is commonly known, these kinds of algorithms are time complex. On the other hand, most of the mobile platforms for running this software, especially older ones, are limited with computational power. In these cases, how to develop applications using the nature-inspired algorithms in real-time on this limited hardware is very important. However, running these kinds of applications on mobile devices may open new possibilities for their usage. For instance, development of some fitness & health applications for athletes allow monitoring of several load indicators during the training that can contribute to improving the performances of the athlete online.

Therefore, this paper tries to find an answer to the following question: Are the cheaper smartphones able to run population-based nature-inspired algorithms in near real-time? In line with this, the BA nature-inspired population-based algorithm was developed that was applied to a benchmark function suite consisting of 10 functions that was run on two platforms: PC, and Android smartphone.

In a nutshell, the paper is organized as follows. Section II presents an analysis of the main nature-inspired algorithms‘ components, and outlines some features of the Bat Algorithm, Section III proposes a new implementation of the Bat Algorithm. In Section IV, experiments and results are presented, while Section V concludes the paper and outlines the future directions.

II. Analysis of nature-inspired algorithm's components

The main components of population-based SI-based algorithms operating without crossover can be summarized as follows [3]:

- representation,
- initialization,
- fitness function evaluation,
- variation operators,
- replacement,
- termination condition.

Representation is intended for the representation of individuals. Mostly , individuals are represented as real numbers. However, binary representation or a tree structure presentation is also very suitable for nature-inspired algorithms. Initialization serves for setting the control parameters and generating the initial population. Usually, a population is initialized randomly, while, in some cases, authors also use other heuristic methods. Fitness function evaluates the quality of individuals in a population and presents a connection with the problem to be solved. Variation operators, such as, for example, crossover or mutation, change the genetic material of individuals. Replacement takes care of eliminating the bad solutions. Finally, the termination condition defines when the algorithm must be terminated.

A. Bat algorithm

The Bat Algorithm (BA) [12] is an SI-based algorithm created for solving optimization problems. It consists of a population of bats that are governed by the physical rules of echolocation. The BA was shown to perform well by solving continuous, as well as discrete optimization problems of smaller dimensions. A very interesting aspect of the BA is its low time complexity compared to the other algorithms from this family. A pseudo-code of the canonical BA is depicted in Algorithm 1,

Algorithm 1 Canonical Bat algorithm

```
1: init_bats();
2: evaluate_the_new_population;
3: find_the_best_solution($\mathbf{x}_{best}$);
4: while termination_condition_not_meet do
5:    for $i = 1$ to $Np$ do
6:       generate_new_solution($\mathbf{x}_i$);
7:       if $\mathrm{rand}(0,1) > r_i$ then
8:          improve_the_best_solution($\mathbf{x}_{best}$)
9:       end if
10:      evaluate_the_new_solution($\mathbf{y}$);
11:      if $f_{new} \leq f_i$ and $\mathrm{N}(0,1) < A_i$ then
12:         $\mathbf{x}_i = \mathbf{y}$; $f_i = f_{new}$;
13:      end if
14:      find_the_best_solution($\mathbf{x}_{best}$);
15:   end for
16: end while
```

from which it can be seen that the BA searches for the good solutions in the search space using two strategies: (1) Exploration, implemented in 'generate_new_solution', and (2) Exploitation, implemented in 'improve_the_best_solution' function. The former modifies the trial solution according to the physical rules of echolocation, while the latter is an implementation of the local search in the vicinity of the best solution. Both strategies are balanced using the parameter r_i.

B. Bat algorithm implementation issues

As mentioned before, most of the SI-based algorithms consist of six components that are not equally computationally expensive. The past studies in [4], [13] showed that the BA can be considered as a very fast and computationally inexpensive algorithm. However, there may still be some issues that can lower the execution time of the BA. If we refer to Sec. II, we can observe all the advantages and disadvantages of the BA implementation.

