mathematics

Article

A Shuffle-Based Artificial Bee Colony Algorithm for Solving Integer Programming and Minimax Problems

Ivona Brajević

Faculty of Applied Management, Economics and Finance, University Business Academy in Novi Sad, Jevrejska 24, 11000 Belgrade, Serbia; ivona.brajevic@mef.edu.rs

Abstract: The artificial bee colony (ABC) algorithm is a prominent swarm intelligence technique due to its simple structure and effective performance. However, the ABC algorithm has a slow convergence rate when it is used to solve complex optimization problems since its solution search equation is more of an exploration than exploitation operator. This paper presents an improved ABC algorithm for solving integer programming and minimax problems. The proposed approach employs a modified ABC search operator, which exploits the useful information of the current best solution in the onlooker phase with the intention of improving its exploitation tendency. Furthermore, the shuffle mutation operator is applied to the created solutions in both bee phases to help the search achieve a better balance between the global exploration and local exploitation abilities and to provide a valuable convergence speed. The experimental results, obtained by testing on seven integer programming problems and ten minimax problems, show that the overall performance of the proposed approach is superior to the ABC. Additionally, it obtains competitive results compared with other state-of-the-art algorithms.

Keywords: artificial bee colony; swarm intelligence; shuffle mutation operator; integer programming problems; minimax problems; global optimization

Citation: Brajević, I. A Shuffle-Based Artificial Bee Colony Algorithm for Solving Integer Programming and Minimax Problems. *Mathematics* **2021**, *9*, 1211. https://doi.org/10.3390/math9111211

Academic Editor: Erik Valdemar Cuevas, Francisco G. Montoya and Alfredo Alcayde

Received: 25 March 2021
Accepted: 25 May 2021
Published: 27 May 2021

1. Introduction

A wide variety of problems from different areas can be formulated as integer programming and minimax problems. Some applications in which integer programming problems appear are system-reliability design, scheduling, capital budgeting, warehouse location, portfolio analysis, automated production systems, mechanical design, transportation and cartography [1–5]. Furthermore, minimax optimization problems are found in many applications, such as optimal control, engineering design, game theory, signal and data processing [6–10].

Since integer programming is known to be NP-hard, solving these problems is considered a challenging task. Dynamic programming and branch-and-bound (BB) are well-known exact integer programming methods [11,12]. These methods divide the feasible region into smaller sub-regions or problems into sub-problems. The main drawback of dynamic programming is that the amount of computation necessary for an optimal solution exponentially grows as the number of variables rises. Branch-and-bound techniques have a high computational cost when solving large-scale problems that require the exploration of a search tree containing hundreds of nodes [11].

Metaheuristic optimization algorithms provide high-quality solutions in an acceptable amount of time. These techniques do not make any presumptions about the problem and can be used to solve a broad class of challenging optimization problems [13–18]. One of the most notable classes of metaheuristics, swarm intelligence (SI) algorithms, has foundations in imitating the collective behavior of biological agents. Particle swarm optimization (PSO) [19], artificial bee colony (ABC) [20], harmony search (HS) [21], firefly algorithm (FA) [22], gravitational search algorithm (GSA) [23], cuckoo search (CS) [24], whale optimization algorithm (WOA) [25] and bat algorithm (BA) [26] are some of the notable SI algorithms. In the last two decades, many SI algorithms were applied to solve integer programming problems. For instance, the PSO was employed to solve integer programming problems in [2]. On standard test problems, the PSO outperformed the branch-and-bound method in most cases.

Sequential quadratic programming (SQP) and smoothing techniques are common strategies for solving minimax problems [6]. These methods perform local minimization and require derivatives information for the objective function, which in most applications are not analytically available. Furthermore, SQP and smoothing techniques struggle to achieve satisfactory solutions when the objective function is discontinuous. On the other hand, metaheuristics are problem-independent optimization methods. Search operators of these methods use some randomness, which enables the algorithm to move away from a local optimum to search on a global scale [27]. Hence, metaheuristic optimization algorithms are considered an adequate alternative for minimax problems.

Since their invention, the original variants of metaheuristic algorithms have been modified to improve their performances. In [28], a memetic PSO algorithm that integrates local search methods to the basic PSO was developed. The local and global variants of the memetic PSO scheme were tested to solve minimax and integer programming problems. The experimental results showed that the memetic PSO outperformed the corresponding variants of the PSO algorithm in the majority of benchmarks. A hybrid cuckoo search algorithm with the Nelder Mead method, named HCSNM, for solving integer programming and minimax problems is proposed in [29]. In [29], it was concluded that the use of the Nelder Mead method enhances the convergence speed of the basic CS technique. A hybrid bat algorithm (HBDS) to solve integer programming is proposed in [30]. The HBDS incorporates direct search methods in the BA to enhance the intensification ability of the BA. Recently, a new hybrid harmony search algorithm with the multidirectional search method, called MDHSA, is developed to enhance the performance of the standard HS algorithm for solving integer programming and minimax problems [31].

The efficiency of the basic ABC algorithm for integer programming problems was investigated in [11]. To our knowledge, the ABC is not tested on a minimax test function in any of the studies. Therefore, investigating the performance of the standard ABC algorithm for solving minimax problems and proposing suitable modifications with the aim to further improve its performance for integer programming and minimax problems is a research problem.

Motivated by these reasons, this paper presents a shuffle-based artificial bee colony algorithm (SB-ABC) for solving integer programming and minimax problems. Although ABC has achieved success in different research fields, it was noticed that the exploitation ability of the ABC is deficient because of a randomly picked neighborhood food source in its solution search equation [32]. Therefore, the ABC algorithm has a slow convergence rate when it is applied to solve complex optimization problems. In order to enhance the exploitation ability of the ABC algorithm, the proposed approach employs a modified ABC search operator, which exploits the useful information of the current best solution in the onlooker phase. Furthermore, in certain iterations, the shuffle mutation operator is applied to the newly created solutions in both bee phases. In that way, the proposed algorithm provides useful diversity in the population, which is crucial in finding a good balance between exploitation and exploration. The SB-ABC algorithm is tested on seven integer programming problems and ten minimax problems. The obtained results for integer programming problems are compared to those of the ABC, BB method and 12 other metaheuristics. For minimax problems, the achieved results are compared to those of the ABC, SQP method and 11 other algorithms. Experimental results indicated that the SB-ABC algorithm obtained highly competitive results in comparison with the other algorithms presented in the literature.

The paper is organized as follows. In Section 2, definitions of minimax and integer programming problems are given. The standard ABC is presented in Section 3. The proposed shuffle-based artificial bee colony approach is explained in Section 4. In Section 5, the optimization results are presented and analyzed. In Section 6, the influence of the proposed modifications on the performance of the SB-ABC algorithm is discussed. Section 7 provides concluding remarks.

2. Problem Statements

An integer programming problem is a discrete optimization problem where all of the variables are limited to integer values. A general integer programming problem can be stated as [11]:

$$min\ f(x),\ x \in S \subseteq \mathbb{Z}^n \tag{1}$$

where S is the feasible region and $\mathbb{Z}$ denotes the set of integers. A problem where some variables are constrained to integers while some variables are not is a mixed integer programming problem. A special instance of the integer programming problem is that in which the variables are restricted to be either 0 or 1. This case is called the 0–1 programming problem or the binary integer programming problem.

Minimax optimization deals with a composition of an inner maximization problem and an outer minimization problem. A general form of the minimax problem can be stated as [31]:

$$min\ F(x) \tag{2}$$

where

$$F(x) = max\ f_i(x),\ i = 1, \ldots, m \tag{3}$$

with $f_i(x)$: $S \subset \mathbb{R}^n \to \mathbb{R}$, $i = 1, \ldots, m$.

Furthermore, a nonlinear programming problem, with inequality constrains, of the form

$$\begin{aligned} &min\ F(x), \\ &g_i(x) \geq\ 0,\ \ i = 1, \ldots, m, \end{aligned} \tag{4}$$

can be transformed to minimax problems as follows:

$$min\ max\ f_i(x),\ i = 1, \ldots, m \tag{5}$$

where

$$\begin{aligned} &f_1(x) = F(x), \\ &f_i(x) = F(x) - \alpha_i \cdot g_i(x) \\ &\alpha_i >\ 0 \end{aligned} \tag{6}$$

for $i = 2, \ldots, m$. It has been shown that when α_i is large enough, the optimum point of the minimax problem coincides with the optimum point of the nonlinear programming problem [6].

3. Artificial Bee Colony Algorithm

Foraging behavior of a honey bee swarm motivated the development of the ABC algorithm [20]. The population of artificial bees is made of employed bees, onlooker bees and scout bees. One-half of the population consists of employed bees. Onlookers and scouts make the other half of the population. In the basic ABC, each food source represents a possible solution for the problem, and the number of the employed bees is equal to the number of food sources. All bees that are presently exploiting a food source are employed bees. The onlooker bees aim to choose promising food sources from those discovered by the employed bees according to the probability proportional to the quality of the food source. After the selection of the food source, the onlookers further seek food in the vicinity of the selected food source. The scout bees are transformed from several employed bees that abandon their unpromising food sources to seek new ones.

The control parameters of the basic ABC algorithm are the size of the population (SP), which is equal to the sum of employed and onlooker bees, the maximum cycle number (MCN), and parameter $limit$, which represents the number of trials for abandoning the food source. In the initialization phase, the ABC creates randomly distributed initial population, which includes SP solutions. Following this step, three phases—employed, onlooker and scout—are repeated for a certain number of iterations. After each iteration, the best-discovered solution is saved.

