The Effect of Elite Pool in Hybrid Population-based Meta-heuristics for Solving Combinatorial Optimization Problems

Article *in* Applied Soft Computing · April 2016

DOI: 10.1016/j.asoc.2016.01.002

Ghaith M. Jaradat
Jerash University
62 PUBLICATIONS **98** CITATIONS
SEE PROFILE

Masri Ayob
Universiti Kebangsaan Malaysia
122 PUBLICATIONS **1,016** CITATIONS
SEE PROFILE

Ibrahim Almarashdeh
Universiti Kebangsaan Malaysia
36 PUBLICATIONS **202** CITATIONS
SEE PROFILE

Applied Soft Computing 44 (2016) 45–56

Contents lists available at ScienceDirect

Applied Soft Computing

journal homepage: www.elsevier.com/locate/asoc

The effect of elite pool in hybrid population-based meta-heuristics for solving combinatorial optimization problems

Ghaith Jaradat[a,*], Masri Ayob[b], Ibrahim AlMarashdeh[c]

[a] Department of Computer Science, Faculty of Information Technology, Jerash University, 26150-311 Jerash, Jordan
[b] Data Mining and Optimization Research Group, Centre of Artificial Intelligence Technology, Faculty of Information Science and Technology, The National University of Malaysia, UKM, 43600 B. B. Bangi, Selangor, Malaysia
[c] School of Information Technology, Faculty of Information Science and Technology, The National University of Malaysia, UKM, 43600 B. B. Bangi, Selangor, Malaysia

ARTICLE INFO

Article history:
Received 17 January 2015
Received in revised form 23 October 2015
Accepted 3 January 2016
Available online 1 April 2016

Keywords:
Hybrid meta-heuristics
Elite pool
Elitist-Ant System
Big Bang-Big Crunch
Scatter search
Combinatorial optimization problems

ABSTRACT

This work investigates the effect of elite pool that has high-quality and diverse solutions in three hybrid population-based meta-heuristics with an elite pool of a hybrid Elitist-Ant System, a hybrid Big Bang-Big Crunch optimization, and a hybrid scatter search. The purpose of incorporating an elite pool in population-based meta-heuristics is to maintain the diversity of the search while exploiting the solution space as in the reference set of the scatter search. This may guarantee the effectiveness and efficiency of the search, which could enhance the performance of the algorithms and generalized well across different datasets. To test the generality of these meta-heuristics via their consistency and efficiency, we use three classes of well-known combinatorial optimization problems as follows: symmetric traveling salesman problem, 0–1 multidimensional knapsack problem, and capacitated vehicle routing problem. Experimental results showed that the performance of our hybrid population-based meta-heuristics, compared to the best known results, is competitive in many instances. This finding indicates the effectiveness of utilizing an elite pool in our hybrid meta-heuristics in diversifying the search and subsequently enhances their performance over different instances and problems.

1. Introduction

The continuing research on building a successful generic problem solver offers the prospect of creating artificial intelligent systems that can generate practical solutions to many combinatorial optimization problems (COPs). Seeking for computational effectiveness and efficiency, research in meta-heuristics has evolved rapidly in an attempt to find good quality solutions to these problems within a desired time frame [1]. A meta-heuristic is a high-level strategy which guides other problem-specific heuristics to search for solutions in a possibly wide set of problem domains [1]. Meta-heuristics are usually effective and flexible; they are increasingly gaining popularity [1]. There are a wide variety of meta-heuristics that have been introduced. They are classified as single solution vs. population-based approaches. Single solution approaches focus on modifying and improving a single candidate solution; e.g. simulated annealing, iterated local search, and variable neighborhood search [2]. On the other hand, population-based approaches maintain and improve multiple candidate solutions (population of solutions), e.g. evolutionary computation, genetic algorithms, and particle swarm optimization [2]. Talbi [3] illustrated another class of meta-heuristics, that is swarm intelligence, which is a collective behavior of decentralized, self-organized agents in a population or swarm; e.g. ant colony optimization, particle swarm optimization, and artificial bee colony.

The population-based method is used because of its capability to explore search space and can be easily combined with local search method in order to enhance solution exploitation mechanism [4]. On the other hand, some common local search methods that have been applied to COPs are tabu search, simulated annealing, and iterated local search. The local search method is utilized because of its capability to exploit solution space [2].

The strength of population-based method is relied on the capability of recombining solutions to obtain new ones [2]. In population-based algorithms such as genetic algorithm, memetic (hybrid genetic) algorithm and scatter search, a structured solution recombination of elite solutions is performed explicitly (which

* Corresponding author. Tel.: +962 796888035.
E-mail addresses: ghaith_jaradat@yahoo.com (G. Jaradat), masri@ukm.edu.my (M. Ayob), ibramars@gmail.com (I. AlMarashdeh).

http://dx.doi.org/10.1016/j.asoc.2016.01.002
1568-4946/© 2016 Published by Elsevier B.V.

involve moving or swapping assignments or permutations in a solution, representing the information exchange between generations) using one or more recombination operators, such as crossover and mutation [2]. The term explicit means that a solution is represented directly by the actual assignment/permutation/allocation and fitness values of solutions. Another kind of recombination is the one performed implicitly, where new solutions are generated using a distribution over the search space which is a function of earlier populations representing the search experience [2]. Ant systems and Big Bang-Big Crunch use implicit recombination. The term implicit means that a solution is represented indirectly by the fitness values of the assignments or values of their contribution to search (e.g. constructing a solution). This enables the search process to perform a guided sampling of the search space.

However, in general, the intensification mechanism in a population-based method still needs to be improved in order to obtain high-quality solutions. Hence, in order to enhance the intensification process, a meta-heuristic that has a capability in exploiting the solution space (e.g. hill climbing) is usually hybridized with population-based method. Many studies have recommended this hybridization, such as [2,3] and [5–7]. Local search method is a meta-heuristic that has the capability in exploiting the solution space that can be complemented with population-based approach to overcome the weakness in the population-based method and to further enhance the quality of solution in the pool.

Generally, we can define the term elite pool as an – adaptive – memory structure with a set of diverse and high-quality solutions that stores useful information about the global optima in the form of a diverse and elite set of solutions. This structure gives the search process the ability to recombine samples from the elite set, so as to exploit useful information about the global optima.

In addition, the utilization of an elite pool or reference set (that has high-quality and diverse solutions), to control the diversity of search, and a dynamic manipulation of the population size are also recommended for better performance of hybrid meta-heuristics [4]. A good performance (w.r.t. consistency, efficiency, effectiveness, and may be generality) is presented by maintaining a balance between diversification and intensification of the search [8]. Therefore, we have chosen the scatter search (SS), Elitist-Ant System (Elitist-AS), and Big Bang Big Crunch (BB-BC) for this study that had been hybridized (in our previous studies) with some diversification and intensification mechanisms to enhance their exploration and exploitation of the solution space. This has been achieved (in our previous works) by incorporating an elite pool (containing both diverse and high-quality solutions) and employing a local search heuristic to intensify the search around elite solutions, while a degree of diversity is maintained. These were applied to the course timetabling problems (see [9–11]). The reason of choosing those three meta-heuristics (Elitist-AS, BB-BC, and SS) are [8,12,13]:

- SS provides a deterministic selection of reference set or pool of elite solutions in terms of quality and diversity. This performs a systematic neighborhood search in the Euclidean or Hamming spaces.
- SS and BB-BC have structured solution combinations using diversification strategies that do not merely rely on randomization.
- SS evolves a strategy of updating in a form of exploiting an adaptive memory to preserve good quality and diversity.
- Elitist-AS, SS, and BB-BC provide useful information about the collection of elite or diverse solutions. However, Elitist-AS and BB-BC do not originally possess elite pool of high-quality and diverse solutions.
- SS and BB-BC support direct solution representation in a Hamming or Euclidean space that is easy to manipulate.
- BB-BC has an elitism strategy and a fast convergence toward high-quality or diverse solutions. However, this process needs to be controlled in order to maintain a balance between diversity and quality of the search.
- BB-BC uses the Euclidean distance to measure similarities between solutions which help to point elite solutions (high-quality and diverse solution).
- Elitist-AS has an elitism strategy with a memory (pool) of diverse solutions. It is sufficient for exploring and exploiting the solution space but still insufficient to balance between diversification and intensification in the search.

In addition, we have chosen both Elitist-AS and BB-BC to experiment the effect of employing an elite pool alongside their implicit solution recombination. That is, we also aim to compare both meta-heuristics against the SS as an explicit recombination-based meta-heuristic. Therefore, this study aims to investigate the effect of elite pool to the performance of the population approaches in terms of their generality in solving a variety of COPs. Based on the utilization of an elite pool, the study investigates the performance (consistency, efficiency, effectiveness and generality) of the three hybrid meta-heuristics by testing them on three classes of NP-hard COPs. The three COPs are: (i) packing problems: the 0–1 multidimensional knapsack problem; (ii) permutation optimization problems: symmetric traveling salesman problem; and (iii) routing problems: Capacitated Vehicle Routing problem.

In this study, we have intentionally fixed the size of the memory structures in our hybrid meta-heuristics and maintained the same update strategy. We have also compared them to similar hybrid approaches and standalone methods such as genetic algorithm, particle swarm optimization and ant systems. In addition, it is worth mentioning that some studies that are similar to our study give us confidence about investigating the effect of elite pools. These studies include [14] who applied a learning procedure (based on gene-expression programming hyper-heuristic) to a number of COPs, including vehicle routing problems. Other examples include [15–17]. Those studies were applied to the course timetabling problems but their interesting contributions are related to the effect of permutations in the first place which might somehow affect the performance or even the importance of an elite pool. As mentioned above, we are motivated, thus, to investigate the effect of the elite pool in hybrid population-based meta-heuristics.

