2014 IEEE Congress on Evolutionary Computation (CEC)
July 6-11, 2014, Beijing, China

Bee Colony Algorithm for the Routing of Guided Automated Battery-operated Electric Vehicles in Personal Rapid Transit Systems

Ezzeddine Fatnassi*, Olfa Chebbi*and Jouhaina Chaouachi Siala†
* Higher Institute of Management of Tunis
Tunis University
41, Liberty street - Bouchoucha - 2000 Bardo, Tunisia
Email: ezzeddine.fatnassi;olfaa.chebbi@gmail.com
†Institute of Advanced Business Studies of Carthage
Carthage University
2016 - Carthage-Presidency Tunis, Tunisia
Email: siala.jouhaina@gmail.com

***Abstract*—Personal rapid transit (PRT) systems are on-demand transportation services that use guided automated vehicles. The present study proposes an artificial bee colony (ABC) heuristic for solving the routing problem associated with PRTs. ABC is a swarm-based heuristic that mimics the behavior of bees. The present study proposes an enhanced version of this algorithm, which includes a specific method for escaping from local optima. Experimental results based on 1320 randomly generated instances are also presented and analyzed.**

I. Introduction

In congested cities, the rapid and safe movement of people among locations is a major problem for city planners, governments, and other stakeholders. Fortunately, scientists and engineers have developed a variety of different transportation tools that can relieve congestion in cities. In particular, the present study considers personal rapid transit (PRT) systems. A PRT is a futuristic system, which mainly comprises driverless electric vehicles (also called pods) that run on dedicated guideways(see figure 1). The aim of a PRT system is to transport up to six people on a direct nonstop transit service to any of its stations. The stations are located off the main line, thus each vehicle can proceed directly to its destination without unnecessary intermediate stops. A PRT is also an on-demand transportation service where the vehicles only move passengers upon request. Early PRTs were proposed in the 1960s, but only a few PRT projects have been completed or are in the planning stages (e.g., Morgan Town PRT in West Virginia, USA; Heathrow PRT in London; and Masdar PRT in Abu Dhabi, UAE). The on-demand feature of this system means that there is a high level of unused capacity and wasted energy because of empty vehicle movements.

A variety of PRT-related optimization problems have been reported previously. In 2006, Won *et al.* [2] presented four different optimization problems relate to the design process and costs of PRT traffic, where they formulated an empty vehicle allocation problem, a vehicle routing problem, a station design problem, and a guideway network design problem for PRTs. The problem of optimizing the size of PRT fleets by minimizing the number of vehicles in a PRT system has also been studied [3]. In 2012, Lees-Miller *et al.* [4] presented two proactive empty vehicle redistributions for PRT systems where the objective was minimizing the passenger waiting time. The present study addresses the routing problem associated with PRTs, where the objective is to minimize the energy consumption and empty vehicle movements given the assumption that the vehicles are battery-operated. This problem is challenging because it involves an additional empty movement, i.e., to and from the depot to charge the vehicles' batteries. This problem has only been address previously by Mrad and Hidri [5], who analyzed the optimal electric energy consumption for a PRT transportation system. The present study proposes an artificial bee colony (ABC) algorithm to solve the PRT problem. Computational experiments were performed using a set of 1320 randomly generated instances to verify the efficacy of the proposed method.
The remainder of this paper is organized as follows. Section 2 outlines the problem addressed in this study. Section 3 presents the major components of the proposed ABC algorithm. Section 4 analyzes the results obtained using the algorithm. Finally, Section 5 concludes this paper.

Fig. 1: PRT vehicle [1]

II. Problem Definition and Mathematical Formulations

The present study focuses specifically on the static problem of routing electric vehicles in a PRT system for a predefined list of trips. This problem is important in several respects. The Morgan Town PRT system has two basic modes of operation: demand mode and schedule mode. It operates in the demand mode outside the rush hours, where vehicles react dynamically to passenger demands. It operates in the schedule mode at other times, where vehicles run on fixed roads with known demand based on a predefined list of known origin and destination pairs. The static solution developed in the present study can be applied to problems related to the schedule mode of operation. This static solution is also useful for the demand mode because it can be used to evaluate various dynamic strategies for routing PRT pods. Indeed, the use of static solutions to evaluate dynamic strategies is common in previous studies of the vehicle routing problem (see [6] and [7] for more details).