Each employed bee seeks a better food source in the employed phase. The search operator used to create a novel food source v_i from the old one x_i is given by:

$$v_{ij} = x_{ij} + \varphi \cdot (x_{ij} - x_{lj}) \tag{7}$$

where j is arandomly picked index of a parameter, x_l is a randomly selected food source that is different from x_i and φ is a uniform random number between $(-1, 1)$. Greedy selection between old and new food sources decides whether the old food source will be replaced by the new one.

In the onlooker phase, each onlooker bee chooses a food source according to the probability that is proportional to the fitness value. The same search strategy, which is given by Equation (7), is used to generate a candidate food source from the picked one. Greedy selection between old and new food sources decides whether the old food source (solution) will be updated. In the scout phase, a solution that can not be updated through a predetermined number of trials is replaced with a randomly created solution.

Many variants of ABC for solving continuous optimization problems were proposed [33–40]. For instance, an enhanced version of ABC, which introduces modifications related to elite strategy and dimension learning, is invented in [33]. The ABC variant, which uses novel search strategies in employed and onlooker bee phases, is developed in [34]. In [35], a hybrid method, which combines firefly and multi-strategy ABC, is developed for solving numerical optimization problems. An enhanced ABC based on the multi-strategy fusion is ABC variant and is proposed to improve the search ability of ABC with a small population [40].

Although the standard ABC was initially invented for continuous optimization problems, the modified variants have also been proposed for combinatorial and discrete problems [41–46]. Akay and Karaboga modified the ABC algorithm in order to solve integer programming problems. In this version of the ABC, a new control parameter called modification rate (MR) is employed in its solution search strategy [11]. The modification rate parameter controls the possible modifications of optimization parameters. In [41], an ABC algorithm with a modified choice function for the traveling salesman problem is developed. Two novel ABC algorithms in which a multiple colonies strategy is adopted are proposed to solve the vehicle routing problem [43]. The ABC technique that integrates the initial solutions, an elitism strategy, recovery and local search schemes is a newly developed variant of ABC for solving the operating room scheduling problem [45]. An improved ABC algorithm for solving the strength–redundancy allocation problem is presented in [46]. In general, application fields of the ABC method are data mining, neural networks, image processing, cryptanalysis, data clustering and engineering [47–53].

4. The Proposed Approach: SB-ABC

Important characteristics of each metaheuristic algorithm are exploitation and exploration [54]. Exploitation refers to the process of visiting areas of a search space in the neighborhood of previously found satisfactory solutions. Exploration is the process of generating solutions with ample diversity and far from the current solutions. A balanced combination of these conflicting processes is essential for successful optimization performance. According to Equation (7), the new individual is generated by moving the old solution to a randomly picked solution, and the direction of the search is random. Consequently, the solution search equation given by Equation (7) has good exploration tendency, but it is not promising at exploitation. Since too much exploration tends to decrease the convergence speed of the algorithm [35], the proposed approach uses modified ABC search equations in employed and onlooker bee phases. To obtain useful diversity in the population, in each bee phase, the shuffle mutation operator is applied to new candidate solutions.

To create a new solution v_i from the solution x_i in the employed bee phase, the SB-ABC algorithm uses a search strategy that is described by [11]:

$$v_{ij} = \begin{cases} x_{ij} + \varphi_i \cdot (x_{ij} - x_{kj}) & \text{, if } R_{ij} < MR \\ x_{ij} & \text{, otherwise} \end{cases} \tag{8}$$

where $j \in \{1, 2, \ldots, D\}$ and D is the number of optimization parameters or dimensions of the problem. In Equation (8), x_k is a randomly selected food source that is different from x_i, φ_i is a uniform random number between $(-1, 1)$, R_{ij} is a randomly chosen real number in range (0, 1) and MR is modification rate control parameter whose value is in the range (0, 1). A higher value of the MR parameter will enable more parameters to be changed in the parent solution with the aim to increase the convergence speed of the basic ABC algorithm.

In the onlooker bee phase of the SB-ABC algorithm, the solutions are chosen according to the probability, which is given by [51]:

$$p_i = 0.9 \cdot (fit_i / maxfit) + 0.1 \tag{9}$$

where the best fitness value in the population is denoted by *maxfit*, while fit_i marks the fitness value of the ith solution.

Inspired by the variant of the ABC proposed to solve numerical optimization, gbest-guided artificial bee colony (GABC) algorithm [55], we modify the search equation described by Equation (8) as follows:

$$v_{ij} = \begin{cases} x_{ij} + \varphi_{ij} \cdot (x_{ij} - x_{kj}) + \phi_{ij} \cdot (y_j - x_{ij}) & \text{, if } R_{ij} < MR \\ x_{ij} & \text{, otherwise} \end{cases} \tag{10}$$

where $j \in \{1, 2, \ldots, D\}$ and D is the number of optimization parameters, i.e., dimension of the problem. In Equation (10), v_i is a new candidate solution, x_i is parent solution, φ_{ij} is a uniform random number in range $(-1, 1)$, ϕ_{ij} is a uniform random number in the segment [0, 1.5], x_k is a randomly selected food source that is different from x_i, y_j is the jth parameter of the best solution found so far, and R_{ij} is a randomly chosen real number within (0,1). According to Equation (10), the third term can move the new potential solution towards the global best solution. Hence, the modified search strategy given by Equation (10) can enhance the exploitation tendency of the basic ABC algorithm.

The right amount of population diversity is of great significance in achieving a proper balance between exploitation and exploration. In the SB-ABC algorithm, the exploitation is increased by using the modified search equation in the onlooker bee phase. Thus, the differences among individuals of a population are decreased since the search process is quite focused on a local region of good solutions. To promote diversity at certain stages of the search process, a new parameter called random permutation production interval ($RPPI$) is introduced in the SB-ABC. This parameter is used as follows: after each $RPPI$th cycle, the shuffle mutation operator is applied to new candidate solutions at employed and onlooker bee phases. The shuffle mutation is a mutation operator where the mutated solution takes the components of the original solution, applying a permutation to them [56]. Usage of the shuffle mutation operator enables a better exploration of solutions but only every $RPPI$ iterations.

The proposed approach computes a value $trial_i$ for each solution x_i during the search process. A value $trial_i$ characterizes the non-advanced number of the solution x_i used for the abandonment. In the scout phase of the SB-ABC algorithm, one solution with the highest $trial$ value that is greater than the value of $limit$ control parameter, if such solution occurs, is exchanged with a randomly generated solution.

The pseudo-code of the employed bee phase is presented in Algorithm 1, while the procedure of the onlooker bee phase is described in Algorithm 2. The input of Algorithm 1 involves the current solutions x_i with corresponding values $trial_i$, $i = 1, 2, \ldots, SP/2$, current $cycle$ value, values of MR and $RPPI$ parameters, and the objective function f. The output of Algorithm 1 is the updated population of solutions x_i and $trial_i$ values, $i = 1, 2, \ldots, SP/2$, which will be employed in the onlooker bee phase. The input of Algorithm 2 includes the current population of solutions x_i with corresponding values $trial_i$, $i = 1, 2, \ldots, SP/2$, current $cycle$ value, values of MR and $RPPI$ parameters, and the objective function f. The output of Algorithm 2 is the updated population of solutions x_i and $trial_i$ values, $i = 1, 2, \ldots, SP/2$, which will be used in the next iteration. The pseudo-code of the SB-ABC algorithm is presented in Algorithm 3. The input of Algorithm 3 includes the values of SP, MCN, MR, $limit$ and $RPPI$ control parameters and the objective function f. The output of Algorithm 3 is the best solution found.

It is important to mention that the proposed approach SB-ABC introduces two modifications in comparison with the ABC algorithm adjusted for integer programming problems: use of the modified ABC search operator described by Equation (10) and the application of the shuffle mutation operator. The crucial difference between these two approaches consists in the different balance of exploitation and exploration. Exploitation is enhanced in the onlooker phase by applying the global best solution to guide the search process. Useful diversity of the population and better exploration of solutions is achieved on the global level by applying the shuffle mutation operator every $RPPI$th iteration.

The SB-ABC algorithm employs three specific control parameters to manage the search process: modification rate MR, $limit$ and $RPPI$, which determines the cycles in which the shuffle mutation operator is applied to candidate solutions. It also uses standard control parameters for all population-based metaheuristics, the population size and maximum number of cycles. In order to solve the integer programming problems, the SB-ABC rounds the parameter values to the closest integer after evolution according to Equations (8) and (10). Solutions were also rounded after the initialization phase and scout phase of the algorithm. Therefore, they were considered as integer numbers for all operations.

