[Egyptian Informatics Journal 22 (2021) 213–223](https://doi.org/10.1016/j.eij.2020.08.003)

Contents lists available at [ScienceDirect](http://www.sciencedirect.com/science/journal/11108665)

Egyptian Informatics Journal

journal homepage: [www.sciencedirect.com](http://www.sciencedirect.com/)

[](http://crossmark.crossref.org/dialog/?doi=10.1016/j.eij.2020.08.003&domain=pdf)A new evolutionary algorithm: Learner performance based behavior algorithm

Chnoor M. Rahman [a](#_bookmark0),[b](#_bookmark1),[⇑](#_bookmark3), Tarik A. Rashid [c](#_bookmark2)

a *Applied Computer Department, College of Medicals and Applied Sciences, Charmo University, Sulaimany, Iraq*

b *Technical College of Informatics, Sulaimany Polytechnic University, Sulaimany, Iraq*

c *Computer Science and Engineering Department, University of Kurdistan Hewler, Erbil, Iraq*

a r t i c l e i n f o

*Article history:*

Received 1 October 2019

Revised 10 January 2020

Accepted 12 August 2020

Available online 1 September 2020

*Keywords:*

Evolutionary algorithms Genetic algorithm

LPB

Learner performance based behavior algorithm

Optimization

Metaheuristic optimization algorithm

a b s t r a c t

A novel evolutionary algorithm called learner performance based behavior algorithm (LPB) is proposed in this article. The basic inspiration of LPB originates from the process of accepting graduated learners from high school in different departments at university. In addition, the changes those learners should do in their studying behaviors to improve their study level at university. The most important stages of opti- mization; exploitation and exploration are outlined by designing the process of accepting graduated learners from high school to university and the procedure of improving the learner’s studying behavior at university to improve the level of their study, respectively. To show the accuracy of the proposed algo- rithm, it is evaluated against a number of test functions, such as traditional benchmark functions, CEC- C06 2019 test functions, and a real-world case study problem. The results of the proposed algorithm are then compared to the DA, GA, and PSO. The proposed algorithm produced superior results in most of the cases and comparative in some others. It is proved that the algorithm has a great ability to deal with the large optimization problems comparing to the DA, GA, and PSO. The overall results proved the ability of LPB in improving the initial population and converging towards the global optima. Moreover, the results of the proposed work are proved statistically.

© 2021 THE AUTHORS. Published by Elsevier BV on behalf of Faculty of Computers and Artificial Intel-

ligence, Cairo University. This is an open access article under the CC BY-NC-ND license ([http://creative-](http://creativecommons.org/licenses/by-nc-nd/4.0/)

[commons.org/licenses/by-nc-nd/4.0/](http://creativecommons.org/licenses/by-nc-nd/4.0/)).

1. Introduction

The computational intelligence (CI) term as a branch of artificial intelligence (AI) was first invented by Bezdek in the early 1990s [[1]](#_bookmark16), which motivated a new field in computer-based intelligence. CI in principle consists of any technologies and science- supported approaches for creating, analyzing, and developing intelligent systems [[2]](#_bookmark18). It mainly depends on a set of nature- inspired computational patterns and a numerical collection of data [[3]](#_bookmark20). The study of optimization techniques is one of the main subjects of CI. Optimization is part of any problem that requires decision making, either in economic or engineering fields. Decision-making tasks involve making the best decision to choose

\* Corresponding author.

*E-mail addresses:* [Chnoor.rahman@charmouniversity.org](mailto:Chnoor.rahman@charmouniversity.org) (C.M. Rahman), [tarik.](mailto:tarik.ahmed@ukh.edu.krd) [ahmed@ukh.edu.krd](mailto:tarik.ahmed@ukh.edu.krd) (T.A. Rashid).

Peer review under responsibility of Faculty of Computers and Information, Cairo University.

between different alternatives. Numerous optimization algorithms exist; however, no single algorithm fits all the different problems. It is crucial for the appropriate optimizer to guarantee that the optimal solution is always reachable. NP-hard problems, for exam- ple, are usually not easy to be solved. However, most combinatorial optimization problems, for example, N-Queens, traveling salesper- son, and 0/1 Knapsack are NP-hard. To solve this type of problem and depending on the size of the problem, two approaches exist, namely; exact methods and metaheuristic methods [[4]](#_bookmark21). Exact methods are useful when the number of decision variables is small. These methods find the optimal solution for the problem. Exam- ples for exact methods are branch and bound algorithm [[5]](#_bookmark14), dynamic and linear programming, and so on. The problem with these methods is that they are known as time expensive methods, so that it is not recommended to use them for solving difficult or NP-hard problems. Likewise, where the decision space is discrete or when a large number of decision variables exist, which occurs in most if not in all practical problems of optimization, exact meth- ods cannot show good performance, instead, metaheuristics can be used [[4]](#_bookmark21).

<https://doi.org/10.1016/j.eij.2020.08.003>

1110-8665/© 2021 THE AUTHORS. Published by Elsevier BV on behalf of Faculty of Computers and Artificial Intelligence, Cairo University. This is an open access article under the CC BY-NC-ND license (<http://creativecommons.org/licenses/by-nc-nd/4.0/>).

Depending on the characteristics, metaheuristic optimization algorithms can be classified in various ways. They can be classified into population-based algorithms and trajectory-based or single- point search algorithms. In the latter case, the algorithm uses a sin- gle solution, which means in each iteration only a single solution will manipulate. Hill climbing, tabu search, and simulated anneal- ing are examples of this class of algorithms. On the other hand, population-based algorithms use a population of agents and the whole population is modified in each iteration. Examples for population-based algorithms are genetic algorithm, particle swarm optimization, ant colony optimization, and so on [[4]](#_bookmark21).

* 1. *Related works*

During the 1960s and 1970s, the metaheuristic optimization algorithms were bloomed. At the beginning of the 1960s, the genetic algorithm (GA) [[6]](#_bookmark15) was developed by John Holland and his collaborators. GA is a search technique; it is based on Darwin’s theory of evolution and selection of biological systems. The ability of the GA for optimization makes the researchers use it in optimiz- ing a wide range of problems. Since then, it has been modified and hybridized with other techniques to solve various problems. In [[7]](#_bookmark17) GA was combined with an active set technique (AST). The hybrid technique was used for optimizing the unsupervised artificial neu- ral network. The aim of this work was to accurately estimate the temperature profiles of the heat conduction model in the head of humans. The results revealed that the hybrid technique produced better and accurate results comparing to standalone approaches, such as GA and AST. Additionally, in [[8]](#_bookmark19), GA combined with an inte- rior point technique to optimize a new approach. The approach was solving the initial value of the equation of a Painlev´ e II, and its variants, utilizing the feed-forward artificial network. Moreover, in [[9]](#_bookmark22), GA combined with IPT to optimize a feed-forward artificial neural network for solving porous fin equation. Better accuracy achieved comparing to other numerical techniques. Similarly, ref- erence [[10]](#_bookmark23) designed a neuro-heuristic schema for non-linear sec- ond order Thomas-Fermi system. To optimize the schema, GA and sequential quadratic programming was utilized. It was discovered that the examined schema was feasible, precise, and effective. Hol- land’s work encouraged many to adopt and develop identical tech- niques in their research works. Later, in 1966, Fogel et al. developed an evolutionary programming technique [[11]](#_bookmark23). In this work, finite state machines were used to represent the solution, and stochastically one of the machines was mutated. Afterward, in 1983, Kirkpatrick et al. developed simulated annealing (SA) [[12]](#_bookmark23). SA mimics the process of annealing that utilized for crystal- lization, which is a physical process in metals and glasses to harden the material. Furthermore, at the beginning of the 1990s, Marco Dorigo completed his Ph.D. thesis on optimization and nature- inspired algorithms. In his thesis, he examined a novel idea known as an ant colony optimization algorithm (ACO) [[13]](#_bookmark23). ACO was inspired by the swarming behavior of social ants utilizing the pher- omone to find the source of food and bring the food back to their nest. Later, in 1995, James Kennedy proposed particle swarm opti- mization (PSO) [[14]](#_bookmark23). PSO can be counted as another significant improvement in the field. It mimics the behaviors of the school of birds or fish. A particle represents a single solution that has a position in the search space. In 2005, Karaboga introduced an arti- ficial bee colony (ABC) [[15]](#_bookmark23). ABC mimics the behaviors of honey- bees. It provides well-balanced exploitation and exploration ability. Thereafter, in 2007, Chu and Tsai proposed a new swarm- based optimization algorithm named cat swarm optimization (CSO) algorithm. CSO mimics the behaviors of cats [[16]](#_bookmark23). Yang in 2010 introduced the bat algorithm, which is based on the echolo- cation behavior of micro bats [[17]](#_bookmark23). In 2014, Mirjalili on the base of hunting behavior, and social hierarchy of grey wolf proposed a

new optimization algorithm named as grey wolf optimization (GWO) algorithm [[18]](#_bookmark24). In 2015, the same author proposed the dragonfly algorithm (DA). DA was mainly inspired by the hunting and migrating behaviors of a dragonfly. The latter is called a dynamic (migratory) swarm, and the former is called static (feed- ing) swarm [[19]](#_bookmark25). Finally, in 2019 fitness dependent optimizer (FDO) developed. It is inspired by the bee swarming reproductive process. FDO mimics the PSO in utilizing velocity to update search agent’spositions. However, FDO uses the fitness function of the problem to produce weights, and these weights are then used to guide the agents in the exploration and exploitation phases [[20]](#_bookmark26).