Therefore, our research question boils down to the following: "*Does the use of elite pool (a pool of diverse and high-quality solutions) in a population-based meta-heuristic enhance the performance of a meta-heuristic when compared to the one that only uses the diverse pool?*" Therefore, our objectives include the following:

1. Propose a population-based meta-heuristic by incorporating a memory structure (e.g. elite pool) which contains a set of diverse and high-quality solutions to strike a balance between diversification and intensification – exploration and exploitation – within the search space; and
2. Test the performance, i.e. generality and consistency, of the proposed hybrid population-based metaheuristic over three different COP domains, that are of a very different nature. We also compare the results with the state-of-the-art population-based meta-heuristics.

The paper is organized as follows: Section 2 describes the problems chosen for the study. Some related works are presented in Section 3. The proposed hybrid meta-heuristics and their design are presented in Section 4. Section 5 shows and discusses experimental results obtained by the proposed hybrid metaheuristics. Finally, the conclusions are presented in Section 6.

2. Problems description

Before describing the problems, the reason behind choosing the above mentioned three COPs (although they have been intensively studied in the literature) is that, they are very common and are derived from real world applications. Hence, the three COPs provide a very good platform for testing the impact of an elite pool on the performance and generality (consistency and efficiency) of our hybrid population-based meta-heuristics.

2.1. Symmetric traveling salesman problem

The symmetric traveling salesman problem (STSP) is an important optimization problem that represents many fields of real world situations such as those found in the areas of transportation, logistics and semiconductor industries [18]. TSP is a very popular NP-hard COP [19]. The basic principle of TSP is that the salesman starts travel from a city to his tour and returns to the same city while trying to obtain a closed tour with a minimum cost. When he takes his tour, the salesman must visit every city once. The cost of his tour directly depends on its length.

Given a set of cities and their positions (pairwise distances), the aim in its classic form is to find the shortest path where each city is visited only once and make sure that the path ends at the starting city. The set of initial solutions is created by randomly generating permutation sequences. The quality of solution is represented by the total distance of the route.

In this study, we adopt the concept of calculating the distances between cities from [20], where the distance between city i and city $(i+1)$ is calculated as the Euclidean distance using the following equation:

$$d(T[i], T[i+1]) = \sqrt{(x_i - x_{i+1})^2 + (y_i - y_{i+1})^2} \tag{1}$$

The aim of solving TSP is to minimize the total closed tour length (see [21] for detail formulation). In this study, our hybrid meta-heuristics generate populations of initial solutions using Nearest Neighbor constructive heuristic as in [22], and the distances between cities are measured by integer numbers as in [20]. The instances used in this work are taken from the TSPLIB benchmark library that was introduced by [21,23]. These instances have been used in many studies and some were extracted from practical applications of the TSP.

2.2. 0–1 multidimensional knapsack problem

A Knapsack Problem (KP) requires a subset of some given items to be chosen such that the corresponding sum of profit is maximized without exceeding the capacity of the knapsack(s). There are several types of KP, depending on the distribution of the items and knapsacks such as: Single KP (e.g. 0–1 KP and Bounded KP) and Multiple KP (e.g. 0–1 MKP and Bin-Packing problem). In the Single KP there is one container (or knapsack) must be filled with an optimal subset of items without exceeding the capacity of the container. In the MKP, more than one container is available.

Each item in the 0-1KP may be chosen at most once, while in the Bounded KP we have a bounded amount of each item type. The Multiple-choice KP occurs when the items should be chosen from disjoint classes and, if several knapsacks are to be filled simultaneously, then it becomes a Multiple KP. In this study, we will consider the most commonly studied KP, which is the 0–1 Multidimensional KP (0–1 MKP). The 0–1 MKP can be described as follows: The problem consists of a set of items j, each with a weight w_j and a value v_j. The items must be packed into one knapsack with capacity c. Not all of the pieces can fit into the knapsack, so the objective is to maximize the value of the pieces chosen to be packed. Refer to [24] for detailed formulation.

2.3. Capacitated vehicle routing problem

The vehicle routing problem is a well-known and challenging COP [25]. With a set of customers associated with demand and serving time, and a fleet of vehicles with a maximum capacity, the aim is to design a least-cost set of routes to serve all customers, provided that each vehicle starts and ends at the depot, the total demand of each route does not exceed the vehicle capacity, and each customer is visited only once and by one vehicle only during its time window(s). The set of initial solutions are generated as follows: first create an empty route, then loop through all customers and add any one to the current route that does not violate the designated constraints. If no customer can be added to the current route, create a new one. The process is repeated until all customers have been assigned to a route. The quality of solution represents the total travel distance.

The capacitated vehicle routing problem (CVRP) is a variant of the classical and static VRP. The CVRP can be defined as the process of designing a least-cost set of routes to serve a set of customers (see [25] for detailed formulation). A fixed fleet of delivery vehicles of uniform capacity must service a set of known customer demands for a single commodity from a common depot. A solution is feasible if the total quantity assigned to each route does not exceed the capacity of the vehicle which services the route.

The goal is to find a feasible set of routes that do not violate any hard constraints and minimize the travel distance as much as possible. The hard constraints that must be satisfied are [25]: (i) each vehicle starts and ends its route at the depot; (ii) the total demand of each route does not exceed the vehicle capacity; (iii) each customer is visited only once by exactly one vehicle; and (iv) the duration of each route does not exceed a global upper bound. The quality of the generated solution is represented as the total traveling distance. Refer to [25] for more details.

3. Related work

A large variety of methodologies were applied to different classes of TSPs, KPs and VRPs. Most common and interesting methodologies are presented in the following subsections. While a variety of heuristics and meta-heuristics has been widely used and successfully applied to solve the STSP, 0–1 MKP and CVRP for several years, the Elitist-AS and BB-BC have not yet been applied to solve the those COPs. The SS was applied to solve the STSP and 0–1 MKP but not the CVRP.

3.1. STSP

Some examples of meta-heuristics and hybrid approaches applied on the STSP include combinatorial an artificial bee colony [20], an ant system with a cooperating agents strategy [26], a genetic algorithm with a generalized chromosome strategy [27], a scatter search with a particle filter strategy [28]; a particle swarm optimization multiple phase neighborhood search and greedy randomized adaptive search [7], an ant-based hyper-heuristic [29] and a hybrid (2-stage) ant colony optimization [30].

3.2. 0–1 MKP

Recently, a variety of exact meta-heuristics hybrid approaches and hyper-heuristics have been proposed in the literature to solve the 0–1 MKP, including a scatter search with a surrogate relaxation

[31], genetic programming and hyper heuristics [32], and a double genetic algorithm with a flipping local search [33].

3.3. CVRP

As the CVRP is an NP-hard problem, only instances of small sizes can be solved to optimality using exact solution methods (e.g. [25]), yet this might not even be possible if it requires to use limited amount of computing time. As a result of this, heuristic methods are used to find good, but not necessarily guaranteed optimal solutions using reasonable amount of computing time. During the 25 years, an increasing number of publications on heuristic approaches have been developed to tackle the CVRP. Some examples of meta-heuristics and hybrid approaches applied on the CVRP include Mathematical programming model [38], tabu search techniques [39], a genetic algorithm with multi-parent insertion crossover [40], a hyper-heuristic with a dynamic multi-armed bandit-gene expression programming [14], a hyper-heuristic with grammatical evolution as an online solver [41], a path-relinking algorithm with tabu search [42], an artificial bee colony algorithm [43], parallel iterative search methods [34], an evolutionary algorithm with active-guided search strategies [35], and a probabilistic population-based method with diversification and intensification methods in local search [36].

3.4. Elite pool

From the above brief review of different methods, it is concluded that many attempts have been made to solve the three COPs by combining different approaches (hybridization). Two main properties of all the above methods in the literature can be summarized as follows: (i) First, use a heuristic method to obtain a feasible initial solution; (ii) second, hybridize the meta-heuristic method with another heuristic method to improve the solution during the iteration process. Those works have shown significant improvements to the three COPs' optimal solutions with the implementation of, mainly, population-based hybridization. For example, a combination of population-based methods with multiple phase neighborhood search, or greedy randomized adaptive search, or local search, or heuristics to select or to generate heuristics, have been effectively solved the three COPs. The purpose of this hybridization is for expanding the neighborhood strategy in the population-based method.

In addition, we believe that an adaptive memory structure is a major component of an efficient and effective hybrid meta-heuristic (e.g. scatter search and tabu search algorithms). It stresses out the concepts of memory, intensification – exploitation, and diversification – exploration. A memory stands for the information collected by the algorithm on the objective function distribution. It can be represented as complex structures, like pheromone trails in the Elitist-AS. Intensification exploits the information obtained, in order to improve the current solutions. This is typically a local search routine, while diversification aims at collecting new information, by exploring the search space.

The three components presented (e.g. memory, intensification and diversification) are not always clearly distinct, and are strongly interdependent in an algorithm. Therefore, we prefer to utilize their advantages via a complex data structure that updates the search information more effectively, called the elite pool.

It aims at taking full advantage of an adaptive memory; we use it as an improvement method of the best solutions obtained after combinations.