Algorithm 1 Employed bee phase of the SB-ABC algorithm

```
for $i = 1$ to $SP/2$ do
    Create candidate solution $v_i$ for $x_i$ by Equation (8);
    if ($cycle$ mod $RPPI = 0$) then
        Create a random permutation $rp$ of $\{1, 2, \ldots, D\}$;
        for $j = 1$ to $D$ do
            $v_{i,j} = v_{i,rp_j}$;
        end for
    end if
    Evaluate the solution $v_i$;
    if $f(v_i) < f(x_i)$ then
        $x_i = v_i$;
        $trial_i = 0$;
    else
        $trial_i = trial_i + 1$;
    end if
end for
```

Algorithm 2 Onlooker bee phase of the SB-ABC algorithm

```
for $i = 1$ to $SP/2$ do
    Calculate the probability $p_i$ by Equation (9);
end for
t = 1;
i = 1;
while $(t \leq SP/2)$ do
    if $(rand(0,1) < p_i)$ then
        $t = t + 1$;
        Create candidate solution $v_i$ for $x_i$ by by Equation (10);
        if ($cycle$ mod $RPPI = 0$) then
            Create a random permutation $rp$ of $\{1, 2, \ldots, D\}$;
            for $j = 1$ to $D$ do
                $v_{i,j} = v_{i,rp_j}$;
            end for
        end if
        Evaluate the solution $v_i$;
        if $f(v_i) < f(x_i)$ then
            $x_i = v_i$;
            $trial_i = 0$;
        else
            $trial_i = trial_i + 1$;
        end if
    end if
    $i = i + 1$;
    if ($i = SP/2$) then
        $i = 1$;
    end if
end while
```

Algorithm 3 Pseudo-code of the SB-ABC algorithm

```
Initial control parameters of the SB-ABC;
Generate initial population of solutions $x_i$, $i = 1, 2, \ldots, SP/2$ randomly in the search space;
Calculate objective function value of each solution $x_i$, $i = 1, 2, \ldots, SP/2$;
Set $trial_i = 0$, $i = 1, 2, \ldots, SP/2$;
$cycle = 0$;
repeat
    Execute Algorithm 1;
    Execute Algorithm 2;
    One solution with the highest $trial$ value that is greater than the abandonment threshold, if such solution occurs, is exchanged with a randomly generated solution;
    Save the current best solution;
    $cycle = cycle + 1$;
until ($cycle = MCN$)
```

5. Experimental Study

The performance of the SB-ABC algorithm is evaluated through seven integer programming problems and ten minimax problems widely used in the literature. The proposed algorithm is implemented in Java, and it was run on a PC with an Intel(R) Core(TM) i5-4460 3.2 GHz processor. In order to show the efficiency of the SB-ABC algorithm, it is compared with several algorithms that were previously applied to solve these problems. In the next subsections, brief descriptions of the used benchmark problems and results of a comparison between the SB-ABC and other state-of-the-art approaches are presented.

5.1. Benchmark Problems

In this section, the integer programming and minimax optimization test problems are described. To test the performance of the SB-ABC algorithm on integer programming problems, seven problems widely used in the literature are employed. The mathematical models of these problems can be found in [11,28,31]. These problems are presented below:
Test problem FI_1 is defined in [28]:

$$FI_1(x) = |x_1| + |x_2| + \ldots + |x_D|$$

where D is the dimension of the problem or number of optimization parameters. The global minimum is $FI_1(x^*) = 0$.
Test problem FI_2 is defined in [28]:

$$FI_2(x) = x^T x = \begin{bmatrix} x_1 & x_2 & \ldots & x_D \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_D \end{bmatrix}$$

where D is the dimension of the problem. The global minimum is $FI_2(x^*) = 0$.
Test problem FI_3 is defined in [28]:

$$FI_3(x) = -\begin{bmatrix} 15 & 27 & 36 & 18 & 12 \end{bmatrix} x + x^T \begin{bmatrix} 35 & -20 & -10 & 32 & -10 \\ -20 & 40 & -6 & -31 & 32 \\ -10 & -6 & 11 & -6 & -10 \\ 32 & -31 & -6 & 38 & -20 \\ -10 & 32 & -10 & -20 & 31 \end{bmatrix} x$$

The global minimum is $FI_3(x^*) = -737$.
Test problem FI_4 is defined in [28]:

$$FI_4(x) = (9x_1^2 + 2x_2^2 - 11)^2 + (3x_1 + 4x_2^2 - 7)^2$$

The global minimum is $FI_4(x^*) = 0$.
Test problem FI_5 is defined in [28]:

$$FI_5(x) = (x_1 + 10x_2)^2 + 5(x_3 - x_4)^2 + (x_2 - 2x_3)^4 + 10(x_1 - x_4)^4$$

The global minimum is $FI_5(x^*) = 0$.
Test problem FI_6 is defined in [28]:

$$FI_6(x) = 2x_1^2 + 3x_2^2 + 4x_1x_2 - 6x_1 - 3x_2$$

The global minimum is $FI_6(x^*) = -6$.
Test problem FI_7 is defined in [28]:

$$FI_7(x) = -3803.84 - 138.08x_1 - 232.92x_2 + 123.08x_1^2 + 203.64x_2^2 + 182.25x_1x_2$$

The global minimum is $FI_7(x^*) = -3833.12$.

To investigate the efficiency of the SB-ABC algorithm on minimax problems, ten benchmark functions are considered [6,28,31]. These benchmarks are presented as follows:
Test problem FM_1 is defined in [31,57]:

$$FM_1(x) = max \;\; f_i(x), \;\; i = 1,2,3$$

$$f_1(x) = x_1^2 + x_2^4$$
$$f_2(x) = (2 - x_1)^2 + (2 - x_2)^2$$
$$f_3(x) = 2exp(-x_1 + x_2)$$

The desired error goal for this problem is $FM_1(x^*) = 1.9522245$.
Test problem FM_2 is defined in [31]:

$$FM_2(x) = max \;\; f_i(x), \;\; i = 1,2,3$$

$$f_1(x) = x_1^4 + x_2^2$$
$$f_2(x) = (2 - x_1)^2 + (2 - x_2)^2$$
$$f_3(x) = 2exp(-x_1 + x_2)$$

The desired error goal for this problem is $FM_2(x^*) = 2$.
Test problem FM_3 is defined in [6,57]:

$$FM_3(x) = x_1^2 + x_2^2 + 2x_3^2 + x_4^2 - 5x_1 - 5x_2 - 21x_3 + 7x_4$$

$$g_2(x) = -x_1^2 - x_2^2 - x_3^2 - x_4^2 - x_1 + x_2 - x_3 + x_4 + 8$$
$$g_3(x) = -x_1^2 - 2x_2^2 - x_3^2 - 2x_4^2 + x_1 + x_4 + 10$$
$$g_4(x) = -x_1^2 - x_2^2 - x_3^2 - 2x_1 + x_2 + x_4 + 5$$

The desired error goal for this problem is $FM_3(x^*) = -40.1$.
Test problem FM_4 is defined in [31]:

$$FM_4(x) = max \;\; f_i(x), \;\; i = 1,2$$

$$f_1(x) = |x_1 + 2x_2 - 7|$$
$$f_2(x) = |2x_1 + x_2 - 5|$$

The desired error goal for this problem is $FM_4(x^*) = 10^{-4}$.
Test problem FM_5 is defined in [31]:

$$FM_5(x) = max \;\; f_i(x)$$

$$f_i(x) = |x_i|, \;\; i = 1, \ldots, 10$$

The desired error goal for this problem is $FM_5(x^*) = 10^{-4}$.

Five other test problems were selected from [57]. The name of the minimax benchmark problems, the dimension of the problem, the number of $f_i(x)$ functions and desired error goal are reported in Table 1.

Table 1. Properties of the minimax test problems FM_1–FM_{10}.

Function	Dimension (D)	$\#f_i(x)$	Desired Error Goal
FM_1 (CB2)	2	3	1.9522245
FM_2	2	3	2
FM_3 (Rosen-Suzuki)	4	4	–40.1
FM_4	2	2	10^{-4}
FM_5	10	10	10^{-4}
FM_6 (SPIRAL)	2	2	10^{-4}
FM_7 (Polak 6)	4	4	-40.1
FM_8 (Wong 1)	7	5	680.9
FM_9 (OET6)	4	21	0.1
FM_{10} (Filter)	9	41	$0.61852848 \cdot 10^{-2}$

5.2. The General Performance of the SB-ABC for Integer Programming Problems

Because the SB-ABC is an improved variant of the ABC, in this section, a comparison between the SB-ABC and ABC algorithm adjusted to solve integer programming problems through seven integer programming problems is presented. The common traditional technique, the branch-and-bound (BB) method, is also included in the comparison with the proposed approach.

The preliminary testing of the SB-ABC was done with the aim of obtaining suitable combinations of parameter values. The SP parameter was set to 20. This value was detected to be a proper selection for all executed tests. The increasing value of this control parameter will increase the computational cost without any enhancement in the reached results. Our tests verified the previous reasoning that a value 0.8 for the MR parameter is a good choice for solving these optimization problems [11]. Additionally, it was experimentally determined that a value of 50 for the parameter $limit$ and a value of 3 for the parameter $RPPI$ are suitable for the SB-ABC algorithm. It was observed that significantly lower or higher values of the $limit$ parameter can deteriorate the obtained results. Higher values of the $RPPI$ parameter would lead to the less frequent use of the shuffle mutation operator and consequently weaker performance of the SB-ABC algorithm. In the SB-ABC, during the initialization step, $SP/2$ solutions are evaluated, and there are $SP/2$ employed bees, $SP/2$ onlookers and a maximum of one scout bee per iteration. Therefore, the maximum number of function evaluations for the SB-ABC is $SP/2 + (SP+1) \cdot MCN$. The maximum number of function evaluations executed by the SB-ABC for all benchmarks was set to 20,000 and the SB-ABC was terminated when the global minimum was reached.

The results of the BB method and ABC algorithm were taken from their original papers [2,11]. For these comparisons, in the BB method and ABC algorithm, the maximum number of function evaluations was set to 25,000. When an accuracy of 10^{-6} was achieved, these methods were terminated. The BB method, ABC and SB-ABC algorithms are conducted for 30 independent runs for each benchmark problem.