Since introducing these algorithms for optimization, many researchers utilized them to optimize problems in various fields. However, some other researchers aimed at improving those algo- rithms. The satisfactory results produced by these algorithms for different optimization problems proved the importance and neces- sity of them [[21–26]](#_bookmark31). Consequently, researchers continue to pro- pose new algorithms in the field. Many of these algorithms do not have a good balance between exploitation and exploration. Having high exploitation traps the algorithm in the local optimum. Moreover, a high degree of exploration raises the probability of finding global optima but decreases the efficiency. Therefore, hav- ing a good balance between exploration and exploitation can make an algorithm perform better compared to the other algorithms [[27]](#_bookmark27).

* 1. *Innovative contribution*

In this paper, a new optimization method, learner performance behavior based algorithm is proposed. The LPB method mimics the process of accepting graduated learners from high school in differ- ent colleges and the behaviors of learners that affect their perfor- mance during the college study, and the factors that may help the learners to change their high-school study behaviors that are not effective anymore for studying in the college. To implement this, multi-populations can be utilized to demonstrate the learners that have a GPA in different ranges. Consequently, this causes a good balance between exploration and exploitation [[27]](#_bookmark27). The most important features of the proposed work are:

It is a population based algorithm.

●

The initial population is created randomly. A percentage of the population is separated.

●

●

The population is divided into a number of sub-populations. The highest fitness in the separated group is used to divide the population into sub-populations.

●

●

The sub-populations that contain the best individuals have pri- ority to go through the optimization process first.

●

Mutation and crossover operators are used to make changes in the structure of new individuals.

●

* 1. *Organization*

The rest of the paper is organized as follows: [Section 2](#_bookmark4) shows the inspiration of the proposed algorithm. [Section 3](#_bookmark5) presents the features (operators) of GA that are utilized in the proposed tech- nique. The LPB operators and techniques along with the pseudo code are presented in section 4. Furthermore, the results of the algorithm and an inclusive and comparative study on some bench- mark test functions along with a real-world problem are presented in section 5. Finally, the conclusion of the work and directions for future researches are shown in section 6.

1. Inspiration

Every year groups (population) of learners finish their high school and apply to the universities. The applications for some of these learners are accepted and the rest are rejected. Depending on their GPA, the learners are divided into different groups. The process of transferring learners from high school to university starts with a group of graduated learners *M* from high school. Departments from universities specify the number of learners that they want to accept to study in their department. Furthermore, each department specifies the minimum GPA that the learners should have in order to study in that department. This is like grouping the graduated learners from high school *M* to a number of different departments (groups) according to their GPA. The department accepts the applications of the learners if the learner’s GPA is in the range of the required GPAs by that department. Among the learners who apply to a specific department, there are a number of learners, which their GPA is under the required GPA. The application for those learners will be rejected. Further- more, there are learners that their GPA is much higher than the required GPA, thus, these applications have a priority and they will be accepted first, and then the lower ranges, and so on, until the number of accepted applications is equal to the number of learners specified by the department. Furthermore, sometimes it happens that in general, the GPA of learners is low. Thus, some of the departments cannot have a specified number of learners with the required GPA. At these situations before finalizing the list of accepted learners, the department, and the university should decide whether they want to accept learners with less GPA or not. After accepting the graduated learners from high school in dif- ferent colleges and departments, the learners go through a number of difficulties. Because the environment they came from was differ- ent from the environment they are in now. In addition, the study- ing behaviors that they had as high school learners may not be effective anymore. It is normal that many fresh learners are not prepared either academically nor in terms of study skills for college-level study. Working on the learner’s studying behaviors, such as seeking help and group working will help them to study more effectively and will result in improving their score during their study in the college [[28,29]](#_bookmark27). Studying behaviors of a learner can be affected by studying behaviors of learners in the same

department or learners in any other department.

The level of learning of transition learners from high school to college can be improved by adopting some effective strategies, which are quite different from those in high school. A number of behaviors have been considered to judge between the strong and weak learners, they include (level of interest, deep processing, effective note taking, problem-solving, group working, seeking help, and self-study). Additionally, according to [[30]](#_bookmark27) the learners with a high level of creativity are always strong learners. Depend- ing on the previously mentioned resources, it can be concluded that learners who have a good level of the aforementioned behav- iors are good learners.

Moreover, it has been noticed that the quality of metacognition is another key-difference between strong and weak learners. Metacognition refers to the learner’s awareness level of under- standing of a topic. Those who have poor metacognition are confi- dent and believe that they have done well on exams while it is not and their low score shocks them. When learners get low marks on exams, they often believe that they should spend more time study- ing subjects. In addition to studying more (although that often helps), however, learners with poor metacognition should change the way they study [[31]](#_bookmark27). Learners with poor metacognition levels usually have poor study strategies, which rise false confidence that they have studied the material well without increasing their actual

level of learning. Most fresh learners at colleges have learned to study skills in high school that are no longer effective. They might have a proper sense of metacognition, which accurately informed them when they studied sufficiently during high school, but it is not accurate anymore. This means that entering college requires overcoming the old study strategies with new ones [[29,32]](#_bookmark27). Besides, having an adequate level of metacognition can cause a good improvement in the learner’s level of study and it may have an effect on all the strategies used by the learner. The main inspi- ration of this algorithm originates from the following steps that a learner goes through:

1. The strategies used to group the learners according to their GPA, and almost all the learners that are accepted in a department have a GPA in a specific range.
2. After accepting the learners in the departments, their study- ing behaviors can be improved to make them good college learners. The learner’s behaviors influenced by each other while they study together.
3. The level of metacognition for learners has a big impact on all the studying behaviors.

In this algorithm, the first step is used to choose individuals from the population. The importance of this step is dividing the main population to some sub-populations and then the individuals will be selected from the sub-populations depending on their fit- ness. This prevents converge to local optima because selecting individuals will start from the perfect sub-population. The latter two steps are used to improve the individuals by letting the learn- ers work in groups and ask for help from each other. Furthermore, having a good level of metacognition will influence the overall studying behaviours of a learner in a stochastic way (mutation). On the other hand, learners affect the studying behaviours of each other when they study together (crossover).

1. Genetic algorithm operators

The genetic operators imitate the procedure of the heredity of genes to produce new individuals at each generation. The opera- tors are utilized to make changes in the structure of individuals during the representation. The common genetic operators are crossover, mutation, and selection. Here, we only define crossover and mutation operators.

* 1. *Crossover*

Crossover is the most fundamental genetic operator. It works on two individuals at the same time and produces offspring by inte- grating features of both individuals. Various crossover techniques are available; however, the most used one is choosing a stochastic cut-point to produce the offspring by integrating the part of one parent to the right of the cut-point with the part of the second par- ent to the left of the cut-point. For example, one-cut point cross- over, two-cut point crossover, multi-cut point crossover, etc. [[33]](#_bookmark27).

* 1. *Mutation*

Mutation creates random changes in different individuals. The simplest form of mutation is altering one or more genes. Mutation in the genetic algorithm has a great role of either a) restoring the lost genes during the selection process, hence, they can be used in another context or b) serving the genes that were not available in the initial population. Various ways of mutation are available for different representations of individuals. For example, uniform

mutation, replacement mutation, dynamic mutation, boundary mutation, and so on [[33]](#_bookmark27).

1. Learner performance based behavior algorithm

As the first step in the algorithm, randomly create a population of graduated learners *M* who want to apply for different depart- ments in different universities. Furthermore, we have an operator and we call it division probability *dp*. As discussed, every depart- ment accepts learners that have a GPA greater than or equal to the minimum required GPA. To show this in the algorithm, at first, we use the *dp* parameter to randomly choose a percentage of ele- ments from *M*. Afterwards, we calculate the fitness of each of the chosen individuals and sort them. Then we divide them into two groups, good and bad, depending on their fitness. The former con- tains the individuals that have a higher GPA and the bad group contains the rest. After this, the fitness of the individuals in the main population *M* is calculated and then filtered. Those individu- als that have fitness smaller or equal to the highest fitness (best fit- ness) in the bad population will be moved to the bad population. The rest of the individuals will be divided into two groups. Those who have fitness smaller or equal to the highest fitness (best fit- ness) in the good population will be moved to a good population, and those who have fitness higher than the highest fitness in the good population will be moved to the perfect population. Then the number of learners specified by the department will be chosen from the perfect population and good population. If the number of individuals in these two populations was smaller than the number of specified learners by the department this is when the number of learners that have got the required GPA is small and the depart- ment should decide whether they should accept other learners with less GPA or not. If they decided to accept other learners, the rest of the individuals will come from the bad population.