Thus, the BA is initialized randomly using a uniform distribution. Hence, there are not any special computational loads that might affect its performance on a mobile device. Additionally, individuals are presented as real-valued vectors. Therefore, no explicit genotype-phenotype mapping is needed for a fitness function evaluation. The fitness function depends on the complexity of the problem to be solved, while the most important parts of the algorithm are variation operators. The operators in the BA are not time consuming, especially when it is assumed that there is still a possibility to remove local search step [4], [13]. Original local search is actually a random walk with direct exploitation. Replacement in the Bat Algorithm is conducted in the fashion of a simulated annealing heuristic [7]. The termination condition in the BA can be either the number of generations [2] or a number of function evaluations.

A summary of the BA's components and its alternatives is presented in Table I.

TABLE I
SUMMARY OF BAT ALGORITHM MAIN COMPONENTS WITH SOME ALTERNATIVES IN IMPLEMENTATION.

Component	Bat algorithm	Better alternative
Representation	real-valued vectors	N/A
Initialization	uniform	N/A
Fitness function evaluation	N/A	N/A
operators	mutation according to physical rules of bat echolocation, use of local search	without local search
replacement	in the fashion of simulated annealing	without simulated annealing step
termination condition	iterations	N/A

III. DESIGN AND IMPLEMENTATION OF NATURE-INSPIRED ALGORITHMS FOR ANDROID

The Android studio was used for the development of the Android application. This is a very convenient, robust, easy to use development kit for developing Android applications. The front end of the application was developed in Java, while the BA was implemented fully in C++. Android supports C++ for the development of applications. Thus, the C++ code was placed in a cpp directory. It was compiled by CMake. The code was compiled into a native library that the build system Gradle can package within APK (Android Package Kit) [3]. Java communicates with the native library thorough the Java Native Interface (JNI).

Basically, the layout of the smartphone application is very simple, where each button on the screen represents a definite benchmark function that needs to be optimized. Let us mention that 10 well known benchmark functions are used in our study, i.e., Griewank, Rastrigin, Rosenbrock, Ackley, Schwefel, DeJong, Easom, Michalewicz, Xin-She, and Zakharov. The layout of the Android application is shown in Figure 1. Implementation on the PC was run in a Linux terminal without GUI.

Let us mention that execution time was measured using the chrono library that comes with C++ 11 standard.

[2]Let us mention that counting every function evaluation may be much more time consuming that iterating through a number of generations.

[3]https://developer.android.com/studio/projects/add-native-code

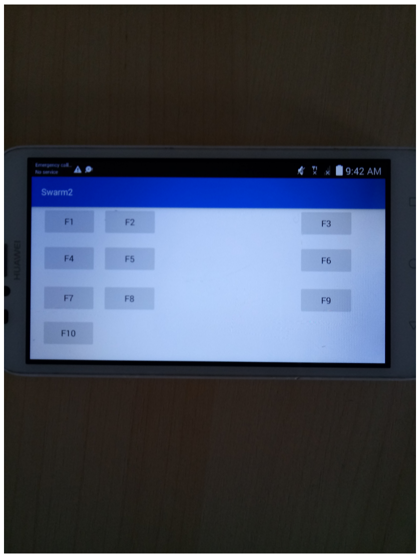

Fig. 1. Android application.

IV. EXPERIMENTAL WORK

In this experiment, we used C++ application that was rewritten from the main implementation of the Bat Algorithm in Matlab programming language. It also corresponds with the basic publication about Bat Algorithms [12]. The same algorithm was run on the following two different platforms: Smartphone and PC.

The mobile application was installed on a Huawei smartphone Y625-U32 with the following specifications:

- CPU model: Qualcomm MSM8212
- Number of cores: Quad-core
- CPU frequency: 1.2 GHz
- ROM: 4GB
- RAM: 1GB
- Android version 4.4.2; Kernel version 3.4.0

For comparison of the same implementation, we used a personal computer of the following characteristics:

- Processor: 8x Intel(R) Xeon(R) CPU E3-1240 v5 @ 3.50GHz
- Memory: 16344MB
- Operating System: Ubuntu 16.04.5 LTS

Parameter settings of the BA on both platforms were set the same, as follows: $D = 10$, $NP = 10$, $MAX\_RUNS = 25$, $MAX\_GEN = 1,000$, $A = 0.5$, $r = 0.5$, $Q_{min} = 0.0$, $Q_{max} = 2.0$.