The following metrics are used to estimate the performances of the BB, ABC and SB-ABC. The convergence speed of each algorithm is compared by recording the mean number of function evaluations (mean) required to reach the acceptable value. If the mean value is smaller, the convergence speed is faster. Since SI algorithms are stochastic, the obtained mean results are not the same in each run. To examine the stability of each method, standard deviation (SD) values are measured. The performance of an algorithm is more stable if the standard deviation value is lower. The success rate (SR) is used as a metric for robustness or reliability of methods. This rate is defined as the ratio of successful runs in the total number of executed runs. A run is considered successful if an algorithm obtains a solution for which the value of the objective function is less than the corresponding acceptable value. If the value of SR is greater, the reliability of the algorithm is better. In Table 2, the mean, corresponding standard deviation (SD) values and SR values of the BB method, ABC and SB-ABC for the benchmark problems FI_1 with 5, 10, 15, 20, 25 and 30 variables and test problems FI_2–FI_7 over 30 runs are given. The best mean results are indicated in bold.

As shown in Table 2, with respect to the SR results reached by these methods, the SB-ABC performs the most reliably since, for each test case, the obtained SR result of the SB-ABC is 100%. The BB method performance is less robust than the SB-ABC for problem FI_1 with 30 variables, since it achieved only 14 successful runs out of 30, while in the remaining test cases, both approaches obtained the same SR results. The SB-ABC and ABC algorithms obtained the same SR results on all test problems, with the exception of problem FI_3, where the ABC performance was less robust. With respect to the mean results, from Table 2, it can be observed that the SB-ABC performs better than its rivals in the majority of cases. To be exact, the SB-ABC is better than the BB method and ABC in 12 and 11 test cases, respectively. On the other hand, the BB method has better mean results for problem FI_2, while the ABC outperformed the SB-ABC for test functions FI_5 and FI_7. With respect to the standard deviation results, from Table 2, it can be seen that the SB-ABC performance is more stable than the BB and ABC methods in most cases.

Table 2. Comparison results of the BB method, ABC and SB-ABC for the FI_1–FI_7 integer programming problems.

		BB			ABC			SB-ABC		
Prob	**D**	**Mean**	**SD**	**SR**	**Mean**	**SD**	**SR**	**Mean**	**SD**	**SR**
FI_1	5	1167.83	659.8	30/30	376	64.6	30/30	**216.0**	62.05	30/30
	10	5495.8	1676.3	30/30	727.3	64.4	30/30	**381.33**	51.62	30/30
	15	10,177.1	2393.4	30/30	974	60.5	30/30	**508.67**	65.05	30/30
	20	16,291.3	3797.9	30/30	1275.3	97.7	30/30	**624.0**	86.01	30/30
	25	23,689.7	2574.2	20/30	1554.7	108.6	30/30	**725.33**	93.94	30/30
	30	25,908.6	755.5	14/30	1906	129.9	30/30	**796.67**	77.13	30/30
FI_2	5	**139.7**	102.6	30/30	449.3	56.7	30/30	239.33	52.53	30/30
FI_3	2	4185.5	32.8	30/30	13850	6711.3	24/30	**3916.67**	1773.67	30/30
FI_4	4	316.9	125.4	30/30	240.7	79.4	30/30	**90.0**	62.34	30/30
FI_5	2	2754	1030.1	30/30	**193.3**	53.5	30/30	421.33	163.62	30/30
FI_6	2	211	15	30/30	258.7	113.6	30/30	**140.67**	57.38	30/30
FI_7	2	358.6	14.7	30/30	**106.7**	44.8	30/30	177.33	130.20	30/30

5.3. Comparison against Other State-of-the-Art Algorithms for Integer Programming Problems

To further demonstrate the efficiency of the SB-ABC, it is benchmarked against 12 other metaheuristic algorithms that were previously successfully used to solve integer programming problems. These algorithms are the basic PSO and its four variants RWMPSOg, RWMPSOl, PSOg, PSOl [28], standard cuckoo search (CS), firefly algorithm (FA), gravitational search algorithm (GSA), whale optimization algorithm (WOA), hybrid cuckoo search algorithm with Nelder Mead method (HCSNM) [29], hybrid bat algorithm (HBDS) [30] and the recently proposed hybrid harmony search algorithm with multidirectional search method (MDHSA) [31].

The results obtained by the RWMPSOg, RWMPSOl, PSOg, PSOl are taken from [28], the results reached by the HCSNM are taken from [29], the results achieved by HBDS are taken from [30], while the results of the MDHSA and basic PSO, CS, FA, GSA and WOA are taken from [31]. In Table 3, the mean, corresponding standard deviation values and SR values of the RWMPSOg, RWMPSOl, PSOg, PSOl, HCSNM, MDHSA and SB-ABC for the benchmark problems FI_1 with 5 variables and test problems FI_2–FI_7 over 50 runs are given. Table 4 presents the mean and standard deviation values obtained by the PSO, FA, CS, GSA, WOA, HBDS and SB-ABC for problem FI_1 with 5 variables and test problems FI_2–FI_7 over 50 runs. The best mean results are in bold. The metaheuristics used for comparison with the SB-ABC also performed the maximum number of function evaluations of 20,000. Since the results of these 12 algorithms are achieved over 50 runs, the statistical results of the SB-ABC over 50 runs are presented in Tables 3 and 4.

The results from Tables 3 and 4 show that the proposed algorithm obtained better mean results on the majority of benchmark problems in comparison with its competitors. Precisely, the SB-ABC is better than RWMPSOg, RWMPSOl, PSOg, PSOl, MDHSA, HCSNM, PSO, FA, CS, GSA, WOA and HBDS in six, six, seven, seven, five, four, seven, seven, seven, seven, seven, and six test problems, respectively. In contrast, the SB-ABC is outperformed by the RWMPSOg, RWMPSOl, PSOg, PSOl, MDHSA, HCSNM, PSO, FA, CS, GSA, WOA and HBDS in one, one, zero, zero, two, three, zero, zero, zero, zero, zero and one test problems, respectively. From the standard deviation values presented in Tables 3 and 4, it can be observed that the proposed SB-ABC has lower standard deviation values on the majority of benchmark problems in comparison with RWMPSOg, RWMPSOl, PSOg PSOl, PSO, CS, GSA and WOA. On the other hand, the HCSNM, MDHSA, FA and HBDS have lower standard deviations compared to the SB-ABC for most of the cases. In addition, the SR results demonstrate that the SB-ABC achieved a 100% success rate on all benchmark problems.

Table 3. Comparison results of the RWMPSOg, RWMPSOl, PSOg, PSOl, MDHSA, HCSNM and SB-ABC for the FI_1–FI_7 integer programming problems.

Prob	Metric	RWMPSOg	RWMPSOl	PSOg	PSOl	HCSNM	MDHSA	SB-ABC
FI_1	Mean	27,176.3	30,923.9	29,435.3	31,252	638.3	**176.01**	218.0
	SD	8657	2405	42,039	1818	4.34	4.265	60.95
	SR	50/50	50/50	34/50	50/50	50/50	50/50	50/50
FI_2	Mean	578.5	773.9	606.4	830.2	232.64	**152.48**	240.8
	SD	136.5	285.5	119	206	4.28	2.565	64.74
	SR	50/50	50/50	50/50	50/50	50/50	50/50	50/50
FI_3	Mean	6490.6	9292.6	12,681	11,320	1668.1	**531.4**	4034.8
	SD	6913	2444	35,067	3803	43.2	30.74	2852.2
	SR	50/50	50/50	50/50	50/50	50/50	50/50	50/50
FI_4	Mean	215	218.7	369.6	390	174.04	182.74	**107.6**
	SD	97.9	115.3	113.2	134.6	6.21	41.60	63.48
	SR	50/50	50/50	50/50	50/50	50/50	50/50	50/50
FI_5	Mean	1521.8	2102.9	1499	2472.4	884.48	449.12	**425.2**
	SD	360.7	689.5	513.1	637.5	56.24	2.413	174.11
	SR	50/50	50/50	43/50	50/50	50/50	50/50	50/50
FI_6	Mean	**110.9**	112	204.8	256	155.89	188.3	144.0
	SD	48.6	48.7	62	107.5	5.16	41.63	68.35
	SR	50/50	50/50	50/50	50/50	50/50	50/50	50/50
FI_7	Mean	242.7	248.9	421.2	466	210.3	192.6	**185.2**
	SD	132.2	134.4	130.4	165	6.39	37.33	123.21
	SR	50/50	50/50	50/50	50/50	50/50	50/50	50/50

Table 4. Comparison results of the PSO, FA, CS, GSA, WOA, HBDS and SB-ABC for the FI_1–FI_7 integer programming problems.