Some of the relationships are mentioned as follows. A pool is defined as a data structure used to store a number of solutions which have turned out to be potentially useful throughout the search [37]. Members of a pool are called elite solutions. Hence, the concept of elite pool can be presented as an adaptive memory, which [36] used the idea of genetic algorithms of combining solutions to construct new solutions and employed the tabu search of [34] as an improvement procedure. Tabu search and unified tabu search were used by [43]; where infeasible solutions are considered by extending the objective function with a penalty function and the use of continuous diversification. Another similar approach that uses the concept of elite pool is the Granular tabu search [25], which restricts the neighborhood size by removing edges from the graph that are unlikely to appear in an optimal solution. Later, other researchers have also applied this idea on their CVRP heuristics (e.g. [35]).

Except for the scatter search, path-relinking and hyper-heuristic, all of the above-mentioned approaches in the subsections have no elite pool of good quality and diverse solutions, whilst our hybrid Elitist-AS and BB-BC possess an incorporated elite pool. The above-mentioned approaches are selected to be compared to our proposed hybrid methods in our paper due to the fact that they are applied on the same instances we select. See Section 5.

It is worthwhile to evaluate the performance of the proposed hybrid algorithms (Elitist-AS, BB-BC, SS) for solving the STSP, 0–1 MKP and CVRP.

4. Proposed algorithms

In this work, we extend the investigation on the impact of an elite pool on the performance and generality of hybrid population-based meta-heuristics (from [9–11]) by testing them on STSP, 0–1 MKP, and CVRP instances.

4.1. Hybrid Elitist-Ant System

The Elitist-AS was originally proposed by [44]. Since, the Elitist-AS is struggled to maintain a balance between diversity and quality of the search, in our previous work [9], we enhanced its capability by hybridizing the Elitist-AS with an Iterated Local Search (ILS), diversification and intensification mechanisms and, an external memory to store elite solutions. The ILS is employed as an intensification mechanism to improve the individual ant solution. The diversification mechanism is employed by restarting the ant search (when it stagnates) to explore different regions (when no further possible improvements) of the search space. Those mechanisms help in strengthen the ability of the pheromone deposition (intensification) and evaporation (diversification) in diversifying the search while maintaining the quality. Based on a collection of diverse and good quality solutions (in the external memory) and a pre-defined number of non-improvements (in the ILS), the intensification mechanism will be commenced to improve solutions obtained from the ILS further. Meanwhile, the diversification mechanism will be applied when the ILS fails to improve the quality of solutions by reinitializing the pheromone trails. Pheromone trail can be considered as a memory structure but an elite one at the same time. That is why we incorporated an external memory to the structure of Elitist-AS. A generic pseudo code of our hybrid Elitist-AS is illustrated in Fig. 1 [9].

Our hybrid Elitist-AS algorithm starts the search by constructing a population of initial solutions using a constructive heuristic (each problem use different constructive heuristic). Each ant presents a solution that will be improved using the ILS (as in [5]) for a significant enhancement of its quality. Once an elite solution (better quality and more diverse) is found, it will be stored in the external memory. This solution will be utilized in the successive iterations as a reference to guide the search toward a global solution. If the elite solution is updated (found better solution), the intensification phase will be proceeded to further explore neighbors around

```
Step 1: Initialization phase
While StoppingCriterion is not met do
Step 2: Construction phase    // Problem dependent
For each ant //solution construction
  Assign/Allocate elements into feasible routes/allocation via
   probabilistic rules;
End for
Step 3: Improvement phase
While non-improvement stopping criterion is not met do
        Locally improve each constructed solution; //employ ILS
        Update size & content of external memory;
End While
If the best solution is updated then
  Step 4: Intensification phase;
   Randomly explore the neighbors of best solution found (elite solution);
  Step 5: Global Pheromone update phase;    // Problem dependent
  Update pheromone trails for allocation/route appearing in solution;
Else
  Step 6: Diversification phase;
   Pheromone evaporation;  // diversity control
   Reinitialize pheromone trails;
   Perturb elite solutions in external memory to generate new population;
End If
End While
Step 7: Return Best ant // best solution
```

Fig. 1. A generic pseudo code of the hybrid Elitist-AS.

the new elite solution in order to generate a better elite solution. If there is no improvement for a predefined number of iterations (stagnation state), the intensification phase will be skipped and the diversification phase is commenced. The diversification phase will reinitialize the pheromone trail values to restart the search. The whole steps will be repeated until the stopping criterion is met, which is either the maximum number of iterations or a global solution is found.

4.2. Hybrid Big Bang-Big Crunch

The hybrid BB-BC (the BB-BC was originally proposed by [13]) is basically a search algorithm that is inspired by the theory of the universe evolution (expansion and shrinking). It mainly characterized by a fast search space exploration and aggressive solution space exploitation [45]. This is presented by a population size reduction. A generic pseudo code of our hybrid BB-BC is illustrated in Fig. 2 [10].

The hybrid BB-BC generates initial populations using the same constructive heuristic (w.r.t. the tackled problem) as used in the hybrid Elitist-AS. A large population of initial solution is generated in the Big Bang phase, and the Euclidean distances among solutions are calculated. That is to measure their attractiveness and their diversity toward/from the elite solution; the differences among fitness values of solutions are used. Unlike the Euclidean distance used in the TSP or VRP (problem formulation), the Euclidean distance measure in the BB-BC is generally used to measure the similarity between solutions (algorithm's behavior). In the Big Crunch phase, a number of neighbors for all solutions in the population are generated. The parent solutions are replaced by some good quality off-springs in order to enforce the population converge toward better quality solution(s). An elite solution (*centre of mass*) is determined based on its quality, which is the best quality among solutions in the population. Then, a simple descent heuristic (as a local search) is applied to the *centre of mass* to enhance its quality. The new *centre of mass* will be stored into an elite pool acting as a reference for the search. That is, for the purpose of guiding the search toward better solution's quality, we generate new successive population(s) by utilizing those elite solutions (i.e. *centre of mass*). This is achieved by performing some perturbations to the elite solutions. In the Big Crunch phase, the population size will be gradually reduced (every iteration) into a single solution by eliminating poor quality solutions. The successive Big Bang phase will generate new population from the solutions of the elite pool rather than generating them from scratch. The whole process is repeated until the stopping criterion is met.

```
Big Bang phase (solutions construction):    // Problem dependent
  Step 1: Generate population (construct solutions from scratch for the 1st
          generation, or else generate new population from elite pool) &
          measure Euclidean distances among solutions in the population;
Big Crunch phase (Local Search move):  // Step 2 is Problem dependent
Repeat
   Step 2: Generate some neighbors for all solutions in the population &
           replace the parent with its best offspring for each solution in
           the population;
  Step 3: Find the centre of mass;
  Step 4: Apply local search to the centre of mass;
  Step 5: Update the elite pool and the best found solution;
  Step 6: Eliminate some poor quality solutions;
Until population size is reduced to a single solution;
  Step 7: Return to Step 1 If stopping criterion is not met;
  Step 8: Return the best found solution
```

Fig. 2. A generic pseudo code of the hybrid BB-BC.

4.3. Hybrid scatter search

The hybrid SS performs structured combinations of elite collection (that has high-diversity and high-quality solutions) in a dynamic memory. The elite collection is the key element to converge the search toward good quality solutions while diversifying the search. As in genetic algorithms, SS concerns with producing a solution from the combination of two or more solutions to yield better solutions than the original ones. SS became a popular method for solving hard COPs [46]. A generic pseudo code of our hybrid SS is shown in Fig. 3.

The hybrid SS starts the search with the diversification method by generating a small population of initial solutions from scratch using constructive heuristics (e.g. Savings algorithm [47] for the CVRP). The whole population is then improved in the improvement method using a hill climbing search. This aims to direct the search toward the local optima.

```
Step1: Start with $P = 0$.  //Steps 1, 6 & 9 are Problem dependent
       Use Diversification Generation Method to construct a solution.
Step2: Apply Improvement Method. Let $x$ be the resulting solution.
If $x \notin P$ Then add $x$ to $P$ (i.e., $P = P \cup x$). Otherwise, discard $x$.
      Repeat this step Until $|P| = PSize$. Otherwise, discard $x$.
       Repeat this step Until $|P| = PSize$.
 Step3: Use Reference Set Update Method to build RefSet (divided
       into two sets) with the "best quality" $b1 = \{x^1, , x^b\}$ solutions
       and; the "most diverse" $b2 = \{y^1, \ldots, y^b\}$ solutions in $P$.
       Order solutions in $b1$ according to their objective function values
        such that $x^1$ is the best solution and $x^b$ is the worst.
       Order solutions in $b2$ according to their dissimilarity values
        (using $d_{\min}(p, q)$) such that $y^1$ is the best and $y^b$ is the worst.
       Make NewSolutions = TRUE.
While (NewSolutions && Stopping criterion is not met) do
  Step4: Generate NewSubsets with the Subset Generation Method.
         Make NewSolutions = FALSE.
  While ($NewSubsets \neq \emptyset$) do // repeat 5 times at most to update RefSet
Step5: Select the next subset $s$ in NewSubsets.
Step6: Apply Solution Combination Method to $s$ to obtain one or more
        new solutions $x$.    // apply crossover to the pair of elite solutions
Step7: Apply Improvement Method to the trial solutions.
Step8: Apply Reference Set Update Method.
If (RefSet has changed) Then
Step9: Make NewSolutions = TRUE. Delete $s$ from NewSubset.
Else Employ Diversification Generation Method to construct a new
population by perturbing elite solutions in RefSet.
Apply the Improvement Method to the new population. // hill climbing
End if
Delete $s$ from NewSubset.
  End While
End While
Step10: Return the best found Solution
```

Fig. 3. A generic pseudo code of the hybrid SS.

