Applying the big bang-big crunch metaheuristic to large-sized operational problems

Yousef K. Qawqzeh[1], Ghaith Jaradat[2], Ali Al-Yousef[3], Anmar Abu-Hamdah[4], Ibrahim Almarashdeh[5], Mutasem Alsmadi[6], Mohammed Tayfour[7], Khalid Shaker[8], Firas Haddad[9]
[1]Department of Computer Science and Information, College of Science-Zulfi, Majmaah University, Saudi Arabia
[2, 3]Department of Computer Science, Faculty of Computer Science and Information Technology, Jerash University, Jordan
[4]Department of Management Information System, College of Business, Taibah University, Saudi Arabia
[5, 6, 7]Department of Management Information Systems, College of Applied Studies and Community Service, Imam Abdulrahman Bin Faisal University, Dammam, Saudi Arabia
[8]Department of Computer Science, Faculty of Computer Science and Information Technology, Anbar University, Iraq
[9]Department of General courses, College of Applied Studies and Community Service, Imam Abdulrahman Bin Faisal University, Dammam, Saudi Arabia

Article Info

Article history:

Received Jun 27, 2019
Revised Nov 07, 2019
Accepted Nov 25, 2019

Keywords:

Big bang-big crunch metaheuristic
Bin packing
Combinatorial optimization
Job shop scheduling
Quadratic assignment

ABSTRACT

In this study, we present an investigation of comparing the capability of a big bang-big crunch metaheuristic (BBBC) for managing operational problems including combinatorial optimization problems. The BBBC is a product of the evolution theory of the universe in physics and astronomy. Two main phases of BBBC are the big bang and the big crunch. The big bang phase involves the creation of a population of random initial solutions, while in the big crunch phase these solutions are shrunk into one elite solution exhibited by a mass center. This study looks into the BBBC's effectiveness in assignment and scheduling problems. Where it was enhanced by incorporating an elite pool of diverse and high quality solutions; a simple descent heuristic as a local search method; implicit recombination; Euclidean distance; dynamic population size; and elitism strategies. Those strategies provide a balanced search of diverse and good quality population. The investigation is conducted by comparing the proposed BBBC with similar metaheuristics. The BBBC is tested on three different classes of combinatorial optimization problems; namely, quadratic assignment, bin packing, and job shop scheduling problems. Where the incorporated strategies have a greater impact on the BBBC's performance. Experiments showed that the BBBC maintains a good balance between diversity and quality which produces high-quality solutions, and outperforms other identical metaheuristics (e.g. swarm intelligence and evolutionary algorithms) reported in the literature.

Corresponding Author:

Yousef K. Qawqzeh,
Department of Computer Science and Information,
Majmaah University,
Majmaah University 66, Majmaah (Al Zulfi), 11962, Saudi Arabia, KSA.
Email: y.qawqzeh@mu.edu.sa

1. INTRODUCTION

Operational problems (OP) management is an important concept of dealing with complex real-world industrial problems (e.g. assignment and scheduling). As illustrated in the literature in the past three decades, Metaheuristics poses a potential impact on tasks of managing such operational problems. They proved to be very effective and promising techniques in industrial simulation modeling (SM), combinatorial optimization (CO), and machine learning (ML). This is due to the capabilities of a metaheuristic to adapt to a problem's specifications via parameter tuning and input/output analysis. Due time, Metaheuristics have become even more sophisticated in dealing with large-sized problem scenarios or rather huge datasets.

It is well known that the difficulty in achieving an optimal solution for any operational problem, in many cases, is a considerable challenge. Therefore, the last decade has seen the application of numerous metaheuristics in solving numerous operational problems such as CO and ML. Two classes of metaheuristics as mentioned by [1] are population-based and local search metaheuristics. Genetic algorithm [2] and the ant colony optimization [3] are among the generally utilized population-based metaheuristics in solving these problems. There have been comprehensive investigations on population-based metaheuristics. This type of metaheuristics is popular due to its ability to explore search space exploration, aside from being easily combined with local search methods for improving the process of solution exploitation [4]. Among the general methods of local search methods used on the problem include simulated annealing [5] and tabu search [6]. Their usage is factored by their ability in exploiting the solution space.

The strength of population-based metaheuristics is being anchored by the ability to recombine solutions in acquiring new ones. Within population-based algorithms, the recombination of elite solutions is implicitly conducted. This entails moving and swapping of assignments within a solution that denotes the exchange of information between generations of a good quality solution [1]. This refers to the generation of new solutions via a distribution over the search space which comprises a function of previous populations that signify the search experience [1]. Meanwhile, ‘implicit’ means that a solution is indirectly signified by the assignments’ fitness values or their contribution’s values to search such as in solution creation. With implicit recombination, [1] stated that the process of search could conduct a guided sampling of the search space. Using this recombination technique, potential areas of the search space can be effectively located [1]. The explicit recombination is one more recombination type. It is employed by the genetic algorithm, memetic (hybrid genetic) algorithm as well as by scatter search. Here, structured solution recombination of elite solutions is conducted in an explicit manner. This involves moving or swapping assignments within a solution that denotes the exchange of information exchange between generations via one or more recombination operators including mutation and crossover[1]. ‘Explicit’ means that a solution is directly signified by the actual assignment or the solutions’ allocation and fitness values. The selection of the solution recombination is influenced by nature as well as the construction of the problem and also by the metaheuristic chosen.

Nonetheless, in intensifying the search for solutions of higher quality, the population-based metaheuristic is regarded as weak. As such, specialized metaheuristics in the solution space exploitation (e.g. hill climbing) is generally hybridized with the population-based metaheuristics. This improves the process of intensification. In relation to this, hybridization between a population-based and other local search metaheuristics has been recommended in many studies (e.g., [1, 4, 7]). Local search metaheuristics could overcome the shortcoming (in the population-based) of solution space exploitation by improving the quality of solution more. Also, to generate better performance of hybrid metaheuristics, the usage of explicit memory such as the use of the elite pool, control on search diversity, and dynamically manipulating the population size are also recommended by [4].

A good performance can be attained if diversification and intensification of the search stay balanced. As mentioned by [8], a good performance (w.r.t. consistency, efficiency, effectiveness, and perhaps generality) can be seen via the maintenance of balance between the search’s diversification and intensification. This has led to the use of the big bang-big crunch (BBBC) in our study. Further, to achieve better performance, the use of an elite pool containing diverse solutions of high quality for controlling the search in terms of diversity, and dynamic manipulation of the size of the population, are also recommended considered by [4]. As demonstrated in the works of [9, 10], the BBBC comprises hybridization with some mechanisms of diversification and intensification for improving its solution space’s exploration and exploitation of the, where, an elite pool and a local search were used in combination for intensifying the search around elite solutions, with the diversity level maintained. It also possesses a dynamic population size manipulation besides the diversity control strategy.

The elite pool is generally referred to as an adaptive memory structure containing a set of diverse and high-quality solutions that keep valuable information about the global optima in the shape of a diverse and elite set of solutions. Using this structure, the process of search could recombine samples from the elite set and this allows the exploitation of valuable information pertaining to the global optima. The use of BBBC in this study is factored by its: easy implementation, provision of a deterministic choice of pool of elite solutions both quality and diversity wise which conducts a systematic neighborhood search within the Euclidean space, performance of pseudo-random diversification strategies for the combinations of structured solution, evolution of a renewed strategy via the exploitation of an adaptive memory for the preservation of good quality and diversity, provision of valuable information of elite or diverse solutions even without initial elite pool, support on representation of direct solution within a Euclidean space which can be manipulated easily, capacity in distributing the search over several solutions rather than only one solution as well as the capacity of quick convergence even when multiple local minima are present [11] which allows the search to quickly locate the elite solutions within diverse regions, elitism strategy with a pool of only diverse solutions which is enough for the solution space exploration, measuring Euclidean distances for similarities between solutions which assists in pointing the elite solutions, and less parameterized structure [11] which means freedom from issues of parameter tuning.

Therefore, this study attempts to find an answer to the research question “Does the diversity control strategy in the BBBC is sufficient enough to deal with large-sized problems?" As such, this study aims to fulfill the main objective, which is “To test the performance of BBBC in terms of generality and consistency, over a variety of COPs (e.g. quadratic assignment, bin packing, and job shop scheduling problems) and against advanced population-based metaheuristics”. The arrangement of this paper is as follows: Section 2 highlights the study’s problems, Section 3 discusses several works pertinent to the subject under study, Section 4 illustrates the proposed BB-BC metaheuristic as well as its design, Section 5 elaborates the outcomes of the experiment, and Section 6 concludes the study.

2. PROBLEM STATEMENT

This section demonstrates the problems considered in this study as testing platforms for our BBBC.

2.1. Quadratic assignment problem (QAP)

The QAP is a standard assignment problem in location theory, and categorized as an NP-hard COP, which was first introduced by [12] in the context of locating "indivisible economic activities". The term quadratic stems from the formulation of the QAP as an integer optimization problem with a quadratic objective function [1]. The QAP is defined by [13] as the problem of assigning a set of facilities to a set of locations with given distances between the locations and given flows between the facilities. The goal is to place the faculties on locations in such a way that the sum of products between flows and distances is minimal. A mathematical formulation of the QAP is shown in (1), taken from [13].

$$Minimize \quad \sum_{i=1}^{n} \sum_{j=1}^{n} a_{ij}\, b_{\Phi(i)\Phi(j)} \tag{1}$$

Given n facilities and n locations, two n x n matrixes A[aij] and B[brs], where aij is the distance between locations i and j, and brs is the flow between facilities and s, Φ is an arbitrary permutation of the set of integers 1,... ., n (corresponding to an assignment of facilities to locations), and Φ(i) gives the location of facility i in Φ. Both a and b represent the cost contribution of simultaneously assigning facility i to location Φ(i) and facility j to location Φ(j). The datasets on which we will test our algorithm are taken from QAPLIB [13] benchmark library. Some datasets such as the els describe distances between a set of facilities in a hospital and the corresponding patient flows, and the bur is among several datasets consisting of optimizing placements of a set of letters on a keyboard for steno-typists.