Prob	Metric	PSO	FA	CS	GSA	WOA	HBDS	SB-ABC
FI_1	Mean	20,000	1617.13	11,880.15	2020	18,436.36	656.56	**218.0**
	SD	0.00	114.77	623.41	112.45	568.47	88.65	60.95
FI_2	Mean	17,540.17	834.15	7176.23	1060	10,134.53	344.22	**240.8**
	SD	1054.56	146.85	637.75	78.69	483.25	43.32	64.74
FI_3	Mean	20,000	1225.17	6400.25	5160	2946.63	**1137.56**	4034.8
	SD	0.00	128.39	819.94	214.25	24.25	85.61	2852.2
FI_4	Mean	16,240.36	476.16	4920.35	1680	9255.42	260.8	**107.6**
	SD	1484.96	31.29	247.19	89.41	857.36	10.39	63.48
FI_5	Mean	13,120.45	1315.53	7540.38	7250	6272.47	1177.12	**425.2**
	SD	1711.83	113.01	440.82	425.36	925.35	111.6	174.11
FI_6	Mean	1340.14	345.71	4875.35	1520.23	18,420.18	149.08	**144.0**
	SD	265.21	35.52	865.11	231.56	869.25	8.21	68.35
FI_7	Mean	1220.46	675.48	3660.45	1100.24	9248.12	222.91	**185.2**
	SD	177.19	36.36	383.23	85.23	962.35	11.19	123.21

5.4. The General Performance of the SB-ABC for Minimax Problems

In this section, the performance of the SB-ABC for solving minimax problems is investigated. The performance of the SB-ABC is compared to the performance of the SQP method and standard ABC algorithm. The fair comparison is ensured since the SQP, ABC and SB-ABC algorithms employed the maximum number of function evaluations of 20,000. The run is counted as successful when the desired error goal is reached within the maximum number of function evaluations.

The specific parameter settings of the SB-ABC are kept the same, as mentioned in Section 5.2. Since the standard ABC is not tested to solve minimax problems in any of the studies, we have tested its performance in solving problems FM_1–FM_{10}. The ABC employed the following parameter settings, SP is 20, MR is 0.8 and $limit$ is $5 \cdot SP \cdot D$. These values of control parameters were used in the standard ABC, adjusted to solve integer programming problems [11]. The results of the SQP method were taken from the respective paper [2].

In Table 5, the mean, corresponding standard deviation (SD) values and SR values of the SQP method, ABC and SB-ABC for the benchmark problems FM_1–FM_{10} over 30 runs are presented. The best mean results are in bold. The mark (-) for FM_{10} in the SQP method means that the results are not reported in its original paper. From Table 5, it can be noticed that the SB-ABC converges faster to the global minimum in comparison with the SQP method and ABC for the majority of test problems.

Table 5. Comparison results of the ABC and SB-ABC for the FM_1–FM_{10} minimax problems.

	SQP			**ABC**			**SB-ABC**		
Problem	**Mean**	**SD**	**SR**	**Mean**	**SD**	**SR**	**Mean**	**SD**	**SR**
FM_1	4044.5	8116.6	24/30	7522.0	5486.06	29/30	**964.67**	319.07	30/30
FM_2	8035.7	9939.9	18/30	1997.2	741.59	30/30	**586.67**	110.55	30/30
FM_3	**135.5**	21.1	30/30	600.8	130.71	30/30	314.67	88.38	30/30
FM_4	**140.6**	38.5	30/30	1854.0	400.80	30/30	736.67	114.20	30/30
FM_5	**611.6**	200.6	30/30	19,022.8	2028.13	24/30	1614.67	176.86	30/30
FM_6	15,684.0	7302.0	10/30	2215.2	2197.82	30/30	**348.66**	214.45	30/30
FM_7	20,000	0.0	0/30	2986.8	2327.84	30/30	**422.0**	148.80	30/30
FM_8	20,000	0.0	0/30	18,442.8	3707.93	18/30	**7288.0**	4827.87	29/30
FM_9	4886.5	8488.4	22/30	3244.4	2562.20	30/30	**852.0**	740.80	30/30
FM_{10}	-	-	-	20,000.0	0.0	0/30	**6584.0**	5648.24	27/30

From the obtained mean values, it can be observed that the SB-ABC has better performance than the SQP and ABC methods in 6 and 10 test problems, respectively. On the other hand, the SB-ABC is outperformed by the SQP and ABC on three and zero benchmarks, respectively. Furthermore, the SR results indicate that the SB-ABC performance is more robust in comparison with the SQP on six test problems (FM_1, FM_2, FM_6, FM_7, FM_8 and FM_9), while both methods reached the same SR results for the rest of the benchmarks. With respect to the standard deviation results, from Table 5, it can be noticed that the SB-ABC performance is more stable than the SQP and ABC methods in most cases. Furthermore, from Table 5, it can be seen that the SB-ABC performs more reliably than the ABC on four benchmark problems (FM_1, FM_5, FM_8 and FM_{10}), while both algorithms achieved the same SR results for the remaining problems.

5.5. *Comparison against Other State-of-the-Art Algorithms for Minimax Problems*

In order to further examine the efficiency of the SB-ABC for minimax problems, its performance is compared to the performance of 10 other algorithms that were previously successfully used to solve these problems. These methods are the heuristic pattern search algorithm HPS2, the basic PSO and its two variants RWMPSOg and UPSOm [58], HC-SNM [29], MDHSA [31], CS, FA, GSA and WOA.

The results obtained by the RWMPSOg are taken from [28], the results reached by the HPS2 are taken from [59], the results achieved by UPSOm are taken from [58], the results obtained by the HCSNM are taken from [29], while the results of the MDHSA, PSO, FA, CS, GSA and WOA are taken from [31]. In Table 6, the mean, standard deviation values and SR values of the HPS2, UPSOm, RWMPSOg, HCSNM, MDHSA and SB-ABC for the benchmark problems FM_1–FM_{10} over 50 runs are given. Table 7 presents the mean and standard deviation values obtained by the PSO, FA, CS, GSA, WOA and SB-ABC for benchmark problems FM_1–FM_{10} over 50 runs. The best mean results are in bold. The metaheuristics used for comparison with the SB-ABC also performed the maximum number of function evaluations of 20,000. The SB-ABC was configured with the specific parameter values, as described in Section 5.2. Since the results of these 10 algorithms are achieved over 50 runs, the statistical results of the SB-ABC over 50 runs are presented in Tables 6 and 7. In Table 6, the mark (-) indicates that the results are not presented in the corresponding paper.

The results from Tables 6 and 7 show that the proposed algorithm obtained better mean results on the majority of benchmark problems in comparison with its competitors. Concretely, the SB-ABC outperformed the HPS2, UPSOm, RWMPSOg, HCSNM, MDHSA, PSO, FA, CS, GSA and WOA in 5, 8, 6, 6, 7, 9, 8, 10, 9 and 10 test problems, respectively. In contrast, the SB-ABC is outperformed by the HPS2, UPSOm, RWMPSOg, HCSNM, MDHSA, PSO, FA, CS, GSA and WOA in three, one, zero, three, three, one, two, zero, one and zero benchmark problems, respectively. From standard deviation values presented in Tables 6 and 7, it can be seen that the SB-ABC has lower standard deviation values for most of the cases in comparison with the UPSOm, RWMPSOg, CS and WOA. On the other hand, the HPS2, HCSNM, MDHSA, PSO, FA and GSA have lower standard deviations compared to the SB-ABC in most cases. With respect to the SR results reached by these methods, the SB-ABC performs the same or more reliably in comparison with the HPS2, UPSOm, RWMPSOg, HCSNM and MDHSA in these minimax problems.

Table 6. Comparison results of the HPS2, UPSOm, RWMPSOg, HCSNM, MDHSA, HCSNM and SB-ABC for the FM_1–FM_{10} minimax problems.

Problem	Metric	HPS2	UPSOm	RWMPSOg	HCSNM	MDHSA	SB-ABC
FM_1	Mean	1848.7	1993.8	2415.3	**705.62**	1564.86	986.4
	SD	2619.4	853.7	1244.2	14.721	56.89	470.92
	SR	99%	100%	100%	100%	100%	100%
FM_2	Mean	635.8	1775.6	-	624.24	**555.64**	599.6
	SD	114.3	241.9	-	20.83	55.33	124.55
	SR	94%	100%	-	100%	85%	100%
FM_3	Mean	**141.2**	1670.4	3991.3	906.28	1839.6	329.6
	SD	28.4	530.6	2545.2	98.24	83.65	119.01
	SR	37%	100%	100%	100%	100%	100%
FM_4	Mean	772.0	1701.6	2947.8	670.22	**633.9**	743.6
	SD	60.8	184.9	257.0	11.07	63.04	125.77
	SR	100%	100%	100%	100%	81%	100%
FM_5	Mean	1809.1	18294.5	18,520.1	4442.76	8382.58	**1722.0**
	SD	2750.3	2389.4	776.9	87.159	198.26	332.82
	SR	94%	100%	100%	95%	75%	100%
FM_6	Mean	4114.7	3435.5	1308.8	1103.86	2064.44	**392.4**
	SD	1150.2	1487.6	505.5	125.36	73.10	359.93
	SR	100%	100%	100%	95%	95%	100%
FM_7	Mean	-	6618.50	-	2629.336	4706.32	**476.0**
	SD	-	2597.54	-	84.80	174.03	113.91
	SR	-	100%	-	75%	80%	100%
FM_8	Mean	**283.0**	2128.5	-	2724.78	4175	8736.0
	SD	123.9	597.4	-	227.24	96.90	5178.29
	SR	64%	100%	-	95%	75%	98%
FM_9	Mean	**324.1**	3332.5	4404.0	977.56	2253.5	934.8
	SD	173.1	1775.4	3308.9	176.82	130.58	765.36
	SR	100%	100%	100%	100%	95%	100%
FM_{10}	Mean	-	-	-	-	9432.24	**6730.4**
	SD	-	-	-	-	156.39	5065.53
	SR	-	-	-	-	-	94%

Table 7. Comparison results of the PSO, FA, CS, GSA, WOA and SB-ABC for the FM_1–FM_{10} minimax problems.