After accepting the graduated learners from high school in the departments, as discussed, they may not have effective studying behaviours [[29,32]](#_bookmark27). However, improving behaviours like help- seeking, group working can have a positive impact on them. In addition, as mentioned in [[28,29]](#_bookmark27) learners can influence each other’s behaviour. For example, when they work in groups or when they ask help from each other their studying behaviours will be affected. To show this in the algorithm crossover operator from genetic algorithm is used. Utilizing a crossover operator will let the individuals exchange some studying behaviours. Consequently, the learner has a set of studying behaviours, which is different from the original studying behaviours owned by the learners. Hence, the overall, behaviours of both individuals will be affected and the produced individuals have different behaviours.

In addition, the level of metacognition has an impact on the overall studying behaviours of a learner. Whenever the metacogni- tion level of a learner is affected, stochastically, the overall study- ing behaviours of the learner will be affected too [[31,34]](#_bookmark27). The level of metacognition according to [[34]](#_bookmark28) is affected by training the lear- ner using a number of strategies. Using these strategies is excluded from this work. Consequently, the level of metacognition of learn- ers can be affected using a rate that can be specified in the algo- rithm. As mentioned the metacognition level may affect the overall behaviours of the learners in a stochastic way. So that, ran- domly changing positions of the behaviours of that individual according to a specific rate or randomly updating the values of studying behaviours of that learner can do this. This is presented in the algorithm by using the mutation operator from the genetic algorithm. Visual 1 shows the pseudo code for LPB algorithm.

Definition of symbols:

*M*: the initial random population

*N*: the number of individuals in the new population

Table 1

Parameter settings for lpb.

Parameters Parameter Value

Crossover rate 2\*round (0.7\*population size) Mutation rate round (0.2\*population size) Population Size 80

dp 0.5

*dp*: the percentage of individuals chosen from *M*

*O*: the sub-population chosen from *M* according to the *dp*

operator.

*BP:* bad population *GP*: good population *PF*: perfect population

*k*: is a counter utilized to count the number of newly created individuals

1. Results and discussion

In this section, a number of standard benchmark functions in the literature are used to examine LPB algorithm. The results are then evaluated against three popular algorithms in the literature: DA, PSO, and GA. The results for 19 classical benchmark functions for PSO, DA, and GA are taken from [[19]](#_bookmark25). Nevertheless, we exam- ined the CEC-C06 2019 test functions to show the ability of the algorithm in solving large scale optimization problems [[37]](#_bookmark32). Addi- tionally, the processing time (PT) in seconds of the algorithm for both groups of the test functions is calculated to show the ability of the algorithm compared to the others in quickly finding the opti- mal results. Furthermore, to prove the significance of the results, the Wilcoxon rank-sum test [[35]](#_bookmark29) is used. Then the algorithm is used to optimize a real-world problem. The parameter settings for LPB algorithm are shown in [Table 1](#_bookmark6).

* 1. *Classical benchmark test functions*

To test the performance of the LPB algorithm a group of bench- mark functions is used. These benchmark functions are divided into three groups: unimodal, multi-modal, and composite test functions [[36–39]](#_bookmark30). Each group has different properties. Unimodal test functions, for example, benchmark the convergence and the exploitation of the algorithm. This group of test functions has a sin- gle optimum. However, multi-modal test functions, as their name implies, have multi optimum. They have one global optimum and multi-local optima. To approach the global optimum an algorithm should avoid the entire local optimal solutions. Hence, this group of test functions can benchmark exploration and avoid local optima.

1. [Initialization]

Randomly create a population *M*

1. [Specify parameters]

Specify the number of required learners *N* for a department, crossover rate and mutation rate

1. [Create Sub-Populations]

Use *dp* parameter to randomly choose a percentage of individuals *O* from *M*

Evaluate the fitness of individuals in *O*

Depending on their fitness, sort the individuals in *O*

(descending order)*,* use one of the sorting methods Divide *O* to two populations, good (individuals with high

fitness) and bad (individuals with low fitness)

(*continued*)

While termination condition is not met

Use *dp* parameter to randomly choose a percentage of individuals *O* from *M*

Evaluate the fitness of individuals in *O*

Depending on their fitness, sort the individuals in *O* (descending order)*,* use one of the sorting methods Divide *O* to two populations, good (individuals with high fitness) and bad (individuals with low fitness)

Find fitness for all individuals in the population *M*

Find the highest fitness in good and bad populations

if an individual from *M* has fitness <= highest fitness in the bad population

Move it to the bad population *BP*

else if an individual from *M* has fitness <= highest fitness in the good population

Move it to the good population *GP*

Else

Move it to the perfect population *PF*

end if

while *k* <= *N*

if *PF* is not empty

Select an individual from *PF*

else if *GP* is not empty

Select an individual from *GP*

Else

Select an individual from *BP*

end if

*k* = *k* + 1;

end while

1. Crossover
2. Mutation
3. [Termination]

Repeat the procedure from step 3 until termination condition is met.

end while

1. [Optimal Solution]

Select the best solution from the perfect population VISUAL 1: PSEUDO CODE FOR LPB ALGORITHM

Finally, the composite test functions are mostly combined, biased, rotated, and shifted versions of the aforementioned groups [[39]](#_bookmark33). They demonstrate the difficulties exist in the real search spaces by providing a huge number of local optima and diverse shapes for various regions. This type of benchmark functions can bench- mark the combined exploitation and exploration of an algorithm. See Appendix A, [Tables 6, 7, and 8](#_bookmark11) for more information about the test functions and their conditions [[19]](#_bookmark25). Ultimately, for each algo- rithm in [Table 2](#_bookmark7), the test functions are solved 30 times, 80 search agents are utilized over 500 iterations. The average and standard deviation are then calculated. Parameters for GA, PSO, and DA are discussed in reference [[19]](#_bookmark25). For all test functions in [Table 1](#_bookmark6), *dp* is set to 0.5. The average and standard deviation of the optimal solu- tion is calculated in the last iteration. These two metrics are used to evaluate the overall performance of the algorithms, and to show the stability degree of the algorithms to solve the test functions.

For each test function in [Table 2](#_bookmark7), superior results are shown in bold. As shown in [Table 2](#_bookmark7), for the first six unimodal test functions (TF1-TF6), the DA algorithm outperforms the LPB algorithm, and also PSO performs better in the (TF1-TF4, and TF6). This proves that the exploitation and the convergence speed of the algorithm are not better than the algorithms used in the comparison. However, the results of the unimodal test functions of the LPB algorithm compar- ing to the GA are evident that LPB algorithm has a greater exploita-

tion rate and convergence speed. In addition, LPB algorithm outperforms both PSO and DA in the last unimodal test function (TF7) and PSO in TF5 as well. Nevertheless, the LPB algorithm pro- vides better results than the other algorithms in all the other test functions. PSO, however, provided a better result in TF12. These results show the ability of the proposed algorithm in avoiding local optima, exploring the search space, and balancing exploration and exploitation. Results of the test functions TF7-TF19 proved that LPB has a superior exploration, and a perfect ability in avoiding local optima, and also it has a superior balance between exploration and exploitation phases comparing to the DA, PSO, and GA. As shown in [Table 2](#_bookmark7), it can be concluded that the LPB algorithm has the first rank among the other algorithms because it outperformed the other algorithms in 12 functions out of 19 functions. [Fig. 1](#_bookmark8) shows the con- vergence curve for the proposed algorithm. In [Fig. 1](#_bookmark8), for each group of the test functions, one function is selected (F2 for unimodal, F9 for multi-modal, and F17 for composite test functions), and cost refers to the fitness value for the global best.

For the traditional benchmark functions, the PT of the LPB is much smaller comparing to the DA. The reason for this is that in the first stage of the LPB, a subset of the population is chosen based on this smaller group other subpopulations are built. The perfect subpopulation has priority to be optimized first, then the good sub- population and so on. Since the subpopulations are much smaller compared to the main population, searching for the solutions in these subpopulations is speeder. This improves the randomness and saves the optimization time simultaneously. However, com- pared to the PSO and GA, the PT of the LPB is higher.