For a detailed discussion and formulation of the QAP, please refer to the original works of the founders of QAPLIB [13] benchmark library and the work of [14]. Generally, the QAP consists of several different types of datasets and that the particular datasets type has a considerable influence on the performance of metaheuristics. According to, the datasets of QAPLIB which we use in this paper can be classified into four classes:

- Unstructured randomly generated dataset: Randomly generated according to a uniform distribution. These are among the hardest to solve.
- Grid-based distance matrix dataset: The distances are defined as the Manhattan distance between grid points.
- Real-life dataset: Small size practical applications of the QAP. The flow matrixes have many zero entries and the remaining entries are clearly not uniformly distributed.
- Real-life like dataset: Randomly generated in such a way that the matrix entries resemble the distributions found for real-life problems.

In this work, we consider only the first class, unstructured randomly generated instances. An instance of the QAP has the following structure of three groups of data:(i) problem size; a Figure that determines the number of rows and columns of locations and facilities matrixes, (ii) distance matrix; a group of locations, and (iii) flow matrix; a group of facilities.

2.2. Bin packing problem (BPP)

The BPP is an assignment problem in which several items of different weights must be packed into several bins all with the same capacity. The aim is to minimize the number of bins required. The classical one-dimensional BPP is similar to the cutting stock and the knapsack problem. The difference is that all the pieces must be packed, and an unlimited number of bins are available. A mathematical formulation of the BPP is shown in (2), taken from [15].

$$Minimize \quad \sum_{i=1}^{n} y_{\mathrm{i}} \qquad (2)$$

where n is the number of pieces (and therefore also the maximum amount of bins necessary), yi is a binary variable indicating whether bin i has been used. For a detailed discussion and formulation of the BPP, please refer to the official website of ORLIB benchmark library. There are three groups of benchmark problems are available from the literature for the one-dimensional bin packing problem:

- The group of OR-Library (created and studied by [16]) which consists of two classes: uniform (uniformly distributed) and triplet (bin capacity and item size are generated deliberately).
- The group of [17]. It contains three sets. They are generated based on the number of the items, bin capacity and the ranges that the items' sizes are drawn from. The third setSch_Set3is considered to be a harder problem dataset because the item size is drawn from a very large range such that no two items have the same size.
- The group of EURO Special Interest Group on Cutting and packing. It represents a collection of a difficult dataset from a large number of test instances.

In this work, we consider only hard instances such as the second group (Scholl's third dataset) to test our BBBC.

2.3. Job shop scheduling problem (JSSP)

The JSSP is a scheduling problem in which number operations, each of which must be processed on exactly one of a number of machines, must be scheduled for processing. Operations are partitioned into jobs and the processing order of operations within each job is specified a priori. The aim is to minimize the makespan. Where the makespan is the total length of the schedule when all the jobs have finished processing. A mathematical formulation of the JSSP is shown in (3), taken from [15, 18, 19].

$$Minimize\ C_{max} \qquad (3)$$

where Cmax is the maximum completion time over all jobs (makespan). For a detailed discussion and formulation of the JSSP, please refer to [19], and the official website of ORLIB.

The subjects of QAP, BPP, and JSSPhave been widely researched, and they are in fact very common problems in real-life applications. They have been intensively studied in the literature with a large number of benchmark datasets being available. As such, they offer a very good platform for the researchers to test the impact of maintaining a balance between diversity and quality of search in the BBBC. So, those solutions to the three problems and analyses will be improved in terms of quality. This study will selectively compare the outcomes of our BBBC with the outcomes of some methods documented in the applicable literature.

3. RELATED WORKS

Diverse methodologies have been used for handling different categories of COPs. Thus, the ensuing subsections will highlight some of the most commonly used ones as well as those interesting ones. It should be noted that there has been a wide and successful usage of diverse types of metaheuristics for QAP, BPP, and JSSP solution. Somehow, the usage of the original BBBC was not identified at all for this purpose. The literature has presented countless metaheuristics. Among them are for the:

3.1. QAP

Numerous studies are still undergoing for four decades for solving different formulations of the QAP. For summarizing such amount, some brief surveys of solution methods for QAP are presented by [20-22]. In addition, a good comparison of evolutionary algorithms for solving the QAP was conducted by [23]. For studying each algorithm in a limited time span and to its speed of convergence, [24] presented excellent experiments of testing GRASP with path relinking approach for solving generalized QAP. [3] Presented a cunning Ant System with a local search and parallelization. [6] Applied an iterated tabu search algorithm which obtained new best results at the time. [25] Investigated the effectiveness of utilizing a memory structure of elite solutions in a tabu search algorithm.

3.2. BPP

In the last two decades, many intensive studies were proposed and presented for both one-dimensional and two-dimensional BPP by [26-29, 30]. Traditional evolutionary algorithms have been applied for the BPP such as ant colony optimization with a local search routine of [31-32], and a genetic algorithm by [33]. Hyper-heuristics have also been applied such as [34-36]. Applied a hyper-heuristic framework called CHAMP for the online BPP [33]. A unified hyper-heuristic framework is developed by [37]. A good survey and classification of approximation algorithms for the BPP were conducted by [38]. A hybrid algorithm was developed by [39] and obtained superior results compared to others. Their hybrid approach has a much-customized constructive schema that meets only the BPP specifications. Conducted a detailed experimental study on the effectiveness of a variable neighborhood search tested on the variable-sized BPP [40].

3.3. JSSP

The JSSP has attracted many researchers to study and test their approaches for many decades. Some examples of those studies are ant colony optimization by [18], scatter search [41]. [42] Presented a hybrid population-based metaheuristic called electromagnetism-like for solving a single machine scheduling problem. A hybrid particle swarm optimization was applied by [43] to a different class of the problem known as the flow shop. [44] Presented a hybrid population-based approach for solving the JSSP with a detailed comparison between a genetic algorithm and a scatter search. [45] Conducted an excellent review paper for the types and specifications of job scheduling problems, as well as a comprehensive review of heuristics and metaheuristics developed to solve the problem. Despite the effectiveness and good results of the abovementioned approaches, they are very problem-specific which makes it hard to adapt to other problems with different solution structures. Most of them tested their generality across several COPs or variable instances. This encourages us to test the generality of our BBBC as well.

3.4. Diversity control strategy

Based on the numerous methods highlighted previously, it can be said that there have been countless efforts of solving the three problems especially via the use of different approaches in combination (hybridization). From all the methods highlighted above, two key properties are salient: (i) First, employ a heuristic method for attaining an initial candidate solution; (ii) second, hybridize the metaheuristic with another heuristic method for improving the solution during the process of iteration. The implementation of primarily population-based hybridization has yielded considerable improvements towards the optimality of the solutions. For instance, population-based methods combined with multiple phase neighborhood search, or greedy randomized adaptive search, or local search, appear to be fairly effective. As stated by [4], such hybridization is to expand the strategy of the neighborhood in the population-based method.

Further, an adaptive memory structure makes up a key building block of an efficient and effective hybrid metaheuristic, for instance, tabu search algorithms and scatter search. The emphasis is on the notions of memory, intensification versus diversification, and exploitation versus exploration. Memory refers to the information gathered by the algorithm on the objective function distribution, and is representable as complex structures including solution recombination, for instance within the BBBC [9, 10]. Meanwhile, intensification exploits the attained information so that the current solutions can be improved. Generally, this entails a local search routine. As for diversification, its aim is to gather fresh information via search space exploration.

These components (e.g., memory, intensification, diversification, elitism, population manipulation, and solution recombination) are not always visibly distinctive. They are also very interdependent in an algorithm. As such, in this study, their advantages are used through a complex structure of data that updates the search information in a more effective manner, known as the elite pool. Here, the aim is to fully exploit the adaptive memory; in this study, it is used as an improved method of the attained best solutions following the combinations. In the context of the relationships: A pool refers to a data structure employed for keeping several solutions found to be possible of value all through the search [46]. A pool member is termed an elite solution and thus, the elite pool is a notion presentable as an adaptive memory. In relation to this, [47] made use of the notion of genetic algorithms of combining solutions for the generation of new solutions using a tabu search as a procedure for improvement. [48] Employed the tabu search and unified tabu search. Here, infeasible solutions are considered via the expansion of the objective function using a penalty function and continuous diversification. The approach taken by [49] was like [48, 50, 51]. Also using the elite pool concept, particularly the Granular tabu search, they limited the size of the neighborhood through the removal of edges from the graph that is not likely to emerge in an optimal solution.

Most of the methods highlighted in sub-sections 3.1-3.3 contain similar structures to the elite pool, e.g. the reference set in the scatter search metaheuristic. However, for the ones clarified in their solution representations, they do not employ implicit solution recombination, unlike our BBBC. This could be a great advantageous feature of our BBBC over other methods in maintaining a clear balance between diversity and quality of the search. [1, 52] demonstrated that solution recombination and adaptive memory in hybrid metaheuristics are essential for effective performance in terms of quality and diversity. To begin with, the fascinating contributions of the studies mentioned previously, have a linkage with the impact of the assignment. One way or another, this might impact the performance or even the significance of an elite pool. Owing to their usage on the same datasets, a comparison will be made between some of the methods highlighted in sub-sections 3.1-3.3 and our BBBC. The performance of BBBC in solving the three problems should be assessed because it would be worthwhile to do so.

4. APPLYING THE BBBC TO OPs

Based on the facts and properties mentioned in section 1, we have implemented our BBBC metaheuristic (from [53]) for more investigation of its capabilities and effectiveness. The BBBC is considered as one of the evolutionary algorithms which are widely used for managing OPs including COPs. However, the BBBC is still obscured to a large portion of the OPs' family.