In the reference set update method, a reference set (*RefSet*) of elite and diverse solutions is created (for the first iteration) and will be iteratively updated for the subsequent iterations once a better quality or diverse solution than that in the *RefSet* is produced. The *RefSet* has two subsets: b1 and b2. Elite solutions are selected based on their quality and then stored in b1, whilst diverse solutions are selected based on their greatest dissimilarity from others in the population and then stored in b2. The solution that has more uncommon i.e. routes from other solutions, the more diverse it becomes. The *RefSet* is updated by replacing the worst elite solution in b1 by a better newly generated solution. Meanwhile, a worst diverse solution is replaced by a newly generated solution that has much dissimilarity from the ones in b2.

Then the subset generation method is proceeded which selects one solutions from each subset in the *RefSet* to be combined and to generate new promising solutions. Those selected two solutions are from b1 and b2. This selection mechanism is called Type-I method [8], which is the combination of all 2-elements subsets. This means, combining all possible unrepeated two solutions.

As in [27], this work performs the solution combination using a single-point crossover operator to generate two off-springs. Feasibility of the off-springs is ensured by a repair function (problem dependent) that rectifies a corrupted solution resulted by the crossover. These off-springs are further be enhanced by the improvement method (e.g. the ILS). The improved off-springs will be compared to the ones in the *RefSet* for updating its contents. Then, a successive diversification generation method is commenced once again with the same population size. The new population is generated by performing some perturbations to the solutions in *RefSet* rather than building them from scratch. The whole process is repeated until the stopping criterion is met.

5. Computational results and discussion

We have tested the three hybrid approaches (Elitist-AS, BB-BC and SS) over twenty nine STSP instances from TSPLIB[1]; fourteen CVRP instances (introduced by [48], which can be downloaded from VRPWEB[2]); and eleven 0–1 MKP instances from [49].[3] As recommended by the literature (e.g. [48]), we ran our approaches 25 times (for each approach) on each instance for 100,000 iterations as a stopping condition. The experiments were performed on Intel Core i7 2.30 GHz processor, 8 GB RAM, and implemented in Java NetBeans IDE v 7.0.1.

5.1. Experimental setup

Parameters shown in Tables 1–3 are determined experimentally (e.g. elite pool size) and based on the literature (e.g. Elitism). For example, the population size in the Elitist-AS and SS is preferred to be relatively small [8], whilst the BB-BC follows a typical population size as of the genetic algorithms.

There are too many instances provided for the three COPs (e.g. 112 instances for the STSP). Therefore, we have determined to test our hybrid approaches on some common instances tested across the literature. In the following subsections, Tables 4–6 show our results obtained for our test on the STSP, 0–1 MKP and CVRP, respectively. These tables also show the best known results with those obtained by our hybrid meta-heuristics highlighted in bold. The standard deviation (Std.) for each methodology is also presented. Many published works reported the deviation percentage – $\Delta(\%)$ – (or brief statistics) of their best solutions from best known solutions (e.g. [43]). Therefore, in this work we report our results similar to their illustration. The deviation percentage – $\Delta(\%)$ – 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 25 independent runs by each of our hybrid meta-heuristics, and b is the best known result found in the related literature.

Table 1
Parameters settings used by the hybrid Elitist-AS.

Parameter	Value
Population size	Number of cities/knapsacks/ customers
Number of Iterations	100,000
Number of non-improvement iterations	100
Pheromone initial values	0.5 for 0–1 *MKP*; 0.01 for *STSP* & *CVRP*
Evaporation rate	0.25 ∊ [0,1] for 0–1 *MKP*; 0.1 ∊ [0.04, .06] for *STSP* & *CVRP*
Controlling ratio (exploration vs. exploitation)	1.0 ∊ [1.0,1.1] for 0–1 *MKP*; 0.95 ∊ [0,1] for *STSP* & *CVRP*
Importance of distances/constraints (penalty)	2.0
Number of employed neighborhood structures per solution	5
Initial external memory (elite pool) size	5
Local search routine	Iterated Local Search
Search update	Use the best ant to update global pheromone

Table 2
Parameters settings used by the hybrid BB-BC.

Parameter	Value
Population size	100
Number of iterations	100,000
Number of non-improvement iteration	30
Reduction rate	0.8
Elite pool size	10
Local search routine	Simple descent heuristic
Number of neighbors created in each generation	5
Search update	Last population solution is forced to be always the best

Table 3
Parameters settings used by the hybrid SS.

Parameter	Value
Population size	50
Maximum number of iterations	100,000
Number of non-improvement iterations	30
Size of *RefSet* (elite pool)	20 ($b1 = 10$, $b2 = 10$)
Local search routine	Hill climbing (in *Step 2*)
	Iterated Local Search (in *Steps 7 & 9*)
Subset selection	Type-I selection, 2-elements subsets
Crossover	One-point
Least similar	Best diverse
Crossover rate	0.8 (problem dependent)
Mutation rate	0.04 (problem dependent)
Search update	Dynamic, once a better solution is found

5.2. Experimental results

5.2.1. Symmetric traveling salesman problem

Table 4 shows our results against the best known results.

Based on Table 4, our hybrid meta-heuristics have shown to be competitive (in most cases) with the best known results. The table

[1] OR library: http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/.
[2] http://neo.lcc.uma.es/radi-aeb/WebVRP/.
[3] http://people.brunel.ac.uk/~mastjjb/jeb/orlib/mknapinfo.html.

Table 4
Results of our hybrid approaches applied to the STSP[a].

Instance[b]	Optimal solution	Elitist-AS			BB-BC			SS		
		Best	Δ(%)	*Std.*	*Best*	Δ(%)	*Std.*	*Best*	Δ(%)	*Std.*
att48	10,628	**10,628**	0	0	**10,628**	0	0	**10,628**	0	0
att532	27,686	28,147	1.66	19.22	29,884	7.9	34.01	28,147	1.66	28.72
a280	2579	**2579**	0	13.03	2628	1.89	49.6	**2579**	0	23.04
bays29	2020	**2020**	0	0	**2020**	0	0	**2020**	0	0
berlin52	7542	**7542**	0	0	**7542**	0	0	**7542**	0	0
bier127	118,282	**118,282**	0	325.5	118,759	.4	422.1	**118,282**	0	361.02
ch130	6110	**6110**	0	22.1	**6110**	0	18.19	**6110**	0	17.02
ch150	6528	**6528**	0	11.22	**6528**	0	14.98	6528	0	12.7
d198	15,780	15,888	.68	64.83	15,954	1.1	36.93	15,954	1.1	22.01
d2103	80,450	80,688	.29	1066.3	80,688	.29	1126.01	80,688	.29	1072.4
d15112	1,573,084	1,573,162	.004	25,576.1	1,582,184	.57	25,798.8	1,573,162	.004	25,613.2
eil51	426	**426**	0	0	**426**	0	0	**426**	0	0
eil76	538	**538**	0	0	**538**	0	0	**538**	0	0
fl1577	22,249	22,977	3.27	217.22	23,577	5.96	223.4	22,977	3.27	217.1
gr24	1272	**1272**	0	0	**1272**	0	0	**1272**	0	0
hk48	11,461	**11,461**	0	2.5	11,491	.26	6.33	**11,461**	0	3.12
kroA100	21,282	**21,282**	0	0	**21,282**	0	0	**21,282**	0	0
kroB150	26,130	26,185	.21	0	26,185	.21	.86	26,185	.21	.73
kroA200	29,368	29,420	.17	1.03	29,868	1.7	1.45	29,420	.17	1.3
lin318	42,029	44,169	5.09	18.63	44,169	5.09	21.03	44,169	5.09	18.2
pcb442	50,778	51,286	1.0	908.1	51,799	2.01	1006.2	51,286	1.0	902.6
pr76	10,8159	**108,159**	0	0	**108,159**	0	0	**108,159**	0	0
rat575	6773	**6773**	0	0	**6773**	0	0	**6773**	0	0
rat783	8806	**8806**	0	1.27	**8806**	0	3.32	**8806**	0	2.85
rd100	7910	**7910**	0	0	**7910**	0	0	**7910**	0	0
rd400	15,281	**15,281**	0	0	15,788	3.31	6.91	**15,281**	0	4.25
st70	675	**675**	0	0	**675**	0	0	**675**	0	0
tsp225	3919	**3919**	0	2.63	**3919**	0	2.98	**3919**	0	2.68
vm1084	239,297	**239,297**	0	1859.9	239,364	.02	2103.1	**239,297**	0	1867.2
Average deviation		.43			1.1			.44		

[a] The instances of STSP (TSPLIB95) have been officially declared in 2007 (by its founder [21]) have been solved to optimality.
[b] The number in the name of an instance gives the instance dimension/number of cities.

Table 5
Results of our hybrid approaches and the best known results in the literature applied to the 0–1 MKP.