The PRT problem was defined previously by Mrad and Hidri[5]. It can be stated as follows: consider the set of trips to perform T, where T has a cardinality $|T|$. Suppose that a PRT network has M stations, where movement is possible between any two stations. The depot is defined as D, which contains an unlimited number of vehicles with an initially battery capacity of B. Each trip $t_i \in T$ is characterized by the triplet (origin station (OS_i), destination station (DS_i), departure time (OT_i)). The arrival time (AT_i) is calculated using the following formula:

$$AT_i = OT_i + Distance(OS_i, OT_i) \tag{1}$$

where $Distance$ is a matrix cost that defines the direct cost (consumed electric energy) between each pair of stations. In a PRT network, this cost is calculated using the Floyd Warshall algorithm[8]. The PRT problem requires the assignment of each vehicle to a set of trips that does not exceed the battery capacity of each vehicle. Thus, an asymmetric graph $G = \{\vec{V}, \vec{E}\}$, where $\vec{V} = \{v_0, v_1, v_2, ..., v_n\}$ defines a set of nodes, where v_0 is the depot and $v_1, v_1, v_2, ..., v_n$ are n different trips that the PRT system must complete. In addition, $\vec{V}^* = \vec{V}/v_0$ is defined. Let $\vec{E} = \{(v_i, v_j); v_i, v_j \in \vec{V}\}$ be the set of arcs defined as follows:

- If $(v_i, v_j) \in V^*$ with $AT_i + Distance(DS_i, OS_j) \leq OT_j$, then the arc (i, j) exists and it has a cost c_{ij} that represents the energy consumed between the arrival station for trip i (DS_i) and the arrival station for trip j (DS_j). Thus, each edge has a combined cost: the cost of the movement among trips and the cost of the trip.
- For each node $i \in V^*$, an arc $(0, i)$ is added. The cost of this arc is c_{0i} and it represents the energy used to reach the arrival station for trip i from the depot.
- For each node $i \in V^*$, an arc $(i, 0)$ is added with a cost c_{i0}, which represents the energy used between the arrival station for trip i and the depot.

In addition, $\vec{E}' = \{(v_i, v_j); v_i, v_j \in \vec{V}^*\}$ is defined. This problem can be viewed as a classical node routing problem, i.e., the asymmetrical distance-constrained vehicle routing problem (ADCVRP), where the total travel distance allowed for each vehicle should be equal to or less than a certain distance. This problem is proven to be NP-Hard [9]. However, the major difference between this problem and the ADCVRP is that the graph G is not complete. In fact, if arc (ij) exists, then the opposite arc (ji) does not exist. A heuristics approach was used to solve this problem because it is NP-hard, which would have an exponential computational time using exact methods.

III. ABC Algorithm

Various metaheuristics are available for simulating and modeling different aspects of the lives of insects such as bees and ants. Bee colonies can be considered recent advances in this field. ABC is a relatively new heuristic technique, which was first proposed by Karaboga and modified subsequently by Karaboga and Akay. Apart from the modification proposed by Karaboga and Akay, many recent interesting modifications could be found such as [10],[11],[12],[13]. ABC is a heuristic that aims to simulate the life and behavior of bees, where the simulation is applied to solve various real world problems. A number of researchers have used ABC to solve combinatorial problems. For example, Wong *et al.* [14] and Chong *et al.* [15] applied ABC to the job shop problem, Rebrayand et al. [16] to the multiprocessor scheduling problem, and Szeto *et al.* [17] solved the capacitated vehicle routing problem using a bee swarm algorithm. For more details on ABC algorithm and its applications the reader is refereed to [18]. The components of the ABC algorithm are described as follows.

In the ABC algorithm, there are three bee colonies (groups): employed bees, onlooker bees, and scout bees. The employed bees can exploit food sources (solutions). Thus, the number of food sources is equal to the number of employed bees. In this algorithm, each employed bee searches for a food source (solution) in its neighborhood during the employed bee phase.

