A Bee Colony Optimization Algorithm for the Long-Term Car Pooling Problem

Mouna Bouzid, Ines Alaya and Moncef Tagina
COSMOS-ENSI, National School of Computer Sciences, University of Manouba, Manouba 2010, Tunisia

Keywords: Metaheuristic, Bee Colony Optimization, The Long-Term Car Pooling Problem.

Abstract: Recently, the big number of vehicles on roadways and the increase in the rising use of private cars have made serious and significant traffic congestion problems in large cities around the world. Severe traffic congestion can have many detrimental effects, such as time loss, air pollution, increased fuel consumption and energy waste. Public transportation systems have the capacity to decrease traffic congestion and be an answer to this increasing transport demand. However, it cannot be the only solution. Another recommended solution for reducing the harmful factors leading to such problems is car pooling. It is a collective transportation system based on the idea that a person shares his private vehicle with one or more people that have the same travel destination.

In this paper, a Bee Colony Optimization (BCO) metaheuristic is used to solve the Car Pooling Problem. The BCO model is based on the collective intelligence shown in bee foraging behavior. The proposed algorithm is experimentally tested on benchmark instances of different sizes. Computational results show the effectiveness of our proposed algorithm when compared to several state of the art algorithms.

1 INTRODUCTION

Swarm intelligence is a field of nature inspired metaheuristics that was successfully applied to many optimization problems. We can cite the Ant colony optimization (Dorigo et al., 2006; Zouari et al., 2017), the Particle swarm intelligence metaheuristic (Huang, 2015; Wu et al., 2011; Liu and Qin, 2014; Nouiri et al., 2018; Karaboga, 2005), and the swarm intelligence based on bees (Karaboga and Basturk, 2008; Karaboga and Akay, 2011; Bacanin et al., 2010; Subotic et al., 2010; Xue et al., 2018; Jadon et al., 2018).

The Bee Colony Optimization (BCO) is a metaheuristic in which the principal idea is taken from the analogy between the natural behavior of bees searching for food, and the behavior of optimization algorithms searching for an optimum of combinatorial optimization problems. Based on exploration and exploitation ability, bees are able to intensify and to diversify the search at the same time.

In fact, the BCO metaheuristic is a competitive approach for solving hard combinatorial optimization problems, such as the stochastic Vehicle Routing Problem (Lučić and Teodorović, 2003b; Teodorović, 2008), Job Shop Scheduling Problem (Chong et al., 2006; Chong et al., 2007; Wong et al., 2010b), p-Median Problem (Teodorovic and Šelmic, 2007) and Traveling Salesman Problem (Wong et al., 2010a).

In this paper, our research is focused in another problem named car pooling. This problem is a collective transportation system based on the idea that sets of car owners having the same travel destination share their vehicles. It has emerged to be a viable possibility for reducing private car usage around the world.

Car pooling has already been considered as an important alternative transportation service throughout the world. In fact, many organizations such as large companies or public administrations encourage their citizens or employees to pick up or take back colleagues while driving to or from a common site. The benefits which can be obtained are particularly relevant both in terms of reduction of the use of private cars and of the parking space required.

Thanks to massive use of internet, car pooling becomes more and more popular. Therefore, the number of applications proposing car pool increase day after day, but they are not optimized, and use generally the FiFo ”First in First out” principle. In addition, after bibliographical studies, it appears that the BCO algorithm has never been used for solving the Long-term Car Pooling Problem (LTCPP). The aim of this work is to propose an intelligent approach based on the BCO algorithm to solve the Long-term Car Pooling Problem (LTCPP), which can be deployed in such application. In order to test the proposed algorithm we used various benchmark instances.

The different parts of this paper are organized as follows. In the next section, we define the Long-term Car Pooling Problem. We recall in section 3 the basic idea of BCO. In section 4, we describe the proposed BCO algorithm for solving our problem. Experiments and results are shown in section 5. Finally, conclusions and perspectives are presented.

2 THE LONG-TERM CAR POOLING PROBLEM

In this section, we give an overview of the Car Pooling Problem and we present a mathematical formulation for the LTCPP.

2.1 An Overview of Car Pooling Problem

Car pooling (Bruglieri et al., 2011; Manzini et al., 2012; Correia and Viegas, 2011; Yan et al., 2011; Vargas et al., 2008) is a collective transportation system. In this transport service, several people share a common vehicle simultaneously, in order to reach common destinations. The principal objective of this mobility service is to decrease the number of private vehicles.

By joining the car pooling system, many advantages are offered. In fact, it saves time, parking spaces, and decreases the number of accidents. Furthermore, it encourages sociability between people since it creates social interaction between colleagues, friends and neighbors. By sharing journey expenses, such as, tolls fuel, costs, and road stress, travel costs can be reduced. Car pooling can make our environment more friendly, ecological and sustainable way to travel as sharing journeys reduces carbon emissions and traffic congestion on the roads.