Instance		Best known	Elitist-AS			BB-BC			SS		
n	*m*		*Best*	Δ(%)	*Std.*	*Best*	Δ(%)	*Std.*	*Best*	Δ(%)	*Std.*
100	15	3766	**3766**	0	0	**3766**	0	0	**3766**	0	0
100	25	3958	**3958**	0	0	**3958**	0	0	**3958**	0	0
150	25	5650	**5650**	0	0	**5656**	0	0	**5656**	0	0
150	50	5764	**5764**	0	0	**5764**	0	0	**5764**	0	0
200	25	7557	**7557**	0	0	**7557**	0	0	**7557**	0	0
200	50	7672	**7672**	0	0	**7672**	0	0	**7672**	0	0
500	25	19,211	19,711	0	.024	19,215	0	.017	19,215	0	.014
500	50	18,801	18,806	.02	.009	18,806	.02	.006	18,806	.02	.006
1500	25	58,085	58,094	.01	.021	**58,085**	0	.016	**58,085**	0	.016
1500	50	57,292	57,294	.003	.007	57,298	.01	.004	57,294	.003	.004
2500	100	95,231	95,240	.009	.017	95,236	.005	.011	**95,231**	0	.002
Average deviation				.0038			.0031			.002	

also shows the gap between the best known solution and our hybrid meta-heuristics. A zero gap indicates that best known (or rather optimal) solutions were obtained. Overall, in comparison to the best known solutions, our hybrid meta-heuristics can produce very good solutions. Elitist-AS has obtained optimal solutions for 20 instances out of 29; SS obtained 19 optimal solutions; and BB-BC obtained 15 optimal solutions. Across 29 instances – 25 runs for each instance – the average $\Delta(\%)$ is only 0.43% for the Elitist-AS; 0.44% for the SS; and 1.1% for the BB-BC. In addition, the consistency of our hybrid methods can be defined by the Std. over 25 runs, where the Std. produced by Elitist-AS is smaller than those from BB-BC and SS for all instances of the STSP (except ch130, d198, fl1577, and lin318). These indicate that our methodologies are very efficient and competitive to solve the STSP compared in terms of solution quality and consistency. Hence, meeting those criteria leads to the generality of our hybrid meta-heuristics over different sizes of instances.

5.2.2. 0–1 multidimensional knapsack problem

Table 5 presents the results we have obtained and compared them against best known results in the literature.

Table 5 shows that our hybrid meta-heuristics can find the optimal solution in many instances. The SS and BB-BC are better than the Elitist-AS in term of quality (for most instances). Overall, in comparison to the best known solutions, our hybrid meta-heuristics can produce very good solutions. Elitist-AS has obtained optimal solutions for 6 instances out of 11; SS obtained 7 optimal solutions; and BB-BC obtained 8 optimal solutions. Across 11 instances – 25 runs for each instance – the average $\Delta(\%)$ is only 0.002% for the SS; 0.0031% for the BB-BC; and 0.0038% for the Elitist-AS. In addition, the consistency of our hybrid methods can be defined by the Std. over 25 runs, where the Std. produced by SS is smaller than those from BB-BC and Elitist-AS for all instances of the 0–1 MKP. These indicate that our methodologies are very efficient and competitive

Table 6
Results of our hybrid approaches applied to the CVRP.

Instance	n[f]	Q[g]	s[h]	L[i]	m[j]	Best known	Elitist-AS			BB-BC			SS		
							Best	Δ(%)	*Std.*	*Best*	Δ(%)	*Std.*	*Best*	Δ(%)	*Std.*
vrpnc1	50	160	0	∞	5	524.61[a]	**524.61**	0	0	**524.61**	0	0	**524.61**	0	0
vrpnc2	75	140	0	∞	10	835.26[a]	**835.26**	0	0	**835.26**	0	0	**835.26**	0	0
vrpnc3	100	200	0	∞	8	826.13[b]	826.14	.001	1.07	826.14	.001	1.21	826.14	.001	1.1
vrpnc4	150	200	0	∞	12	1028.42[a]	**1028.42**	0	0	1029.32	.08	.26	**1028.42**	0	0
vrpnc5	199	200	0	∞	17	1291.29[c]	1292.57	.09	4.96	1292.57	.09	5.24	1292.57	.09	4.1
vrpnc6	50	160	10	200	6	555.43[a]	**555.43**	0	0	**555.43**	0	0	**555.43**	0	0
vrpnc7	75	140	10	160	11	909.67[b]	**909.67**	0	0	**909.67**	0	0	**909.67**	0	0
vrpnc8	100	200	10	230	9	865.94[d]	**865.94**	0	0	**865.94**	0	0	**865.94**	0	0
vrpnc9	150	200	10	200	14	1162.55[a]	1163.52	.08	3.8	1168.78	.53	3.9	1168.25	.49	3.1
vrpnc10	199	200	10	200	18	1395.85[e]	1405.26	.67	4.2	1410.58	1.5	4.3	1408.23	.88	1.02
vrpnc11	120	200	0	∞	7	1042.11[a]	**1042.11**	0	0	**1042.11**	0	0	**1042.11**	0	0
vrpnc12	100	200	0	∞	10	819.55[b]	819.56	.001	.98	819.56	.001	1.9	819.56	.001	1.9
vrpnc13	120	200	50	720	11	1541.14[a]	1541.14	0	.57	1552.54	.73	2.14	1548.88	.5	1.42
vrpnc14	100	200	90	1040	11	866.36[b]	866.37	.001	1.61	866.37	.001	1.94	866.37	.001	1.71
Average deviation						.06			.2			.14			

Note:
[a] [34] – parallel iterative search
[b] [41] – a grammatical evolution hyper-heuristic.
[c] [35] – evolutionary algorithm with active-guided local search.
[d] [43] – an improved artificial bee colony.
[e] [36] – probabilistic diversification & intensification methods in local search.
[f] n is the number of customers.
[g] Q is the vehicle capacity.
[h] s is the services time for each customer.
[i] L is the route maximum length.
[j] m is the number of vehicle routes.

to solve the 0–1 MKP compared in terms of solution quality and consistency. Hence, meeting those criteria leads to the generality of our hybrid meta-heuristics over different sizes of instances.

5.2.3. Capacitated vehicle routing problem

We have tested our methodologies on a common set of 14 instances from the literature, summarized in Table 6. We also compare our results against the best known results as shown in the table.

Table 6 shows that our hybrid meta-heuristics can find the optimal solutions in many instances or near-optimal ones in the rest (i.e. obtained by SS). Overall, in comparison to the best known solutions, our hybrid meta-heuristics can produce very good solutions as well. Elitist-AS has obtained optimal solutions for 20 instances out of 29; SS obtained 19 optimal solutions; and BB-BC obtained 15 optimal solutions. Across 14 instances – 25 runs for each instance – the average $\Delta(\%)$ is only 0.06% for the Elitist-AS; 0.2% for the SS; and 0.14% for the BB-BC. In addition, the consistency of our hybrid methods can be defined by the Std. over 25 runs, where the Std. produced by Elitist-AS is smaller than those from SS and BB-BC for all instances of the CVRP (except vrpnc5, vrpnc9, and vrpnc10). These indicate that our methodologies are very efficient and competitive to solve the CVRP compared in terms of solution quality and consistency. Hence, meeting those criteria leads to the generality of our hybrid meta-heuristics over different sizes of instances. Overall, the Std. indicates that Elitist-AS is more consistent than SS and BB-BC across all tested problem domains.

5.3. Comparison with previous works

This section is devoted to assessing the performance of our hybrid meta-heuristics against other conventional and hybrid methods in the literature. So, we aim at (i) assessing the benefit of integrating an elite pool within our hybrid methods, and (ii) testing the generality and consistency of our hybrid methods over three different COPs and compare them to others.

In order to support our hypothesis of the impact/effect of the elite pool on the performance of population-based meta-heuristics, we have investigated this issue by comparing our hybrid meta-heuristics with a number of conventional and hybrid meta-heuristics which have no elite pool. For example, a typical genetic algorithm has a pool (specifically an explicit memory) of diverse solutions but it does not possess a pool of elite solutions (diverse and high-quality) [2,3]. That is why it has a great search diversification mechanism but it lacks an efficient intensification mechanism [2]. In some algorithms such as honey bee mating, grammatical evolution hyper-heuristic, and memetic algorithms, the use of elite pool can effect the performance of meta-heuristic algorithms in solving variety of optimization problems [2] and [50–52].

Tables 7–9 show the results (optimal results are shown in bold) obtained by our hybrid meta-heuristics for the three problems: STSP, 0–1 MKP and CVRP compared to similar meta-heuristics that do not possess an elite pool of high-quality and high-diversity solutions. The computational time of the compared methods ranges from 500s to 1500s.

Compared to other population-based and hybrid methods in Tables 7–9, we can see that, almost across all instances, our hybrid methods outperform other methods (e.g. GA in Table 7, DGALS in Table 8, and ABC in Table 9). For example, in Table 7, our hybrid Elitist-AS (with an elite pool) has obtained a much better result (e.g. berlin52 = 7542) than the conventional AS and CEAS. Similar examples are those of our hybrid BB-BC (with an elite pool) compared to the conventional GA (e.g. att48 = 10,628) and the hybrid SS (with an elite pool) compared to GCGA-LS (e.g. eil51 = 426).

To see the effect in terms of the quality of utilizing an elite pool, we implemented the Elitist-AS without an elite pool and compared it to our hybrid Elitist-AS with an elite pool; please see the last column in Tables 7–9. The conventional BB-BC is somewhat similar to the genetic algorithm, and the conventional SS is a complex meta-heuristic which already possesses a hybrid structure and incorporates an elite pool within. Therefore, we have chosen to implement the CEAS which has no elite pool.

Many of the methodologies applied to the three COPs tested in this work (STSP, CVRP and 0–1 MKP) did not use an explicit memory. This may cause the lack of maintaining a balance between diversity and quality of the search. They also lack a systematic selection

Table 7
Our best results on STSP compared to population-based methods without an elite pool.