By contrast, onlooker bees wait in the hives for the arrival of the employed bees, which share their information about food sources by dancing in the dance area of the hives. The dance is proportional to the nectar content of the food source discovered. In general, the number of onlooker bees is also equal to the number of employed bees. Onlooker bees watch the dance and choose a food source according to probabilities determined by the qualities of the food sources. Therefore, good food sources have a greater chance of being selected and exploited. In the algorithm, the onlooker bee phase follows the employed bee phase. In general, the onlooker bees will search all the information stored by the employed bees and select a specific food source with a certain probability related to its fitness. The onlooker bees will then produce a modification of the selected food source and greedy selection is applied to retain the food source with the best nectar food. Scout bees search for new food sources in the neighborhood around the hives. Normally, the scout bees intervene only after the employed and onlooker bees finish their searches. The scout bees consider each food source and decide whether to abandon it or not. In general, the ABC algorithm assigns a counter to each food source, which is updated during the search phases [19]. If this counter exceeds a specific value, this indicates that the food source is exhausted and it should be abandoned. The scout bees replace the abandoned food source with new, fresh, and randomly generated food sources.

The ABC algorithm can be formalized as follows.

Algorithm 1 ABC algorithm

```
1: Initialize-the-population()
2: cycle ⟵ 1
3: while $cycle <= max - cycle$ do
4:     Employed-Bee-Phase
5:     Calculate probabilities for Onlookers
6:     Onlooker bee phase
7:     Scout bee phase
8:     Memorize best solution
9:     Cycle++
10: end while
```

Algorithm 2 Self-recombination (Solution S)

```
1: evaluate S using the Split function
2: for $i = 1 \to nb - Road$ do
3:     for $j = 1 \to nb - Road$ do
4:         $R_1 \longleftarrow ChooseAtRandom$
5:         $R_2 \longleftarrow ChooseAtRandom$
6:         if $cost(Q_{R_1}; H_{R_2}) < cost(Q_{R_1}; D) + cost(D; H_{R_2})$ then
7:             $RECOMBINE(R_1,\ R_2)$
8:         end if
9:     end for
10: end for
```

A. Representation and Evaluation Method

To maintain the simplicity of the algorithm, the solutions are represented as permutation of trips, as shown in Figure 2.

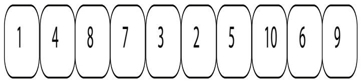

Fig. 2: Permutation representation

In the algorithm, the search space is also restricted to feasible solutions. As a consequence, an adaptation of the split function initially introduced by Prins [20] is applied to vehicle routing problems to evaluate the different solutions. This function can extract a set of roads from a permutation and find the cost associated with the permutation's optimal splitting. (More details of the split function can be found in [20]).

An enhancement to the proposed algorithm is used with the evaluation function. Due to the low connectivity rate of the graph G, the split function can generate many permutations for different roads with a low battery capacity usage B for each vehicle. Therefore, a self-recombining function is introduced that takes the available information for each road and it tries to combine this information to reduce the interlinkages between the generated roads. This method operates as follows.

Assuming that each generated road has a head trip (H) and a queue trip (Q), and that the depot is represented by a node D, the self-recombining algorithm is as follows.

The $RECOMBINE$ function combines two different roads to obtain one road.

1) Initial Solutions: At the start of the algorithm, it is possible to choose either to initialize the first solutions randomly or to select a relatively structured solution. A better initial solution for the algorithm can yield a better final solution.

The first permutation is selected to start the search, where the SLP solution(Solution of a Linear Program) requires the solution of the following linear program.

First, the following integer variable is introduced:

$$x_{ij} = \begin{cases} 1 & \text{if node } j \text{ is visited after node } i \\ 0 & \text{Otherwise} \end{cases}$$

$$\textbf{PRT:} \qquad \text{Min} \sum_{(i,j)\in E} c_{ij}x_{ij}$$

$$\sum_{j\in\delta^+(i)} x_{ij} = 1 \forall i \in V^*$$

$$\sum_{j\in\delta^-(i)} x_{ji} = 1 \forall i \in V^*$$

$$x_{ij} \in \{0,1\} \forall \ \ (i,j) \in E$$

This linear program outputs various roads without considering the battery capacity but it yields a relatively good solution. The use of the split function facilitates the generation of a feasible solution from all the unfeasible roads that are generated. The use of this initialization ensures that the first generation yields relatively good solutions, as well as ensuring that the algorithm converges rapidly to a good final solution.