4.1. BBBC description

Initially introduced by [54, 55], BB-BC is essentially a search algorithm inspired by the universe evolution theory which revolves around expansion and shrinking. As described by [11], this algorithm is primarily characterized by a fast search space exploration and aggressive exploitation of solution space. This is signified by the shrinking of the population in terms of size. The works presented by [54] and [11] provide the details. This research comprises further investigation on the effect of the diversity control following the inclusion of the performance and generality of the BBBC metaheuristic (from [11]) by having it tested on large-sized datasets of QAP, BPP, and JSSP. Figure 1 which comprises a generic pseudo-code of our BB-BC can be referred. There are many other methods inspired by nature that have been applied to the QAP, BPP, and JSSP, such as genetic algorithm, ant colony optimization, particle swarm optimization, scatter search, local search, hybrids, and hyper-heuristics. The BB-BC has been applied to a limited number of combinatorial optimization problems. For example, [54] first proposed and applied the original BBBC to the truss optimization problem, and compared it against the genetic algorithm (GA) and an improved version of it called combat-GA. They showed that the BBBC had outperformed the combat-GA in most of the test functions instances in terms of quality and speed. In another work, [56] compared the BBBC to particle swarm optimization (PSO), harmony search (HS) and ant colony optimization (ACO) over the size optimization of space trusses. They showed that the performance of the BBBC demonstrates superiority over PSO, HS, and ACO in computational time and quality of solutions. The BBBC was also applied to several optimization problems, such as target tracking for underwater vehicle detection and tracking [11]; and engineering optimization [57]. Lately, [9] applied an enhanced version of the BBBC to solve course timetabling problems, where it outperformed several similar methods and showed a consistent and fast convergence towards optimality. According to [10] tested the same BBBC on three classes of COPs; e.g. vehicle routing, traveling salesman, knapsack problems. It proved that it is effective and consistent in solving these problems to optimality in most cases. The BBBC has been applied twice for the data clustering problem by [58, 59]. It showed good performance as well as generated good quality results.

```
Big Bang phase (solutions construction):
Step 1: Employ Constructive heuristics to Generate $N_{pop}$;
    //for the 1st generation
    OR else
    Perform Permutations to elite pool to Generate $N_{newpop}$;
    THEN Measure Euclidean distances among solutions;
Big Crunch phase (Local Search move):
Repeat
    Step 2: FOR EACH $C_i$ in $N_{pop}$
        Generate some neighbors $N_s$ for all solutions in $N_{pop}$;
        THEN Replace parent with its best offspring $C_i^{new}$;
    Step 3: Find center of mass $C_c$;
    Step 4: Apply local search to $C_c$;
    Step 5: Update elite pool & best found solution $C_{best}$;
    Step 6: Eliminate some poor quality solutions; //reduction rate
Until population size is reduced to a single solution;
Step 7: Return to Step 1 If stopping criterion is not met;
Step 8: Return best found solution $C_{best}$;
```

Figure 1. A generic pseudo code of our BBBC

As mentioned, BBBC is grounded on a theory relating to universe evolution in the realms of physics and astronomy. The theory elucidates the creation, evolution and the ending of the universe. BBBC theory comprises two phases, namely the big bang (BB) and the big crunch (BC). The BB phase comprises a set of procedures of energy dissipation in nature with regard to disordering and randomness while the BC phase involves a procedure that arbitrarily dispenses particles and draws these particles into an order. The phases of BB and BC both signify large exploration of search space and best exploitation of solution space, respectively. The BB phase (a.k.a. energy dissipation) involves the random creation of an initial population of feasible solutions, and this is akin to GA in terms of the creation of a random initial population.

Gradually, the populations generated in the BB phase will be reduced in the BC phase. Such reduction is for decreasing the computational time and attaining fast convergence, while the solutions' diversity remains the same. The cost function value of a solution within the population signifies a mass, and as remarked by [54], the best solution is signified as the center of mass which will attract other solutions. Such a state is attributable to the notion that solutions with bigger massare are possibly much closer to the center of the search space (the universe), or to the point in which the convergence of the big crunch will occur.

According to [11], the BBBC specifically works with variable population size, for instance, stellar objects. BBBC can maintain search diversity. Thus, the problem of being trapped in a local optimum can be prevented while convergence within a reasonable speed can be obtained [57]. BBBC is akin to memetic algorithms but there is no combination of solutions (e.g. crossover) in BBBC, while the mutation is denoted by perturbations of solution. To demonstrate the characteristics of our BBBC, a summarized comparison between memetic, original BBBC, and our BBBC algorithms are highlighted in Table 1.

Table 1. A comparison between memetic algorithm and BBBC

Features	Traditional Memetic	Original BBBC	Our BBBC
Population	Chromosomes Large size	Stellar objects Large size	Stellar objects Large size
Reproduction	Probabilistic selection in Hamming space	Probabilistic selection in Euclidean space	Probabilistic selection in Euclidean space
Combination	Randomized Crossover and Mutation	No, Mutation is a perturbation	No, Mutation is a perturbation
Evolution	Survival of the fittest	Strongest gravitational pole	Strongest gravitational pole
Local search	Significant intensification	Significant intensification	Significant intensification
Search update	Randomized selection	Pseudo- random selection	Pseudo-random selection
Memory use	No, only the Population pool	No, only the Population pool	Adaptive explicit memory (e.g. elite pool)
Search experience	Limited information about solution's components	Limited information about solution's components	Information about the solution's components via the elite pool
Diversification	Mutation	Euclidean distances	Euclidean distances, Perturbation (Diverse replacement)
Intensification	Selection, Crossover, Replacement	Centre of mass	Centre of mass,Local Search
Solution recombination	Explicit	Implicit	Implicit

In essence, our BBBC presented in this study is distinct from the original BBBC that [54] had introduced, particularly with respect to its representation of exploration and exploitation phases (solution construction and improvement). In particular, an assembly of elite solutions for the creation of a new promising population in successive *BB* phases is exploited in this study. Here, the elite collection comprises solutions of good quality. On the other hand, the original BBBC reconstructs new solutions from scratch in the creation of the new generation. Also, variable neighborhood structures and simple descent heuristic (as a local search) are used in this study, On the other hand, [54] scrutinized solution neighbors employing either greedy descent or steepest descent. Additionally, in determining the boundaries (allowable space) of the successive population, this study employs the quality of the produced solutions and the minimum Euclidean distance in representing the *center of mass*, that is, the best quality solution, and maximum, minimum cost values of solutions within the elite pool which contains solutions of local optima. Comparatively, in the original BBBC, the positions of solutions which are denoted by the Euclidean distances and the population distribution's standard deviation are computed relatively to the *center of mass* within the search space, and the magnitude of gravitational attraction that impacts the population to converge toward the *center of mass* within the Euclidean space [54]. The boundary of the search space was initially ascertained using the summation of the Euclidean distances of all solutions within the population. Somehow, to efficiently control new solutions' production within a desirable quality limits for the convergence toward good quality solutions, the measurement of the Euclidean distance of the entire population is also considered.

The Euclidean distance assists in the determination of the search space's boundaries and distribution. Actually, in BBBC, the Euclidean distance is irreplaceable. In other words, no other distance measurements, for instance, the Manhattan distance, can be used in this context. Normally, the distribution of the new offsprings for the successive iteration *BB* phase as well as in *BC* phase, is around the *center of mass* (C_c) (as in [54]) see (4):

$$C_i^{new} = C_c + \sigma \tag{4}$$

Here, C_i^{new} denotes the new produced solution i; while σ signifies a standard deviation of a normal distribution. The standard deviation decreases following the elapse of iterations based on the formula below (5) [54]:

$$\sigma = \frac{r\alpha(C_{\max} - C_{\min})}{k}, 0 < \frac{\alpha}{k} < 1 \tag{5}$$

Here, r represents a random number between [0,1], α denotes a rate of reduction of the search space size, C_{max} and C_{min} represent the elite pool's upper and lower boundaries while k represents the number of *BB* phase iterations. As such, the production of the new offspring is according to (4) within the upper and lower limits. The production of offsprings is via the performance of some perturbations to the solutions in the elite pool. It is necessary to have lower and upper boundaries to enable control to the distribution of solutions. In this paper, our preliminary experiments showed that r has no significant impact on the process of population reduction. Thus, it is taken out by having its value fixed to 1.

In the last part of the BC phase signified by the reduction of the population size to one solution, a new generation is created from the earlier generations' elite pool with a similar population size (as in the first generation), beginning with the earlier C_c. Here, through shakings performed to the solution, a new population from the elite pool is recreated by the algorithm where the maximum and minimum of the earlier generation's solutions' cost values become the limits e.g. bounded with (4). The inclusion of potential good quality solutions is assured through the allowance of an extended lower bound, meaning that, the enhanced solutions are all allowed even those outside the bound, while the upper limit is limited so that the obtainment of worse solution can be limited.

In this paper, our BBBC starts with the construction phase known as the *BB* phase or the diversification phase. This phase comprises the construction of a population of N_{pop} preliminary candidate solutions C_i from scratch (*Step 1*) for the first generation. For the succeeding *BC* phase, the new population is created from the elite pool, but the elite solutions themselves are not included in the new population. During this step, shaking is performed to solutions in the pool confined by the upper and lower cost values of solutions within the elite pool. Table 2 illustrates the constructive heuristics used in the construction phase.

Table 2. Constructive heuristics used in our BBBC

Problem	Constructive Heuristic
QAP	Nearest Neighbor (NN) heuristic:
	Visit either the closest neighbor or the second or third closest neighbors, with a selection probability of 1/3 for each neighbor.
BPP	Bin-oriented FFD+B2F:
	Bin-oriented (improved) first-fit-decreasing (iterated items packing into the bin of the smallest index with sufficient capacity) with best-2-fit (iterated FFD until the bin is filled)
JSSP	Dispatching (or Priority) Rules:
	Shortest Remaining Processing Time (SRPT); an operation with shortest remaining job processing time.

Also, during this step (*Step 1*), measurement is made to the Euclidean distances among solutions within the population. This is for establishing a diversity control over the search and for estimating an elite solution in terms of its attractiveness. Here, it is possible that the diversity of search is bounded to a certain degree based on the differences between solutions' quality values. As an example, a difference between two solutions namely C_i and C_{i+1} is denoted by the difference of (distance d) between the values of fitness those solutions ($d(C_i, C_{i+1}) = f(C_i)-f(C_{i+1})$). Worded simply, a larger difference between C_i and C_{i+1} denotes the higher probability of solutions to encircle each other in the following iteration. Such occurrence is taken into account so that the search is not diversified too much, and thus, the convergence is toward solutions of good quality effectively as well as efficiently. Solution with the best quality with the minimum Euclidean distance, as the C_c is chosen in this study. The most diverse solution comprises a solution with a larger maximum distance. Such a solution may contain structure and fitness costs that are totally different from the elite solutions. The computation of Euclidean distances among solutions in the population as shown in 6, as well as the distances between solutions in the population and solutions in the elite pool as demonstrated in 7 are as follows (similar to [54, 7]):

$$d_{\min}(C_i, C_{i+1}) = \sqrt{\sum_{i=1}^{N}(C_i - C_{i+1})^2} \tag{6}$$

where $i \in N_{pop}$ $\{C_1, C_2\ldots,C_N\}$

$$d_{\min}(p,q) = \sqrt{\sum_{i=1}^{n}(p_i - q_i)^2} \tag{7}$$

Here: $d_{min}(p, q)$ denotes the distance between each solution (p) in the population and every solution (q) presently in the elite pool (best quality solutions C_{best}, one or more C_c also included). For instance, a distance between two solutions is stated as ($f(p_1)$-$f(p_2)$), where a solution's fitness value (quality) is subtracted from the other, while the distance between a solution and a C_c is computed as ($f(p_i)$-$f(q_i)$) [7]. The Euclidean distance basically looks into the square root of differences between solutions. [7] Mentioned that in the nature-inspired algorithms, the population diversity or the solution space's density estimator is assessable with the sum of the Euclidean distances between a solution and with the rest of other solutions in the population as an assessment of how much that candidate solution contributes to the diversity. The attractiveness of a solution containing a minimum distance from the elite solution is greater toward that elite solution (C_c).

