Parallelized Multiple Swarm Artificial Bee Colony (PMS-ABC) Algorithm for Constrained Optimization Problems*

Miloš SUBOTIC, Aleksandar MANASIJEVIC, Aleksandar KUPUSINAC
Faculty of Technical Sciences University of Novi Sad, Trg Dositeja Obradovića 6, Novi Sad, 21000, Serbia
milos.subotic@gmail.com (**Corresponding author*), aleksandarmanasijevic@uns.ac.rs, sasak@uns.ac.rs

Abstract: Since their introduction, bio-inspired algorithms, especially the ones based on the social behaviour of the animals that live in colonies have demonstrated great potential in finding near-optimal solutions for both unconstrained and constrained hard optimization problems. In this research, a parallel version of the popular Artificial Bee Colony (ABC) algorithm for optimization of constrained problems, has been introduced. An island-based model, in which the whole population is divided into subpopulations, is used. Subpopulations execute the serial version of the original algorithm and occasionally exchange the obtained results. The proposed algorithm has been tested based on a set of well-known constraint benchmark functions and five real-world engineering design problems. The results demonstrate clear improvements compared with those obtained with the original ABC algorithm.

Keywords: Artificial bee colony, Optimization metaheuristics, Swarm intelligence, Parallelized algorithms, Nature inspired algorithms, Constraint optimization.

1. Introduction

Optimization is a mathematical problem often met in practical engineering disciplines. Optimization is often defined as a process or methodology of finding the best possible solution. Bioinspired meta-heuristics are trying to mimic the behaviour of living organisms that could be found in nature and have demonstrated very good results in solving complex optimization problems. The bioinspired heuristics, especially the swarm intelligence algorithms, have significantly gained popularity over the past 20 years, due to their efficiency in finding suboptimal solutions to intractable optimization problems. Swarm intelligence represents the cooperative behaviour of dispersed, self-organized systems, natural or artificial. Swarm is defined as a very large number of small animals, especially insects that are collaborating and interacting with each other. Swarm intelligence algorithms are inspired by the social behaviour of insects, fish or birds. Algorithms based on the social behaviour of honey bees are among the most popular ones, especially the one presented by Karaboga (Karaboga, 2005) and later developed by Karaboga and Bastruk (Basturk & Karaboga, 2006), (Karaboga & Basturk, 2007b).

Among different types of optimization problems, continuous constrained numerical functions are one of the most computationally demanding. The optimum solution must be feasible, hence it must satisfy the given equality and/or inequality constraints. The mathematical form is given below.

$$\begin{aligned}
&\text{minimize } \mathrm{f}(x),\ x=(x_1,...,x_n)\in \mathbf{R}^n \\
&l_i \le x_i \le u_i, i=1,...,n \\
&\text{subject to: } g_j(x)\le 0,\ \text{for j =1,...,q} \\
&h_j(x)=0,\ \text{for } j=q+1,...,m
\end{aligned} \tag{1}$$

The original ABC algorithm has been successfully modified so it could be applied to constrained optimization problems (Karaboga & Basturk, 2007a), (Karaboga & Akay, 2011). Motivated by good results, (Akay & Karaboga, 2012) have tested modified ABC on 5 real-life engineering problems and managed to improve the quality of the results. Later, further modifications have been introduced, which additionally improved the version of the ABC algorithm for constrained optimization problems (Akay & Karaboga, 2017).

Another extension of the original ABC has been oriented towards discrete optimization problems (Ozturk et al., 2015).

* This research is the continuation of our manuscript "Parallelized Multiple Swarm Artificial Bee Colony Algorithm (MS-ABC) for Global Optimization" published in *Studies in Informatics and Control, vol. 23(1), 2014.* In the previous research modified parallelized version of the original Artificial Bee Colony (ABC) algorithm has been applied to the optimization of unconstrained functions. The current paper presents parallelized version of ABC variant modified for the optimization of constrained functions. In the original paper the name of the algorithm was Multiple Swarm Artificial Bee Colony (MS-ABC) while in this paper the name Parallelized Multiple Swarm Artificial Bee Colony (PMS-ABC) is used.

During this period the ABC algorithm has been applied to different real-world optimization problems that would be very hard or impossible to solve by the conventional, deterministic technics. There have been various successful attempts in solving them with ABC. (Secui, 2015) and (Marzband et al., 2017) have used the ABC algorithm to solve the economic/emission dispatch problem. Additionally, portfolio optimization problems were solved by the ABC algorithm (Chen, 2015) and optimization of an industrial process as demonstrated by (Zhang et al., 2017).