2) Neighborhood Operator: In general for heuristics, there exists a distinctive type of neighborhood operator [21] Three different neighborhood operators are used to obtain the new solution for both employed and onlooker bees: exchange operator, displacement operator, and inversion operator. The exchange operator choose at random two different trips and exchange their positions. The displacement operator choose at random a subsequence of trips and change its position at random. The inversion operator choose at random a subsequence of trips and inverse the order of the trips in it.

During both the onlooker and employed bee phases, one of these three mutation operators is selected randomly to obtain the new solutions.

IV. Computational Results

This section describes the results of experiments where the ABC algorithm was applied to a set of randomly generated problems. The proposed heuristics developed in the previous sections were implemented in the C++ language. The experiments were performed on a laptop with an Intel i5 2.3 GHz CPU and 6 GB of RAM, with the Microsoft Windows 7 operating system.

A. Test problems

The tests were conducted using 33 different classes (40 instances for each class), where the number of trips, n, was given as
$n \in \{10; 15; 20; 25; 30; 35; 40; 45; 50; 55; 60; 65; 70; 75; 80; 85;$

$90; 95; 100; 110; 120; 130; 140; 150; 160; 170; 180; 190; 200; 250; 300; 350; 400\}$ Each instance was generated based on the following assumptions:

- Number of stations = 12
- Cost of the arcs of the network: Generated randomly between 1 and 15 min
- Departure stations: Generated randomly between 1 and 12
- Arrival stations: Generated randomly between 1 and 12
- Departure time for each trip: Generated randomly between 1 and 3600 s
- Battery capacity: 40 min
- Max-wait: the maximum amount of time that a vehicle could wait at a station before starting a trip was 1000 s.

The set of generated instances was based on assumptions employed in [22].

GAP was used to verify the quality of each algorithm, which is defined by the following formula.

- $$GAP = (\frac{(SOL_{heuristic} - Lowerbound)}{lowerbound}) * 100$$

For each instance, the lower bound was taken as the linear relaxation of the mathematical model given in [23] to resolve the ADCVRP coded using Cplex12.1. It should also be noted that the SLP solution was calculated using Cplex12.1.

B. Results Obtained Using the Proposed Heuristic

Tuning the parameters of the ABC algorithm to improve its obtained results is a widely studied issue in combinatorial optimization [24],[25],[26]. The objective is to obtain the best possible results by testing on a relatively small number of instances a restricted number of parameter combinations. For that purpose, we used a simple algorithm based on three simple steps[27]:

1) choose the test instances.
2) select a parameter values using a simple method.
3) apply a statistical test to choose the best parameters option.

As for the first step, we took the 40 instances of size 200 trips as they present a good testing instances bed due to their relative complexity. As for second step, and starting from a promising parameters configuration, the ABC algorithm is tested on the instances used as a test bed. Next, we choose randomly one or two parameters and change them. The ABC algorithm is tested again on the testing bed. The new obtained results are compared against the old results. If they are better, the new configuration replaces the old configuration. If the two results are almost equivalent, then the parameter combination that yields the small CPU run time is kept. We used a Wilcoxon's matched-pairs signed rank test to know if a parameter combination performs better than another parameter combination. This procedure is run for 20 iterations. Although this method would not present an optimal parameters tuning, it presents a good alternative to find quickly a good parameter'values for an heuristic algorithm.

Finally, the obtained parameters were as follows:

- Onlooker bee size: 10.
- Employer bee size: 10.
- Max-Cycle: 1000.
- Limit for the scout bee: 15.

To test the importance of integrating the various components of the algorithm, the ABC algorithm was run with and without different operators to determine the proportions of the final results that were attributable to each component. The results of this analysis are shown in Table II. Table II confirms that the proposed ABC algorithm performs better than all other versions of the ABC. The results also confirm the importance of combining all the components to obtain better quality results.