Car pooling problem can be operated in two different ways. It can be either a Daily Car Pooling Problem (DCPP) or a Long-term Car Pooling Problem (LTCPP).

In the case of DCPP (Baldacci et al., 2004; Calvo et al., 2004; Swan et al., 2013), on each day a number of users, considered as servers, are available for picking up and later bringing back colleagues (clients) on that particular day. Then, the problem asks to assign clients to servers and to identify which routes should be driven by the servers in order to minimize the number of unassigned clients, subject to time window and car capacity constraints.

However, in the case of LTCPP, each user is available to act both as a server and as a client. The LTCPP requires to find user pools or crews where each user will in turn, on different days, picks up the remaining pool members and to identify the routes to be driven by each member in the car pool. The main objective of this problem is to minimize the number of vehicles and the length of the path traveled by all users when acting as servers subject to car capacity and time window constraints (Guo, 2012).

The Long-term Car Pooling Problem can be seen as a compound problem. It is a combination of a clustering problem and a routing problem. This model is frequently used by universities and large compa-

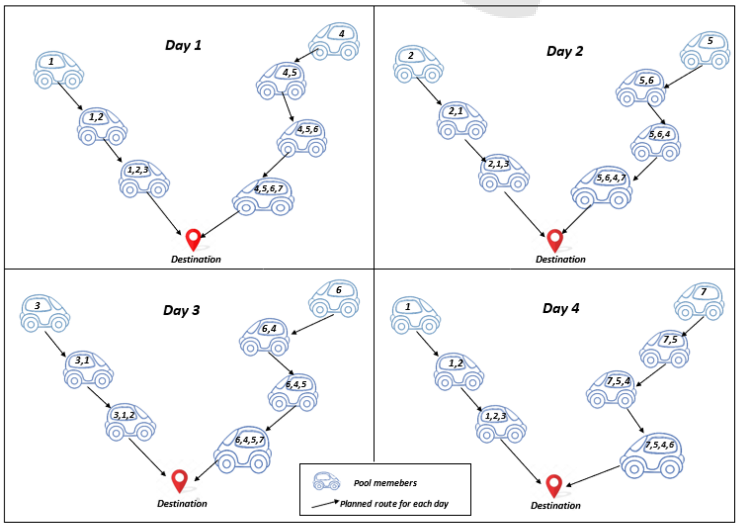

Figure 1: An example of the LTCPP.

nies which provide long-term carpool service for their students or employees, since it is the most stable car pooling model. Indeed, in the LTCPP users will not usually be changing (in a relatively long period). Therefore, the maximum number of users in a pool is equal to the capacity of the smallest car among those owned by all pool members, because finally each member should pick up all the other ones. Figure 1 presents an example of the LTCPP. In this example we suppose that we have 7 users and the maximum car capacity is 4. In the first day, users 1, 2 and 3 are pooled together where 1 is considered as server. Users 4, 5, 6 and 7 construct another pool and the user 5 is considered as server. In the other days, each member could be considered as server.

In this article, our study focuses on the Long-term Car Pooling Problem, which is an NP-complete problem (Varrentrapp et al., 2002). Due to its hardness and significance it has constantly been studied by researchers who approached it from various perspectives. After bibliographical study, we noticed that different heuristics and metaheuristics have been developed by different authors. We can cite the Saving Functions Based Algorithm (Ferrari et al., 2003), an Approximated Non-deterministic Tree Search (ANTS) algorithm (Maniezzo et al., 2004), a Simulation Based Approach (SB) (Correia and Viegas, 2008), a Multi-Matching System (Yan et al., 2011), a Clustering Ant Colony Algorithm (CAC) (Guo et al., 2012), a Guided Genetic Algorithm (GGA) (Guo et al., 2011).

2.2 LTCPP Formulation

The LTCPP can be modeled, by a direct graph $G = \{U \cup \{0\}, A\}$, where U is the set of users, and $A = \{arc(i,j)/i \in U, j \in U \cup \{0\}\}$ is the set of directed weighted arcs where each $arc(i,j) \in A$ is associated with a non-negative travel cost d_{ij} and a travel time t_{ij}. Every user $i \in U$ for the Long-term Car Pooling is characterized by: the origin (home), the node 0 that corresponds to the destination, the earlier time e_i for leaving home; the acceptable time r_i for arriving at destination; the capacity Q_i of his car; furthermore the maximum time T_i that the user will accept to drive.

The LTCPP can be seen as a multi-objective problem, looking for minimizing the total number of vehicles traveling to the common destination and the distance to be driven by each user, when acting as a driver.