Over time, our BBBC documents the diversity of the population. The calculation in terms of (6) comprises the minimum average distance of a solution from all other solutions within the population, which is also termed by [60] as the average distance from all candidate solutions. In terms of (7), the computation comprises the minimum distance between a solution in the population and the C_c which is also termed by [60] as the distance from the best candidate solution of the population. [60] Further mentioned that the problem of getting trapped in local optima can also be prevented.

Step 2 involves the *BC* phase (improvement) which is also known as the intensification phase or a local search move. First, several neighbors of all solutions in the population plus C_c are produced through simple perturbations. The best offspring will replace each solution. This results in better quality solutions in the following population, while the diversity of the search remains the same. Such is done so that premature convergence of the search can be prevented, that is, the search diversity is conserved by the retaining some of the poor-quality solutions, considering that some of these are taken out from the population that went beyond the upper boundary. The entire BBBC cycle denotes the balance between diversity and quality of the search. Here, the *BC* phase (solution space exploitation) gradually shrinks the population into a single elite solution. On the other hand, the big bang (search space exploration) produces an entirely new population of diverse solutions from among those within the elite pool.

Step 3 comprises the determination of C_c according to the discovered best solution cost value (C_{best}) and the minimum average distance from the remainder of the population. The use of a simple descent heuristic is for a predefined number of non-improvement iterations (*Step 4*) to further improve C_c. Meanwhile, *Step 5* involves creating and updating the elite pool (collection). Here, the best solutions $C_c$of the earlier generations are kept within the elite pool and used as reference solutions for the *BB* phase in succeeding iterations. Fixed-size of the elite pool is used in this study; during the first iteration, several good solutions were chosen to be added into the pool. Every iteration the elite pool is updated, and this is done through the replacement of the worst solution cost in the present C_c and solutions. As can be seen (4), the reduction of the population size (*Step 6*) leads to a gradual convergence of the search into a single solution. Here, poor quality solutions around C_c are taken out. The *BC* phase is done over and over until singularity is achieved (i.e., the population size is shrunk to a single solution).

A new *BB* phase starts after the reduction of the population size into a single solution in the *BC* phase (*Step 7*). Here, the first step is repeated; a new population is produced from the elite pool via the addition of elite solutions into the new population and the creation of several neighbors from them for the establishment of the new population, instead of creating new solutions from nothing as was laid down by [51]. All C_c solutions (in the elite pool) are included in the new population if the elite pool is completely occupied. The purpose of conducting this step is to sustain a higher diversity level so that premature convergence can be avoided. However, in the initial big bangs where the elite pool is yet to be C_c solutions obtained from earlier big bangs, *centers of mass* in the elite pool were all excluded from the new population. The processes of search in our BBBC are done over and over until the stopping criterion is satisfied. In other words, the processes will stop when either the maximum number of iterations is achieved, or when the best quality solution is located. Lastly, BBBC returns the best-discovered solution (*Step 8*).

In this work, neighborhood structures (N_s) are randomly employed to the entire population N_{pop}, C_c is included (i.e. in *Step 1* and *Step 3*). A number of neighbors are created for every $C_c$solution at each iteration. Here, the best neighbor is selected as a replacement to its parent solution for the ensuing generation of N_{newpop}. Table 3 illustrates the N_s employed in our BBBC for each problem.

Table 3. Problem specific neighborhoods

N_s	QAP	BPP	JSSP
		Description	
N_1	Swap the locations of two facilities in the same assignment.	Select items from the bin with the largest residual capacity and try to shift them to the rest of the bins.	Revers a set of jobs of random length in a machine.
N_2	Transfer a facility in one location to another in the same assignment.	Move half the items randomly selected from the current bin to a new bin if the number of items in the current bin exceeds the average item numbers per bin.	Swap two jobs from the same machine.
N_3	Swap the locations of three facilities in the same assignment.	Select the largest item from the bin with the largest residual capacity and exchange this item with another smaller item (or several items whose capacity sum is smaller) from another randomly selected non-fully-filled bin.	Transfer a job from its position in one machine to another position in the same machine.
N_4	Transfer two facilities in two locations to another location in the same assignment.	Similar to N_3, but randomly select two non-fully-filled bins with a probability proportional to the number of their residual capacities.	Swap three jobs from the same machine.
N_5	Revers a set of facilities in the same assignment.	Select the biggest item from a probabilistically selected bin.	Transfer two jobs from their positions in one machine to another position in the same machine.

Any local search routine can be employed in our BBBC. A local search is employed to improve the quality of solutions by exploring their neighborhoods without sacrificing the diversity of the search. A simple exploration of some neighborhoods of a solution (e.g. simple shaking of assignments in case of QAP, or bins in case of BPP, or machines in case of JSSP) is employed in the *BC* phase, as it may be sufficient enough to escape the local optima. Through our experiments, we observed that employing N_s in Table 3 is quite sufficient and fast to explore some good neighbors of an elite solution. Generally, the structures of the neighborhood comprise relocating a randomly chosen neighbor solution around one C_c; swapping two randomly chosen neighbor solutions from two randomly chosen C_c, and swapping all neighbor solutions around two randomly chosen C_c. As a substantial mechanism intensification, a simple descent heuristic local search is used. This improves the quality of solutions as their neighborhoods are explored without foregoing the diversity of the search. In the *BC* phase, a simple exploration of several neighborhoods of a solution is used. For instance, simple shaking is performed, see Table 3. This may be sufficient in escaping the local optima.

5. COMPUTATIONAL RESULTS AND DISCUSSION

Our BBBC is also comprised to experiment the impact of using an elite pool together with its recombination of implicit solution. This means that comparison will also be made between this method and others that also employ explicit recombination. As such, the aim of this study is to investigate the effect of the elite pool on the performance of the BBBC and how it maintained a balance between quality and diversity. With the use of an elite pool, the performance of the BBBC metaheuristic, in terms of consistency, efficiency, effectiveness as well as a generality, is examined by having the method tested on QAP, BPP, and JSSP.

5.1. Experimental setup

The parameter tuning of our BBBC is a combinatorial problem itself. It is quite easy to find an approximate optimum for a given parameter, but since they are all correlated, the problem becomes harder and highly non-linear. Thus, we focused on testing different general improvements for the BBBC rather than fine-tuning each parameter in order to get a slight improvement for a particular instance. It is possible that better solutions would be found by using a set of instance-dependent parameters. However, our aim is to design a robust solver that is able to efficiently solve a large variety of instances.

To test the consistency of our BBBC, it was run 10 times for every instance from each benchmark dataset with a number of iterations as a stopping requirement which is relaxed running time, or a predefined running time in seconds (depends on instance's size), or once the optimal solution is found. A platform ofIntel Core i7 2.30 GHz processor, 8GB RAM, and Java NetBeans IDE v8.2 was employed for the experiments.

Parameters are experimentally established (e.g. elite pool size) and are grounded by the literature as well (e.g. Elitism). For example, in terms of genetic algorithms, BBBC adheres to the classic population size. Table 4 can be referred to. These parameters are tuned in a way that we first calibrate the elite pool size, reduction rate, and non-improvement iterations number. Then we fixed the perturbation operator. Based on preliminary testing, we observed that these parameter settings give satisfying results. The size of the memory structures in our BBBC metaheuristic was intentionally fixed in this study (depending on the instance's characteristics). As for the update strategy, it was maintained. A comparison was also made between this method and other similar metaheuristics and standalone methods, including the original BBBC. The effect of the elite pool in BBBC was thus explored in this study.

Table 4. Parameters settings used by our BBBC for QAP, BPP, JSSP

Parameter	Description
Population size	100
Elite pool size	10
Reduction rate	0.8
Neighbors created for each generation	5
Search update	Elitism
	Last population solution is forced to be always the best
Local search routine	Simple descent heuristic
Stopping criterion local search	500 iterations
Local search's Non-improvement	30 iterations
Stopping criterion per run	100,000 iterations

5.2. Experimental results

There are hundreds of instances provided for the three COPs. Therefore, we have determined to test our approach to some common hard and large-sized instances tested across the literature. So we didn't consider the small size benchmark instances from the ORLIB or the QAPLIB (e.g. QAP instances whose sizes range from n=500 to 1000) since they are solved very easily within at most 30 seconds by our BBBC algorithm and are also solved relatively easily by most recent metaheuristics. In the following subsections, Tables 5-7 show our results obtained for our test on the QAP, BPP, and JSSP, respectively. These tables also show the best-known results with those obtained by our BBBC highlighted in bold. The standard deviation (*Std.*) is also presented. Many published works reported the deviation percentage-Δ (%)-(or brief statistics) of their best solutions from best-known solutions. Therefore, in this work, we report our results similar to their illustration. The deviation percentage-Δ(%)-from the best-known result found in the literature is calculated for each instance as follows: $\Delta(\%)=((a-b)/b) * 100$,where a is the best result returned over 10 independent runs, and b is the best-known result found in the related literature.

Tables 5-7 further demonstrate that the results generated by our BBBC supersede those of other compared algorithms in QAP, BPP, and JSSP. Many of which are optimal; e.g. Tai15a=388214 in QAP; or equal to the best-known solutions reported in the literature; e.g. Tai73=5552 in JSSP. For some instances of QAP, BPP, and JSSP, the solutions generated by our BBBC are similar to those generated by other algorithms. Further, with respect to the time spent in locating the best solution; our BBBC is far better than other algorithms in almost all of the instances, where they have reported a range of their computational time between 30-500 seconds for the JSSP, 500-1500 seconds for the QAP, and for the 50-300 seconds BPP. In some studies, they reported an extensive running time exceeds 18 hours or even a couple of days, especially the exact methods or constructive heuristics. These methods may guarantee the feasibility of initial solutions but fail to improve them into optimality, if applicable, in a reasonable time.