Problem	Metric	PSO	FA	CS	GSA	WOA	SB-ABC
FM_1	Mean	3535.46	1125.61	5375.52	1620.4	10,126.36	**986.4**
	SD	491.66	189.56	613.35	126.25	1583.65	470.92
FM_2	Mean	20,000	785.17	6150.34	1980.5	10,263.45	**599.6**
	SD	0.00	31.94	519.65	253.69	758.58	124.55
FM_3	Mean	2920.15	695.54	3745.19	1800.7	1523.36	**329.6**
	SD	269.48	50.03	878.09	45.58	121.89	119.01
FM_4	Mean	5680.17	782.52	5845.23	1680.4	10,253.58	**743.6**
	SD	937.44	86.77	804.36	58.78	980.45	125.77
FM_5	Mean	20,000	13,692.13	7895.14	11,800.6	11,458.36	**1722.0**
	SD	0.00	900.12	1077.07	25.36	1785.36	332.82
FM_6	Mean	5643.65	2685.25	11,915.24	1860.6	1235.69	**392.4**
	SD	-	610.07	341.45	253.69	48.69	359.93
FM_7	Mean	20,000	7659.45	20,000	7200.4	19,465.35	**476.0**
	SD	0.00	583.21	1788.18	1986.25	2568.39	113.91
FM_8	Mean	**6220.25**	8147.45	14,754.14	8500.6	9186.25	8736.0
	SD	727.44	1026.22	1391.58	1453.67	485.79	5178.29
FM_9	Mean	6680.19	**748.17**	6765.24	1440.7	3648.69	934.8
	SD	509.34	98.59	843.49	245.36	896.47	765.36
FM_{10}	Mean	18,125.360	11,124.55	10,436.22	11,254.6	13,242.24	**6730.4**
	SD	2356.58	1254.58	23.15	2145.25	2536.36	5065.53

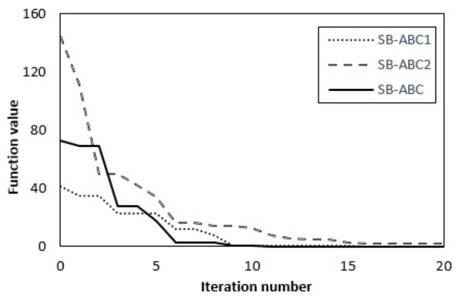
6. Discussion

The impact of the introduced modifications on the SB-ABC will be examined in this section. Seven integer programming problems and ten minimax problems were solved by two diverse variants of the SB-ABC. The obtained results of each variant are compared with respect to the same of the developed SB-ABC approach. In the following text, these variants are presented:

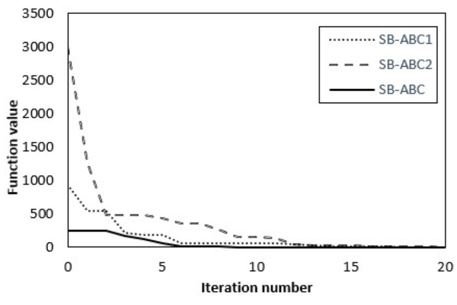
1. *Variant 1*: To examine the effectiveness of employing the modified ABC operator in the onlooker bee phase given by Equation (10), an SB-ABC version that uses the standard ABC search equation is tested. The label SB-ABC1 is used for this variant.
2. *Variant 2*: To investigate the effectiveness of employing the shuffle mutation operator in employed and onlooker bee phases, an SB-ABC version that does not include this operator is tested. The label SB-ABC2 is used for this variant.

Each SB-ABC version is run 50 times for each test problem. The maximum number of function evaluations was 20,000 for each method. The tested methods were configured with specific parameter values, as described in Section 5.2. The mean, standard deviation values and SR values obtained by the SB-ABC1, SB-ABC2 and SB-ABC for seven integer programming problems and ten minimax problems are presented in Tables 8 and 9. A result in boldface denotes the best mean result. The convergence graphs achieved by the SB-ABC1, SB-ABC2 and SB-ABC on the four picked integer programming functions and the four selected minimax optimization problems are given in Figures 1 and 2, respectively.

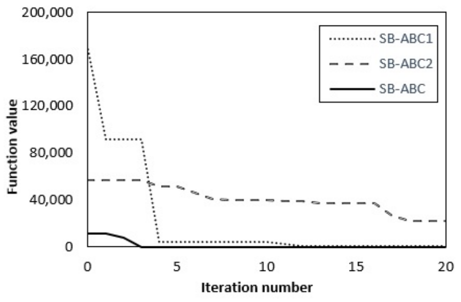
From the mean results presented in Table 8, it can be seen that the SB-ABC results outperform SB-ABC1 and SB-ABC2 versions on all integer programming benchmark problems. With respect to the SR values, it can be noticed that each algorithm achieved a 100% success rate on all FI_1–FI_7 integer programming problems. From the standard deviation values presented in Table 8, it can be observed that the SB-ABC has lower standard deviation values for most of the cases in comparison with the SB-ABC1 and SB-ABC2 versions.



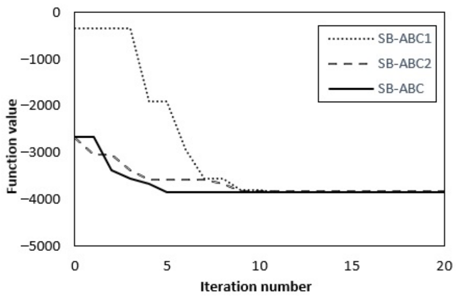
(**a**) FI_1



(**b**) FI_2

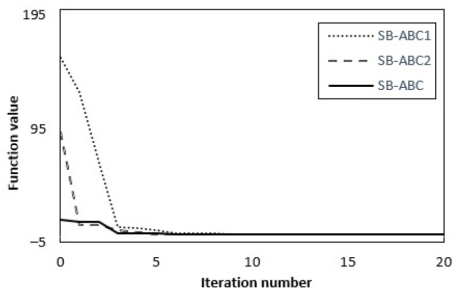


(**c**) FI_5

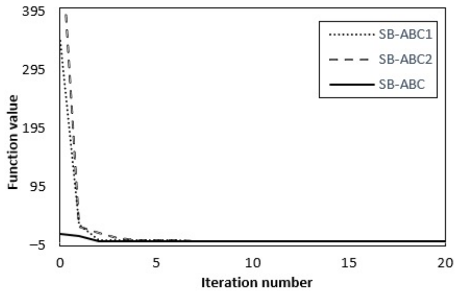


(**d**) FI_7

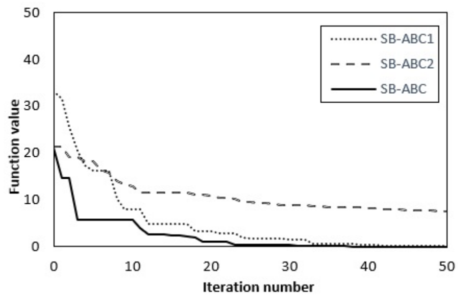
Figure 1. Convergence graphs of SB-ABC1, SB-ABC2 and SB-ABC for the selected integer programming problems.



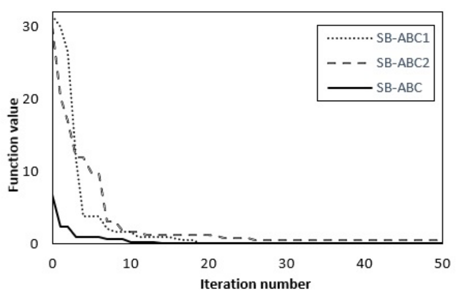
(**a**) FM_1



(**b**) FM_2



(**c**) FM_5



(**d**) FM_9

Figure 2. Convergence graphs of SB-ABC1, SB-ABC2 and SB-ABC for the selected minimax problems.

Table 8. Comparison results of the SB-ABC1, SB-ABC2 and SB-ABC for the FI_1–FI_7 integer programming problems.

		SB-ABC1			SB-ABC2			SB-ABC		
Prob	**D**	**Mean**	**SD**	**SR**	**Mean**	**SD**	**SR**	**Mean**	**SD**	**SR**
FI_1	5	284.8	68.30	100%	427.6	163.60	100%	**218.0**	60.95	100%
	10	506.4	88.67	100%	1690.8	653.42	100%	**387.6**	49.13	100%
	15	653.2	71.95	100%	2896.0	940.17	100%	**511.2**	53.54	100%
	20	815.33	124.75	100%	4967.33	1347.21	100%	**644.4**	70.68	100%
	25	893.2	137.78	100%	6350.0	1369.01	100%	**732.8**	82.43	100%
	30	1014.8	140.96	100%	8222.0	1491.48	100%	**816.8**	76.03	100%
FI_2	5	326.8	84.25	100%	511.6	241.86	100%	**240.8**	64.74	100%
FI_3	2	7524.4	5462.02	100%	4165.6	2647.41	100%	**4034.8**	2852.2	100%
FI_4	4	131.2	77.68	100%	163.6	107.68	100%	**107.6**	63.48	100%
FI_5	2	472.8	190.06	100%	1188.8	522.33	100%	**425.2**	174.11	100%
FI_6	2	157.2	64.37	100%	147.6	70.64	100%	**144.0**	68.35	100%
FI_7	2	224.8	192.56	100%	190.8	53.99	100%	**185.2**	123.21	100%

Table 9. Comparison results of the SB-ABC1, SB-ABC2 and SB-ABC for the FM_1–FM_{10} minimax problems.