However, based on the research of Varrentrapp, Maniezzo and Stützle (Varrentrapp et al., 2002), we can combine these two objectives in a single objective function. Then, the LTCPP can be considered as a single-objective problem.

Each user of pool k, will use on different days his car to pick up his pool mates and drive to the common destination. Therefore, each driver must know his Hamiltonian path starting from his origin (his home), then linking all other nodes corresponding to other pool members' homes exactly once, until reaching the common destination.

Let $path(i,k)$ be a feasible Hamiltonian path, starting from $i \in k$, linking all $j \in k \backslash \{i\}$ and ending in 0. Suppose $|k| \leq Q_k$, where $|k|$ is the size of this pool, Q_k is the minimum car capacity of pool k and all constraints are satisfied. d_{i0} denote the travel cost from user's home directly to the destination, while ρ_i is a penalty imposed for a client driving alone.

The cost of a pool k is then computed as follows:

$$Cost(k) = \begin{cases} \sum_{i \in k} \frac{cost(path(i,k))}{|k|}, if |k| > 1, \\ \sum_{i \in k} d_{i0} + \rho_i, otherwise. \end{cases}$$

The total cost of a complete solution to the LTCPP is then equal to the sum of the costs of the pools in it. By this view, both objective functions can be optimized at the same time. In fact, provided that the penalty of a client driving alone is sufficiently greater than zero, it is better to pool clients together than to leave them alone.

In order to formulate this problem, we present the following notations (Guo et al., 2012):

- K : Set of indices of all pools;
- U : Set of indices of all users;
- A : Set of indices of all arcs;
- X_{ij}^{hk}: Binary variable equals to 1 if arc(i,j) is traveled by a server h of a pool k;
- Y_{ik}: Binary variable equals to 1 if user i is in pool k;
- S_i^h: Positive variable indicating the pick-up time of user i by server h;
- F_i^h: Positive variable for denoting the arrival time of user i at destination when traveling with server h;
- d_{ij}: Positive value indicating the travel cost between users i and j;
- t_{ij}: Positive value indicating the travel time between users i and j;
- ρ_i: Positive value for denoting the penalty for user i when he travels alone;
- φ_i: Binary variable equals to 1 if user i is not pooled with any other user;
- e_i: Positive value denoting the earlier time for leaving home of user i;

- r_i: Positive value indicating the acceptable time for arriving at work of user i;
- Q_k: Positive value denoting the minimum car capacity of pool k;
- T_h: Positive value for denoting the extra driving time specified by server h;

Objective Function:

F= $\min(\sum_{k\in K}\frac{\sum_{h\in U}\sum_{(i,j)\in A} d_{ij}X_{ij}^{hk}}{\sum_{i\in U} Y_{ik}}+\sum_{i\in U}\rho_i\varphi_i)$

Constraints:

$$\sum_{j\in U\setminus\{h\}} X_{ij}^{hk}=Y_{ik} \qquad i,h\in U; k\in K; \quad (1)$$

$$\sum_{j\in U} X_{ji}^{hk}=Y_{ik} \qquad i,h\in U; k\in K; \quad (2)$$

$$\sum_{j\in U} X_{ij}^{hk}=\sum_{j\in U} X_{ji}^{hk} \qquad i,h\in U; k\in K; \quad (3)$$

$$\sum_{k\in K} Y_{ik}+\varphi_i=1 \qquad i\in U; \quad (4)$$

$$\sum_{(i,j)\in A} X_{ij}^{hk}\leq Q_k \qquad h\in U; k\in K; \quad (5)$$

$$\sum_{(i,j)\in A} X_{ij}^{hk}t_{ij}\leq T_h \qquad h\in U; k\in K; \quad (6)$$

$$S_i^h\geq e_i \qquad i,h\in U; \quad (7)$$

$$S_j^h-S_i^h\geq t_{ij}-M*(1-\sum_{k\in K} X_{ij}^{hk}) \quad (i,j)\in A; h\in U; \quad (8)$$

$$F_i^h\geq S_i^h+t_{i0}-M*(1-\sum_{k\in K} X_{i0}^{hk}) \qquad i,h\in U; \quad (9)$$

$$F_i^h\leq r_i+M*(1-\sum_{k\in K}\sum_{j\in U} X_{ij}^{hk}) \qquad i,h\in U; \quad (10)$$

$$X_{ij}^{hk}\in\{0,1\} \qquad h\in U; k\in K; (i,j)\in A; \quad (11)$$

$$Y_{ik}\in\{0,1\} \qquad i\in U; k\in K; \quad (12)$$

$$\varphi_i\in\{0,1\} \qquad i\in U; \quad (13)$$

$$S_i^h\geq 0 \qquad i,h\in U; \quad (14)$$

$$F_i^h\geq 0 \qquad i,h\in U; \quad (15)$$

If there is an arc-connecting user i and user j or user j and user i, equations (1) and (2) force a user i to be added in a pool k. Constraint (3) makes sure the continuity of the path. Equation (4) shows that each user must be assigned to a pool or be penalized. The car capacity and the extra driving time constraints are translated respectively in (5) and (6). Equations (7) and (8), where M is a big constant, collectively set feasible pick-up times, while (9) and (10) are minimum and maximum values of feasible arrival times, respectively. Constraints (11) and (12) and (13) are binary constraints while (14) and (15) are positivity constraints.