ABC has gained a lot of popularity in researchers' community, hence a lot of effort has been invested to further improve the original algorithm. The self-adaptive search process, in which the best strategy for producing candidate solutions is selected dynamically, has been implemented by (Xue et al., 2018) and a similar approach has been used by (Kiran et al., 2015). The use of crossover operator in scout bee phase has been proposed by (Brajevic, 2015). Aiming to improve the quality of the results, (Sharma et al., 2016) has added Lévy flight to the search process of the original ABC. Another improved version of the original ABC, in which infeasible solutions with small constraint violation are allowed in the early stages of the algorithm execution, has been introduced by (Liang et al., 2017).

As (Shi et al., 2016) have pointed out, ABC has strong global search ability but slow convergence speed which has been a weakness. Parallelization can be one of the methods that could be utilized to compensate for this shortcoming.

In the last two decades, there has been a dramatic increase in the number of computers based on multicore architecture. It has begun with super computers and later has shifted toward PC and handheld devices like tablets and smart phones. A dramatic change can be witnessed in computer architecture due to the multicore paradigm shift, as every electronic device from cell phones to supercomputers confronts parallelism of unprecedented scale (Williams et al., 2007). The main idea is to divide the problem into smaller chunks that will be solved at the same time. In general, a system of *n* processors, each of speed *k*, is slower than the system with one processor of speed $n*k$, given that the same Central Processing Unit (CPU) architecture is used. The development tempo of modern semiconductor industry has slowed down due to the physical limitations of manufacturing process. In order to further increase the performance of produced chips, it is not possible to rely solely on increase of clock speed any more. With higher clock speeds, issues like excessive power consumption, heat dissipation, and current leakage are becoming more apparent and harder to deal with. Heat dissipation and power consumption are especially important for the development of handheld devices which, at the moment are more used than classic PCs. A parallel system is usually cheaper to build and it is energy-efficient. Modern hardware is composed of a greater number of less power cores. However, for this type of hardware, it is more complicated to develop software.

Swarm intelligence algorithms have natural parallelism but practical implementations in parallel and distributed computational systems are nontrivial. Execution time for swarm intelligence algorithms has always been long since they are usually applied to complex problems with a large number of parameters. It is advisable, due to the heuristic nature of swarm intelligence algorithms, to run them multiple times so that more accurate results may be obtained. In every run of the algorithm, each individual swarm member has to compute the value of the function that is being optimized multiple times. The clear segregation of these computational tasks makes swarm intelligence algorithms very suitable for parallelization. Parallelization of swarm intelligence algorithms reduces the execution time. In addition to this apparent benefit, another effect has been observed. Division of the whole population into subpopulations often produces synergetic effect that allows the whole population to obtain the higher quality of the results compared with those obtained by the non-parallelized version of the same algorithm.

There are different ways to parallelize a population-based algorithm. Table 1 shows taxonomy applied to Ant Colony Optimization algorithm, that has been presented by (Pedemonte et al., 2011). In Table 1, strategies are differentiated according to the number of colonies and to the existence of cooperation between units. A very similar classification can be applied to all population-based heuristics.

Table 1. Different approaches to parallelization of population-based algorithm

Parallel approach				
no. colonies	One colony		Several Colonies	
cooperation	no	yes	no	yes
model	master slave	cellular	parallel independent runs	multi-colony

In master-slave model, there is the main process, master that delegates computational tasks and synchronizes their execution between sub processes – slaves. In cellular model, the whole colony is divided into cells - small units that consist of a very small number of individual agents. Each cell communicates with adjacent cells only. In multi-colony model, the whole population is divided into a few sub colonies that communicate between themselves. Approaches without the communication among subpopulation are trying to shorten the execution time, while models which are emphasizing communication try to improve the quality of results compared to the ones obtained by the serial version of the algorithm. Granularity level can differ from very fine, like the one where one thread/core is assigned to only one agent, to very course, where the whole population is divided into several subpopulations and each of them has its thread. Finer grained population is more suitable for devices with greater number of less powerful cores like modern GPUs, while courser grained parallelization is more appropriate for systems with fewer, more powerful cores, like modern CPUs. If the decision about the granularity is based on the computational demands of the problem being optimized, finer granulation is a better option for the optimization problems that don't require a high amount of computation, while courser granularity is more suitable for the computationally intensive optimization problems. In master-slave model, the computational intensity of delegated jobs can range from the evaluation of objective function and its constraints to running the serial version of an algorithm on a part of the search space. Cellular models are usually very fine-grained and every population agent has its dedicated execution thread. All interactions between agents occur between neighbours to avoid communication load. This approach is very suitable for optimization problems with high computational demands. In the parallel independent runs approach, there is no communication between threads. Each thread represents independent run of original, non-parallelized swarm intelligence algorithm. The main goal of this approach is to speed up the execution of the algorithm, hence there is no influence on the quality of the obtained results. The parallel independent runs approach can achieve speedups up to n times where n represents the number of execution cores.