Instance	Elitist-AS	BB-BC	SS	GA	AS	GCGA-LS	ACS-2opt	CEAS	Elitist-AS vs. CEAS *p*-Value
att48	**10,628**	**10,628**	**10,628**	10,782	*N/A*	10,638	*N/A*	10,736	+
att532	28,147	29,884	28,147	*N/A*	*N/A*	*N/A*	*N/A*	31,861	+
a280	**2579**	2628	**2579**	*N/A*	2687	*N/A*	2618	3092	+
bays29	**2020**	**2020**	**2020**	*N/A*	2077	*N/A*	**2020**	2022	+
berlin52	**7542**	**7542**	**7542**	*N/A*	7601	*N/A*	**7542**	7548	+
bier127	**118,282**	118,759	**118,282**	120,978	*N/A*	119,363	*N/A*	120,516	+
ch130	**6110**	**6110**	**6110**	*N/A*	*N/A*	*N/A*	*N/A*	6265	+
ch150	**6528**	**6528**	**6528**	*N/A*	6696	*N/A*	6570	6759	+
d198	15,888	15,954	15,954	16,108	*N/A*	15,940	*N/A*	16,066	+
d2103	80,688	80,688	80,688	*N/A*	*N/A*	*N/A*	*N/A*	80,882	+
d15112	1,573,162	1,582,184	1,573,162	*N/A*	*N/A*	*N/A*	*N/A*	1,591,630	+
eil51	**426**	**426**	**426**	430	*N/A*	427	*N/A*	431	+
eil76	**538**	**538**	**538**	552	*N/A*	548	*N/A*	551	+
fl1577	22,977	23,577	22,977	*N/A*	*N/A*	*N/A*	*N/A*	30,102	+
gr24	**1272**	**1272**	**1272**	*N/A*	*N/A*	*N/A*	*N/A*	**1272**	~
hk48	**11,461**	11,491	**11,461**	*N/A*	*N/A*	*N/A*	*N/A*	14,541	+
kroA100	**21,282**	**21,282**	**21,282**	21,554	*N/A*	21,292	*N/A*	22,802	+
kroB150	**26,185**	**26,185**	**26,185**	26,698	*N/A*	26,556	*N/A*	26,906	+
kroA200	**29,420**	29,868	**29,420**	30,166	31,662	30,191	29,470	30,478	+
lin318	44,169	44,169	44,169	43,607	43,651	44,396	**42,086**	45,716	+
pcb442	**51,286**	51,799	**51,286**	52,923	53,469	54,670	51,790	58,231	+
pr76	**108,159**	**108,159**	**108,159**	109,132	*N/A*	108,308	*N/A*	109,729	+
rat575	**6773**	**6773**	**6773**	*N/A*	*N/A*	*N/A*	*N/A*	7579	+
rat783	**8806**	**8806**	**8806**	*N/A*	8950	*N/A*	8815	9554	+
rd100	**7910**	**7910**	**7910**	8203	8603	7986	8159	8328	+
rd400	**15,281**	15,788	**15,281**	15,933	*N/A*	16,215	*N/A*	17,413	+
st70	**675**	**675**	**675**	683	752	**675**	722	**675**	~
tsp225	**3919**	**3919**	**3919**	*N/A*	*N/A*	*N/A*	4102	4007	+
vm1084	**239,297**	239,364	**239,297**	*N/A*	*N/A*	*N/A*	*N/A*	241,758	+

Note: GA: genetic algorithm by [27]. ACS-2opt: two-stage ant colony system and 2opt by [30]. AS: ant system by [30]. GCGA-LS: generalized chromosome GA with local search by [27].

Table 8
Our best results on 0–1 MKP compared to population-based methods without an elite pool.

Instance		Elitist-AS	BB-BC	SS	DGALS	CEAS	Elitist-AS vs. CEAS
n	*m*						*p*-Value
100	15	**3766**	**3766**	**3766**	**3766**	3832	+
100	25	**3958**	**3958**	**3958**	**3958**	4655	+
150	25	**5650**	5656	5656	5656	5780	+
150	50	**5764**	**5764**	**5764**	**5764**	5846	+
200	25	**7557**	**7557**	**7557**	**7557**	7775	+
200	50	**7672**	**7672**	**7672**	**7672**	8266	+
500	25	19,711	19,215	19,215	**19,211**	22,557	+
500	50	**18,806**	**18,806**	**18,806**	18,796	19,686	+
1500	25	58,094	**58,085**	**58,085**	58,078	59,278	+
1500	50	**57,294**	57,298	**57,294**	57,295	77,785	+
2500	100	95,240	95,236	**95,231**	95,378	103,690	+

Note: DGALS: double genetic algorithm with local search by [33].

Table 9
Our best results on CVRP compared to population-based methods without an elite pool.

Instance	Elitist-AS	BB-BC	SS	ABC	TS-PR	CEAS	Elitist-AS vs. CEAS *p*-Value
vrpnc1	**524.61**	**524.61**	**524.61**	526.74	**524.61**	**524.61**	~
vrpnc2	**835.26**	**835.26**	**835.26**	865.23	836.37	886.39	+
vrpnc3	**826.14**	**826.14**	**826.14**	842.47	828.26	**826.14**	~
vrpnc4	**1028.42**	1029.32	**1028.42**	1065.71	1034.08	1063.35	+
vrpnc5	1292.57	1292.57	1292.57	1382.85	1311.78	1420.00	+
vrpnc6	**555.43**	**555.43**	**555.43**	560.33	**555.43**	**555.43**	~
vrpnc7	**909.67**	**909.67**	**909.67**	945.56	909.68	909.68	+
vrpnc8	**865.94**	**865.94**	**865.94**	887.71	866.71	887.71	+
vrpnc9	1163.52	1168.78	1168.25	1235.37	1177.01	1170.32	+
vrpnc10	1405.26	1410.58	1408.23	1497.92	1420.66	1492.11	+
vrpnc11	**1042.11**	**1042.11**	**1042.11**	1183.92	1042.97	1146.60	+
vrpnc12	**819.56**	**819.56**	**819.56**	843.17	**819.56**	821.12	+
vrpnc13	1541.14	1552.54	1548.88	1592.88	1568.79	1650.88	+
vrpnc14	**866.37**	**866.37**	**866.37**	881.41	**866.37**	886.62	+

Note: ABC: the original artificial bee colony algorithm by [43]. TS-PR: tabu search and path re-linking algorithm by [42].

strategy or a solution combination strategy, where 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 works reported in the literature used some memory to store the best solution (such as the bee colony algorithm in [20], or the ant colony algorithm in [53]), but unfortunately they did not provide a clear description of the memory.

From Tables 7–9, for most instances (of the three COPs), it can be clearly seen that our methodologies (especially the SS) are better than those without elite pools (e.g. see att48 in Table 7). This demonstrated the effectiveness of using an elite pool (of diverse and high-quality solutions) in a population-based method. Some of our results are exactly the same as others (e.g. see vrpnc1 in Table 9). In few cases, our methodologies obtained worse results than those of the original approaches (elite pool-less); this might be due to the pseudorandom behavior of a population-based meta-heuristic. However, the SS meta-heuristic is the only standalone method so far that has an explicit memory, a systematic selection, and an explicit solution combination. Note that in order to investigate the effectiveness of having pool in the meta-heuristic algorithms; we have embedded similar strategies and components of basic SS in our hybrid Elitist-AS, BB-BC and SS. This structure may enable our approaches to maintain a balance between diversity and quality of the search. Across all instances, it appeared that the SS is the best approach compared to our hybrid Elitist-AS, and BB-BC. It could be due to the utilization of an explicit-elite pool structure (namely *RefSet*) that is already embedded within the SS by default (unlike its counterparts). This *RefSet* is well structured in a way that it interacts effectively with the solutions combination and the diversification generation methods to provide an adaptive search update. Therefore, it may guarantee a relatively fast convergence toward high-quality (or optimal) solutions without losing diversity of the search. As mentioned before, both Elitist-AS and BB-BC meta-heuristics possess an implicit memory to store high-quality and diverse 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. We believe that the transition between implicit and explicit solution representations during the search process is a hard task. However, all of our methods are fairly effective and consistent across three problems (STSP, MKP, and CVRP).

In addition to the results above, it is worthwhile to draw some statistically significant conclusions regarding the performance of Elitist-AS and the CEAS. Therefore, the Wilcoxon test (pairwise comparisons) with significant level of 0.05 is performed. The *p*-value of the Wilcoxon test of Elitist-AS versus CEAS is presented in the last column of Tables 7–9. Where "+" indicates Elitist-AS is statistically better than CEAS (p-value <0.05), "−" indicates Elitist-AS outperformed by CEAS (p-value >0.05) and "∼" indicates both algorithms have the same performance (p-value = 0.05). The results in Tables 7–9 (last column) show that Elitist-AS is statistically better than CEAS on 49 instances, not statistically better than CEAS on none of the instances, and perform the same as CEAS on 5 instances out of 54 tested instances of the considered problem domains.

To summarize, the results demonstrate that Elitist-AS is better than CEAS in terms of consistency, efficiency and generality with regards to the tested instances of the considered problem domains. This is mainly due to the use of elite pool within Elitist-AS which has a positive effect on the ability of Elitist-AS in producing good quality and consistent results compared to CEAS. In all domains, the Std. and the $\Delta(\%)$ of Elitist-AS, BB-BC, and SS reveal that their results are stable and very close to the best results obtained by other population-based meta-heuristic methods. All of the observations above serve as evidence that the Elitist-AS, BB-BC, and SS are capable of producing good quality results over all instances, instead of just a few ones.

As shown in this experimental study, our hybrids SS, Elitist-AS and BB-BC obtained in all three problems (STSP, CVRP and 0–1 MKP) competitive results, if not optimal ones in some instances, when compared against the best known results reported in the literature. For the three problems, the percentage deviation demonstrates that, the results of our hybrid meta-heuristics are very close to the best known ones. This is proven based on their consistency and generality across the three problems. We believe this is due to the factor of hybridizing an explicit memory structure (e.g. reference set) in each of our meta-heuristics to diversify the search by exploring different regions of the search space, or rather by escaping local optima, while preserving high-quality solutions. Overall, the result implies that hybridizing an explicit memory structure with a population-based meta-heuristic has a significant impact on the performance of population based meta-heuristics (specifically SS, Elitist-AS and BB-BC) in solving STSP, CVRP and 0–1 MKP.