3 THE BEE COLONY OPTIMIZATION ALGORITHM

The BCO is a population-based algorithm inspired by bees' behavior in the nature. It was proposed for dealing with hard combinatorial optimization problems. The main idea of this approach is to build the multi agent system (colony of artificial bees) where each artificial bee called agent, can generate one solution to the problem. Looking for the best feasible solutions, artificial bees investigate through the search space. After that, they cooperate and exchange information. Using collective knowledge and information sharing, artificial bees update their current solution and then, concentrate on the more promising areas and incrementally abandon solutions from the less promising ones. Step by step, this population of agents collectively generate and/or yield their solutions. The BCO algorithm runs iteratively until a stopping condition is met.

The basic principles of collective bee intelligence in solving combinatorial optimization problems were proposed for the first time by Lucic and Teodorovic (Lučić and Teodorović, 2003b; Lučić and Teodorović, 2003a).

The algorithm is composed of two alternating phases: forward pass and backward pass. During every forward pass, each artificial bee is exploring the search space. It makes a predefined number of local moves, which gradually construct and/or improve the solution, yielding to a new solution. Having created various partial solutions, they return to the nest and perform the second phase, called backward pass. In the hive, the artificial bees share information about their solutions.

In nature, bees communicate through a waggle dance which would notify the other bees about the quantity and the quality of nectar they have discovered, and the proximity of the path to the hive. In the BCO algorithm, the artificial bees publicize the quality of the solution created, i.e. the objective function value. During the backward pass, bees compare all partial generated solutions. Based on a certain probability, each bee decides whether to stay faithful to its created partial solution or not. Note that, bees with higher objective function value have greater chance to continue their own exploration. Hence, if bees dance and thus recruit the nestmates before returning to the created partial solution, they are considered as recruiters. Moreover, if a bee chooses to become uncommitted follower, it must select a new solution from recruiters by the roulette wheel (better solutions have higher opportunities of being chosen for exploration).

The two phases (forward and backward pass) alternate, until a stopping condition is met. There are different possible stopping conditions, like, the maximum total number of forward/backward passes without the improvement of the objective function, the maximum total number of forward/backward passes, etc.

In the first stage of the search, all the bees are located in the hive.

4 THE PROPOSED BCO ALGORITHM FOR SOLVING THE LTCPP

The basic idea of the Bee Colony Optimization (BCO) is to model the problem to solve as the search for a minimum cost path in a graph, and to use artificial bees to search for good paths. Let's represent each user for the Long-term Car Pooling Problem by a node. In our proposed algorithm, each bee is allowed to explore and search for a car pool. The bee starts to build a car pool by randomly determining the first user to be added to its car pool initially empty. We decompose our problem into stages. Each stage is associated to an unpooled user. In fact, the first user in the car (driver) represents the first stage; the second user to join the pool represents the second stage, etc. In every stage, the bee chooses to visit one node.

Thus during the forward pass, every bee travels to a node. It behaves according to a selection process named the roulette wheel selection. Based on the probability values, this model of choice is employed to aid the bee in its decision making on which user to be pooled. The probability gives to the bee the likelihood to move from user i to user j. In fact, if constraints presented previously (in section 2.1) are satisfied, the probability value between two different users i and j, P_{ij} , is defined as follows:

$$P_{ij} = \frac{\frac{1}{d_{ij}}}{\sum_{k \in U} \frac{1}{d_{ik}}} \qquad (16)$$

Where d_{ij} represents the travel cost from user i to user j. Note that the travel cost is inversely proportional to P_{ij}. In other words, the shorter the distance, the higher the likelihood of that user to be chosen.

After that, the bee returns to the hive (in the backward pass), where it participates in a decision making process. Then, every bee decides whether to abandon their created partial solution and become again uncommitted follower, or to continue its own exploration and become recruiter.

During the second forward pass, each bee visits a new node, creates a partial solution, and after that performs again a backward pass and returns to the hive.

Then, it performs a third forward pass, etc.The two phases of the search algorithm, forward and backward pass, are alternated iteratively, until the total number of forward/backward passes reaches the car pool size. Then, all generated pools are evaluated and the best one is added to the current solution.