The main goal of a multi colony approach is to improve the quality of the obtained results. The reduction of the execution time is a beneficial side effect. Every colony runs a serial version of an algorithm at the same time and each of the colonies has its dedicated process. Colonies communicate among themselves and, depending on migration policy, exchange the results. After the exchange of the results, each colony continues with the execution of an algorithm, but with the results from other colonies in its matrix. It is observed that this approach has synergetic effect and that n colonies with population size k, often produce better results than one colony with n x k agents in the population.

A lot of researchers have tried a parallel approach, aiming to improve the quality of the results. (Subotic & Tuba, 2014) have been successfully used the parallelization to improve the quality of the results of unconstrained optimization problems. (Asadzadeh, 2016) has improved the quality of the solutions of job shop scheduling problem by using parallel ABC, (Huo et al., 2018) tried to obtain better parameters for parameter calibration of hydrological model by dividing original ABC population into subpopulations and (Hei et al., 2015) have improved the solution of cooperative spectrum sensing problem by using parallel ABC. The author of the original ABC also saw parallelization as a way to increase the quality of the results (Karaboga & Aslan, 2015).

The rest of the paper is structured in the following manner. Section 2 briefly summarizes the original ABC algorithm, and an adjustment of the original ABC algorithm, which allows it to be applied on the constrained optimization problems, following with detailed presentation of the proposed modification. Section 3 describes benchmark functions used for testing and algorithm parameters. Section 4 presents the results of the paper while section 5 draws the conclusions.

2. PMS-ABC (The proposed modification)

Artificial Bee Colony Algorithm is inspired by the foraging behaviour of honey bees. One thing that distinguishes ABC from other swarm intelligence algorithms is the representation of the solutions. In ABC solutions are not represented by individual bees, but by the food source. The nectar amount shows the quality of the solution and it is represented by the fitness value of the objective function. The population in ABC contains three types of bees, employed bees, onlooker bees, and scout bees. The role of employed and onlooker bees is to exploit existing food sources, while scout bees are exploring for new, promising food sources. There is an equal number of employed and onlooker bees in one colony and their number is equal to the number of food sources. Every employed bee is assigned to one of the food sources and it leads the search process in the proximity of their food source. Employed bees share information about the richness of their food source with the onlooker bees helping them to increase search intensity around good food sources. Sometimes food source becomes depleted and in that case employed bee abandons it and becomes scout bee. Scout bees perform a random search for new food sources. Like other population-based algorithms, ABC performs search process through numerous iterations.

Two modifications of the original ABC are performed, so it could be applied to the optimization of the constrained functions. In the original ABC, for the unconstrained optimization problems, only one, randomly selected, parameter of the objective function is modified by employed and onlooker bees. The rest of the parameters are copied from the original solution. In the ABC for the constrained problems, a new control parameter is introduced. It is called modification rate (MR). For every parameter of the objective function, a uniformly distributed pseudo-random number in the range [0, 1] is generated. If this number is smaller than MR, this parameter will be modified. After a new solution is produced, it has to be compared with the selected solution and only one of them will be kept in the solutions matrix. In the original ABC greedy selection process is applied and a solution with a higher fitness value is kept. In modified ABC, instead of greedy selection, Deb's rule is applied. Deb's rule promotes feasible solutions compared to infeasible ones and drives the whole population towards feasible region. Deb's rule compares two solutions at the time, using tournament selection. It is based on three criteria:

1. Any feasible solution is preferred to any infeasible solution
2. If both solutions are feasible, the one with a better value of objective function is kept
3. If both solutions are infeasible, the one with smaller constraint violation is kept.

Population-based algorithms are very suitable for conversion to a parallel version. The CPU load is already naturally and logically divided among the population individuals, hence one of the key decisions is how many individuals are going to be assigned to each of CPU threads (cores). This is called the granularity of the parallelization. Granularity has the most significant impact on the performance of parallel implementation of the algorithm, both in terms of the quality of the obtained results and the execution times. Granularity could vary from scenario where each population unit has dedicated execution thread, on one end of the spectrum, to scenario where the whole population is divided into only two subpopulations and each subpopulation is bounded to one execution thread, at the other end of the spectrum and everything in between. The number of individuals that will be assigned to each of the execution cores is determined by the complexity of an evaluation function and speed of the individual core. The main goal is to avoid excessive thread creation and synchronization. If the number of CPU cycles used for algorithm calculation is denoted by nca, the number of CPU cycles used for managing the threads by ncm and the efficiency of CPU utilization by eu, the following equation can be formulated:

$$eu = nca / ncm \tag{2}$$

It is clearly shown that is desirable to have a denominator as small as possible when compared to the numerator. As the complexity of evaluation function increases, the suitability for finer grained parallelization also increases. On the other hand, functions that require a smaller number of CPU cycles for the evaluation are more suitable for courser granularity since that requires fewer threads to create and manage. Regarding the power of individual cores, systems with a greater number of slower cores are a better fit for finer granulation of parallelized algorithm, while systems with more powerful cores are more appropriate for courser granularity. The first type of system is very well represented by Graphical Processing Units (GPU) and smartphones while the second type of system is well illustrated by modern multi-core CPUs and workstations with multiple CPUs. If fine granulation would be used for the fairly simple objective function, great overhead in the creation and synchronization of threads would be introduced. Since modern CPUs have a significant number of instructions per clock (IPC), hence great efficiency per clock, a small number of cycles is used for calculating an objective function while a large portion of CPU cycles is used for thread management. Modern GPUs have a very large number of slow and inefficient (compared to CPU) cores, which makes them a good match for fine grained parallelization. In this study, a very coarse-grained parallelization model is implemented. This parallelization model is also known as the island-based model, which means that the whole population is divided into a very few subpopulations. According to the parallelization pattern used in this study, the main population has been divided into N subpopulations and the serial version of the algorithm has been run by each of the subpopulations simultaneously. Subpopulations are called swarms, hence the name of this modification, Parallelized Multiple Swarm ABC, (PMS-ABC). This model mainly aims at improving the quality of the results, but a shortening of the execution time is a welcomed side effect.

The other two important issues that arise, when serial algorithm is converted to a parallel version, are which communication pattern and which exchange policy should be used. Quality of the results and execution times are strongly influenced by the communication strategy among subpopulations. Two main communication types are widely used, synchronous and asynchronous. Synchronous communication type is determined by the number of execution steps of an algorithm, hence for every n execution steps, subpopulation will exchange results. In this model, at a predetermined point of an algorithm, all subpopulations will wait until all of them reach that same point of the algorithm, exchange the results, and then continue at the same time. In the asynchronous parallelization model, synchronizations are performed when subpopulation reaches a plateau in improving the quality of the solution. In this study, synchronous communication model is used. Migration policy is another key factor that affects the quality of the obtained solutions. Migration policy describes the way solutions are exchanged between subpopulations. There are different migration policies used, but the most popular and widely used is the one where the best solutions are exchanged. This will ensure that each of the subpopulations contains the best solution found in the whole population. PMS-ABC uses a variation of migration policy based on exchanging the best solutions. Solutions from all subpopulations are collected, and then n randomly selected solutions in each of the subpopulations are replaced with the n best from the whole pool of the solutions.

The original ABC algorithm modified for the optimization of the constrained function has four control parameters: population size, number of iterations, limit and modification rate. Limit is parameter that is used in process of replacement of depleted food sources. If some solution has not been improved for the certain predefined number (limit) of attempts, it will be replaced by the randomly generated solution. This parameter is used to prevent the population from being stuck in the local minimum. PMS-ABC relies on the same parameters as the original ABC but requires a few additional ones, that are specific for the parallelization. The additional parameters are used to tune swarms' communication strategy and they determine when the swarms start to communicate and how often they will communicate. Parameters used for controlling communication strategy are First Exchange Cycle (FEC) and Exchange Cycle Rate (ECR). FEC shows in which iteration of the algorithm communication will start, in other words when the first communication is going to be performed. FEC should be very carefully chosen, since if it is too small, the number of iterations of the serial version of the algorithm, that each of the swarms is performing prior to the first communication, becomes too small to find the meaningful solutions. In this way, there is an increased probability that subpopulations will start to exchange poor quality solutions. But if FEC is too big, the algorithm loses the ability to search multiple spaces and becomes similar to the multiple independent runs approach. ECR denotes the frequency of the communications, hence it shows the number of iterations (cycles) between two communications among subpopulations. The ECR has a great impact on the final results. If communication occurs very often, the risk of premature convergence rises, since there won't be enough cycles for swarms to improve exchanged solutions. However, if communication rarely occurs, the chance of being trapped in local optimum increases. This parameter should be very well balanced. The third control parameter used in PMS-ABC is number of solutions that won't be replaced by random ones in the scout bee phase and it is called Elitism Factor (EF). This parameter is introduced due to the use of elitism, a concept introduced and established in Genetic Algorithms at the first, and later widely used in other bio-inspired heuristics. It is used to preserve the best genes and ensure their presence in the next generation. PMS-ABC uses elitism to ensure that the best solutions won't be replaced by the random ones, hence the algorithm will try to improve the best EF solutions more times than the other solutions. Each solution in the solution matrix has a counter that is increased every time an original solution is better compared to the candidate solution. This counter is called number of trials. At the end of each cycle, for each of the solutions, number of trials is compared to a limit. If it is larger than a limit, a solution will be replaced by a new, randomly generated, solution. For EF best solutions, counter will not be increased if a candidate solution is not better compared to the original one. The use of elitism wouldn't be recommended in the original ABC since it could cause colony to be trapped in the local minimum, but in PMS-ABC other subpopulations will exchange the results with the subpopulation that has been stuck allowing it to leave local optimum.