A global statistical analysis of the results obtained was also conducted because a simple comparison of the overall average GAP using different versions of the ABC algorithm cannot yield the same conclusions and results as an appropriate statistical comparison. The statistical analysis determined whether the results were significant and not simple consequences of chance or sampling errors.

First, it was necessary to determine whether to apply a parametric test or a nonparametric test. Thus, tests of normality were performed, i.e., the Kolmogorov-Smirnov normality (K-S) test and the D'Agostino-Pearson normality test. The results of these two tests were significant because the $P - value$ was less than 0.05, thus the data were not normally distributed Therefore, Wilcoxon's matched-pairs signed rank test was used for the comparative study. An R program was developed in RStudio to compare the different ABC versions based on the trip size for each class. This facilitated an evaluation of the performance based on the trip size. Table III shows the results of this test.

TABLE III: Comparative results based on Wilcoxon's matched-pairs signed rank test

	ABC vs. ABC Ver1	ABC vs. ABC Ver2	ABC vs. ABC Ver3	ABC vs. ABC Ver4
All	< 0.001	< 0.001	< 0.001	< 0.001
10	NaN	NaN	NaN	NaN
15	NaN	NaN	NaN	NaN
20	NaN	0.0679	0.0679	0.0180
25	NaN	0.3173	0.0180	0.0007
30	NaN	0.0077	0.0015	< 0.001
35	0.8927	0.0033	0.0010	< 0.001
40	0.0431	0.0010	0.0004	< 0.001
45	0.8927	0.0022	0.0030	< 0.001
50	0.4008	< 0.001	< 0.001	< 0.001
55	0.0117	< 0.001	< 0.001	< 0.001
60	0.0926	< 0.001	< 0.001	< 0.001
65	0.0401	< 0.001	< 0.001	< 0.001
70	0.0152	< 0.001	< 0.001	< 0.001
75	0.0017	< 0.001	< 0.001	< 0.001
80	0.0130	< 0.001	< 0.001	< 0.001
85	0.0003	< 0.001	< 0.001	< 0.001
90	0.0008	< 0.001	< 0.001	< 0.001
95	< 0.001	< 0.001	< 0.001	< 0.001
100	< 0.001	< 0.001	< 0.001	< 0.001
110	< 0.001	< 0.001	< 0.001	< 0.001
120	< 0.001	< 0.001	< 0.001	< 0.001
130	< 0.001	< 0.001	< 0.001	< 0.001
140	< 0.001	< 0.001	< 0.001	< 0.001
150	< 0.001	< 0.001	< 0.001	< 0.001
160	< 0.001	< 0.001	< 0.001	< 0.001
170	< 0.001	< 0.001	< 0.001	< 0.001
180	< 0.001	< 0.001	< 0.001	< 0.001
190	< 0.001	< 0.001	< 0.001	< 0.001
200	< 0.001	< 0.001	< 0.001	< 0.001
250	< 0.001	< 0.001	< 0.001	< 0.001
300	< 0.001	< 0.001	< 0.001	< 0.001
350	< 0.001	< 0.001	< 0.001	< 0.001
400	< 0.001	< 0.001	< 0.001	< 0.001

TABLE I: Results obtained using the ABC algorithm

	Average Gap %	Average Time
10	1.105	0.882
15	0.793	1.198
20	0.577	1.565
25	0.883	1.870
30	0.782	2.308
35	1.102	2.569
40	0.804	2.946
45	0.697	3.155
50	0.658	3.517
55	0.936	3.926
60	0.713	4.193
65	0.810	4.493
70	1.243	4.814
75	0.921	5.153
80	1.082	5.714
85	1.165	5.844
90	1.032	6.492
95	1.209	7.145
100	1.457	7.125
110	1.233	7.931
120	1.490	8.967
130	1.932	9.383
140	1.277	10.173
150	1.403	11.072
160	1.894	12.252
170	1.879	12.781
180	1.713	14.514
190	1.928	15.143
200	2.245	15.558
250	2.006	22.519
300	2.539	26.574
350	2.900	37.782
400	3.523	46.444
Average	1.392	9.879