One iteration is finished, when all the users are clustered by bees. The algorithm terminates when a maximal number of iterations is reached.

The overall structure of the BCO-LTCPP is outlined in Algorithm 1.

Algorithm 1: BCO-LTCPP Algorithm.

```
Initialize parameters: Number of bees n and Maximum Iteration;
While Maximum Iteration is not met do
Initialize the car pool of every bee b $G_b \leftarrow \emptyset$;
Initialize the current solution $S_{cur} \leftarrow \emptyset$;
Initialize the current set of users not yet pooled $U_{cur} \leftarrow U$;
   Repeat
     Select randomly a new user u;
     $NC_u$ ←the car capacity of user u;
     For every bee b do{
        Insert u into $G_b$ and eliminate it from $U_{cur}$;}
     $j \leftarrow 1$;
     While $(j < NC_u)$do{
        (a) The forward pass
         For every bee b do{
             Evaluate all unserved users in $U_{cur}$;
             Choose an unserved user i who satisfies
              constraints using the roulette wheel
             selection based on probability $P_{ij}$(16);
             Insert user i into $G_b$ and eliminate it
             from $U_{cur}$;}
        (b)The backward pass
        For every bee b do{
             Every bee decides randomly whether to
             continue its own exploration and
             becomes recruiter, or to become
             uncommitted;
             If (b is uncommitted) then
                 Randomly chooses a recruiter to
                  follow;}
        $j \leftarrow j+1$;}
     Find the best pool $G-best_u$ from all pools $G_b$;
     $S_{cur} \leftarrow S_{cur} \bigcup G-best_u$;
   Until all users are clustered;
Update the best solution;
End
Output the best solution;
```

5 EXPERIMENTAL STUDY AND DISCUSSION

This section describes, first, the benchmark problems used in our experiments. To test the efficiency and the performance of our algorithm, we have chosen to compare our experimental results with four approaches for solving the LTCPP: the GGA, the Ants, the CAC and the simulation based approach (SB), since we haven't found any other BCO in the literature for the LTCPP and because all these approaches have been proven to have the ability to solve our problem and provide high solution quality. Since the CAC is one of the most recent swarm intelligence metaheuristic for solving the Long-term Car Pooling Problem, and, in order to compare both of the algorithms in the same environment, we have implemented the CAC algorithm and tested it exactly as it was described in (Guo et al., 2012), and we have maintained the same parameter values. The results of GGA, Ants and SB are obtained directly from (Guo, 2012).

5.1 Benchmark Problems

Since there are no benchmarks for the LTCPP in the literature, we have chosen to use in our experimental study, the benchmarks developed in (Guo, 2012), which are originally derived from the Pickup and Delivery Problems with Time Windows (PDPTW) benchmarks by Li and Lim (Li and Lim, 2003).

These benchmarks include two sets of instances where each one is composed of 9 instances with the number of users ranging from 100 to 400. The first set has the users clustered distributed, therefore is named with C. The users in the second set are allocated randomly, so the set is named with R.

For all instances, we considered the depot in the original instances as the destination, while we retained the coordinates of the customers, who become the users. The cost d_{ij} was assumed to be equal to the Euclidean distance between user i and j. Travel times t_{ij} were set equal to the distances divided by 50 km/h (average travel speed). For each user, the penalty ρ_i was calculated as $\rho_i = 2d_{i0}$, where we denoted by d_{i0} the travel cost from user i's home directly to the destination, and the car capacity Q_k was set to 4. The maximum ride time T_k was computed as $T_k = 1.5t_{i0}$, where t_{i0} represents the time needed to travel from the user i's home to the destination. The latest arrival times r_i is an integer value randomly selected in the interval [510,540], and the earliest departure time of user i was estimated to be equal to $e_i = r_i - max(t_{i0} + 30, 2t_{i0})$.

5.2 Computational Results

For both algorithms, there are some control parameters, which are used for its efficient performance. After some times of test runs, the parameters settings of our algorithm are specified as follows:
-Number of iterations: IT = 1000,
-Number of bees: n = number of users in the instance.
The BCO-LTCPP algorithm was implemented in JAVA, and all the tests were performed on computer with Intel core i5-3317U, 1.70 GHz and 6 Go of RAM.

5.2.1 Comparison of BCO-LTCPP with State-of-the Art Algorithms

We have performed 10 runs for each instance. The results presented in this study are the averages of ten runs (AVG). We also report the best result (Best) among the ten replications. In addition, the results are also analyzed through the non-parametric Wilcoxon rank-signed test (Sheskin, 2003) (W-test) in order to verify if the difference between the compared algorithms is statistically significant. Note that for W-Test, the level of significance considered is 0.05. We use (+) and (-) to denote if the BCO-LTCPP result is, respectively, significantly or not significantly better than the other algorithms.