A. Results

The numerical results of the experiments are presented in Table II,

which is divided into two parts, where the former presents the results obtained by optimizing the benchmark functions on the personal computer, while the latter the same obtained on the observed smartphone. Obviously, the results are presented according to the specific functions from the benchmark suite.

TABLE II
RESULTS OF IMPLEMENTATIONS.

Function	Personal computer						Smartphone					
	t_{best}	t_{worst}	t_{mean}	t_{median}	t_{stdev}	t_{total}	t_{best}	t_{worst}	t_{mean}	t_{median}	t_{stdev}	t_{total}
Griewank	0.0472	0.0567	0.0478	0.0472	0.0017	1.48	0.4433	0.5028	0.4633	0.4611	0.0132	14.36
Rastrigin	0.0482	0.0614	0.0491	0.0485	0.0023	1.52	0.4288	0.4603	0.4393	0.4367	0.0072	13.62
Rosenbrock	0.0453	0.0578	0.0460	0.0456	0.0022	1.43	0.4670	0.5094	0.4796	0.4796	0.0094	14.87
Ackley	0.0479	0.0621	0.0490	0.0488	0.0024	1.52	0.4176	0.4593	0.4308	0.4283	0.0102	13.35
Schwefel	0.0455	0.0601	0.0467	0.0463	0.0024	1.45	0.4336	0.4824	0.4461	0.4459	0.0094	13.83
DeJong	0.0438	0.0571	0.0445	0.0439	0.0023	1.38	0.4225	0.4616	0.4320	0.4307	0.0074	13.39
Easom	0.0487	0.0616	0.0499	0.0498	0.0023	1.55	0.5153	0.5639	0.5250	0.5238	0.0086	16.27
Michalewicz	0.0412	0.0563	0.0424	0.0420	0.0026	1.31	0.4670	0.4998	0.4832	0.4851	0.0079	14.98
Xin-She	0.0462	0.0586	0.0467	0.0462	0.0022	1.45	0.4229	0.4577	0.4366	0.4360	0.0094	13.54
Zakharov	0.0452	0.0587	0.0460	0.0457	0.0023	1.43	0.4321	0.4684	0.4452	0.4438	0.0088	13.80

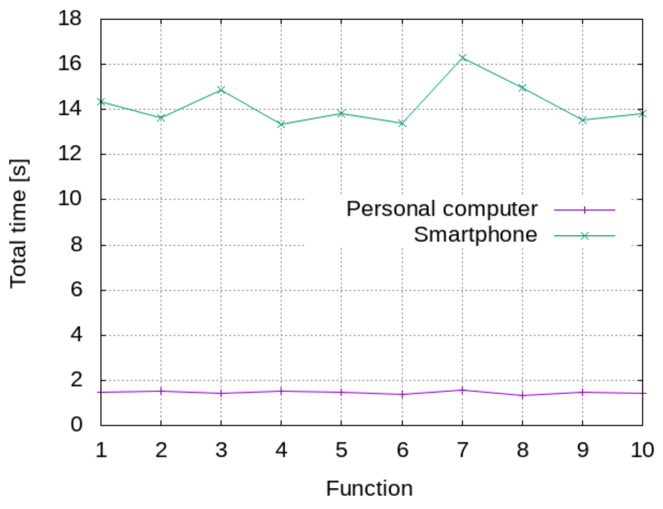

Fig. 2. Comparison of the total time t_{total} between PC and SmartPhone.

The results obtained by comparison of the total time needed for executing the whole benchmark function suite on both observed platforms are depicted graphically in Fig. 2.

As can be seen from the Table as well as the Figure, the same algorithm on the smartphone needs approximately 10-times more time than its counterpart running on the personal computer. However, with producing the results on the smartphone platform we showed that this is also a promising platform for solving the most complex problems with which humans are confronted today. Actually, this fact proves that mobile platforms could also become interesting for solving the NP-hard problems in real-time environments for the future.