At the beginning of the PMS-ABC algorithm, the whole population is divided into subpopulations (swarms) of an equal size and all subpopulations continue to execute serial version of the original ABC. Subpopulations start the search process with different random seeds. After the FEC number of iterations, communication between subpopulations starts and it is performed every ECR iteration. In each communication, random solutions in every subpopulation will be replaced by the best solutions from the whole colony. Pseudo-code for the PMS-ABC algorithm is presented:

```
initialize swarms with different random seeds
for each swarm:
    evaluate population
    while MFEC is not reached:
        send employers
        calculate probabilities
        send onlookers
        send scouts
        increase noTrial except for EF best solutions
        if (cn > FEC and cn mod ECR = 0) exchange
        memorize bestSolution
    end while
end for
find the best solution from all swarms
```

3. Benchmark functions and parameter settings

The set of 13, well known, benchmark functions (Karaboga & Akay, 2011) has been used to evaluate the performance of PMS-ABC. The dimensionality of functions is in the range from 2 up to 20 parameters. To cover various types of constrained optimization problems, different forms of functions are used, like linear, nonlinear and quadratic. Besides, five real-world engineering problems (Akay & Karaboga, 2012) have also used for testing.

For the comparison, the original Artificial Bee Colony Algorithm, modified for the optimization of constrained functions, presented by (Karaboga & Akay, 2011) (Akay & Karaboga, 2012) is used. In Karaboga's study, modified ABC for constrained optimizations uses four parameters: population size, number of iterations of an algorithm, limit, and modification rate. For the first group of experiments, colony size, for serial ABC, has been set to 40, the number of iterations has been set to 6000, which gives 240000 evaluations. For evaluating PMS-ABC, an equal number of function evaluation has been used. Colony is divided into three swarms, each having 20 bees. The number of iterations has been set to 4000. Both algorithms use the following equation (3) to calculate the limit:

$$\text{limit} = SN * D \tag{3}$$

where D is the dimension of the problem and SN is the number of food sources. For both ABC and PMS-ABC MR has been 0.8. PMS-ABC requires a few additional parameters. The first one is FEC and finding the right FEC value requires good balance. For the constrained optimization problems, it has been experimentally shown that the best results are obtained when FEC is set to 600. Communication frequency should be determined when island based parallelization model is used. The empirical experiments have demonstrated that the best performance of an algorithm, in terms of result quality, is reached when swarms communicate more often, hence, in this study, ECR has been set to 80. Elitism factor (EF) shows how many solutions will be copied to the next iteration of the algorithm regardless of their trial counter value. In this study, EF has been set to 2 and the best results are obtained when, on each exchange of the results, 3 solutions in each of the swarms are replaced by the best solutions from all of the swarms. The parameters used for testing the engineering optimization problems are shown in Table 2.

Table 2. Parameter values used for testing

Parameter	Value
Colony size	30
Iterations	1000
Swarms	3
MR	0.9
FEC	200
ECR	40
EF	2

4. Results

For the performance analysis of PMS-ABC algorithm, for each of the test functions, 30 independent runs of the algorithm have been conducted. For each of the runs, a different pseudo-random seed has been used. The results of the experiments are presented in tables 3 and 4. Quality of the results is compared by using best value, mean value, worst value and standard deviation. Best, mean and worst results are used to illustrate the ability of the algorithm to reach quality solutions, while the standard deviation has bee n included to show the consistency and robustness of algorithms. Quality of the best and mean results will be analysed first. In 7 out of 13 functions (G01, G03, G04, G06, G08, G11, G12), both algorithms have performed the same in terms of the best results and both algorithms have reached known optimums. The suboptimal result has been reported for G5 function when an original ABC has been used. This might be due to violated constraints. PMS-ABC has reached the known optimum for G5 function. PMS-ABC has reached optimum for G09 also.