Table 1 summarizes the results of the C set instances. On this set of instances, the BCO-LTCPP outperforms the GGA, Ants, SB and CAC on all instances in best found solution. Considering the average solution quality, our algorithm outperforms the GGA and Ants on 8 instances, and outperforms the CAC and SB on all instances. Besides, the superiority of our algorithm is statistically clear. In fact, the W- test shows that statistically our algorithm outperforms widely the other algorithms (GGA, Ants and SB). Also, statistically, the BCO-LTCPP is significantly better than the CAC in seven instances among the nine instances.

Table 2 shows the experimental results of set R instances. In fact, we can draw that the BCO-LTCPP outperforms the GGA, Ants, SB and CAC on all instances in best-found solution. Also, the BCO-LTCPP outperforms the GGA and Ants on 8 instances and the CAC and SB on all instances, considering the average solution. Furthermore, our algorithm is statistically a clear winner when compared to the GGA, the Ants and the SB. In addition, the W-test shows that the results of the BCO-LTCPP algorithm are significantly better than the CAC algorithm on eight instances.

Table 1: Experimental results of set C instances for BCO, GGA, Ants, SB and CAC algorithms.

Instance	**Size**	**BCO-LTCPP**		**GGA**		**Ants**		**SB**		**CAC**		
		Best	**AVG**	**Best**	**AVG**	**Best**	**AVG**	**Best**	**AVG**	**Best**	**AVG**	W-test
C101	100	1491	1576.1	1585,5	1599,3	1585.5	1592,9	1647,4	1669,2	1523	1595.4	(+)
C102	100	1617	1696	1701,9	1712	1711.4	1748,5	1717,5	1724,8	1652	1722.9	(-)
C103	100	1503	1545	1513,7	1543,9	1512.6	1535,1	1532,2	1599,4	1524	1559.8	(+)
C201	200	2600	2653.5	2672,2	2749,4	2784.4	2854,2	2761,7	2868,6	2660	2754.3	(+)
C202	200	2765	2817.7	2836,7	2876,5	2936.1	3004,5	3081,7	3114,1	2816	2912.6	(+)
C203	200	2709	2785	2716	2891,8	2845.9	3003,5	2975,1	3182,4	2789	2874.9	(+)
C401	400	5308	5496.2	5489,4	5690,6	5833.5	6281,4	6174,2	6860,3	5661	5727	(+)
C402	400	4350	4586.2	4548,3	4786,4	4893.5	5153,2	5383,7	5524,5	4552	4590.9	(-)
C403	400	5776	5917.7	5909,6	6085,2	6125.6	6742,1	6675,2	6994,5	5803	5974.5	(+)
W-test				(+)	(+)	(+)	(-)	(+)	(+)			

Table 2: Experimental results of set R instances for BCO, GGA, Ants, SB and CAC algorithms.

Instance	**Size**	**BCO-LTCPP**		**GGA**		**Ants**		**SB**		**CAC**		
		Best	**AVG**	**Best**	**AVG**	**Best**	**AVG**	**Best**	**AVG**	**Best**	**AVG**	W-test
R101	100	2136	2197,2	2207,1	2235,9	2207.1	2281,5	2235,1	2265,4	2199	2254.4	(+)
R102	100	1801	1898,2	1824,5	1867,5	1834.6	1864,2	1832,8	2091,7	1906	1976.2	(+)
R103	100	2098	2146,7	2209,1	2286	2299.2	2438,7	2204,7	2418,5	2102	2177.9	(+)
R201	200	3933	4054,4	4034,8	4188,3	4101.5	4253,5	4425	4567,1	4190	4272.2	(+)
R202	200	3598	3655,3	3646,8	3751,7	3772.2	4071,9	3952,4	4283,3	3698	3771.2	(+)
R203	200	3852	4017,8	3923,2	4158,4	4368.5	4541,5	4092,4	4257,5	4242	4372.9	(+)
R401	400	7441	7659,7	7514,9	7799,5	8396.1	8580,4	8787,8	8993,8	7777	7894.6	(+)
R402	400	6023	6233,6	6172,7	6254	6512.7	6893,3	7258,7	7417,5	6045	6186.5	(-)
R403	400	7403	7617,5	7670,2	7872,9	8113.1	8338,9	8841,9	8933,5	7885	7978.8	(+)
W-test				(+)	(+)	(+)	(+)	(+)	(+)			

Table 3: CPU time of the different algorithms (in seconds).