TABLE II: Results of the comparison of different versions of the ABC algorithm

	ABC	ABC Ver1	ABC Ver2	ABC Ver3	ABC Ver4
Number of values	1320	1320	1320	1320	1320
Sum	990.1	1425	2189	1493	3413
Minimum	0	0	0	0	0
25% Percentile	0	0.1342	0.3343	0.1649	0.5525
Median	0.4494	0.7067	1.241	0.7944	1.946
75% Percentile	1.194	1.649	2.589	1.731	3.981
Maximum	6.231	6.911	9.104	6.231	14.87
Mean	0.7501	1.079	1.658	1.131	2.586
Std. Deviation	0.8916	1.173	1.594	1.165	2.54
Std. Error of Mean	0.02454	0.03228	0.04388	0.03207	0.0699
Lower 95% CI of mean	0.702	1.016	1.572	1.068	2.448
Upper 95% CI of mean	0.7983	1.143	1.744	1.194	2.723
ABC Ver1	ABC Without the SLP				
ABC Ver2	ABC Without The recombine function				
ABC Ver3	ABC Without Onlooker Bee phase				
ABC Ver4	ABC Without Employed Bee phase				

In this test, $H0$ assumed that the percentage deviations of the results obtained using the two algorithms would be the same. By contrast, $H1$ hypothesis assumed that the average deviation of one algorithm was better than that of the other algorithm.

Table III shows that the proposed ABC algorithm performed better than all the other versions because the $P-value$ was less than 0.05.

The statistical analyses confirmed that the different versions of the algorithm delivered significantly different performance. The results also confirmed that the proposed ABC algorithm performed better according to the statistical analysis. In particular, the results of the statistical tests show that for small size

TABLE IV: Results of the runs test

	Nonlinear equation	Linear equation
Points above curve	14	14
Points below curve	19	19
Number of runs	21	3
P-value (runs test)	0.945	¡ 0.0001
Deviation from Model	Not significant	Significant

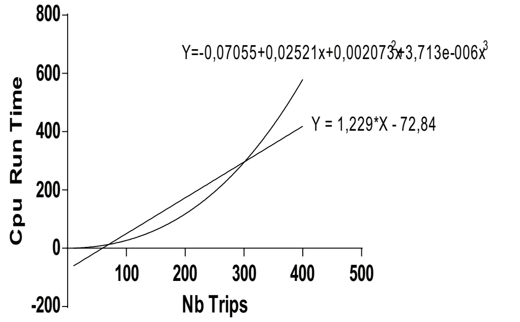

Fig. 3: Linear and nonlinear regressions for the CPU runtime

instances the results were almost the same, especially with ABC ver1 and ver2. However, as the size of the problem (and therefore the difficulty) increased, it was necessary to combine all of the different ABC components to obtain high quality results. In particular, the recombining method had an especially strong effect in improving the results obtained. This was demonstrated clearly in the experiments because the version of the ABC algorithm without the recombining method yielded an average GAP of 1.658, whereas the proposed ABC algorithm had an average GAP of 0.908.

The previous tables show the comparative results with different ABC algorithms. The following table provides a comparative analysis of the proposed ABC algorithm.
The ABC algorithm differs from traditional evolutionary algorithms because it exploits an intensive neighborhood policy to search for and generate new solutions. The ABC algorithm has also the advantage of inserting new solutions into the population during the scout bee phase, which can help the algorithm to escape from local optima. The integration of the proposed ABC algorithm with the recombining method also helps to escape from local optima because it allows the possibility of mixing the solutions in a uniform manner by rearranging the various roads generated.

Finally, the CPU runtime was estimated as a function of the trip size. This analysis aimed to determine the relationship between these two variables. First, a correlation test was performed to determine the statistical relationship between the two variables. The result of the Spearman's rank correlation coefficient test was significant (P value < 0.05), thus it was concluded that there was a relationship between the two variables. Next, linear and nonlinear (cubic function) regression models were used to estimate the exact CPU runtime as a function of the trip size. The results obtained using these two models are shown in Figure 3 and post-analysis results for the two models are shown in Table IV. Based on the results of the runs test, it was concluded that the cubic function was a better estimator of the CPU runtime, where the P value was 0.945. This could be explained by the fact that we have several $O(n^2)$ steps in our algorithm. In fact, the evaluation function (The split procedure) have a complexity of $O(n^2)$ [20].