5.4. Computational complexity

It is interesting to consider the computational complexity of the algorithm in order to know which portion is most compute-intensive in our proposed hybrid meta-heuristics. Based on our analysis, the computational complexity of the search in our hybrid meta-heuristics is not high when compared to most meta-heuristics. This presents another benefit of our hybrid meta-heuristics – i.e., the general and computational efficiency for a range of problems. To provide some indication where most of the time is being spent, the computational complexity or cost of the hybrid meta-heuristics is briefly analyzed below:

1. The overall computational complexity for each of the meta-heuristics (without hybridization) is:
   i $O(p * n)$ for Elitist-AS.
   ii $O(k * n)$ for BB-BC.
   iii $O(p * n + n_L\,(RefSize^2/2)\,n)$ for SS. where p is the population size, n is the number of iterations, n_L is the number of loops, k is the number of crossovers (perturbations), and *RefSize* is the memory size.
2. However, our proposed hybrid meta-heuristics start by initializing the population of initial solutions:
   iv Initialization = $O(m^2 + p)$. Where p is the population size, and m is the dimension of the solution.
3. Constructive heuristics are employed in the population initialization:
   v Nearest Neighbor complexity $O(n^2)$ for *STSP*.
   vi Greedy method complexity $O(n^2 \log_2(n))$ for 0–1 *MKP*.
   vii Savings algorithm complexity $O(n^2 \log_2(n))$ for *CVRP*.
   Regardless a specific type of constructive heuristics, generally the complexity is = $O(m^2 * p)$.
4. Next, they generate a population of solutions for the adaptive memory mechanism (e.g. external memory, elite pool, *Refset*) which has complexity $O(s * m)$, where s is the memory size and m is the dimension of the solution (e.g. number of cities in *STSP*).
5. The proposed hybrid meta-heuristics will then employ local search routines to generate new better quality solutions:
   viii Hill Climbing complexity = $O(n)$.
   ix Iterated local search = $O(n)$.
   x Simple descent heuristic = $O(n)$.
6. Each hybrid meta-heuristic will call a local search routine every n iterations (every n consecutive non-improvement iterations) to generate a new population from scratch or re-initializing the search as follows:

xi Selection = O(m).

xii Crossover = O($m-1$) for BB-BC and SS.

xiii Mutation = O($m(1+q)$) BB-BC and SS, where q is the mutation probability.

xiv Perturbations = O($m-1$) for Elitist-AS.

xv Deposit pheromone trail = O($m^2 * p$) for Elitist-AS.

xvi Update pheromone trail = O(m^2) for Elitist-AS.

7. The hybrid meta-heuristics will update the adaptive memory mechanisms (e.g. external memory, elite pool, *Refset*) and sort the solutions of the adaptive memory mechanism which has the complexity O($s\log(s)$).
8. The search is updated by the elitism. The search will examine solutions in the memory whether to loop or to stop:

xvii Elitism = O($m * p^2$).

xviii Loop or Exit (stopping criterion) = O($m * p$).

Thus, the overall complexity for each proposed hybrid meta-heuristic is:

- Hybrid Elitist-AS = O(m^2+p) + O($m^2 * p$) + O($s * m$) + O(n) + O($m-1$) + O($m^2 * p$) + O(m^2) + O($s\log(s)$) + O($m * p^2$) + O($m * p$).
- Hybrid BB-BC = O(m^2+p) + O($m^2 * p$) + O($s * m$) + O(n) + O(m) + O($m-1$) + O($m(1+q)$) + O($s\log(s)$) + O($m * p^2$) + O($m * p$).
- Hybrid SS = O(m^2+p) + O($m^2 * p$) + O($s * m$) + O(n) + O(m) + O($m-1$) + 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 one of the following: O(n^2) for *STSP*, or O($n^2 \log_2(n)$) for 0–1 *MKP*, or O($n^2 \log_2(n)$) for *CVRP*.

Intentionally, we have not simplified them as we wish to draw out the complexity of the various stages. Since the proposed hybrid meta-heuristics have been tested on three different problem domains, and each one uses a variety of local search routines, we have not presented in detail the complexity of the utilized local search routine. By inspection, we can observe that the complexity of the local search routine (O(n)) is very low, e.g. w.r.t. to the parameters of n, m and p, etc., therefore does not present to be a critical factor in the proposed hybrid meta-heuristics. The crucial factors are presented in the memory structures (external memory, elite pool, *Refset*) and the search update (see steps 4 and 7, steps 6 and 8, respectively). In some cases, constructive heuristics could be presented as crucial factors too (see step 3); such as the case of CVRP.

Overall, the advantages of our hybrid methods are the ability to utilize the heuristic information about diverse and high-quality solutions during instance solving – via elite pool – to diversify the search while intensifying the improvement of a high-quality solution. Results demonstrate that our hybrid methods provide a general mechanism regardless the nature and complexity of the instances and can be applied to other domains without many changes (i.e. the user only needs to change the constructive heuristics and neighborhood structures). Applying a methodology to other problem domains or even different instances of the same problem usually requires a considerable amount of modification (e.g. change algorithm parameters or structures). We have demonstrated the generality of our hybrid methods across three different problem domains. We would hope that the proposed methodologies would also generalize to other domains.

We evaluate the performance of our hybrid meta-heuristics by considering the following three criteria:

Generality: We define generality as the ability of our hybrid methods to work well, not only across different instances of the same problem, but also across three different problem domains.

Consistency: The ability of our hybrid methods to produce stable results when executed several times for every instance. Typically, consistency is one of the most important criteria in evaluating any algorithm. This is due to the fact that many search algorithms have a stochastic component, prompting thus different solutions over multiple runs even if the initial solution is the same. We measure the consistency of our hybrid methods based on the average and the standard deviation over 25 independent runs.

Efficiency: The ability of our hybrid methods to produce good results that are close or better than the best known value in the literature. We measure the efficiency of our hybrid methods by reporting, for each instance, the best and the percentage deviation, $\Delta(\%)$, from the best known results in the literature.

For all tested instances, we compared our hybrid methods' results with similar methods in terms of solution quality rather than computational time. This can be attributed to the fact that different computer resources used made the comparison very difficult. Therefore, we set the number of iterations as the termination criteria as a result of the use of the adaptive memory (e.g. 20 min) in our hybrid methods. As a result, the execution time of our hybrid methods is within the range of those published in the literature.

6. Conclusion

In this study, we have demonstrated the generalization, and consistency of three hybrid population-based meta-heuristics (namely, SS, Elitist-AS, and BB-BC) in three different classes of COPS (i.e. STSP, CVRP, and 0–1 MKP) using the same parameters settings. This was achieved by investigating and testing the impact of an elite pool on the general performance of a population-based meta-heuristic. The three hybrids population-based meta-heuristics utilize an elite pool (namely, reference set in the SS, elite pool in the BB-BC, and external memory in the Elitist-AS) which contains a collection of diverse and high-quality solutions. These memory structures help 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 the three hybrid meta-heuristics produce high-quality solutions, if not optimal, and their performance are or can be generalized well across the three problems. The study concluded that the hybridization of an elite pool (which contains a collection of high-quality and diverse solutions) within a population-based meta-heuristic can enhance the performance of population-based meta-heuristics that is generalized well across different problems and producing high-quality solutions which are either competitive or optimal in some cases. The analysis and discussion on the computational complexity is another proof that our hybrid meta-heuristics are efficient across three COPs.

The main contributions of this work are as follows:

- The development of the hybrid methods (Elitist-AS, BB-BC, and SS) that possess elite pool and perform heuristic perturbations, demonstrating that strengths of different search algorithms can be merged into one hybrid population-based meta-heuristic methodology (e.g. constructive heuristics and meta-heuristics; population-based and local search methods).
- The hybridization of an adaptive memory mechanism (e.g. elite pool) that contains a collection of high-quality and diverse solutions, with a population-based meta-heuristic, 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 other similar methods in some cases.
- The development of a hybrid meta-heuristic which 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 hybrid meta-heuristics can start their search process for better solutions. The elite pool also provides a mean of implement cooperation and to achieve faster convergence. In future work, we intend to investigate the effectiveness of our hybrid meta-heuristics across other COPs.

References

[1] I.H. Osman, J.P. Kelly, Meta-heuristic: an overview, in: I.H. Osman, J.P. Kelly (Eds.), Meta-Heuristic: Theory and Applications, Kluwer Academic Publishers, 1996.

[2] C. Blum, 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 114, 2008, pp. 1–30.

[3] E.G. Talbi, Metaheuristics: from Design to Implementation, Wiley, USA, 2009.

[4] E.G. Talbi, A taxonomy of hybrid metaheuristics, J. Heuristics 8 (2002) 541–564.

[5] O. Rossi-Doria, M. Samples, M. Birattari, M. Chiarandini, M. Dorigo, L.M. Gambardella, J. Knowels, M. Manfrin, M. Mastrolilli, B. Paechter, L. Paquete, T. Stultzle, A comparison of the performance of different metaheuristics on the timetabling problem, in: E.K. Burke, P. De Causmaecker (Eds.), PATAT 2002, LNCS 2740, Springer, Heidelberg, 2003, pp. 329–354.

[6] R. Qu, E.K. Burke, B. McCollum, L.T.G. Merlot, S.Y. Lee, A survey of search methodologies and automated system development for examination timetabling, J. Sched. 12 (2009) 55–89.