Instances	BCO	GGA	Ants	SB	CAC
C101	2	13	17	91	8
C102	3	9	14	94	7
C103	2	12	18	85	8
C201	14	31	57	329	39
C202	14	28	64	473	43
C203	14	42	58	394	39
C401	46	248	424	934	231
C402	42	203	357	683	243
C403	46	295	511	1257	263
R101	2	18	18	100	6
R102	2	15	17	97	7
R103	3	17	21	80	7
R201	15	43	108	430	35
R202	16	41	84	231	33
R203	12	38	116	540	34
R401	59	392	581	1106	217
R402	49	277	479	896	231
R403	55	311	631	1037	237
Computer \ Langage	Intel core i5-3317U 1.70GHz (JAVA)	Intel core i7 740QM 2.9GHz (JAVA)	Intel core i7 740QM 2.9GHz (JAVA)	Intel core i7 740QM 2.9GHz (JAVA)	Intel core i5-3317U 1.70GHz (JAVA)

5.2.2 Comparison of CPU Time of the Different Algorithms

Table 3 presents the elapsed CPU time to obtain the best solution. In this table, we can see that, the computing time of the BCO-LTCPP is much lower than other approaches. We note that these values are given as an indication since the different algorithms are tested on different computers. However, we remark that the proposed BCO-LTCPP algorithm, despite it was run on the less powerful processor, takes much less CPU time.

Therefore, we can conclude that our algorithm is able to reach significant better solutions in a short time.

6 CONCLUSIONS

In this paper, we proposed a Bee Colony Optimization algorithm for the Long-term Car Pooling Problem. The proposed BCO algorithm, called BCO-LTCPP, was tested on various benchmark instances. Based on the experimental results, we can conclude that the BCO-LTCPP is an efficient approach to solve the Long-term Car Pooling Problem. In fact, these results show that the proposed algorithm is able to reach significantly better solutions in a very short computational time when compared to other competitive approaches from literature on all instances.

The effectiveness of the developed BCO algorithm encourages its application, for future works, to other transportation problems, such as the daily carpooling problem or the academic vehicle routing Problem.

REFERENCES

Bacanin, N., Tuba, M., and Brajevic, I. (2010). An object-oriented software implementation of a modified artificial bee colony (abc) algorithm. In *Proceedings of the 11th WSEAS international conference on nural networks and 11th WSEAS international conference on evolutionary computing and 11th WSEAS international conference on Fuzzy systems*, pages 179–184. World Scientific and Engineering Academy and Society (WSEAS).

Baldacci, R., Maniezzo, V., and Mingozzi, A. (2004). An exact method for the car pooling problem based on lagrangean column generation. *Operations Research*, 52(3):422–439.

Bruglieri, M., Ciccarelli, D., Colorni, A., and Luè, A. (2011). Poliunipool: a carpooling system for universities. *Procedia-Social and Behavioral Sciences*, 20:558–567.

Calvo, R. W., de Luigi, F., Haastrup, P., and Maniezzo, V. (2004). A distributed geographic information system for the daily car pooling problem. *Computers & Operations Research*, 31(13):2263–2278.

Chong, C. S., Low, M. Y. H., Sivakumar, A. I., and Gay, K. L. (2006). A bee colony optimization algorithm to job shop scheduling. In *Proceedings of the 2006 winter simulation conference*, pages 1954–1961. IEEE.

Chong, C. S., Low, M. Y. H., Sivakumar, A. I., and Gay, K. L. (2007). Using a bee colony algorithm for neighborhood search in job shop scheduling problems. In *21st European conference on modeling and simulation (ECMS 2007)*.

Correia, G. and Viegas, J. M. (2011). Carpooling and carpool clubs: Clarifying concepts and assessing value enhancement possibilities through a stated preference web survey in lisbon, portugal. *Transportation Research Part A: Policy and Practice*, 45(2):81–90.

Correia, G. H. d. A. and Viegas, J. M. (2008). Structured simulation-based methodology for carpooling viability assessment. Technical report.

Dorigo, M., Birattari, M., and Stutzle, T. (2006). Ant colony optimization. *IEEE computational intelligence magazine*, 1(4):28–39.

Ferrari, E., Manzini, R., Pareschi, A., Persona, A., and Regattieri, A. (2003). The car pooling problem: Heuristic algorithms based on savings functions. *Journal of Advanced Transportation*, 37(3):243–272.

Guo, Y. (2012). *Metaheuristics for solving large size long-term car pooling problem and an extension.* PhD thesis, Artois.

Guo, Y., Goncalves, G., and Hsu, T. (2011). A guided genetic algorithm for solving the long-term car pooling problem. In *2011 IEEE Workshop On Computational Intelligence In Production And Logistics Systems (CIPLS)*, pages 1–7. IEEE.

Guo, Y., Goncalves, G., and Hsu, T. (2012). A clustering ant colony algorithm for the long-term car pooling problem. *International Journal of Swarm Intelligence Research (IJSIR)*, 3(2):39–62.