However, these time indicators (in seconds) are reported for small instances on relatively old platforms, while the reported ones in hours of running time are for the very huge and hard instances such as the datasets we tested in our study. For example, in the QAP, some hard instances were solved in about 5 hours or less than 30 minutes in some studies. While in our study, our BBBC solved them in time ranges between 15-50 minutes. In preliminary experiments, our BBBC solved all small instances in a matter of seconds. Many studies in the literature reported their results in average or percentage, or even difference values; which make it hard for us to present clear and fair comparisons.

Based on Table 5-7 (see Appendix), our BBBC has obtained optimal solutions for some instances or near-optimal for most instances. It also obtained better results compared to other methods, specifically the original BBBC. The tables also show the gap between the best-known solution and our BBBC. A zero gap indicates that the best known (or rather optimal) solution was obtained. Overall, in comparison to the best-known solutions, our BBBC can produce very good solutions. For example, it has obtained optimal solutions for 8 instances out of 15 in the QAP and the rest equal to the best known results. In the BPP, it has obtained the best results for all instances. In the JSSP, it has also obtained the best results for all instances. Across all instances, -10 runs for each instance– the average Δ(%) is only 0%. In addition, the consistency of our method can be defined by the *Std.* over 10 runs, where the *Std.* is relatively small for all instances of the three problems.

In general, as opposed to other best-known solutions, our BBBC generates high-quality solutions and a small standard deviation for every instance. As opposed to the best-known results, the results obtained in this study are either optimal or the same, which means that our BBBC converges to the global optimum in every run, whereas the problem of getting trapped in local optimum solutions may be faced by other algorithms. Based on the outcomes obtained, it can thus be said that this study is using a very efficient and competitive methodology in solving the problems of QAP, BPP, and JSSP particularly with respect to solution quality and consistency. As such, the fulfillment of those criteria leads to the generality of this study's proposed BBBC over diverse sizes of datasets. Overall, our BBBC is better and more consistent than the original BBBC across all tested problem domains.

We observed that when utilizing the elite pool alongside the local search routine in our BBBC, the search converges very quickly towards a high-quality solution. This suggests that once the local search employed a smaller population size is may be more effective in exploring the search space, which may also lead to better performance than an equivalent number of neighborhood structures from randomly constructed solutions. That is why we have utilized the strategy of generating new solutions from the ones in the elite pool rather than from scratch.

5.3. Comparison with previous works

This section elaborates on the performance assessment of our BBBC against other conventional and hybrid algorithms reported in the literature. In specific, this section will elaborate on: (i) the evaluation of the benefit of integrating an elite pool within the BBBC, and (ii) the testing on the generality and consistency of the BBBC over three problems and comparing between the BBBC and other well-known algorithms. For supporting this study's hypothesis on the impact of the diversity control strategy (via the elite pool and the local search routine) on the performance of the BBBC, we compared our BBBC with several conventional and hybrid metaheuristics containing no elite pool. As an example, a genetic algorithm usually has a pool (an explicit memory specifically) of diverse solutions, but it has no pool of elite solutions (diverse and high-quality) [1, 4]. This explains why this algorithm possesses a great mechanism of search diversification while lacking an efficient intensification mechanism [1]. For certain algorithms including memetic and swarm optimization algorithms, the usage of the elite pool can improve the performance of metaheuristic algorithms in the resolution of various problems of optimization [61].

A lot of methodologies used in QAP, BPP, and JSSP did not include the use of an explicit or implicit memory, which may lead to the lack of sustaining a balance between the search's diversity and quality. A systematic selection strategy is also lacking, making the current study's outcomes outstanding. This might be the reason behind their poor results or (more importantly) their ability to tackle only one specific problem (a customized methodology for solving a specific problem). On the other hand, some work reported in the literature used some memory to store the best solution (such as the particle swarm optimization, or the ant colony optimization), but unfortunately, they did not provide a clear description of the memory. The elite pool is structured in a manner that effectively interacts with the strategy of implicit solution recombination while the Euclidean distance measurement makes available an adaptive search update. Hence, a fairly quick convergence towards high-quality solutions through the simple descent heuristic may be certain without having the search diversity sacrificed. As indicated, our BBBC has an explicit memory to enable the storage of high-quality and diverse solutions. Nonetheless, having to directly apply assignments and perturbations can be exhaustive, e.g. apply neighborhood structures that are problem dependent, to good quality or diverse solutions for more quality improvements could be time-consuming.

From Tables 5-7, for most instances (of the three COPs), it can be clearly seen that our methodology is better than those without elite pools. This demonstrated the effectiveness of using an elite pool in a population-based method. In a few cases, our methodology obtained the same results as those of the methodologies without elite pool; this might be due to the pseudorandom behavior of a population-based metaheuristic. As mentioned before, our BBBC metaheuristic possesses an explicit memory to store elite solutions; yet it could be exhaustive to directly apply permutations and perturbations, e.g. apply problem-dependent neighborhood structures, to good quality or diverse solutions for more quality improvements. As the case of ant colony optimization, we believe that the transition between implicit and explicit solution representations during the search process is a hard task. The effect of the quality of using an elite pool has been determined. Specifically, we applied the conventional BBBC with no elite pool and employ an iterated local search routine. Then, we compare it to our BBBC with an elite pool and a simple descent heuristic as the local search routine. Some statistically significant conclusions on the performance of our BBBC are worth discussing.

Therefore, the Wilcoxon test (pairwise comparisons) with a significant level of 0.05 is performed. The *p*-value of the Wilcoxon test of our BBBC versus original BBBC is presented in the last column of Tables 5–7. Where "+" indicates our BBBC is statistically better than the original (*p*-value <0.05), "-" indicates that ours is outperformed by the original (*p*-value >0.05) and "∼" indicates that both algorithms have the same performance (*p*-value=0.05). The results inTables 5-7 (last column) show that our BBBC is:

- Statistically better than the original on 13 instancesforQAP, 10 instances for BPP, and 30 instances for JSSP.
- Not statistically better than the original on none of the instances for all the considered problems, and
- Perform the same as the original on 2 instances forQAP.

Briefly stated, the obtained outcomes demonstrate the superiority of our BBBC with respect to consistency, efficiency, and generality, particularly in terms of the tested datasets. This is primarily factored by the usage of the elite pool within the BBBC which imparts a positive impact on the capacity of our BBBC in generating outcomes of good quality that are also consistent as opposed to the conventional BBBC. In all datasets, *Std.* and Δ(%) of our BBBC shows stable and better outcomes or outcomes that are equal to those generated by other metaheuristics. These observations are proofs of the capacity of our BBBC in generating good quality outcomes overall datasets, rather than just a few ones. From the experiments, it is clear that the outstanding performance of our BBBC is primarily factored by the hybridization of BBBC with:

- Explicit memory structure such as elite pool
- Simple descent heuristic as a local search of 5 neighborhood structures,
- Implicit solution recombination, and
- Euclidean distance.

The purpose is to diversify the search through the exploration of diverse regions of the search space, or rather, by avoiding local optima, while the high-quality solutions are maintained. The result generally shows the significant impact of the hybridization of the elite pool and the local search routine with the BBBC on its performance in solving the problems of QAP, BPP, and JSSP. As such, it is clear that our BBBC and the original BBBC (proposed by [54] differ from one another. Firstly, there is no elite pool in the original BBBC and thus, it is not effective when exchanging search experiences between the BB and BC phases. Additionally, having a rate of reduction of 10% in the population size is not enough to attain better convergence; while speed is incredible, there would still be no considerable enhancement. The reduction is conducted by taking out the worst solution from the population at each iteration. Lastly, the original BBBC contains Euclidean distances measurement and an iterated local search.

Comparatively, our BBBC has both an elite pool and a local search. With respect to the reduction rate of the population size, as well as the measurement of the Euclidean distance, it is performed by taking out the solutions of poor quality around C_c from the population at every iteration. It also employs a simple descent heuristic. As can be viewed in Table 5-7 (see Appendix), our BBBC shows the best performance and consistency when it comes to acquiring solutions of good quality in nearly all of the runs. Such is evidenced by the maintenance of a balance between the search's diversity and quality via the interaction between solutions in the elite pool, Euclidean distance, implicit solution recombination, reduction rate, the restart of a new population as well as the local search routine. It is thus deducible that the inclusion of an elite pool alongside the local search into the BBBC has majorly contributed to the improvement of the search particularly in terms of intensification and diversification. Also, the Euclidean distance affects the process of intensification.

As evidence, our BBBC can reliably generate good quality results (see *Std.* and Δ (%)). Its results are fairly comparable to some of those attained using other metaheuristics documented in the literature, where smaller difference denotes a more consistent algorithm. The superiority of our BBBC in terms of the results generated could be linked to some factors. Firstly, a reduction in the population size may assist the convergence of the search to local minima or C_c in the phase of *BC.* Meanwhile, the recreation of a new population in a new *BB* phase may assist in the diversification of the search. The search of certain neighbors inside the boundaries of the search space in the *BC* phase may likely to assure a considerable improvement to the solution. Our BBBC includes the exploitation of an elite pool in creating a new promising population in succeeding *BB* phases. Here, good information about elite solutions is transferred to the next generations so that the recombination of good quality solutions can be performed.

Nonetheless, the usage of elite solutions is for producing new potential solutions (instead of doing it from zero) for restarting the search with a new diversified population but with quality almost identical to that of the current C_c. Valuable information is provided by the elite pool particularly in terms of the location of the global solution (the sought-after C_c) that is shown by the Euclidean distances between solutions in the population and the *center of mass*(s). This information is then utilized by the local search to explore the solution space around the current $C_c$by generating a set of neighborhood structures. The experiments conducted in this study show the effectiveness of adding the elite pool, a local search, as an attempt to improve the original BBBC. Here, the elite pool is exploited so that a balance between diversity and quality of the search can be preserved. At the same time, the Euclidean distance and implicit solution recombination provide assistance in the process of search update. With local search, the process of enhancing the solutions' quality becomes more significant.