	SB-ABC1			SB-ABC2			SB-ABC		
Prob	**Mean**	**SD**	**SR**	**Mean**	**SD**	**SR**	**Mean**	**SD**	**SR**
FM_1	1150.8	381.05	100%	996.8	322.20	100%	**986.4**	470.92	100%
FM_2	896.8	217.01	100%	770.8	169.52	100%	**599.6**	124.55	100%
FM_3	446.4	193.76	100%	365.6	162.83	100%	**329.6**	119.01	100%
FM_4	1098.8	217.16	100%	**706.8**	129.77	100%	743.6	125.77	100%
FM_5	2515.6	429.91	100%	20,000.0	0.0	0%	**1722.0**	332.82	100%
FM_6	674.8	1027.67	100%	4123.6	6755.41	88%	**392.4**	359.93	100%
FM_7	595.2	233.15	100%	2196.0	3016.41	100%	**476.0**	113.91	100%
FM_8	10,949.2	4522.99	94%	9770.4	6020.82	86%	**8736.0**	5178.29	98%
FM_9	956.8	689.11	100%	3199.6	4290.61	98%	**934.8**	765.36	100%
FM_{10}	13,504.8	6895.94	54%	**6451.2**	6825.51	86%	6730.4	5065.53	94%

Compared with the SB-ABC1, it can be seen from Table 9 that the proposed algorithm reached better mean results and the same or better SR results for all FM_1–FM_{10} minimax problems. When comparing the SB-ABC with respect to the SB-ABC2, it can be noticed from Table 9 that the SB-ABC obtained better mean values for eight minimax problems and slightly worse mean results for the remaining two benchmarks (FM_4 and FM_{10}). From the standard deviation values presented in Table 9, it can be noticed that the SB-ABC has lower standard deviation values for most of the cases in comparison with the SB-ABC1 and SB-ABC2 variants. According to the SR results presented in Table 9, the SB-ABC outperformed the SB-ABC2 in five test problems (FM_5, FM_6, FM_8, FM_9 and FM_{10}), while the SB-ABC and SB-ABC2 achieved the same results in the remaining benchmarks. As shown in Figures 1 and 2, SB-ABC converges faster to the optimum on the selected problems in comparison with its two variants.

These observations indicate that each introduced modification contributes to the satisfactory performance of the SB-ABC. Use of the shuffle mutation operator provides useful diversity and consequently helps the SB-ABC to locate the favorable areas within the search space. Adding the modified ABC operator enables an enhanced exploitation ability of the algorithm. Combining these modifications significantly improves the convergence speed and robustness of the SB-ABC algorithm.

7. Conclusions

In this paper, a novel shuffle-based artificial bee colony algorithm (SB-ABC) is proposed with the intention to solve integer programming and minimax problems. The proposed algorithm employs the shuffle mutation operator and modified ABC search strategy with an aim to improve the exploitation tendency of the algorithm to provide a valuable convergence speed. The proposed approach was applied to solve seven integer programming and ten minimax problems taken from the literature. The SB-ABC algorithm obtained highly competitive results in comparison with the standard ABC, BB method and 12 other metaheuristic algorithms in solving integer programming problems. Compared with the standard ABC, SQP method and 10 other state-of-the-art algorithms, the SB-ABC showed better performance for the majority of the minimax test problems.

The effects of introduced modifications related to the shuffle mutation operator and modified ABC search operator have been investigated. It is experimentally validated that the use of each introduced modification is of great importance in achieving a satisfactory performance of the SB-ABC with respect to the convergence speed and robustness. In the proposed SB-ABC method, the balance between the global exploration and local exploitation abilities is addressed by suitable configuration of control parameters. As in many other swarm intelligence techniques, the problem of discovering appropriate values for the control parameters also exists in the SB-ABC algorithm. Extending the SB-ABC method with a self-adaptive control mechanism to reach better exploration and exploitation balance during distinct search phases is a promising direction for future study. Developing a hybrid algorithm that would incorporate different operators of some other well-established metaheuristic methods in the SB-ABC algorithm for solving large scale integer programming and minimax problems will also be examined in future work. Another possible way for creating a hybrid approach is to employ certain metaheuristic methods to assume a role of a local optimizer, while the SB-ABC algorithm would perform a global search. In addition, the application of the proposed SB-ABC approach for solving some combinatorial optimization problems will be investigated.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The author declares no conflict of interest.

References

1. Coit, D.W.; Zio, E. The evolution of system reliability optimization. *Reliab. Eng. Syst. Saf.* **2019**, *192*, 106259. [CrossRef]
2. Laskari, E.C.; Parsopoulos, K.E.; Vrahatis, M.N. Particle swarm optimization for integer programming. In Proceedings of the 2002 Congress on Evolutionary Computation, CEC'02 (Cat. No.02TH8600), Honolulu, HI, USA, 12–17 May 2002; pp. 1582–1587.
3. Kim, T.-H.; Cho, M.; Shin, S. Constrained Mixed-Variable Design Optimization Based on Particle Swarm Optimizer with a Diversity Classifier for Cyclically Neighboring Subpopulations. *Mathematics* **2020**, *8*, 2016. [CrossRef]
4. Agarana, M.C.; Ajayi, O.O.; Akinwumi, I.I. Integer programming algorithm for public transport system in sub-saharan african cities. *Wit. Trans. Built. Environ.* **2019**, *182*, 339–350.
5. Haunert, J.-H.; Wolff, A. Beyond Maximum Independent Set: An Extended Integer Programming Formulation for Point Labeling. *ISPRS Int. J. Geo-Inf.* **2017**, *6*, 342. [CrossRef]
6. Laskari, E.C.; Parsopoulos, K.E.; Vrahatis, M.N. Particle swarm optimization for minimax problems. In Proceedings of the 2002 Congress on Evolutionary Computation, CEC'02 (Cat. No.02TH8600), Honolulu, HI, USA, 12–17 May 2002; pp. 1576–1581.
7. Khakifirooz, M.; Chien, C.; Fathi, M.; Pardalos, P.M. Minimax Optimization for Recipe Management in High-Mixed Semiconductor Lithography Process. *IEEE Trans. Industr. Inform.* **2020**, *16*, 4975–4985. [CrossRef]
8. Razaviyayn, M.; Huang, T.; Lu, S.; Nouiehed, M.; Sanjabi, M.; Hong, M. Nonconvex Min-Max Optimization: Applications, Challenges, and Recent Theoretical Advances. *IEEE Signal Process. Mag.* **2020**, *37*, 55–66. [CrossRef]
9. Zhou, Z.; Yang, Q. An Active Set Smoothing Method for Solving Unconstrained Minimax Problems. *Math. Probl. Eng.* **2020**, *2020*, 1–25. [CrossRef]
10. Ma, G.; Zhang, Y.; Liu, M. A generalized gradient projection method based on a new working set for minimax optimization problems with inequality constraints. *J. Inequal Appl.* **2017**, *51*, 1–14. [CrossRef] [PubMed]
11. Akay, B.; Karaboga, D. Solving Integer Programming Problems by Using Artificial Bee Colony Algorithm. In *AI*IA 2009: Emergent Perspectives in Artificial Intelligence*; Serra, R., Cucchiara, R., Eds.; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2009; Volume 5883, pp. 355–364.
12. Tarray, T.A.; Bhat, M. A nonlinear programming problem using branch and bound method. *Investig. Oper.* **2017**, *38*, 291–298.