Table 3. The results of experiments (1)

Function		Optimum	ABC	PMS-ABC
G01	Best	-15	-15	-15
	Mean		-15	-15
	Worst		-15	-15
	St. Dev.		0	0
G02	Best	-0.803619036	-0.803598	**-0.803613**
	Mean		-0.792412	**-0.792832**
	Worst		-0.749797	**-0.750225**
	St. Dev.		**0.012**	0.013221
G03	Best	-1.0005001	-1	-1
	Mean		-1	-1
	Worst		-1	-1
	St. Dev.		0	0
G04	Best	-30665.53867	-30665.539	-30665.539
	Mean		-30665.539	-30665.539
	Worst		-30665.539	-30665.539
	St. Dev.		0	0
G05	Best	5126.496714	**5126.484**	5126.497
	Mean		5185.714	**5134.298**
	Worst		5438.387	**5169.360**
	St. Dev.		75.358	**12.753**
G06	Best	-6961.813876	-6961.814	-6961.814
	Mean		-6961.813	**-6961.814**
	Worst		-6961.805	**-6961.814**
	St. Dev.		0.002	**0**
G07	Best	24.30620907	24.330	**24.312**
	Mean		24.473	**24.434**
	Worst		25.19	**24.838**
	St. Dev.		0.186	**0.119**
G08	Best	-0.095825041	-0.095825	-0.095825
	Mean		-0.095825	-0.095825
	Worst		-0.095825	-0.095825
	St. Dev.		0	0
G09	Best	680.6300574	680.634	**680.630**
	Mean		680.640	**680.639**
	Worst		**680.653**	680.665
	St. Dev.		**0.004**	0.007
G10	Best	7049.248021	7053.904	**7051.484**
	Mean		7224.407	**7154.818**
	Worst		7604.132	**7428.969**
	St. Dev.		133.87	**104.271**
G11	Best	0.75	0.75	0.75
	Mean		0.75	0.75
	Worst		0.75	0.75
	St. Dev.		0	0
G12	Best	-1	-1	-1
	Mean		-1	-1
	Worst		-1	-1
	St. Dev.		0	0
G13	Best	0.053941514	0.760	**0.054**
	Mean		0.968	**0.296**
	Worst		1	**0.443**
	St. Dev.		**0.055**	0.183

Table 4. The results of experiments (2)

Function		Original ABC	PMS-ABC
Welded beam	Best	1.724852	1.724852
	Mean	1.741913	**1.731123**
	St. Dev.	3.1E-02	**9.5E-03**
Pressure vessel	Best	6059.714736	**6059.714286**
	Mean	6245.308144	**6171.501958**
	St. Dev.	**2.05E+02**	2.25E+02
Ten/comp. spring	Best	**0.012665**	0.012666
	Mean	**0.012709**	0.012792
	St. Dev.	0.012813	1.3E-04
Speed reducer	Best	2997.058412	**2996.347849**
	Mean	2997.058412	**2996.347849**
	St. Dev.	0	0
Gear train	Best	2.700857E-12	2.700857E-12
	Mean	3.641339E-10	**9.385241E-11**
	St. Dev.	5.525811E-10	**2.008979E-10**

For G02, G07, G10, and G13 PMS-ABC has performed better in terms of best results. When analysing the mean values, both algorithms have achieved a global minimum for mean results for 6 functions (G01, G03, G04, G08, G11, and G12), but for the rest of the tested functions, PMS-ABC has always reached better mean result. A good illustration of the PMS-ABC superiority is especially visible when the results of G13 function are observed. If the worst solutions out of 30 runs are observed, it is visible that the original ABC has better worst result only once, for G09 function, while PMS-ABC has reached better worst results for 6 functions, G03, G05, G06, G07, G10 and G13. In terms of result quality, it is clear that PMS-ABC outperforms original ABC in the three observed parameters, best, mean and worst result. Standard deviation is used as a robustness and consistency measure of an algorithm. PMS-ABC has obtained a better standard deviation for 4 functions, while the original ABC has obtained a better standard deviation value for three functions. One of the functions for which ABC has shown better consistency is G13, but the best, mean and worst results obtained by PMS-ABC for G13 are much better. Hence, PMS-ABC has shown better exploration performance, while the whole colony in original ABC has remained stuck around local optimum.