5.4. Diversity control strategy

We turn our attention now to analyzing two important components of our BBBC metaheuristic, the adaptive memory structure (the elite pool) and the local search. This strategy involves maintaining a balance between diversification and intensification of the search. This could also be applicable to search space exploration against solution space exploitation, in other words, the BB and the BC phases. The elite pool adaptively selects and manipulates a number of high quality and diverse solutions. This is, as we believe, essential for our BBBC. To be more confident about this strategy, we carried out an additional experiment to examine the influence of the elite pool and the local search. We applied the original BBBC without adding any other components or tuning its parameters. It can be referred to as Tables 5-7 and notice the differences between our BBBC and the original BBBC. Table 1 summarizes the structural differences as well. In the original BBBC, the best solutions are uniformly selected without using any memory information. In addition, the original one employs an extensive local search, known as the iterated local search. Our BBBC allows the search to gradually converge from performing exploration to exploiting the immediate neighbors of the best solution. In our case, this could be achieved by:

- Determining how often the local search should be applied, this is at most 500 iterations in total and 30 subsidiary iteration in case of consecutive non-improvements.
- Specifying on which solutions the local search should use, which are the best solutions in the elite pool.
- Determining how long the local search should be run, which is the number of created neighbors of the best solution found so far.
- Specifying how efficient does the local search need to be, which depends on the problem's characteristics as well as the size and type of neighborhood structures.

5.5. Computational complexity

The computational complexity of the algorithm should be taken into account so that the part that is most compute-intensive in the proposed BBBC metaheuristic can be determined. The analysis conducted in this study demonstrates that the computational complexity of the search in our BBBC metaheuristic is not high as opposed to most metaheuristics. This is, in fact, another positive attribute of the BBBC, particularly when it comes to the general and computational efficiency for a range of problems. In order to ascertain where the considerable portion of the time is being spent, the computational complexity or cost of our BBBC metaheuristic is briefly described next.

The general computational complexity for the original BBBC is $O(k*n)$. Here, k denotes the number of perturbations, and n denotes the number of iterations. Somehow, our BBBC begins to initialize a population using the complexity of $O(m^2+p)$. Here, p denotes the size of the population while m represents the dimension of the solution. Further, a constructive heuristic, e.g. nearest neighbor, is applied to initialize with the complexity of $O(n^2)$. Irrespective of a specific type of constructive heuristic, the complexity is generally expressed as $O(m^2 * p)$. Afterward, a population of solutions is created for the elite pool with the complexity of $O(s*m)$, where s is the memory size and m is the dimension of the solution (e.g. the number of jobs and machines in JSSP). Then, to create new superior quality solutions with the complexity of $O(n)$, the simple descent heuristic will be used by the BBBC. Further, the neighborhood structures with the complexity of $O(n^2 \log n)$ are employed to search for a solution's neighbors. The simple descent heuristic will be called at every n consecutive, non-improvement iterations. This allows the creation of a new population from scratch, or the search will be re-initialized as follows:

- Selection = $O(m)$.
- Mutation (perturbation) = $O(m\ (1+q))$, where q is the mutation probability.

The elite pool is then updated, and its solutions sorted. Here, the complexity of the solution is denoted by $O(s \log(s))$. The elitism is used to update the search. The search inspects solutions in the memory whether to loop or to stop:

- Elitism = $O(m * p^2)$.
- Loop or Exit (stopping criterion) = $O(m * p)$.

As such, the overall complexity for our BBBC is: $O(m^2+p)+O(m^2*p)+O(s*m)+O(n)+O(n^2 \log n)+O(m)+O(m\ (1+q))+O(s \log(s))+O(m*p^2)+O(m*p)$. Note that $O(m^2*p)$ can be replaced by $O(n^2)$ for the QAP, and by $O(n^2 \log_2 (n))$ for the BPP and JSSP.

There has been no simplification made because this study wishes to draw out the complexity of the many stages. As observed, in Step 5, the complexity of the local search routine (O(*n*)) is very low, for instance, *w.r.t.* to the parameters of *n*, *m* and *p*, and so forth. As such, it is not regarded as a critical factor in our BBBC. The crucial factors are shown in the elite pool and the search update (refer to steps 4 and 7, steps 6 and 8, respectively). In certain cases, constructive heuristics are also presentable as crucial factors (refer to Step 3). Essentially, our BBBC has the capacity to employ the ability the heuristic information regarding diverse and high-quality solutions in instance solving, which is through the elite pool, to allow the diversification of the search while intensifying the enhancement of a high-quality solution. As evidenced by the results, our BBBC provides a general mechanism irrespective of the nature and complexity of the instances. It is also applicable to other domains with no significant amount of changes to be made prior to the usage. In fact, only constructive heuristics and neighborhood structures need to be changed. It should be noted that in general, the application of a methodology to other problem areas or even different instances of the same problem necessitate a significant amount of modification, for example, the modification on algorithm parameters or structures. Comparatively, our BBBC can be simply used across different datasets of the QAP, BPP, and JSSP. It is also hoped our BBBC would be generalizable to other domains as well.

The performance of our BBBC is evaluated using three criteria: generality, consistency, and efficiency. *Generality* refers to the ability to working soundly across different datasets of the same problem as well as different problems. Meanwhile, c*onsistency* refers to the capacity in generating results that are stable when executed a number of times for each dataset. Consistency is generally among the most essential criteria in the evaluation of any algorithm because many search algorithms contain a stochastic component that requires different solutions over multiple runs albeit the same initial solution. Here, the consistency is grounded on the average and the standard deviation over 10 independent runs. *Efficiency* refers to the algorithm's capacity in generating good results that is almost similar to or superior to the best-known value documented in the literature. Our BBBC is measured by reporting, for every dataset, the Best and deviation percentage from the best-known results documented in the literature. For each tested dataset, a comparison was made between our BBBCand those of identical methods with regards to solution quality instead of computational time. This is because different computer resources employed has made comparison very challenging. As such, the number of iterations being the termination criteria from the usage of the adaptive memory in our BBBC was established, resulting in the execution time to be within the range of those documented in the literature.

6. CONCLUSION

In this study, we have demonstrated the generalization, and consistency of our BBBC metaheuristic in three different classes of COPS (e.g. QAP, BPP, and JSSP) using the same parameter settings. This was achieved by investigating and testing the impact of an elite pool and a local search on the general performance of a population-based metaheuristic. It utilizes an elite pool that contains a collection of diverse and high-quality solutions. This memory structure helps maintain a balance between diversity and quality of the search. For example, escaping local optima, i.e. the minima or maxima according to a problem's formulation, can be made through the employment of generating new solutions from those diverse ones in the elite pool. This way, the search might be diversified for new promising areas. In addition, the search can be converged toward better quality solutions by concentrating the search around good quality solutions from the elite pool. Results showed that our BBBC metaheuristic produces high-quality solutions, if not optimal, and its performance can be generalized well across the three problems. The study concluded that the hybridization of an elite pool and a local search within a population-based metaheuristic can enhance dramatically the performance of population-based metaheuristic that is generalized well across different problems and producing high-quality solutions which are either competitive or optimal in most cases. The analysis and discussion on the computational complexity is another proof that our BBBC metaheuristics efficient across three COPs. The main contributions of this work are as follows: The development of our BBBC that possesses elite pool and perform heuristic perturbations, demonstrating that strengths of different search algorithms can be merged into one hybrid methodology (e.g. constructive heuristics, population-based and local search metaheuristics), The hybridization of the elite pool with a population-based metaheuristic, and manages to obtain consistent results, generalized across different problem domains in addition to producing high-quality solutions that are either competitive or better than similar methods in most cases, The developed hybrid metaheuristic can be easily applied to different problem domains without much effort, e.g. the user only needs to change the constructive heuristics and neighborhood structures. In particular, the use of an elite pool provides a diversity of high-quality solutions from which our BBBC can start its search process for better solutions. Besides the local search, the elite pool also provides the means to implement cooperation and to achieve faster convergence, the diversity control strategy used in our BBBC has shown a sufficient capability to solve large-sized and hard datasets for three different problem domains. In future work, to examine other neighborhood structures e.g. the 2-flip moves, which would further enhance the exploitation of a solution space. We may also intend to investigate the effectiveness of our BBBC metaheuristic across other COPs and other domains such as deep machine learning.

COMPLIANCE WITH ETHICAL STANDARDS

Conflict of interest: The authors declare that they have no conflict of interest.

REFERENCES

[1] C. Blum and A. Roli, "Hybrid Metaheuristics: An Introduction, Studies in Computational Intelligence," In: C. Blum, M.J.B. Aguilera, A. Roli, M. Samples (Eds.), *Hybrid Metaheuristics: An Emerging Approach to Optimization, Springer-Verlag Berlin*, Heidelberg, SCI, vol. 114, pp. 1-30, 2008.

[2] S. Asta, and E. Ozcan, "A tensor analysis improved genetic algorithm for online bin packing," *proceedings of the 2015 on Genetic and Evolutionary Computation Conference*, ACM, New York, pp. 799–806, 2015.

[3] S. Tsutsui, "Cunning Ant System for Quadratic Assignment Problem with Local Search and Parallelization," A. Ghosh, R.K. De, S.K. Pal (Eds.): PReMI, *Springer-Verlag Berlin Heidelberg*, LNCS 4815, pp. 269–278, 2007.

[4] E. G. Talbi., *Metaheuristics: from design to implementation*, Wiley, USA, Ch. 1, 2009.

[5] Z. Gungr and A. Unler, "K-harmonic means data clustering with simulated annealing heuristic," *Applied Mathematics and Computation*, vol. 184, pp. 199–209, 2007.

[6] A. Misevicius, "An implementation of the iterated tabu search algorithm for the quadratic assignment problem," *OR Spectrum*, vol. 34, pp. 665-690, 2012.

[7] J. Brownlee, *Clever Algorithms: Nature-Inspired Programming Recipes*, First Edition, LuLu, Australia, 2011.

[8] F. Glover, *et al.*, "Scatter search," A. Ghosh, S. Tsutsui (Eds.), Theory and Applications of Evolutionary Computation: Recent Trends, *Springer*, pp. 519–529, 2002.