13. Brajević, I.; Ignjatović, J. An upgraded firefly algorithm with feasibility-based rules for constrained engineering optimization problems. *J. Intell. Manuf.* **2019**, *30*, 2545–2574. [CrossRef]
14. Stojanović, I.; Brajević, I.; Stanimirović, P.S.; Kazakovtsev, L.A.; Zdravev, Z. Application of Heuristic and Metaheuristic Algorithms in Solving Constrained Weber Problem with Feasible Region Bounded by Arcs. *Math. Probl. Eng.* **2017**, *2017*, 1–13. [CrossRef]
15. Liu, Q.; Li, X.; Liu, H.; Guo, Z. Multi-objective metaheuristics for discrete optimization problems: A review of the state-of-the-art. *Appl. Soft Comput.* **2020**, *93*, 106382. [CrossRef]
16. Ng, K.K.H.; Lee, C.K.M.; Chan, F.T.S.; Lv, Y. Review on meta-heuristics approaches for airside operation research. *Appl. Soft Comput.* **2018**, *66*, 104–133. [CrossRef]
17. Iliopoulou, C.; Kepaptsoglou, K.; Vlahogianni, E. Metaheuristics for the transit route network design problem: A review and comparative analysis. *Public Transp.* **2019**, *11*, 487–521. [CrossRef]
18. Bala, A; Ismail, I.; Ibrahim, R.; Sait, S.M. Applications of Metaheuristics in Reservoir Computing Techniques: A Review. *IEEE Access* **2018**, *6*, 58012–58029. [CrossRef]
19. Kennedy, J.; Eberhart, R.C. Particle swarm optimization. In Proceedings of the ICNN'95—International Conference on Neural Networks, Perth, WA, Australia, 27 November–1 December 1995; pp. 1942–1948.
20. Karaboga, D.; Basturk, B. A powerful and efficient algorithm for numerical function optimization: Artificial bee colony (ABC) algorithm. *J. Glob. Optim.* **2007**, *39*, 459–471. [CrossRef]
21. Geem, Z.W.; Kim, J.H.; Loganathan, G.V. A new heuristic optimization algorithm: Harmony search. *Simulation* **2001**, *76*, 60–68. [CrossRef]
22. Yang, X.S. Firefly algorithms for multimodal optimization. In *Stochastic Algorithms: Foundations and Applications, SAGA 2009*; Watanabe, O., Zeugmann, T., Eds.; Lecture Notes in Computer Science; Springer: Berlin, Germany, 2009; Volume 5792, pp. 169–178.
23. Rashedi, E.; Nezamabadi-pour, H.; Saryazdi, S. GSA: A gravitational search algorithm. *Inf. Sci.* **2009**, *179*, 2232–2248. [CrossRef]
24. Yang, X.S.; Deb, S. Cuckoo Search via Lévy flights. In Proceedings of the World Congress on Nature & Biologically Inspired Computing (NaBIC), Coimbatore, India, 9–11 December 2009; pp. 210–214.
25. Mirjalili, S.; Lewis, A. The whale optimization algorithm. *Adv. Eng. Softw.* **2016**, *95*, 51–67. [CrossRef]
26. Yang, X.S. A New Metaheuristic Bat-Inspired Algorithm. In *Nature Inspired Cooperative Strategies for Optimization (NICSO 2010)*; González, J.R., Pelta, D.A., Cruz, C., Terrazas, G., Krasnogor, N., Eds.; Studies in Computational Intelligence; Springer: Berlin/Heidelberg, Germany, 2010; Volume 284, pp. 65–74.
27. Brajević, I.; Stanimirović, P. An improved chaotic firefly algorithm for global numerical optimization. *Int. J. Comput. Intell. Syst.* **2018**, *12*, 131–148. [CrossRef]
28. Petalas, Y.G.; Parsopoulos, K.E.; Vrahatis, M.N. Memetic particle swarm optimization. *Ann. Oper. Res.* **2007**, *156*, 99–127. [CrossRef]
29. Ali, A.F.; Tawhid, M.A. A hybrid cuckoo search algorithm with Nelder Mead method for solving global optimization problems. *Springerplus* **2016**, *5*, 1–22. [CrossRef]
30. Ali, A.F.; Tawhid, M.A. Solving Integer Programming Problems by Hybrid Bat Algorithm and Direct Search Method. *Trends Artif. Intell.* **2018**, *2*, 46–59.
31. Tawhid, M.A.; Ali, A.F.; Tawhid, M.A. Multidirectional harmony search algorithm for solving integer programming and minimax problems. *Int. J. Bio-Inspir. Com.* **2019**, *13*, 141–158. [CrossRef]
32. Xiang, W.; Meng, X.; Li, Y.; He, R.; An, M. An improved artificial bee colony algorithm based on the gravity model. *Inf. Sci.* **2018**, *429*, 49–71. [CrossRef]
33. Xiao, S.; Wang, W.; Wang, H.; Tan, D.; Wang, Y.; Yu, X.; Wu, R. An Improved Artificial Bee Colony Algorithm Based on Elite Strategy and Dimension Learning. *Mathematics* **2019**, *7*, 289. [CrossRef]
34. Lin, Q.; Zhu, M.; Li, G.; Wang, W.; Cui, L.; Chen, J.; Lu, J. A novel artificial bee colony algorithm with local and global information interaction. *Appl. Soft Comput.* **2018**, *62*, 702–735. [CrossRef]
35. Brajević, I.; Stanimirović, P. S.; Li, S.; Cao, X. A Hybrid Firefly and Multi-Strategy Artificial Bee Colony Algorithm. *Int. J. Comput. Intell. Syst.* **2020**, *13*, 810–821. [CrossRef]
36. Karaboga, D.; Akay, B.; Karaboga, N. A survey on the studies employing machine learning (ML) for enhancing artificial bee colony (ABC) optimization algorithm. *Cogent Eng.* **2020**, *7*, 1855741. [CrossRef]
37. Hussain, K.; Salleh, M.N.M.; Cheng, S.; Shi, Y.; Naseem, R. Artificial bee colony algorithm: A component-wise analysis using diversity measurement. *J. King Saud Univ. Comp. Inf. Sci.* **2020**, *32*, 794–808. [CrossRef]
38. Gorkemli, B.; Karaboga, D. A quick semantic artificial bee colony programming (qsABCP) for symbolic regression. *Inf. Sci.* **2019**, *502*, 346–362. [CrossRef]
39. Aslan, S.; Karaboga, D.; Badem, H. A new artificial bee colony algorithm employing intelligent forager forwarding strategies. *Appl. Soft Comput.* **2020**, *96*, 106656. [CrossRef]
40. Song, X.; Zhao, M.; Xing, S. A multi-strategy fusion artificial bee colony algorithm with small population. *Expert Syst. Appl.* **2020**, *142*, 112921. [CrossRef]
41. Choong, S.S.; Wong, L.-P.; Lim, C.P. An artificial bee colony algorithm with a Modified Choice Function for the traveling salesman problem. *Swarm Evol. Comput.* **2019**, *44*, 622–635. [CrossRef]
42. Karaboga, D.; Gorkemli, B. Solving Traveling Salesman Problem by Using Combinatorial Artificial Bee Colony Algorithms. *Int. J. Artif. Intell. Tools* **2019**, *28*, 1950004. [CrossRef]

43. Ng, K.K.H.; Lee, K.M.; Zhang, S.Z.; Wu, K.; Ho, W. A multiple colonies artificial bee colony algorithm for a capacitated vehicle routing problem and re-routing strategies under time-dependent traffic congestion. *Comput. Ind. Eng.* **2017**, *109*, 151–168. [CrossRef]
44. Sedighizadeh, D.; Mazaheripour, H. Optimization of multi objective vehicle routing problem using a new hybrid algorithm based on particle swarm optimization and artificial bee colony algorithm considering Precedence constraints. *Alex. Eng. J.* **2018**, *57*, 2225–2239. [CrossRef]
45. Lin, Y.-K.; Li, M.-Y. Solving Operating Room Scheduling Problem Using Artificial Bee Colony Algorithm. *Healthcare* **2021**, *9*, 152. [CrossRef]
46. Zhang, J.; Li, L.; Chen, Z. Strength–redundancy allocation problem using artificial bee colony algorithm for multi-state systems. *Reliab. Eng. Syst. Saf.* **2021**, *209*, 107494. [CrossRef]
47. Hancer, E.; Karaboga, D. A comprehensive survey of traditional, merge-split and evolutionary approaches proposed for determination of cluster number. *Swarm Evol. Comput.* **2017**, *32*, 49–67. [CrossRef]
48. Caliskan, A.; Çil, Z.A.; Badem, H.; Karaboga, D. Regression-Based Neuro-Fuzzy Network Trained by ABC Algorithm for High-Density Impulse Noise Elimination. *IEEE Trans. Fuzzy Syst.* **2020**, *28*, 1084–1095. [CrossRef]
49. Akay, B. A Binomial Crossover Based Artificial Bee Colony Algorithm for Cryptanalysis of Polyalphabetic Cipher. *Teh. Vjesn.* **2020**, *27*, 1825–1835.
50. Kumar, A.; Kumar, D.; Jarial, S.K. A Review on Artificial Bee Colony Algorithms and Their Applications to Data Clustering. *Cybern. Inf. Technol.* **2017**, *17*, 3–28. [CrossRef]
51. Brajevic, I. Crossover-based artificial bee colony algorithm for constrained optimization problems. *Neural. Comput. Appl.* **2015**, *26*, 1587–1601. [CrossRef]
52. Aslan S.; Karaboga, D. A genetic Artificial Bee Colony algorithm for signal reconstruction based big data optimization. *Appl. Soft Comput.* **2020**, *88*, 106053. [CrossRef]
53. Pooja, S.G. Innovative Review on Artificial Bee Colony Algorithm and Its Variants. In *Advances in Computing and Intelligent Systems*; Sharma, H., Govindan, K., Poonia, R., Kumar, S., El-Medany, W., Eds.; Algorithms for Intelligent Systems; Springer: Singapore, 2020; pp. 165–176.
54. Črepinšek, M.; Liu, S.-H.; Mernik, M. Exploration and exploitation in evolutionary algorithms: A survey. *ACM Comput. Surv.* **2013**, *45*, 1–33. [CrossRef]
55. Zhu, G.; Kwong, S. Gbest-guided artificial bee colony algorithm for numerical function optimization. *Appl. Math. Comput.* **2010**, *217*, 3166–3173. [CrossRef]
56. Canali, C.; Lancellotti, R. GASP: Genetic Algorithms for Service Placement in Fog Computing Systems. *Algorithms* **2019**, *12*, 201. [CrossRef]
57. Lukšan, L.; Vlček, J. *Test Problems for Non-Smooth Unconstrained and Linearly Constrained Optimization*; Technical Report 798; Institute of Computer Science, Academy of Sciences of the Czech Republic: Prague, Czech Republic, 2000.
58. Parsopoulos, K.E.; Vrahatis, M.N. Unified particle swarm optimization for tackling operations research problems. In Proceedings of the 2005 IEEE Swarm Intelligence Symposium, SIS 2005, Pasadena, CA, USA, 8–10 June 2005; pp. 53–59.
59. Santo, I.A.C.P.E.; Fernandes, E.M.G.P. Heuristics pattern search for bound constrained minimax problems. In *Computational Science and Its Applications—ICCSA 2011*; Murgante, B., Gervasi, O., Iglesias, A., Taniar, D., Apduhan, B.O., Eds.; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2011; Volume 6784, pp. 174–184.