Huang, C.-L. (2015). A particle-based simplified swarm optimization algorithm for reliability redundancy allocation problems. *Reliability Engineering & System Safety*, 142:221–230.

Jadon, S. S., Bansal, J. C., Tiwari, R., and Sharma, H. (2018). Artificial bee colony algorithm with global and local neighborhoods. *International Journal of System Assurance Engineering and Management*, 9(3):589–601.

Karaboga, D. (2005). An idea based on honey bee swarm for numerical optimization. Technical report, Technical report-tr06, Erciyes university, engineering faculty, computer ....

Karaboga, D. and Akay, B. (2011). A modified artificial bee colony (abc) algorithm for constrained optimization problems. *Applied soft computing*, 11(3):3021–3031.

Karaboga, D. and Basturk, B. (2008). On the performance of artificial bee colony (abc) algorithm. *Applied soft computing*, 8(1):687–697.

Li, H. and Lim, A. (2003). A metaheuristic for the pickup and delivery problem with time windows. *International Journal on Artificial Intelligence Tools*, 12(02):173–186.

Liu, Y. and Qin, G. (2014). A modified particle swarm optimization algorithm for reliability redundancy optimization problem. *Journal of Computers*, 9(9):2124–2131.

Lučić, P. and Teodorović, D. (2003a). Computing with bees: attacking complex transportation engineering problems. *International Journal on Artificial Intelligence Tools*, 12(03):375–394.

Lučić, P. and Teodorović, D. (2003b). Vehicle routing problem with uncertain demand at nodes: the bee system and fuzzy logic approach. In *Fuzzy sets based heuristics for optimization*, pages 67–82. Springer.

Maniezzo, V., Carbonaro, A., and Hildmann, H. (2004). An ants heuristic for the long—term car pooling problem. In *New optimization techniques in engineering*, pages 411–430. Springer.

Manzini, R., Pareschi, A., et al. (2012). A decision-support system for the car pooling problem. *Journal of Transportation Technologies*, 2(02):85.

Nouiri, M., Bekrar, A., Jemai, A., Niar, S., and Ammari, A. C. (2018). An effective and distributed particle swarm optimization algorithm for flexible job-shop scheduling problem. *Journal of Intelligent Manufacturing*, 29(3):603–615.

Sheskin, D. J. (2003). *Handbook of parametric and nonparametric statistical procedures*. crc Press.

Subotic, M., Tuba, M., and Stanarevic, N. (2010). Parallelization of the artificial bee colony (abc) algorithm. *Recent advances in neural networks, fuzzy systems & evolutionary computing*, pages 191–196.

Swan, J., Drake, J., Özcan, E., Goulding, J., and Woodward, J. (2013). A comparison of acceptance criteria for the daily car-pooling problem. In *Computer and information sciences III*, pages 477–483. Springer.

Teodorović, D. (2008). Swarm intelligence systems for transportation engineering: Principles and applications. *Transportation Research Part C: Emerging Technologies*, 16(6):651–667.

Teodorovic, D. and Šelmic, M. (2007). The bco algorithm for the p-median problem. *Proceedings of the XXXIV Serbian Operations Research Conferece*, pages 417–420.

Vargas, M. A., Sefair, J., Walteros, J. L., Medaglia, A. L., and Rivera, L. (2008). Car pooling optimization: a case study in strasbourg (france). In *2008 IEEE Systems and Information Engineering Design Symposium*, pages 89–94. IEEE.

Varrentrapp, K., Maniezzo, V., and Stützle, T. (2002). The long term car pooling problem–on the soundness of the problem formulation and proof of np-completeness. *Technische Universitat Darmstadt*.

Wong, L.-P., Low, M. Y. H., and Chong, C. S. (2010a). Bee colony optimization with local search for traveling salesman problem. *International Journal on Artificial Intelligence Tools*, 19(03):305–334.

Wong, L.-P., Puan, C. Y., Low, M. Y. H., Wong, Y. W., and Chong, C. S. (2010b). Bee colony optimisation algorithm with big valley landscape exploitation for job shop scheduling problems. *International Journal of Bio-Inspired Computation*, 2(2):85–99.

Wu, P., Gao, L., Zou, D., and Li, S. (2011). An improved particle swarm optimization algorithm for reliability problems. *ISA transactions*, 50(1):71–81.

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

Yan, S., Chen, C.-Y., and Lin, Y.-F. (2011). A model with a heuristic algorithm for solving the long-term many-to-many car pooling problem. *IEEE Transactions on Intelligent Transportation Systems*, 12(4):1362–1373.

Zouari, W., Alaya, I., and Tagina, M. (2017). A hybrid ant colony algorithm with a local search for the strongly correlated knapsack problem. In *2017 IEEE/ACS 14th International Conference on Computer Systems and Applications (AICCSA)*, pages 527–533. IEEE.