To assess the quality of our ABC, a simple genetic algorithm (GA)[28],[29],[30] was developed in order to compare the results of our ABC with those of an algorithm that focuses more on the global search and the exploration of the search space. The main features of the testing GA are presented in the following algorithm.

Algorithm 3 Genetic Algorithm

```
1: Initialize-parameter()
2: while Termination criterion not satisfied do
3:    for all Individuals in the population do
4:       parent1 ⟵ Select-at-random(pop)
5:       parent2 ⟵ Select-at-random(pop)
6:       Apply the one-point crossover operator to the two parents
7:       Apply the insertion mutation operator to the offspring generated
8:       Recombine the offspring generated
9:       Evaluate the offspring
10:      x⟵ Get-Worst(pop)
11:      $\Delta = evaluation(offspring) - evaluation(Individual - at(x))$
12:      if $\Delta < 0$ then
13:         Insert($offspring$, pop, x)
14:      end if
15:   end for
16: end while
17: individual ⟵ Best-individual(pop)
```

Note that the representation and the evaluation of individuals in the GA as well as the initialization of the initial population are the same as with the ABC algorithm.

For comparing the ABC and the GA, we used the average relative percent deviation (ARPD) computed as follows:

- $$ARPD = (\frac{(SOL_{heuristic} - SOL^*_{heuristic})}{SOL^*_{heuristic}}) \cdot 100$$
  where $SOL^*_{heuristic}$ is either the best value obtained from the GA or the ABC.

Results of table V confirm the superiority of our ABC in comparison with the GA. This could be explained by the fact that the ABC algorithm combine both local search moves in addition to global search moves in order to discover properly the whole search space of the problem.

V. Conclusions

Bee algorithms have been tested successfully using a large set of different problems. In the present study, the ABC algorithm was adapted to solve the problem of reducing empty vehicle movements in PRT systems. The proposed hybrid algorithm benefits from the application of a linear programming technique to find a good solution that is used to start the search, while a specific self-recombination method considers the global information stored in each generated road for

TABLE V: Results of the Comparison between the GA and the ABC Algorithms

Number of trips	Average ARPD for the ABC	Average ARPD for the GA
10	0	0
15	0.034	0.038
20	0	0.125
25	0.012	0.304
30	0.037	0.299
35	0.017	0.716
40	0.015	0.605
45	0.010	0.613
50	0.015	0.726
55	0.031	0.731
60	0	0.697
65	0.026	0.702
70	0.026	0.906
75	0.013	0.900
80	0.026	0.787
85	0.044	0.889
90	0.021	0.760
95	0	0.857
100	0.014	0.852
110	0.008	0.715
120	0.046	0.852
130	0.052	0.570
140	0.003	0.701
150	0.033	0.634
160	0.012	0.735
170	0.026	0.416
180	0.065	0.534
190	0.031	0.724
200	0.039	0.622
250	0.083	0.468
300	0.107	0.411
350	0.060	0.390
400	0.125	0.463
Average	0.031	0.598

each solution to escape from local optima. The computational results demonstrate that the algorithm is competitive because it can find a good GAP within a relatively short computational time. The results of comparative studies conducted using different versions of the algorithm showed that the proposed ABC algorithm obtained better results for the PRT problem compared with the basic ABC algorithm.

References

[1] "Multi-Vehicle Operations Certification Testing Commences Ahead of Schedule," http://www.ultraprt.net/multiVehicle.htm, 2009, [Online; accessed 05-May-2012].

[2] J.-M. Won, H. Choe, and F. Karray, "Optimal design of personal rapid transit," in *Intelligent Transportation Systems Conference*, Sep. 2006, pp. 1489–1494. [Online]. Available: http://dx.doi.org/10.1109/ITSC.2006.1707434

[3] J. Li, Y. S. Chen, H. Li, I. Andreasson, and H. van Zuylen, "Optimizing the fleet size of a Personal Rapid Transit system: A case study in port of Rotterdam," in *International Conference on Intelligent Transportation*, 2010, pp. 301–305.