* 1. *CEC-C06 2019 benchmark test functions*

Many real-world problems exist in which time is not as impor- tant as getting an accurate answer. In addition, practically people tune an algorithm and execute it more than one trail if they wanted. This means users try to find the most successful algorithm for their scenario regardless of time. It is this feature of numerical optimization, which the CEC-C06 benchmark test functions also known as ‘‘The 100-digit challenge” examine. They calculate the values of functions at ‘‘horizontal” slices of the convergence plot [[39]](#_bookmark33). These test functions are considered for use in an annual com- petition of optimization. They are used to evaluate the algorithm for large scale optimization problems. The first three functions, CEC01 to CEC03, have various dimensions as shown in Appendix B [Table 9](#_bookmark13). On the other hand, the CEC04 to CEC10 functions set as 10-dimensional minimization problems in the range [ 100, 100], and they are shifted and rotated. All the CEC functions are scalable and all global optimum of these functions were united towards point 1. The results of the CEC-C06 2019 test functions for the LPB, DA, and PSO are shown in [Table 3](#_bookmark8). For each test func- tion in [Table 3](#_bookmark8), superior results are shown in bold. The test func- tions are solved 30 times utilizing 80 search agents over 500 iterations. The average, standard deviation, and processing time are then calculated. The results of the CEC-C06 2019 benchmark functions for DA and PSO are taken from [[40]](#_bookmark34). As shown in [Table 3](#_bookmark8), the value of metrics, average, and standard deviation for the LPB algorithm in almost all the CEC-C06 2019 test functions are smaller than DA, and PSO. However, PSO showed its superiority in CEC04. Additionally, the results of the LPB and PSO for optimizing CEC05, and CEC09 are comparative. The results of the CEC-C06 2019 benchmark functions revealed that for large scale optimization problems LPB provides better results compared to the DA, and PSO. The processing time for the LPB and DA for the CEC-C06 2019 is also shown in [Table 3](#_bookmark8). As clear, the PT for the LPB for optimizing all the functions is much smaller. The reason for this, as mentioned earlier, is that in the first stage of the LPB, a subset of the popula- tion is chosen based on this smaller group other subpopulations

—

Table 2

Comparison of results of the classical benchmark function between LPB, DA, PSO, and GA.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Test Function |  | LPB |  | DA | PSO | GA |  |
| TF1 | Ave. |  | 0.001877545 | 2.85E-18 | 4.2E-18 |  | 748.5972 |
|  | Std. |  | 0.002093616 | 7.16E-18 | 1.31E-18 |  | 324.9262 |
|  | PT (Seconds) | 160.840946 | | 1445.243327 | 249.665030 | 65.422226 | |
| TF2 | Ave. | 0.005238111 | | 1.49E-05 | 0.003154 | 5.971358 | |
|  | Std. | 0.003652512 | | 3.76E-05 | 0.009811 | 1.533102 | |
|  | PT (Seconds) | 169.076368 | | 1259.496468 | 3.826913 | 55.040008 | |
| TF3 | Ave. | 36.4748883 | | 1.29E-06 | 0.001891 | 1949.003 | |
|  | Std. | 29.22415523 | | 2.1E-06 | 0.003311 | 994.2733 | |
|  | PT (Seconds) | 202.408611 | | 1216.762524 | 12.702411 | 80.126424 | |
| TF4 | Ave. | 0.393866 | | 0.000988 | 0.001748 | 21.16304 | |
|  | Std. | 0.135818 | | 0.002776 | 0.002515 | 2.605406 | |
|  | PT (Seconds) | 191.301934 | | 1399.014810 | 2.877756 | 63.099468 | |
| TF5 | Ave. | 16.76919 | | 7.600558 | 63.45331 | 133307.1 | |
|  | Std. | 22.19251 | | 6.786473 | 80.12726 | 85007.62 | |
|  | PT (Seconds) | 130.846636 | | 1707.285731 | 5.224432 | 55.818782 | |
| TF6 | Ave. | 0.00203173 | | 4.17E-16 | 4.36E-17 | 563.8889 | |
|  | Std. | 0.0027832 | | 1.32E-15 | 1.38E-16 | 229.6997 | |
|  | PT (Seconds) | 157.547318 | | 1550.130722 | 2.795879 | 51.284046 | |
| TF7 | Ave. | 0.004975 | | 0.010293 | 0.005973 | 0.166872 | |
|  | Std. | 0.002965 | | 0.004691 | 0.003583 | 0.072571 | |
| TF8 | PT (Seconds) Ave. | 158.642028  —3747.65 | | 1593.877054  —2857.58 | 8.982616  —7.1E + 11 | 56.555067  —3407.25 | |

Std. 189.0206 383.6466 1.2E + 12 164.4776

PT (Seconds) 162.354305 1738.794894 8.266467 55.234252

TF9 Ave. 0.001567 16.01883 10.44724 25.51886

Std. 0.001842 9.479113 7.879807 6.66936

PT (Seconds) 159.074029 1638.957037 4.816792 84.833759

TF10 Ave. 0.017933 0.23103 0.280137 9.498785

Std. 0.013532 0.487053 0.601817 1.271393

PT (Seconds) 128.431567 1297.325669 8.013542 84.666823

TF11 Ave. 0.066355 0.193354 0.083463 7.719959

Std. 0.030973 0.073495 0.035067 3.62607

PT (Seconds) 130.664299 1210.086084 9.429028 56.656545

TF12 Ave. 2.78659E-05 0.031101 8.57E-11 1858.502

Std. 3.83626E-05 0.098349 2.71E-10 5820.215

PT (Seconds) 140.837076 1464.060419 22.898798 102.745164

TF13 Ave. 0.000309 0.002197 0.002197 68047.23

Std. 0.000512 0.004633 0.004633 87736.76

PT (Seconds) 139.449467 1339.438272 16.752814 103.377836

TF14 Ave. 0.998004 103.742 150 130.0991

Std. 1.26E-11 91.24364 135.4006 21.32037

PT (Seconds) 170.207352 1034.450489 86.298548 152.142368

TF15 Ave. 0.002358 193.0171 188.1951 116.0554

Std. 0.003757 80.6332 157.2834 19.19351

PT (Seconds) 247.224271 1659.652400 8.250347 54.974533

TF16 Ave. —1.03163 458.2962 263.0948 383.9184

Std. 2.46E-06 165.3724 187.1352 36.60532

PT (Seconds) 181.858429 969.827007 4.247415 80.998874

TF17 Ave. 0.397888 596.6629 466.5429 503.0485

Std. 3.16E-06 171.0631 180.9493 35.79406

PT (Seconds) 141.213291 1018.757437 2.607163 50.990811

TF18 Ave. 3.000142 229.9515 136.1759 118.438

Std. 0.000283 184.6095 160.0187 51.00183

PT (Seconds) 180.663489 1001.716543 2.718852 80.273981

TF19 Ave. —3.86278 679.588 741.6341 544.1018

Std. 9.61E-07 199.4014 206.7296 13.30161

PT (Seconds) 169.415055 1312.805448 8.952319 77.905123

are built. The perfect subpopulation has priority to be optimized first, then the good subpopulation and so on. Since the subpopula- tions are much smaller compared to the main population, search- ing for the solutions in these subpopulations is speeder. Consequently, this improves the randomness and saves the opti- mization time simultaneously. However, compared to the PSO and GA, the PT of the LPB is higher.

* 1. *Statistical tests*

The Wilcoxon rank-sum test function [[35]](#_bookmark29) is used to verify the importance of the results statistically. The *p* values reported in [Table 4](#_bookmark9) for classical benchmark test functions prove that for most of the test functions the LPB showed significantly better results

compared to the DA. Again, in reference [[19]](#_bookmark25) it was proved that the results of the DA are statistically significant comparing to the PSO and GA. This means that there is no need to compare the pro- posed algorithm with PSO and GA statistically since it has proved its superiority against DA. As shown in [Table 4](#_bookmark9), all the results except (TF6, TF11, TF12, and TF19) were smaller than 0.05, which proves the importance of the results of the proposed algorithm.

* 1. *Real world application*

In this section, the proposed algorithm is used to optimize a generalized assignment problem. The problem and its representa- tion are discussed in the following two sections.

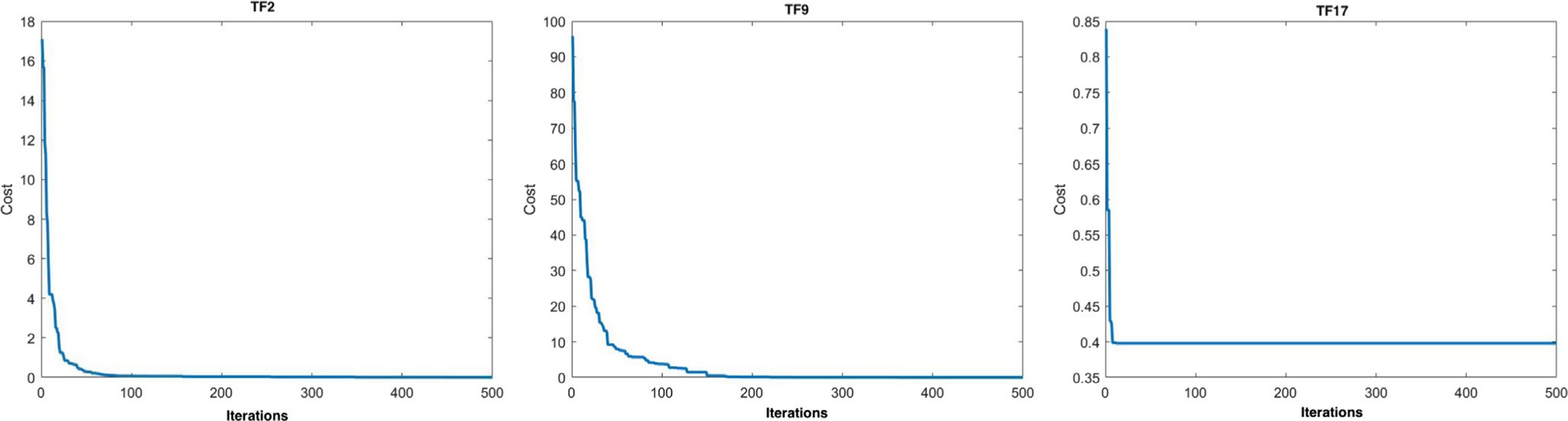


Fig. 1. Convergence curve for LPB on unimodal, multi-modal, and composite benchmark function.