[9] G. M. Jaradat, and M. Ayob, "Effect of elite pool and Euclidean distance in Big Bang-Big Crunch metaheuristic for post-enrolment course timetabling problems," *Int. J. Soft Comput*, vol. 8(2), pp. 96–107, 2013.

[10] G. M. Jaradat, *et al.*, "The effect of elite pool in hybrid population-based meta-heuristics for solving combinatorial optimization problems," *Appl. Soft Comput*, vol. 44, pp. 45-56, 2016.

[11] M. Genc and K. Hocaoglu, "Bearing-Only Target Tracking Based on Big Bang-Big Crunch Algorithm," *The Proceedings of the 3rd International Multi-Conference on Computing in the Global Information Technology ICCGI '08*, vol. 53, pp. 229 – 233, 2008.

[12] T. C. Koopmans and M. J. Beckmann, "Assignment problems and the location of economics activities," *Econometrica*, vol. 25, pp. 53–76, 1957.

[13] E. Burkard, *et al.*, "The official website of the QAPLIB - a quadratic assignment problem library," 2004. [Online], Available: http://www.seas.upenn.edu/qaplib/, [Accessed in 30 July 2019].

[14] A. Puris, *et al.*, "Analysis of the efficiency of a Two-Stage methodology for ant colony optimization: Case of study with TSP and QAP," *Journal of Expert Systems with Applications, Elsevier*, vol. 37, pp. 5443-5453, 2010.

[15] S. Martello and P. Toth, "Lower bounds and reduction procedures for the bin packingproblem," *Discrete Applied Mathematics*, vol. 28, pp. 59–70, 1990.

[16] E. Falkenauer, "A hybrid grouping genetic algorithm for bin packing," *Journal of Heuristics*, vol. 2, pp. 5–30, 1996.

[17] A. Scholl, *et al.*, "BISON: A fast hybrid procedure for exactly solvingthe one-dimensional bin packing problem," *Computers and Operations Research*, vol. 24, pp. 627–645, 1997.

[18] M. L. Pinedo, "Planning and Scheduling in Manufacturing and Services," *Springer*, New York, 2002.

[19] L. Wang, *et al.*, "Flexible Job Shop Scheduling Problem Using an Improved Ant Colony Optimization," *Scientific Programming, Hindawi*, vol. 2017, pp. 1-11, 2017.

[20] A. N. H. Zaied and L. A. E. Shawky, "A Survey of the Quadratic Assignment Problem," *International Journal of Computer Applications*, vol. 101(6), pp. 28-36, 2014.

[21] K. Bhati and A. Rasool, "Quadratic Assignment Problem and its Relevance to the Real World: A Survey," *Int. J. of Computer Applications*, vol. 96(9), pp. 42-47, 2014.

[22] H. M. Nehi and S. Gelareh, "A Survey of Meta-Heuristic Solution Methods for the Quadratic Assignment Problem," *Applied Mathematical Sciences*, vol. 1(46), pp. 2293-2312, 2007.

[23] V. Maniezzo, *et al.*, "Algodesk: An experimental comparison of eight evolutionary heuristics applied to the Quadratic Assignment Problem," *Theory and Methodology, European Journal of Operational Research, Elsevier Science B.V*, vol. 81, pp. 188-204, 1995.

[24] G. R. Mateus, *et al.*, "GRASP with Path-Relinking for the Generalized Quadratic Assignment Problem," *AT&T Labs, Research Technical Report*, pp. 1-35, 2009.

[25] F. Glover, *et al.*, "Diversification-driven tabu search for unconstrained binary quadratic problems," *4OR-A Quarterly Journal of Operations Research*, 2010.

[26] N. P. Hoare and J. E. Beasley, "Placing boxes on shelves: a case study," *Journal of the Operational Research Society*, vol. 52(6), pp. 605-614, 2001.

[27] C. O. Lopez and J. E. Beasley, "A formulation space search heuristic for packing unequal circles in a fixed size circular container," *European Journal of Operational Research*, vol. 251(1), pp. 64-73, 2016.

[28] C. O. Lopez and J. E. Beasley., "Packing unequal rectangles and squares in a fixed size circular container using formulation space search," *Computers & Operations Research*, vol. 94, pp. 106-117, 2018.

[29] C. O. Lopez and J. E. Beasley., "Packing a fixed number of identical circles in a circular container with circular prohibited areas," *Journal of Optimization Letters, Springer-Verlag Berlin*, Heidelberg, vol. 13, pp. 1449, 2019.

[30] T. Stützle., "Iterated local search for the quadratic assignment problem," *European Journal of Operational Research*, vol. 174, pp. 1519-1539, 2006.

[31] F. Ducatelle and J. Levine, "Ant colony optimisation for bin packing and cutting stock problems," *UK Workshop on Computational Intelligence (UKCI-01)*, Edinburgh, 2001.

[32] F. Ducatelle and J. Levine, "Ant colony optimisation and local search for bin packing and cutting stock problems," *Journal of the Operational Research Society*, vol. 55(7), pp. 705-716, 2004

[33] S. Asta, *et al*., "CHAMP: Creating heuristics via many parameters for online bin packing," *Expert Syst. Appl.*, vol. 63, pp. 208-221, 2016.

[34] E. Ozcan, *et al*., "A comprehensive survey of hyperheuristics," *Intelligent Data Analysis*, vol. 12, pp. 3-23, 2008.

[35] N. Pillay., "A study of evolutionary algorithm selection hyper-heuristics for the one-dimensional bin-packing problem," *South African Computer Journal*, vol. 48, pp. 31–40, 2012.

[36] K. Burke, *et al*., "Hyper-heuristics," *J. Oper Res Soc*, vol. 64, pp. 1695–1724, 2013.

[37] E. Lopez-Camacho, *et al*., "A unified hyper-heuristic framework for solving bin packing problems," *Expert Systems with Applications*, vol. 41, pp. 6876–6889, 2014.

[38] Jr. Coffman, *et al*., "Bin packing approximation algorithms: Survey and classification," Pardalos, P.M., Du, D.Z., Graham, R.L. (Eds.), *Handbook of Combinatorial Optimization. Springer*, New York, pp. 455–531, 2013.

[39] A. Alvim, *et al*., "A Hybrid Improvement Heuristic for the One-Dimensional Bin Packing Problem," *Journal of Heuristics*, vol. 10, pp. 205-229, 2004.

[40] V. Hemmelmayr, *et al*., "Variable neighbourhood search for the variable sized bin packing problem," *Computers & Operations Research*, vol. 39, pp. 1097– 1108, 2012.

[41] M. Saravanan and A. NoorulHaq, "A scattersearch algorithm for scheduling optimisation of job shop problems," *International Journal of Product Developmen*t, vol. 10(1/2/3), pp. 259-272, 2010.

[42] S. Chen, *et al*., "A Hybrid Electromagnetism-Like Algorithm for Single Machine Scheduling Problem," *Lecture Notes in Computer Science*, vol. 4682, pp. 543-552, 2007.

[43] M. Eddaly, *et al*., "Combinatorial particle swarm optimization forsolving blocking flowshop scheduling problem," *J. Comput. Des, Eng*, vol. 3, pp. 295–311, 2016.

[44] V. Sels, *et al*., "A hybrid job shop procedure for a Belgian manufacturing company producing industrial wheels and castors in rubber," Technical Report, Ghent University, 2011.

[45] G. Zobolas, *et al*., "Exact, Heuristic and Meta-heuristic Algorithms for Solving Shop Scheduling Problems," *Metaheuristics for Scheduling in Industrial and Manufacturing Applications*, pp. 1-40, 2008.

[46] P. Greistorfer and S. Voss, "Controlled pool maintenance in combinatorial optimization," C. Rego, B. Alidaee (Eds.), *Conference on Adaptive Memory and Evolution: Tabu Search and Scatter Search, Chapter 18*, University of Mississippi, Kluwer Academic Publishers, pp. 387–424, 2005

[47] Y. Rochat and E. Taillard., "Probabilistic diversification and intensification in local search for vehicle routing," *J. Heuristics*, vol. 1, pp. 147–167, 1995.

[48] W. Y. Szeto, *et al*., "An artificial bee colony algorithm for the capacitated vehicle routing problem," *Production, manufacturing and logistics, Eur. J. Oper. Res*, vol. 215, pp. 126–135, 2011.

[49] D. Mester and O. Bräysy., "Active-guided evolution strategies for large-scale capacitated vehicle routing problems," *Comput.Operat. Res*, vol. 34, pp. 2964-2975, 2007.

[50] R. Bai *et al*., "A simulated annealing hyper-heuristic for university course timetabling," E.K. Burke, H. Rudova (Eds.): *PATAT 2006*, pp. 345-350, 2006.

[51] M. Bashiri and H. Karimi, "Effective heuristics and meta-heuristics for the quadratic assignment problem with tuned parameters and analytical comparisons," *Journal of Industrial Engineering International*, vol. 8, 2012.

[52] C. Blum, *et al*., "Hybrid Metaheuristics: An Emerging Approach to Optimization," *Studies in Computational Intelligence Springer*, Heidelberg, vol. 114, 2008.

[53] G. M. Jaradat, and M. Ayob, "Big Bang-Big Crunch Optimization Algorithm to Solve the Course Timetabling Problem," 10th *International Conference on Intelligent Systems Design and Applications (ISDA)*, Cairo, pp. 1448-1452, 2010.

[54] K. Erol and I. Eksin, "A new optimization method: Big Bang–Big Crunch," *Advances in Engineering Software, Elsevie*r, vol. 37, pp. 106–111, 2006.

[55] G. Zäpfel, "Metaheuristic Search Concepts: A Tutorial with Applications to Production and Logistics," *Springer-Verlag Berlin Heidelberg (Chapter 9)*, pp. 159-235, 2010.

[56] A. Kaveh and S. Talatahari, "Size optimization of space trusses using Big Bang-Big Crunch algorithm," *Computers and Structures*, vol. 87(17-18), pp. 1129-1140, 2009.

[57] M. Kripka and M. L. Kripka, "Big Crunch Optimization Method," *Eng Opt 2008-International Conference on Engineering Optimization Rio de Janeiro*, 2008.

[58] A. Hatamlou, *et al*., "Data clustering using big bang–big crunch algorithm," *Communications in Computer and Information Science*, pp. 383–388, 2011.