V. CONCLUSION AND FUTURE DIRECTIONS

In this paper, we analyzed the performance of a stochastic population-based nature-inspired algorithm on a cheap smartphone device running the Android operating system. Experiments showed the great potential of applying stochastic population-based nature-inspired algorithms on smartphone devices. As a result, modern mobile devices based on Android really allowed us to run the Bat Algorithm in near real-time.

There is also a lot of future work like testing the other stochastic population-based nature-inspired algorithms, such as, for example, Genetic Algorithms or Particle Swarm Optimization. It would also be important to verify the performance of implementation that is written fully in Java. A very important work would also include the use of various device sensors that would collect data in combination with nature-inspired algorithms.

ACKNOWLEDGEMENT

Iztok Fister Jr. acknowledges the financial support from the Slovenian Research Agency (Research Core Funding No. P2-0057). Iztok Fister acknowledges the financial support from the Slovenian Research Agency (Research Core Funding No. P2-0041).

REFERENCES

[1] François CJ Allaire, Mohamed Tarbouchi, Gilles Labonté, and Giovanni Fusina. Fpga implementation of genetic algorithm for uav real-time path planning. In *Unmanned Aircraft Systems*, pages 495–510. Springer, 2008.
[2] Lauro CM de Paula, Anderson S Soares, Telma W de Lima, Alexandre CB Delbem, Clarimar J Coelho, and RG Arlindo Filho. A gpu-based implementation of the firefly algorithm for variable selection in multivariate calibration problems. *PloS one*, 9(12):e114145, 2014.
[3] Iztok Fister Jr., Janez Brest, Uroš Mlakar, and Iztok Fister. Towards the universal framework of stochastic nature-inspired population-based algorithms. In *Computational Intelligence (SSCI), 2016 IEEE Symposium Series on*, pages 1–8. IEEE, 2016.
[4] Iztok Fister Jr., Iztok Fister, Xin-She Yang, Simon Fong, and Yan Zhuang. Bat algorithm: Recent advances. In *Computational Intelligence and Informatics (CINTI), 2014 IEEE 15th International Symposium on*, pages 163–167. IEEE, 2014.
[5] David E Goldberg. Genetic algorithms in search, optimization, and machine learning. 1989.
[6] Alwyn V Husselmann and KA Hawick. Parallel parametric optimisation with firefly algorithms on graphical processing units. In *Proc. Int. Conf. on Genetic and Evolutionary Methods (GEM12). Number CSTN-141, Las Vegas, USA, CSREA (16–19 July 2012)*, pages 77–83, 2012.
[7] Scott Kirkpatrick, C Daniel Gelatt, and Mario P Vecchi. Optimization by simulated annealing. *science*, 220(4598):671–680, 1983.
[8] Ernesto Mininno, Ferrante Neri, Francesco Cupertino, and David Naso. Compact differential evolution. *IEEE Transactions on Evolutionary Computation*, 15(1):32–54, 2011.
[9] Luca Mussi, Fabio Daolio, and Stefano Cagnoni. Evaluation of parallel particle swarm optimization algorithms within the cuda architecture. *Information Sciences*, 181(20):4642–4657, 2011.
[10] Ferrante Neri. Memetic compact differential evolution for cartesian robot control. *IEEE Computational Intelligence Magazine*, 5(2):54–65, 2010.

[11] Paul D Reynolds, Russell W Duren, Matthew L Trumbo, and RJ Marks. Fpga implementation of particle swarm optimization for inversion of large neural networks. In *Swarm Intelligence Symposium, 2005. SIS 2005. Proceedings 2005 IEEE*, pages 389–392. IEEE, 2005.

[12] Xin-She Yang. A new metaheuristic bat-inspired algorithm. In *Nature inspired cooperative strategies for optimization (NICSO 2010)*, pages 65–74. Springer, 2010.

[13] Xin-She Yang and Xingshi He. Bat algorithm: literature review and applications. *International Journal of Bio-Inspired Computation*, 5(3):141–149, 2013.

[14] You Zhou and Ying Tan. Gpu-based parallel particle swarm optimization. In *Evolutionary Computation, 2009. CEC'09. IEEE Congress on*, pages 1493–1500. IEEE, 2009.