Table 3

IEEE CEC 2019 benchmark test results.

CEC Function LPB DA PSO

CEC01 Ave. 7494381363.65768 543 × 108 1.47127E + 12

Std. 8138223463.28023 669 × 108 1.32362E + 12

PT (Seconds) 377.373846 2034.958870 382.330436

CEC02 Ave. 17.63898 78.0368 15183.91348

Std. 0.31898 87.7888 3729.553229

PT (Seconds) 140.912536 2122.108475 6.064791

CEC03 Ave. 12.7024 13.7026 12.70240422

Std. 0 0.0007 9.03E-15

PT (Seconds) 144.194876 2223.799974 8.901970

CEC04 Ave. 77.90824 344.3561 16.80077558

Std. 29.88519 414.0982 8.199076134

PT (Seconds) 137.305797 1720.974833 5.179151

CEC05 Ave. 1.18822 2.5572 1.138264955

Std. 0.10945 0.3245 0.089389848

PT (Seconds) 138.406681 1722.243949 5.370252

CEC06 Ave. 3.73895 9.8955 9.305312443

Std. 0.82305 1.6404 1.69E + 00

PT (Seconds) 142.041586 1401.682147 131.167162

CEC07 Ave. 145.28775 578.9531 160.6863065

Std. 177.8949 329.3983 104.2035197

PT (Seconds) 122.135692 1376.289834 5.436392

CEC08 Ave. 4.88769 6.8734 5.224137165

Std. 0.67942 0.5015 0.786760649

PT (Seconds) 138.207450 1802.883649 5.527832

CEC09 Ave. 2.89429 6.0467 2.373279266

Std. 0.23138 2.871 0.018437068

PT (Seconds) 141.699472 1365.799778 4.446880

CEC10 Ave. 20.00179 21.2604 20.28063455

Std. 0.00233 0.1715 0.128530895

PT (Seconds) 147.995515 1699.088096 9.462923

* + 1. *Problem definition*

A generalized assignment problem known as (GAP) is a popular NP-hard combinatorial optimization problem [[41]](#_bookmark35). The main goal in the GAP is assigning a set of tasks to a set of workers with min- imum cost. In this work, we assign cases in the court to justice teams in a way that the cases could be finished within a minimum number of working hours. Assigning cases and justice administra- tion in the judicial system is routine works, however, they are very time-consuming. Increasing caseloads at any time will make the

way that the total hours of assigning cases to the justice teams are minimized. To form the problem mathematically, first we define the following symbols:

i ? row number indicating ith case i e [1, N]

j ? column number indicating jth justice team j e[1, N] C[i][j] ? cost of allocating ith case to the jth team X[i][j] = 1 if jth justice team is assigned to ith case X[i][j] = 0 otherwise.

The problem can be formulated mathematically as:

problem more series. In this work, we use the proposed algorithm *N N*

to assign the right case to the right justice team and to assign a *Min* X X *C*½*i*]½*j*]*X*½*i*]½*j*]

proper time to deliver the decision of the court. The cases should

*i*¼1

*j*¼1

be assigned to the teams in the base of the number of hours required by that team to deal with that case. So that, it can be con- sidered that *N* cases and *N* justice teams are available where we have to assign each case to one and only one justice team in a

Subject to:

X *X*½*i*]½*j*] ¼ 1; 8*i* 2 *N* ¼ f1; 2; ·· · *N*g

*N*

*i*¼1

Table 4

The wilcoxon rank-sum test overall runs for classical benchmark test functions.

Test Function LPB Vs. DA TF1 7.72E-06

TF2 1.07E-10

TF3 5.52E-09

TF4 3.42E-06

TF5 0.006739

TF6 0.75328294

TF7 7.77E-13

TF8 4.23E-27

TF9 1.91E-05

TF10 1.08E-09

two points between 1 and *N* were generated and values of those two positions were swapped. The proposed algorithm was applied to the problem using 80 individuals, for 200 iterations. To verify the ability of the algorithm to solve the problem different size of the matrix was given to the algorithm, as shown in [Table 5](#_bookmark10). To run the program a standard laptop with processor Intel Core i7, 16 GHz was used. The results for different matrix sizes are shown in [Table 5](#_bookmark10).

In all the cases the population size was kept to 80, and the val- ues for the matrix were generated in the range [10 100]. [Fig. 2](#_bookmark12) shows the convergence of the algorithm towards the global mini- mum for solving the aforementioned problem using different size for the matrix. The Figures show that the size of the matrix will

|  |  |  |
| --- | --- | --- |
| TF11 TF12 | 5.96E-17  0.138213 | not affect the accuracy of the proposed algorithm and its conver-  gence towards the global minimum. |
| TF13 | 0.185156 |  |
| TF14 | 0.04631 |  |
| TF15 | 0.025386 | 6. Conclusions |
| TF16 | 0.033765 |  |

TF17 0.089253

TF18 0.007899

TF19 0.35758

X *X*½*i*]½*j*] ¼ 1; 8*j* 2 *N* ¼ f1; 2; ·· · *N*g

*N*

*j*¼1

*X*½*i*]½*j*] 2 f0; 1g

* + 1. *Problem representation*

Representing the problem will be a row from 1 to *N* examining the square cost matrix. Every individual in the population is a per- mutation from 1 to *N*. If the *jth* element in the row is *i*, thus, the *ith* case will be given to the *jth* justice team. For instance, let’s consider the following matrix:

This paper proposed another metaheuristic algorithm based on the process of transferring graduated learners from high school to university and improving the studying behaviors of the learners at colleges. The genetic algorithm inspired this algorithm. The two most important phases of metaheuristic algorithms (exploitation and exploration) were outlined. Mimicking the process of transfer- ring graduated learners from high school to college and dividing them into different groups according to their GPA outlined the for- mer phase. The exploration phase, however, was designed by mim- icking the process of improving the level of learners by utilizing various affective study skills. The parameters used in the LPB were *dp*, crossover, mutation. The *dp* parameter is used in the first steps of the algorithm to divide the population into different groups. The latter two parameters were utilized in the process of improving learners studying skills.

The ability of the proposed work was benchmarked using tradi- tional test function and the CEC-C06 2019 functions. The results

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Case1 | Team1 23 | Team2 21 | Team3 12 | Team4 30 | Team5 19 | were compared to PSO, GA and one of the most recently developed  algorithms, which is DA. It was proven that the LPB performed bet- ter in most of the cases. Moreover, The processing time of the algo- |
| Case2 | 30 | 25 | 13 | 22 | 21 | rithm was compared to the GA, PSO, and DA. the PT of the proposed |
| Case3 | 21 | 23 | 32 | 40 | 15 | work was much smaller compared to the DA. However, it was |
| Case4 | 12 | 32 | 40 | 32 | 29 | found that the processing time of the PSO, and GA is smaller than |
| Case5 | 20 | 15 | 21 | 27 | 22 | the LPB. Additionally, the results of the Wilcoxon rank-sum test |
|  |  |  |  |  |  | function proved the significance of the produced results by the |

If the solution is [4 5 2 3 1] that means case 4 in column 1 with cost 12 will be given to the first justice team, case 5 in column 2 with cost 15 will be given to the second team, case 2 with cost 13 in col- umn 3 will be given to the third team, and so on. Because of the constraint that says every case should be assigned to one and only one team and according to the encoding used, elements in each tuple should be unique. Thus, partially mapped crossover [[42]](#_bookmark36) was used where the individuals are permutations of numbers between 1 and *N*. For mutation, swap mutation was used, randomly

Table 5

Result of the court case assignment problem with varying size.

proposed technique. Furthermore, the ability of the algorithm was tested using a real-world NP-hard problem. Again, the results proved the effectiveness of the proposed algorithm in solving a real-world problem. As per finding of the examined work, it can be concluded that the proposed work is able to outperform most of the algorithms in the literature. However, bigger problem sizes for combinatorial optimization could be a challenge for LPB. There- fore, it is recommended for researchers in different fields to use it as an optimization technique.