For the analysis of the results of real-world engineering problems, the best result out of 30 runs, the mean result and the standard deviation will be used as criteria of comparison. PMS-ABC has obtained better best result twice, for Pressure vessel and Speed reducer problems, while the original ABC has outperformed PMS-ABC once, for Tension/compression spring problem. When comparing the mean results, PMS-ABC has obtained better results for all the optimization problems except the Tension/compression spring problem. When comparing the algorithms in terms of consistency, PMS-ABC has managed to reach better values for standard deviation 3 times, while the original ABC has done it only once. Hence, in terms of consistency, PMS-ABC has an edge when it is compared to the original ABC.

5. Conclusion

This study has presented a parallel implementation of the original ABC algorithm modified for the optimization of the constrained functions. The island-based model of parallelization was used and the entire bee population has been divided into three subpopulations called swarms. Each swarm executed serial version of the algorithm using a different pseudo-random seed. The best results have been periodically exchanged among swarms after a predefined number of cycles. Elitism, a well-known and proven concept from Genetic Algorithms, has been used in this study. Elitism has allowed a greater number of attempts to modify the best solution, without replacing it with a random one. In PMS-ABC the best solution is never going to be replaced by a random one in the scout bee phase. This prevents the abandonment of high-quality solutions. The possibility of trapping swarms in local optimum has been avoided by the communication which occurs among them. A better solution has been chosen by applying Deb's rule, same as in the original ABC for the optimization of constrained functions. Multi swarm artificial bee colony algorithm has been compared to the original ABC modified for the optimization of constrained functions using a set of 13 well-known benchmark functions and a set of five real-world engineering design problems. Algorithms have been compared in terms of best, mean, worst result and standard deviation. Both algorithms have been run 30 times independently using different pseudo-random seeds.

PMS-ABC has consistently demonstrated a better performance both in terms of quality of the results and consistency of the algorithm.

PMS-ABC's advantage is visible especially when the results obtained for the G13 function are analysed. The best result reached by ABC is 0.76 while PMS-ABC has almost managed to find an optimum and has reached 0.054 as the best result.

In the modern multi-core CPU era, it is very desirable to utilize more than one core. By using a separate thread for each of the swarms, PMS-ABC reduces execution time almost linearly. This result is comparable to the research conducted by (Huo et al., 2018).

These results show that parallelization has a great potential to improve the quality of the results compared to the original ABC. An even brother conclusion would be that all swarm intelligence algorithms could benefit from the parallelization. This has been demonstrated by (Dao et al., 2018) who have applied parallelization on bat algorithm and by (Atashpendar et al., 2018) and (Gülcü & Kodaz, 2015) who have demonstrated the parallelization of PSO algorithm. Additionally, some researches have demonstrated the reduction of the execution time like the ones conducted by (Thiruvady et al., 2016) and (Ouyang et al., 2015).

The main conclusions are that parallelization of swarm algorithms can make full use of the multi-core resources and improve the quality of the obtained results.

REFERENCES

Akay, B. & Karaboga, D. (2012). Artificial bee colony algorithm for large-scale problems and engineering design optimization, *Journal of Intelligent Manufacturing*, *23*(4), 1001-1014.

Akay, B. B. & Karaboga, D. (2017). Artificial bee colony algorithm variants on constrained optimization, *An International Journal of Optimization and Control: Theories & Applications (IJOCTA), 7*(1), 14.

Asadzadeh, L. (2016). A parallel artificial bee colony algorithm for the job shop scheduling problem with a dynamic migration strategy, *Computers & Industrial Engineering, 102*, 359-367.

Atashpendar, A., Dorronsoro, B., Danoy, G. & Bouvry, P. (2018). A scalable parallel cooperative coevolutionary PSO algorithm for multi-objective optimization, *Journal of Parallel and Distributed Computing, 112*, 111-125.

Basturk, B. & Karaboga, D. (2006). An artificial bee colony (ABC) algorithm for numeric function optimization. In *Proceedings of the IEEE swarm intelligence symposium*, Indianapolis, IN, USA (pp. 12-14).

Brajevic, I. (2015). Crossover-based artificial bee colony algorithm for constrained optimization problems, *Neural Computing and Applications 26*(7), 1587-1601.

Chen, W. (2015). Artificial bee colony algorithm for constrained possibilistic portfolio optimization problem, *Physica A: Statistical Mechanics and its Applications, 429*, 125-139.

Dao, T.-K., Pan, T.-S., Nguyen, T.-T. & Pan, J.-S. (2018). Parallel bat algorithm for optimizing makespan in job shop scheduling problems, *Journal of Intelligent Manufacturing, 29*(2), 451-462.

Gülcü, Ş. & Kodaz, H. (2015). A novel parallel multi-swarm algorithm based on comprehensive learning particle swarm optimization, *Engineering Applications of Artificial Intelligence, 45*, 33-45.