[59] I. Al-Marashdeh, *et al*., "An Elite Pool-Based Big Bang Big Crunch Metaheuristic for Data Clustering," *Journal of Computer Science, Science Publications*, vol. 14(I12), pp. 1611-1626, 2018.

[60] T. Bui, *et al*., "Tackling Dynamic Problems with Multi objective Evolutionary Algorithms," *Multi objective Problem Solving from Nature: From Concepts to Applications*, Joshua Knowles, David Corne, and Kalyanmoy Deb (Eds), *Springer*, pp. 77-92, 2007.

[61] M. G. C. Resende, *et al*., "Scatter search and path-relinking: fundamentals, advances and applications," M. Gendreau, J.-Y. Potvin (Eds.), *the Handbook of Metaheuristics, 2nd ed., Springer*, 2010.

[62] A. Misevicius., "Restart-based genetic algorithm for the quadratic assignment problem," *Research and Development in Intelligent Systems, Proceedings of AI-2008, the Twenty-eighth SGAI International Conference on Innovative Techniques and Applications of Artificial Intelligence, London: Springer*, pp. 91-104, 2008.

[63] S. Ramkumar, *et al*., "A population-based hybrid ant system for quadratic assignment formulations in facility layout design," *Int J AdvManufTechnol*, vol. 44, pp. 548–558, 2009.

[64] C. Mumford., "An order based memetic evolutionary algorithm for setpartitioning problems," *J. Fulcher & L. C. Jain (Eds.), Computational intelligence: a compendium. Studies in computational intelligence, Springer*, vol. 115, pp. 881–925, 2008.

[65] R. Yesodha and T. Amudha, "A Comparative Study on Heuristic Procedures To Solve Bin Packing Problems," *International Journal in Foundation of Computer Science & Technology (IJFCST)*, vol. 2, pp. 37-49, 2012

[66] A. Layeb and S. Chenche, "A Novel GRASP Algorithm for Solving the Bin Packing Problem," *International Journal of Information Engineering and Electronic Business*, vol. 4, pp. 8-14, 2012.

[67] F. Pezzella and E. Merelli., "A tabu search method guided by shifting bottleneck for the job shop scheduling problem," *Eur. J. Operational Res*, vol. 120(2), pp. 297–310, 2000.

[68] E. Nababan, *et al*., "Branch and bound algorithm in optimizing job shop scheduling problems," *2008 International Symposium on Information Technolog*, pp. 1-5, 2008.

[69] P. Eswaramurthy and A. Tamilarasi, "Hybridizing tabu search with ant colony optimization for solving job shop scheduling problems," *International Journal of Advanced Manufacturing Technology*, vol. 40, pp. 1004-1015, 2009.

APPENDIX

Table 5. Results of our BBBC on QAP compared to some metaheuristics

Instance	Size	Best known	Our BBBC				Orig. BBBC	RGA	HAS	ITS	IFLS	*p*-value
	NxM		*Best*	Δ(%)	*Std.*	*Time*						
Rou12	12	235528*	235528	0	0	1.24	240124	*N/A*	235528	*N/A*	*N/A*	+
Rou15	15	354210*	354210	0	0	1.59	354210	*N/A*	354210	*N/A*	*N/A*	~
Rou20	20	725522*	725522	0	0.03	2.61	735358	*N/A*	725522	*N/A*	*N/A*	+
Tai12a	12	224416*	224416	0	0	1.28	230704	*N/A*	224416	*N/A*	*N/A*	+
Tai15a	15	388214*	388214	0	0	1.26	391522	*N/A*	388250	*N/A*	*N/A*	+
Tai17a	17	491812*	491812	0	0	1.25	502966	*N/A*	494550	*N/A*	*N/A*	+
Tai20a	20	703482*	703482	0	0.05	1.58	716772	703482	709080	703482	*N/A*	+
Tai25a	25	1167256*	1167256	0	0.13	7.52	1187418	1167256	1185587	1167256	*N/A*	+
Tai30a	30	1818146	1818146	0	0.38	11.69	1863880	1818146	1842986	1818146	1818146	+
Tai35a	35	2422002	2422002	0	0.21	3.23	2495202	2422002	2453578	2422002	*N/A*	~
Tai40a	40	3139370	3139370	0	0.36	49.21	3212950	3139370	3192668	3139370	3139370	+
Tai50a	50	4938796	4938796	0	0.32	97.16	5064316	4938796	5061236	4938796	4938796	+
Tai60a	60	7205962	7205962	0	0.35	924.92	7391598	7205962	7377002	7205962	*N/A*	+
Tai80a	80	13499184	13499184	0	0.27	1147.24	13890458	13515450	13848650	13515450	13527910	+
Tai100a	100	21044752	21044752	0	0.19	3098.55	21054656	21054656	*N/A*	21054656	21090402	+

Note: * Optimal solution
N: number of facilities, *C*: number of locations.
RGA: by [62]; HAS: hybrid ant system by [63]; ITS: iterated tabu search by [6]; IFLS: iterated fast local search by Bashiri & Karimi (2012).

Table 6. Results of our BBBC on BPP compared to some metaheuristics

Instance	Size	Best Known	Our BBBC				Orig. BBBC	GSA	IG	FFD	BF	GRASP	*p*-value
Sch_Set3	*N x C*		*Best*	Δ(%)	*Std.*	*Time*							
hard0	200x10^5	56	56	0	0	1.02	59	56	59	59	60	*N/A*	+
hard1	200x10^5	57	57	0	0.01	1.18	58	57	57	60	60	*N/A*	+
hard2	200x10^5	56	56	0	0	1.19	59	57	59	60	59	59	+
hard3	200x10^5	55	55	0	0	1.04	57	56	57	59	60	*N/A*	+
hard4	200x10^5	57	57	0	0.02	1.33	59	57	58	60	59	*N/A*	+
hard5	200x10^5	56	56	0	0	1.29	59	56	57	59	60	*N/A*	+
hard6	200x10^5	57	57	0	0.07	1.69	59	57	57	60	59	59	+
hard7	200x10^5	55	55	0	0	1.83	58	55	58	59	60	58	+
hard8	200x10^5	57	57	0	0.11	2.67	59	57	58	60	60	59	+
hard9	200x10^5	56	56	0	0.01	2.79	60	56	58	60	59	58	+

Note: * Optimal solution
N: number of items, *C*: bin capacity.
GSA: genetic simulated annealing by [64]; IG: iterated greedy by [64]; FFD: first fit decreasing weight by [64]; BF: best fit by [65]; GRASP: greedy randomized adaptive search by [66].

Table 7. Results of our BBBC on JSSP compared to some metaheuristics

Instance	Size	Best known	Our BBBC				Orig. BBBC	TSSB	BB	HTS-ACO	ACOFT-TR	*p*-value
	J x *M*		*Best*	Δ(%)	*Std.*	*Time*						
Tai51	50x15	2760	2760	0	0	0.02	2779	2760	2779	2760	*N/A*	+
Tai52	50x15	2756	2756	0	0	0.01	2800	2756	2779	2756	*N/A*	+
Tai53	50x15	2717	2717	0	0	0.09	2768	2717	2772	2717	*N/A*	+
Tai54	50x15	2797	2797	0	0.01	0.78	3133	2839	3133	2839	*N/A*	+
Tai55	50x15	2679	2679	0	0	1.22	2757	2684	2716	2683	*N/A*	+
Tai56	50x15	2781	2781	0	0	0.49	2948	2781	2948	2781	*N/A*	+
Tai57	50x15	2943	2943	0	0.06	1.33	3086	2943	3086	2943	*N/A*	+
Tai58	50x15	2885	2885	0	0.02	1.08	2928	2885	2912	2885	*N/A*	+
Tai59	50x15	2655	2655	0	0	0.95	2744	2655	2744	2655	*N/A*	+
Tai60	50x15	2723	2723	0	0	1.79	2862	2723	2862	2723	*N/A*	+
Tai61	50x20	2868	2868	0	0.13	0.83	2928	2868	2928	2868	*N/A*	+
Tai62	50x20	2848	2848	0	0.11	1.60	3013	*N/A*	2930	*N/A*	2883	+
Tai63	50x20	2755	2755	0	0	2.81	2899	2755	2899	2755	*N/A*	+
Tai64	50x20	2691	2691	0	0	1.33	2786	2702	2786	2702	*N/A*	+
Tai65	50x20	2725	2725	0	0	1.85	2905	2725	2905	2725	*N/A*	+
Tai66	50x20	2845	2845	0	0.25	1.84	2857	2845	2857	2845	*N/A*	+
Tai67	50x20	2812	2821	0	0.14	1.85	2921	*N/A*	2961	*N/A*	2825	+
Tai68	50x20	2764	2764	0	0	3.15	2907	2784	2869	2784	*N/A*	+
Tai69	50x20	3063	3063	0	0.04	3.52	3120	3071	3120	3071	*N/A*	+
Tai70	50x20	2995	2995	0	0.12	3.65	3143	2995	3143	2995	*N/A*	+
Tai71	100x20	5464	5464	0	0.41	3.58	5582	5464	*N/A*	5464	*N/A*	+
Tai72	100x20	5181	5181	0	0.29	1.82	5231	5181	*N/A*	5181	*N/A*	+
Tai73	100x20	5552	5552	0	0.38	2.93	5618	5568	*N/A*	5568	*N/A*	+
Tai74	100x20	5339	5339	0	0.16	2.28	5389	5339	*N/A*	5339	*N/A*	+
Tai75	100x20	5392	5392	0	0.12	3.21	5438	5392	*N/A*	5392	*N/A*	+
Tai76	100x20	5342	5342	0	0.47	4.43	5403	5342	*N/A*	5342	*N/A*	+
Tai77	100x20	5436	5436	0	0.56	4.68	5486	5436	*N/A*	5436	*N/A*	+
Tai78	100x20	5394	5394	0	0.51	5.18	5459	5394	*N/A*	5394	*N/A*	+
Tai79	100x20	5358	5358	0	0.72	5.67	5608	5358	*N/A*	5358	*N/A*	+
Tai80	100x20	5183	5183	0	0.44	1.23	5278	5183	*N/A*	5183	*N/A*	+

Note: * Optimal solution

J: number of jobs, *M*: number of machines.

TSSB: tabu search guided by shifting bottleneck by [65]; BB: branch and bound by [67]; HTS-ACO: hybrid ant system with tabu search by [68]; ACOFT-TR: ant colony optimization with tabu search by [69].