For future works, a number of research directions can be rec- ommended. First of all, the authors will focus on reducing the processing time of the algorithm. Moreover, implementing the multi objective version of the algorithm is another research direc- tion. Modifying the algorithm to improve the exploitation phase

Size of matrix Optimal

Solution

No. Of Generations

Time Required (Sec.)

of LPB is another area that the authors are planning to implement in the future. Besides, another future work is finding new

10 × 10 218 17 0.14

15 × 15 350 15 0.17

20 × 20 425 34 0.33

30 × 30 676 57 0.53

parameters to replace the parameters from the genetic algorithm. In addition, utilizing the proposed technique to optimize different problems and compare the results with other heuristic techniques.

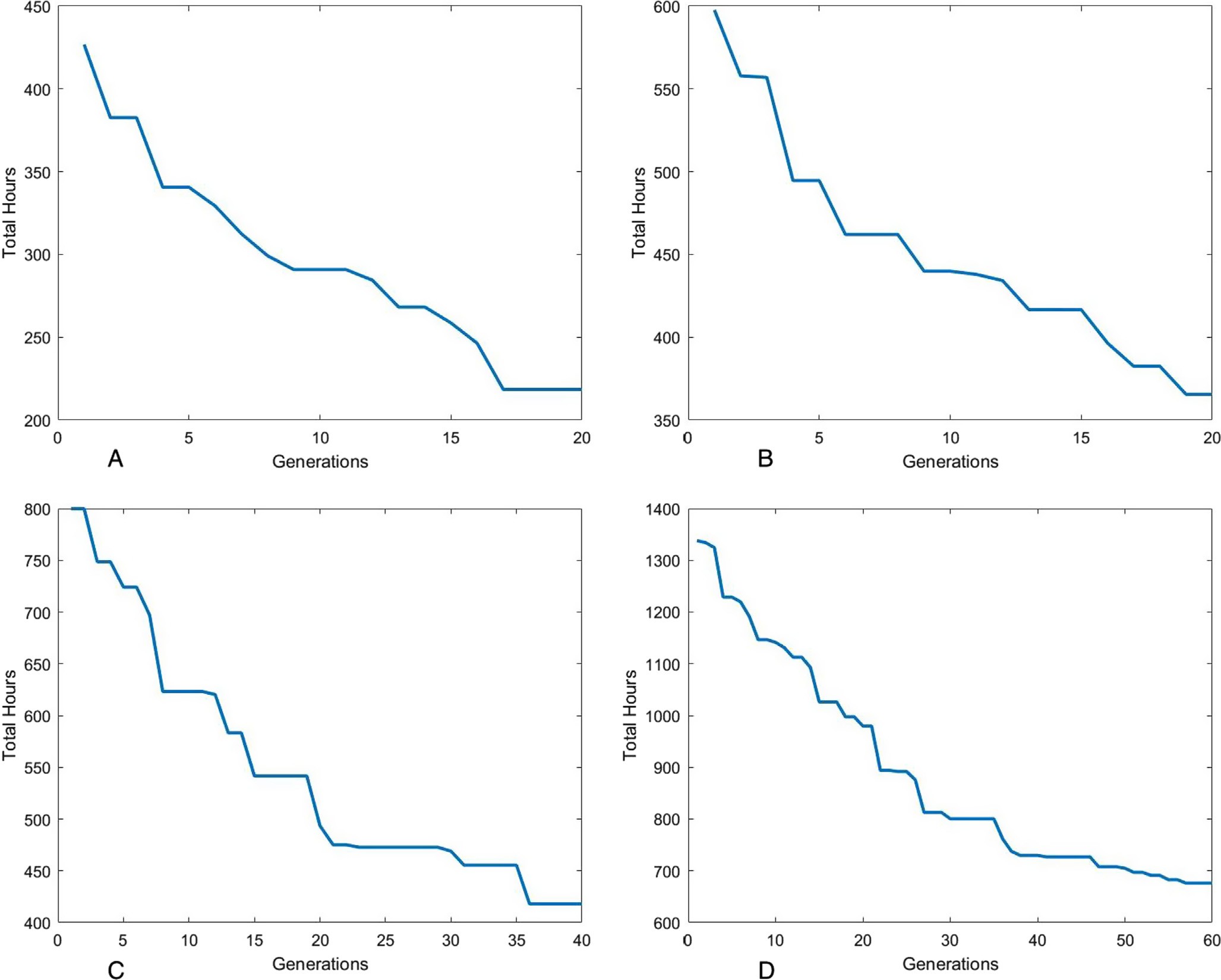


Fig. 2. Convergence to the global minimum using a different number of cases and justice teams, A) 10x10, B) 15x15, C) 20x20, D) 30x30.

Acknowledgment

The authors would like to send special thanks to Mr. Ahmed Saadaldin Qosaeri from the University of Kurdistan Hewler, for his thoughtful ideas and discussion.

Appendix A

Single-objective test problems are used in this work. See [Tables](#_bookmark11) [6, 7, and 8](#_bookmark11) for the mathematical representation of traditional benchmark functions used in this work.

Table 6

Unimodal benchmark functions.

Function Dimension Range Shift position *fmin*

*TF*1(*x*) = P*n x*2

*i*=1 *i*

10 [—100, 100] [—30, —30, .. . —30] 0

*TF*2(*x*) = P*n* |*xi*| + Q*n*

*i*=1

*i*=1

*j*=1 *j*

|*xi*| 10 [—10,10] [—3, —3, .. . —3] 0

*TF*3(*x*) = P*n*

P*i*

*x* 2 10 [—100, 100] [—30, —30, .. . —30] 0

*TF*4(*x*) = *max*{|*x*|, 16*i*6*n*}

*i*

*i*=1

*TF*5(*x*) = P*n*—1 h100(*xi* 1 — *x*2 2

*i*=1

+

1) + (

10 [—100, 100] [—30, —30, .. . —30] 0

*xi* — 1)2 i 10 [—30, 30] [—15, —15, ... —15] 0

*TF*6(*x*) = P*n*

*i*=1

*TF*7(*x*) = P*n*

*i*=1 *i*

([*xi* + 0.5])2 10 [—100, 100] [—750, .. . —750] 0

*ix*4 + *random*[0, 1] 10 [—1.28, 1.28] [—0.25, .. . —0.25] 0

Table 7

Multi-modal benchmark functions.

Function Range Shift Position *fmin*

*i*=1 *i*

|  |  |  |
| --- | --- | --- |
| *TF*8(*x*) = P*n* —*x*2*sin* p|ﬃﬃ*x*ﬃﬃ*i*ﬃ|ﬃﬃ [—500, 500]  *TF*9(*x*) = P*n x*2 — 10 cos (2p*xi*) + 10 [—5.12, 5.12] | [—300, .. . —300]  [—2, —2, .. . —2] | —418.9829 X 5  0  0 |
| *i*=1 *i n i*=1 |  |  |
| *TF*11(*x*) = 1 P*n x*2 — Q*n cos* *xi*ﬃ + 1 [—600, 600] | [—400, .. . —400] | 0 |
| *TF*12(*x*) = p n10*sin*(p*y*1 ) + P*n*—1 (*yi* — 1)2 h1 + 10*sin*2(p*yi* + 1)i + (*yn* — 1)2 o + P*n* 1 *u*(*xi*, 10, 100, 4) [—50, 50] | [—30, 30, .. . 30] | 0 |
| *TF*13(*x*) = 0.1n*sin*2(3p*x*1) + P*n* (*xi* — 1)2 h1 + *sin*2(3p*xi* + 1)i + (*xn* — 1)2 h1 + *sin*2(2p*xn*)io + P*n u*(*xi*, 5, 100, 4) [—50, 50] | [—100, .. . —100] | 0 |

*i*=1

*i*

*TF*10(*x*) = —20*exp* —0.2

qPﬃﬃﬃﬃﬃﬃﬃﬃﬃﬃﬃﬃﬃﬃﬃﬃ P

1

*n*

*x*2

— *exp*

*n*

*cos*(2p*xi*) + 20 + *e* [–32, 32]

4000

*n*

*i*=1 *i*

*i*=1

,*i*

*i*=1 *i*=1

*i*=

*i*=1

Table 8

Composite benchmark functions.