Hei, Y., Li, W., Fu, W. & Li, X. (2015). Efficient Parallel Artificial Bee Colony Algorithm for Cooperative Spectrum Sensing Optimization, *Circuits, Systems, and Signal Processing, 34*(11), 3611-3629.

Huo, J., Liu, L. & Zhang, Y. (2018). An improved multi-cores parallel artificial Bee colony optimization algorithm for parameters calibration of hydrological model, *Future Generation Computer Systems, 81*, 492-504.

Karaboga, D. (2005). An Idea Based on Honey Bee Swarm for Numerical Optimization, *Technical Report - TR06*. Technical Report, Erciyes University.

Karaboga, D. & Akay, B. (2011). A modified Artificial Bee Colony (ABC) algorithm for constrained optimization problems, *Applied Soft Computing, 11*(3), 3021-3031.

Karaboga, D. & Aslan, S. (2015). A new emigrant creation strategy for parallel Artificial Bee Colony algorithm. In *2015 9th International Conference on Electrical and Electronics Engineering (ELECO)* (pp. 689-694).

Karaboga, D. & Basturk, B. (2007a). Artificial Bee Colony (ABC) Optimization Algorithm for Solving Constrained Optimization Problems, *Foundations of Fuzzy Logic and Soft Computing*, 789-798. Springer Berlin Heidelberg. Berlin, Heidelberg.

Karaboga, D. & Basturk, B. (2007b). A powerful and efficient algorithm for numerical function optimization: Artificial bee colony (ABC) algorithm, *Journal of Global Optimization*, *39*, 459-471.

Kiran, M. S., Hakli, H., Gunduz, M. & Uguz, H. (2015). Artificial bee colony algorithm with variable search strategy for continuous optimization, *Information Sciences, 300*, 140-157.

Liang, Y., Wan, Z. & Fang, D. (2017). An improved artificial bee colony algorithm for solving constrained optimization problems, *International Journal of Machine Learning and Cybernetics*, *8*(3), 739-754.

Marzband, M., Azarinejadian, F., Savaghebi, M. & Guerrero, J. M. (2017). An Optimal Energy Management System for Islanded Microgrids Based on Multiperiod Artificial Bee Colony Combined with Markov Chain, *IEEE Systems Journal, 11*(3), 1712-1722.

Ouyang, A., Tang, Z., Zhou, X., Xu, Y., Pan, G. & Li, K. (2015). Parallel hybrid PSO with CUDA for lD heat conduction equation, *Computers & Fluids, 110*, 198-210.

Ozturk, C., Hancer, E. & Karaboga, D. (2015). Dynamic clustering with improved binary artificial bee colony algorithm, *Applied Soft Computing*, *28*, 69-80.

Pedemonte, M., Nesmachnow, S. & Cancela, H. (2011). A survey on parallel ant colony optimization, *Applied Soft Computing*, *11*(8), 5181-5197.

Secui, D. C. (2015). A new modified artificial bee colony algorithm for the economic dispatch problem, *Energy Conversion and Management*, *89*, 43-62.

Sharma, H., Bansal, J. C., Arya, K. V. & Yang, X.-S. (2016). Lévy flight artificial bee colony algorithm, *International Journal of Systems Science*, *47*(11), 2652-2670.

Shi, Y., Pun, C.-M., Hu, H. & Gao, H. (2016). An improved artificial bee colony and its application, *Knowledge-Based Systems*, *107*, 14-31.

Subotic, M. & Tuba, M. (2014). Parallelized Multiple Swarm Artificial Bee Colony Algorithm (MS-ABC) for Global Optimization, *Studies in Informatics and Control*, *23*(1), 117-126. DOI: 10.24846/v23i1y201412

Thiruvady, D., Ernst, A. T. & Singh, G. (2016). Parallel ant colony optimization for resource constrained job scheduling, *Annals of Operations Research*, *242*(2), 355-372.

Williams, S., Oliker, L., Vuduc, R., Shalf, J., Yelick, K. & Demmel, J. (2007). Optimization of sparse matrix-vector multiplication on emerging multicore platforms. In *SC '07: Proceedings of the 2007 ACM/IEEE Conference on Supercomputing* (pp. 1-12).

Xue, Y., Jiang, J., Zhao, B. & Ma, T. (2018). A self-adaptive artificial bee colony algorithm based on global best for global optimization, *Soft Computing*, *22*(9), 2935-2952.

Zhang, R., Chang, P.-C., Song, S. & Wu, C. (2017). A multi-objective artificial bee colony algorithm for parallel batch-processing machine scheduling in fabric dyeing processes, *Knowledge-Based Systems*, *116*, 114-129.