[7] Y. Marinakis, M. Marinaki, A hybrid genetic – particle swarm optimization algorithm for the vehicle routing problem, Expert Syst. Appl. 37 (2010) 1446–1455.

[8] F. Glover, M. Laguna, R. Martí, Scatter search, in: A. Ghosh, S. Tsutsui (Eds.), Theory and Applications of Evolutionary Computation: Recent Trends, Springer, 2002, pp. 519–529.

[9] G. Jaradat, M. Ayob, An Elitist-Ant System for solving the post-enrolment course timetabling problem, in: Y. Zhang, et al. (Eds.), DTA/BSBT 2010, vol. 118, Springer, Heidelberg, CCIS, 2010, pp. 167–176.

[10] G. Jaradat, 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. 8 (2) (2013) 96–107.

[11] G. Jaradat, M. Ayob, Z. Ahmad, On the performance of scatter search for post-enrolment course timetabling problem, J. Comb. Optim. 27 (3) (2014) 417–439, ISSN:1382-6905.

[12] M. Dorigo, T. Stützle, Ant colony optimization: overview and recent advances, in: M. Gendreau, J.-Y. Potvin (Eds.), Handbook of Metaheuristics, International Series in Operations Research & Management Science, Chapter 8, vol. 146, Springer, 2010, pp. 227–263.

[13] K. Erol, I. Eksin, A new optimization method: Big Bang-Big Crunch, Adv. Eng. Softw. 37 (2006) 106–111.

[14] N. Sabar, M. Ayob, G. Kendall, R. Qu, A dynamic multi-armed bandit-gene expression programming hyper-heuristic for combinatorial optimization problems, IEEE Trans. Cybern. 45 (2) (2015) 217–228.

[15] A.L. Bolaji, A.T. Khader, M.A. Al-Betar, M.A. Awadallah, University course timetabling using hybridized artificial bee colony with hill climbing optimizer, J. Comput. Sci. 5 (5) (2014) 809–818.

[16] J.A. Soria-Alcaraz, G. Ochoa, J. Swan, M. Carpio, H. Puga, E.K. Burke, Effective learning hyper-heuristics for the course timetabling problem, Eur. J. Oper. Res. 238 (1) (2014) 77–86.

[17] V. Cacchiani, A. Caprara, R. Roberti, P. Toth, A new lower bound for curriculum-based course timetabling, Comput. Oper. Res. 40 (10) (2013) 2466–2477.

[18] A. Rodriguez, R. Ruiz, The effect of the asymmetry of road transportation networks on the traveling salesman problem, Comput. Oper. Res. 39 (7) (2012) 1566–1576.

[19] G. Laporte, The vehicle routing problem: an overview of exact and approximate algorithms, Eur. J. Oper. Res. 59 (1992) 345–358.

[20] D. Karaboga, B. Gorkemli, A combinatorial artificial bee colony algorithm for traveling salesman problem, in: 2011 international symposium on innovations in intelligent systems and applications (INISTA), 2011, pp. 50–53.

[21] G. Reinelt, The TSPLIB symmetric traveling salesman problem instances, 1995, Available in: http://elib.zib.de/pub/mp-testdata/tsp/tsplib/tsp/index.html.

[22] M. Albayrak, N. Allahverdi, Development a new mutation operator to solve the traveling salesman problem by aid of genetic algorithms, Expert Syst. Appl. 38 (3) (2008) 450–461.

[23] G. Reinelt, TSPLIB – a travelling salesman problem library, ORSA J. Comput. 3 (1991) 376–384.

[24] S. Hanafi, C. Wilbaut, Scatter search for the 0-1 multidimensional knapsack problem, J. Math. Model. Algorithms 7 (2) (2008) 143–159.

[25] P. Toth, D. Vigo, The Vehicle Routing Problem, vol. 9, Society for Industrial Mathematics, 2002.

[26] M. Dorigo, V. Maniezzo, A. Colorni, The ant system: optimization by a colony of cooperating agents, in: IEEE transactions on systems, man, and cybernetics – part B, vol. 26, 1996, pp. 29–42.

[27] J.H. Yang, C.G. Wu, H.P. Lee, Y.C. Liang, Solving traveling salesman problems using generalized chromosome genetic algorithm, Prog. Nat. Sci. 18 (2008) 887–892.

[28] J.J. Pantrigo, A. Duarte, Á. Sánchez, R. Cabido, Scatter search particle filter to solve the dynamic travelling salesman problem, in: G.R. Raidl, J. Gottlieb (Eds.), EvoCOP 2005, LNCS 3448, 2005, pp. 177–189.

[29] P.C. Chen, G. Kendall, G.V. Berghe, An ant based hyper-heuristic for the travelling tournament problem, in: Proceedings of the 2007 IEEE symposium on computational intelligence in scheduling (CI-Sched 2007), 2007.

[30] A. Puris, R. Bello, Y. Martinez, A. Nowe, Two-stage ant colony optimization for solving the traveling salesman problem, in: J. Mira, J.R. Alvarez (Eds.), IWINAC 2007, Part II, LNCS 4528, 2007, pp. 307–316.

[31] C.G. da Silva, J. Clímaco, J. Figueira, A scatter search method for the bi-criteria multi-dimensional {0,1}-knapsack problem using surrogate relaxation, J. Math. Model. Algorithms 3 (2004) 183–208.

[32] E.K. Burke, M.R. Hyde, G. Kendall, G. Ochoa, E. Özcan, J.R. Woodward, Automating the packing heuristic design process with genetic programming, Evol. Comput. 20 (1) (2012) 63–89.

[33] C.L. Alonso, F. Caro, J.L. Montana, A flipping local search genetic algorithm for the multidimensional 0-1 Knapsack problem, in: R. Marın, et al. (Eds.), CAEPIA 2005, LNAI 4177, 2006, pp. 21–30.

[34] E. Taillard, Parallel iterative search methods for vehicle routing problems, Networks 23 (1993) 661–676.

[35] D. Mester, O. Braysy, Active-guided evolution strategies for large-scale vehicle routing problems, Comput. Oper. Res. 34 (2007) 2964–2975.

[36] Y. Rochat, E. Taillard, Probabilistic diversification and intensification in local search for vehicle routing, J. Heuristics 1 (1995) 147–167.

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

[38] J.A. Castillo-Salazar, D. Landa-Silva, Q. Rong, Computational study for workforce scheduling and routing problems, in: International Conference on Operations Research and Enterprise Systems, ICORES 2014, 2014, pp. 434–444.

[39] S. Ceschia, L. Di Gaspero, A. Schaerf, Tabu search techniques for the heterogeneous vehicle routing problem with time windows and carrier-dependent costs, J. Sched. 14 (6) (2011) 601–615.

[40] E.T. Yassen, M. Ayob, M. Zakree, A. Nazri, N.R. Sabar, Multi-parent insertion crossover for vehicle routing problem with time windows, in: 4th conference on data mining and optimization, IEEE, 2012, ISBN:978-1-4673-2718-12.

[41] N. Sabar, M. Ayob, G. Kendall, R. Qu, Grammatical evolution hyper-heuristic for combinatorial optimization problems, IEEE Trans. Evol. Comput. 17 (2013) 840–861.

[42] S.C. Ho, M. Gendreau, Path relinking for the vehicle routing problem, J. Heuristics 12 (2006) 55–72.

[43] W.Y. Szeto, Y. Wu, S.C. Ho, An artificial bee colony algorithm for the capacitated vehicle routing problem. Production, manufacturing and logistics, Eur. J. Oper. Res. 215 (2011) 126–135.

[44] M. Dorigo, V. Maniezzo, A. Colorni, Positive feedback as a search strategy, Technical Report 91-016, Dipartimento di Elettronica e Informazione, Policecnico di Milano, Italy, 1991.

[45] H.M. Genc, A.K. Hocaoglu, Bearing-only target tracking based on Big Bang-Big Crunch algorithm, in: The Proceedings of the 3rd International Multi-Conference on Computing in the Global Information Technology, 2008, pp. 229–233, http://dx.doi.org/10.1109/ICCGI.2008.53.

[46] M. Laguna, Scatter search and path relinking, in: M. Ehrgott, et al. (Eds.), EMO 2009, LNCS 5467, Springer-Verlag, Berlin, Heidelberg, 2009, p. 1.

[47] G. Clarke, J.W. Wright, Scheduling of vehicles from a central depot to a number of delivery points, Oper. Res. 12 (1964) 568–581.

[48] N. Christofides, A. Mingozzi, P. Toth, The vehicle routing problem, Comb. Optim. 11 (1979) 315–338.

[49] J.E. Beasley, OR-Library: distributing test problems by electronic mail, J. Oper. Res. Soc. 41 (11) (1990) 1069–1072.

[50] J. Montgomery, M. Randall, The accumulated experience ant colony for the travelling salesman problem, Int. J. Comput. Intell. Appl. 3 (2) (2003) 189–198.

[51] H. Hoos, T. Stützle, in: H.H. Hoos, T. Stützle (Eds.), Stochastic Local Search: Foundations and Applications, Morgan Kaufmann, Elsevier, 2005, ISBN:1-55860-872-9.

[52] M.G.C. Resende, C.C. Ribeiro, F. Glover, R. Martí, Scatter search and path-relinking: fundamentals, advances and applications, in: M. Gendreau, J.-Y. Potvin (Eds.), The Handbook of Metaheuristics, 2nd ed., Springer, 2010.

[53] B. Li, L. Wang, W. Song, Ant colony optimization for the traveling salesman problem based on ants with memory, in: Fourth international conference on natural computation, IEEE 978-0-7695-3304-9/08, 2008, http://dx.doi.org/10.1109/ICNC.2008.354.