Function Dimension Range *fmin*

*TF*14(*CF*1)*f* 1, *f* 2, *f* 3 ·· · *f* 10 = *Spherefunction*d1, d2, d3 ·· · d10 = [1, 1, 1, ·· · 1]k1, k2, k3, ··· k10 =[ 5 , 5 , 5 , ·· · 5 ]

100 100 100

100

10 [—5, 5] 0

*TF*15(*CF*2)*f* 1, *f* 2, *f* 3 ·· · *f* 10 = *Grienwank*' *sfunction*d1, d2, d3 ·· · d10 = [1, 1, 1, ·· · 1]k1, k2, k3, ·· · k10 =[ 5 , 5 , 5 , ·· · 5 ]

100 100 100

100

10 [—5, 5] 0

*TF*16(*CF*3)*f* 1, *f* 2, *f* 3 ·· · *f* 10 = *Grienwank*'*sfunction*d1, d2, d3 ·· · d10 = [1, 1, 1, ·· · 1]k1, k2, k3, ·· · k10 = [1, 1, 1, ·· · 1] 10 [—5, 5] 0

*TF*17(*CF*4)*f* 1, *f* 2 = *Ackley*' *sfunctionf* 3, *f* 4 = *Rastrigin*' *sfunctionf* 5, *f* 6 = *Weierstrass*' *sfunctionf* 7, *f* 8 = *Griewank*' *sfunctionf* 9,

*f* 10 = *Spherefunction*d1, d2, d3 ·· · d10 = [1, 1, 1, ·· · 1]k1, k2, k3, ·· · k10 =[ 5 , 5 , 1, 1, 5 , 5 , 5 , 5 , 5 , 5 ]

32 32

0.5 0.5 100 100 100 100

*TF*18(*CF*5)*f* 1*f* 2 = *Rastrigin*' *sfunctionf* 3, *f* 4 = *Weierstrass*'*sfunctionf* 5, *f* 6 = *Griewank*'*sfunctionf* 7, *f* 8 = *Ackley*' *sfunctionf* 9,

*f* 10 = *Spherefunction*d1, d2, d3 ·· · d10 = [1, 1, 1, ·· · 1]k1, k2, k3, ·· · k10 = [1 , 1 , 5 , 5 , 5 , 5 ,  5 ,  5 , 5 , 5 ]

5 5 0.5 0.5 100 100 32 32 100 100

10 [—5, 5] 0

*TF*19(*CF*6)*f* 1*f* 2 = *Rastrigin*' *sfunctionf* 3, *f* 4 = *Weierstrass*'*sfunctionf* 5, *f* 6 = *Griewank*'*sfunctionf* 7, *f* 8 = *Ackley*' *sfunctionf* 9,

*f* 10 = *Spherefunction*d1, d2, d3 ·· · d10 = [0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1]k1, k2, k3, ·· · k10

10 [—5, 5] 0

1 1 5 5 5 5 5 5 5 5

= [0.1 \* 5 , 0.2 \* 5 , 0.3 \* 0.5 , 0.4 \* 0.5 , 0.5 \* 100 , 0.6 \* 100 , 0.7 \* 32 , 0.8 \* 32 , 0.9 \* 100 , 1 \* 100]

10 [—5, 5] 0

Table 9

CEC-C06 2019 benchmark functions [[37]](#_bookmark32).

Function Functions Dimension Range *fmin*

CEC01 STORN’S CHEBYSHEV POLYNOMIAL FITTING PROBLEM 9 [—8192, 8192] 1

CEC02 INVERSE HILBERT MATRIX PROBLEM 16 [—16384, 16384] 1

CEC03 LENNARD-JONES MINIMUM ENERGY CLUSTER 18 [—4, 4] 1

CEC04 RASTRIGIN’S FUNCTION 10 [—100, 100] 1

CEC05 GRIENWANK’S FUNCTION 10 [—100, 100] 1

CEC06 WEIERSRASS FUNCTION 10 [—100, 100] 1

CEC07 MODIFIED SCHWEFEL’S FUNCTION 10 [—100, 100] 1

CEC08 EXPANDED SCHAFFER’S F6 FUNCTION 10 [—100, 100] 1

CEC09 HAPPY CAT FUNCTION 10 [—100, 100] 1

CEC10 ACKLEY FUNCTION 10 [—100, 100] 1

Appendix B

The CEC-C06 2019 benchmark functions are shown in the fol- lowing table:

References

1. [Bezdek JC. On the relationship between neural networks, pattern recognition,](http://refhub.elsevier.com/S1110-8665(20)30141-9/h0005) [and intelligence. Int J Approximate Reasoning 1992;6(2):85–107](http://refhub.elsevier.com/S1110-8665(20)30141-9/h0005).
2. [Xing B, Gao WJ. Innovative computational intelligence: A rough guide to 134](http://refhub.elsevier.com/S1110-8665(20)30141-9/h0010) [clever algorithms. Cham, Switzerland: Springer Publication; 2014](http://refhub.elsevier.com/S1110-8665(20)30141-9/h0010).
3. [Du KL, Swamy MNS. Search and optimization by](http://refhub.elsevier.com/S1110-8665(20)30141-9/h0015) [metaheuristics. Switzerland: Springer Publication; 2016](http://refhub.elsevier.com/S1110-8665(20)30141-9/h0015).
4. Jourdan, L., Basseur, M. and Talbi, E. Hybridizing exact methods and metaheuristics: A taxonomy. Eur J Operat Res 2009; [online] 199(3), pp.620-

629. Available at: [https://www.sciencedirect.com/science/article/abs/pii/](http://www.sciencedirect.com/science/article/abs/pii/) S0377221708003597 [Accessed 26 Jun. 2019].

1. Lawler, E. and Wood, D. Branch-and-Bound Methods: A Survey. Operat Res 1966; [online] 14(4), pp.699-719. Available at: https://pubsonline.informs. org/doi/abs/10.1287/opre.14.4.699 [Accessed 18 May 2019].
2. [Goldberg DE, Holland JH. Genetic algorithms and Machine Learning. Mach](http://refhub.elsevier.com/S1110-8665(20)30141-9/h0030) [Learn 1988;3(2–3):95–9](http://refhub.elsevier.com/S1110-8665(20)30141-9/h0030).
3. Raja, M., Umar, M., Sabir, Z., Khan, J. and Baleanu, D. A new stochastic computing paradigm for the dynamics of nonlinear singular heat conduction model of the human head. Eur Phys J Plus, [online]; 2018: 133(9). Available at: https://link.springer.com/article/10.1140/epjp/i2018-12153-4 [Accessed 10

Dec. 2019].

1. Zahoor Raja, M., Shah, Z., Anwaar Manzar, M., Ahmad, I., Awais, M. and Baleanu, D. A new stochastic computing paradigm for nonlinear Painlevé II systems in applications of random matrix theory. Eur Phys J Plus, [online]; 2018: 133(7). Available at: https://link.springer.com/article/10.1140/epjp/

i2018-12080-4 [Accessed 10 Dec. 2019].

1. Ahmad, I., Zahid, H., Ahmad, F., Raja, M. and Baleanu, D. Design of computational intelligent procedure for thermal analysis of porous fin

model. Chin J Phys, [online]; 2019: 59, pp.641-655. Available at: https:// [www.sciencedirect.com/science/article/abs/pii/S0577907318308396?via%](http://www.sciencedirect.com/science/article/abs/pii/S0577907318308396?via%25) 3Dihub [Accessed 10 Dec. 2019].

1. Sabir, Z., Manzar, M., Raja, M., Sheraz, M. and Wazwaz, A. Neuro-heuristics for nonlinear singular Thomas-Fermi systems. Appl Soft Comput, [online]; 2018: 65, pp.152-169. Available at: [https://www.sciencedirect.com/science/article/](http://www.sciencedirect.com/science/article/) abs/pii/S1568494618300152 [Accessed 10 Dec. 2019].
2. Fogel, L. J., Owens, A. J., & Walsh, M. J. Intelligent decision making through a simulation of evolution. Behav Sci 1966; 11(4), 253–272. Available at: https://doi.org/10.1002/bs.3830110403
3. [Kirkpatrick S, Gelatt DC, Vechhi MP. Optimization by simulated annealing.](http://refhub.elsevier.com/S1110-8665(20)30141-9/h0060) [Science 1983;1966(220):671–80. Available at: https://pdfs.semanticscholar.](http://refhub.elsevier.com/S1110-8665(20)30141-9/h0060) [org/e893/4a942f06ee91940ab57732953ec6a24b3f00.pdf](http://refhub.elsevier.com/S1110-8665(20)30141-9/h0060).
4. [Dorigo M, Blum C. Ant colony optimization theory: A survey. Theoret Comput](http://refhub.elsevier.com/S1110-8665(20)30141-9/h0065) [Sci 2005;344(2–3):243–78](http://refhub.elsevier.com/S1110-8665(20)30141-9/h0065).
5. Kennedy, J. Particle Swarm Optimization. Encyclopedia of Machine Learning and Data Mining 2017; [online] pp.967-972. Available at: [https://www.](http://www/) semanticscholar.org/paper/Particle-Swarm-Optimization-Kennedy/ 20a5cda34f158ace5881d3f6635c3eb5b531c199 [Accessed 17 May 2019].
6. Basturk, B., Karaboga, D. An artificial bee colony (ABC) algorithm for numeric function optimi- zation. In: Proceedings of the IEEE Swarm Intelligence Symposium 2006; Indianapolis, Indiana, USA, 12–14 May; 2006.
7. Chu, S., Tsai, P. and Pan, J. Cat Swarm Optimization. Lecture Notes in Computer Science 2006; [online] pp.854-858. Available at: https://link.springer.com/ chapter/10.1007/978-3-540-36668-3\_94 [Accessed 18 May 2019].
8. [Yang XS. A New Metaheuristic Bat-Inspired Algorithm. Nat Insp Cooperat](http://refhub.elsevier.com/S1110-8665(20)30141-9/h0085) [Strateg Optimiz 2010;84:65–74. Available at: https://link.springer.com/](http://refhub.elsevier.com/S1110-8665(20)30141-9/h0085) [chapter/10.1007/978-3-642-12538-6\_6](http://refhub.elsevier.com/S1110-8665(20)30141-9/h0085).
9. Mirjalili, S., Mirjalili, S. and Lewis, A. Grey Wolf Optimizer. Adv Eng Softw 2014; [online] 69, pp.46-61. Available at: https://www.sciencedirect. com/science/article/pii/S0965997813001853 [Accessed 18 May 2019].
10. Mirjalili, S. Dragonfly algorithm: a new meta-heuristic optimization technique for solving single-objective, discrete, and multi-objective problems. Neur Comput Appl 2015; [online] 27(4), pp.1053-1073. Available at: https://link. springer.com/article/10.1007/s00521-015-1920-1 [Accessed 18 May 2019].
11. Abdullah, J., Ahmed, T. Fitness Dependent Optimizer: Inspired by the Bee Swarming Reproductive Process. IEEE Access 2019; [online] 7, pp.43473- 43486. Available at: https://ieeexplore.ieee.org/ielx7/6287639/8600701/ 08672851.pdf?tp=&arnumber=8672851&isnumber=8600701&ref= [Accessed 31 May 2019].
12. Chandra Mohan, B. and Baskaran, R. A survey: Ant Colony Optimization based recent research and implementation on several engineering domain. Exp Syst Appl 2012; [online] 39(4), pp.4618-4627. Available at: https:// [www.sciencedirect.com/science/article/pii/S0957417411013996](http://www.sciencedirect.com/science/article/pii/S0957417411013996) [Accessed 18 May 2019].
13. Karaboga, D., Gorkemli, B., Ozturk, C. and Karaboga, N. A comprehensive survey: artificial bee colony (ABC) algorithm and applications. Artif Intellig Rev 2012; [online] 42(1), pp.21-57. Available at: https://link.springer.com/article/ 10.1007/s10462-012-9328-0 [Accessed 18 May 2019].
14. Zhang, Y., Wang, S. and Ji, G. A Comprehensive Survey on Particle Swarm Optimization Algorithm and Its Applications. Mathemat Probl Eng 2015; [online] Available at: [https://www.hindawi.com/journals/mpe/2015/931256/](http://www.hindawi.com/journals/mpe/2015/931256/) [Accessed 18 May 2019].
15. Chawla, M. Duhan, M. Bat Algorithm: A Survey of the State-of-the-Art. Appl Artif Intellig 2015; [online] 29(6), pp.617-634. Available at: https:// [www.tandfonline.com/doi/abs/10.1080/08839514.2015.1038434?](http://www.tandfonline.com/doi/abs/10.1080/08839514.2015.1038434) journalCode=uaai20 [Accessed 18 May 2019].
16. Lee, N., Li, X. Wang, D. A comprehensive survey on genetic algorithms for DNA motif prediction. Informat Sci 2018; [online] 466, pp.25-43. Available at: [https://www.sciencedirect.com/science/article/pii/S0020025518305206](http://www.sciencedirect.com/science/article/pii/S0020025518305206) [Accessed 18 May 2019].
17. Mohammed, H., Umar, S. Rashid, T. A Systematic and Meta-Analysis Survey of Whale Optimization Algorithm. Computat Intellig Neurosci 2019; [online] 2019, pp.1-25. Available at: [https://www.hindawi.com/journals/cin/2019/](http://www.hindawi.com/journals/cin/2019/) 8718571/cta/ [Accessed 18 May 2019].
18. Cˇrepinšek, M., Liu, S. and Mernik, M. Exploration and exploitation in evolutionary algorithms-A survey. ACM Computing Surveys 2019; [online] Available at: https://romisatriawahono.net/lecture/rm/survey/softcomputing/ Crepinsek%20-%20Exploration%20and%20Exploitation%20in%20Evolutionary% 20Algorithms%20-%202013.pdf [Accessed 27 Jun. 2019].
19. Qayyum, A. Learner help-seeking attitudes and behaviors in a digital era. Int J Educat Technol Higher Education 2018; [online] 15(1). Available at: https:// educationaltechnologyjournal.springeropen.com/articles/10.1186/s41239- 018-0100-7 [Accessed 31 May 2019].
20. CHEW, S. Improving Classroom Performance by Challenging Learner Misconceptions About Learning online 2010 Association for Psychological Science Available at: [https://www.psychologicalscience.org/observer/](http://www.psychologicalscience.org/observer/) improving-classroom-performance-by-challenging-learner-misconceptions- about-learning [Accessed 31 May 2019].
21. Karwowski, M. Are creative learners really welcome in the classrooms? Implicit theories of ‘‘good” and ‘‘creative” learner’ personality among polish teachers. Procedia - Soc Behav Sci 2010; [online] 2(2), pp.1233-1237. Available at: [https://www.sciencedirect.com/science/article/pii/S1877042810002193](http://www.sciencedirect.com/science/article/pii/S1877042810002193) [Accessed 31 May 2019].
22. Roediger, H.L., III, & Karpicke, J.D. Test-enhanced learning: Taking memory tests improves long-term retention. Associat Psychol Sci 2006; Available at: <http://learninglab.psych.purdue.edu/downloads/> 2006\_Roediger\_Karpicke\_PsychSci.pdf , 17, 249-255.
23. [Cao L, Nietfeld J. College Learners’ Metacognitive Awareness of Difficulties in](http://refhub.elsevier.com/S1110-8665(20)30141-9/h0160) [Learning the Class Content Does Not Automatically Lead to Adjustment of](http://refhub.elsevier.com/S1110-8665(20)30141-9/h0160) [Study Strategies. Austral J Educat Dev Psychol 2007;7:31–46](http://refhub.elsevier.com/S1110-8665(20)30141-9/h0160).
24. Gen, M. Lin, L. Genetic Algorithms. Wiley Encyclopedia of Computer Science and Engineering 2008; [online] Available at: https://doi.org/10.1002/ 9780470050118.ecse169 [Accessed 8 Jun. 2019].
25. Millis, B. Using Metacognition to Promote Learning. IDEA Paper #63 2016; IDEA, no. 63. Available at: [https://www.ideaedu.org/Portals/0/Uploads/](http://www.ideaedu.org/Portals/0/Uploads/) Documents/IDEA%20Papers/IDEA%20Papers/PaperIDEA\_63.pdf
26. Derrac, J., García, S., Molina, D. and Herrera, F. A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. Swarm Evolution Comput 2011; [online] 1 (1), pp.3-18. Available at: [https://www.sciencedirect.com/science/article/abs/](http://www.sciencedirect.com/science/article/abs/) pii/S2210650211000034 [Accessed 2 Jun. 2019].
27. Yao, X., Liu, Y. and Lin, G. Evolutionary programming made faster. IEEE Trans Evolut Comput 1999; [online] 3(2), pp.82-102. Available at: https://ieeexplore. ieee.org/document/771163 [Accessed 1 Jun. 2019].
28. [Price KV, Awad NH, Ali MZ, Suganthan PN. The 100-Digit Challenge: Problem](http://refhub.elsevier.com/S1110-8665(20)30141-9/h0185) [Definitions and Evaluation Criteria for the 100-Digit Challenge Special Session](http://refhub.elsevier.com/S1110-8665(20)30141-9/h0185) [and Competition on Single Objective Numerical](http://refhub.elsevier.com/S1110-8665(20)30141-9/h0185) [Optimization. Singapore: Nanyang Technological University; 2018](http://refhub.elsevier.com/S1110-8665(20)30141-9/h0185).
29. Molga, M., Smutnicki, C. Test functions for optimization needs. Test functions for optimization needs 2005; Available at: <http://www.robertmarks.org/> Classes/ENGR5358/Papers/functions.pdf [Accessed 1 Jun. 2019].
30. Liang, J., Suganthan, P., Deb, K. Novel composition test functions for numerical global optimization. Proceedings 2005 IEEE Swarm Intelligence Symposium, 2005. SIS 2005, [online] (Proceedings of 2005 IEEE), pp.68-75. Available at: https://ieeexplore.ieee.org/document/1501604 [Accessed 2 Jun. 2019].
31. Rahman, C. Rashid, T. Dragonfly Algorithm and Its Applications in Applied Science Survey. Computat Intellig Neurosci, [online] 2019, pp.1-21. Available at: https://new.hindawi.com/journals/cin/2019/9293617/ [Accessed 28 Dec. 2019].
32. Sahu, A. Tapadar, R. Solving the Assignment problem using Genetic Algorithm and Simulated Annealing. IAENG Int J Appl Mathemat 2007; [online] Available at: <http://www.iaeng.org/IJAM/issues_v36/issue_1/IJAM_36_1_7.pdf> [Accessed 24 May 2019].
33. Goldberg, D. Lingle, R. Alleles, Loci, and the Traveling Salesman Problem. In: Proceedings of the First International Conference on Genetic Algorithms and Their Application 1985, L. Erlbaum Associates Inc. Hillsdale, NJ, USA ©; 1985, pp.154-159.