[4] J. D. Lees-Miller and R. E. Wilson, "Proactive empty vehicle redistribution for personal rapid transit and taxis," *Transportation Planning and Technology*, vol. 35, no. 1, pp. 17–30, 2012. [Online]. Available: http://www.tandfonline.com/doi/abs/10.1080/03081060.2012.635414

[5] H. L. Mrad Mahdi, "Optimal consumed electric energy while sequencing vehicle trips in a personal rapid transit transportation system," *Computers & Industrial Engineering*, 2014, forthcoming.

[6] G. Berbeglia, J.-F. Cordeau, I. Gribkovskaia, and G. Laporte, "Static pickup and delivery problems: a classification scheme and survey," *Top*, vol. 15, no. 1, pp. 1–31, 2007. [Online]. Available: http://www.springerlink.com/index/10.1007/s11750-007-0009-0

[7] S. Mitrovic-Minica, R. Krishnamurtia, and G. Laporteb, "Double-horizon based heuristics for the dynamic pickup and delivery problem with time windows," *Transportation Research Part B: Methodological*, vol. 38, no. 8, pp. 669 – 685, 2004. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S019126150300095X

[8] R. W. Floyd, "Algorithm 97: shortest path," *Communications of the ACM*, vol. 5, no. 6, p. 345, 1962.

[9] P. Toth and D. Vigo, *The vehicle routing problem*, ser. SIAM monographs on discrete mathematics and applications. Society for Industrial and Applied Mathematics, 2002.

[10] S. Biswas, S. Kundu, D. Bose, S. Das, P. Suganthan, and B. Panigrahi, "Migrating forager population in a multi-population artificial bee colony algorithm with modified perturbation schemes," in *Swarm Intelligence (SIS), 2013 IEEE Symposium on*, April 2013, pp. 248–255.

[11] S. Das, S. Biswas, B. Panigrahi, S. Kundu, and D. Basu, "A spatially informative optic flow model of bee colony with saccadic flight strategy for global optimization," *Cybernetics, IEEE Transactions on*, vol. PP, no. 99, pp. 1–1, 2014.

[12] S. Biswas, S. Das, S. Kundu, and G. Patra, "Utilizing time-linkage property in dops: An information sharing based artificial bee colony algorithm for tracking multiple optima in uncertain environments," *Soft Computing*, pp. 1–14, 2013. [Online]. Available: http://dx.doi.org/10.1007/s00500-013-1138-z

[13] S. Das, S. Biswas, and S. Kundu, "Synergizing fitness learning with proximity-based food source selection in artificial bee colony algorithm for numerical optimization," *Applied Soft Computing*, vol. 13, no. 12, pp. 4676 – 4694, 2013. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1568494613002354

[14] L.-P. Wong, C. Y. Puan, M. Y.-H. Low, and C. S. Chong, "Bee colony optimization algorithm with big valley landscape exploitation for job shop scheduling problems," in *Winter Simulation Conference*, 2008, pp. 2050–2058.

[15] C. S. Chong, A. I. Sivakumar, Malcolm, and K. L. Gay, "A bee colony optimization algorithm to job shop scheduling," in *WSC '06: Proceedings of the 38th conference on Winter simulation*. Winter Simulation Conference, 2006, pp. 1954–1961. [Online]. Available: http://portal.acm.org/citation.cfm?id=1218112.1218469

[16] P. Rebreyend, C. Clugery, and E. Hily, "A heuristic-based bee colony algorithm for the multiprocessor scheduling problem," in *Nature Inspired Cooperative Strategies for Optimization (NICSO 2010)*, ser. Studies in Computational Intelligence, J. Gonzlez, D. Pelta, C. Cruz, G. Terrazas, and N. Krasnogor, Eds. Springer Berlin Heidelberg, 2010, vol. 284, pp. 295–304. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-12538-6_25

[17] W. Szeto, Y. Wu, and S. C. Ho, "An artificial bee colony algorithm for the capacitated vehicle routing problem," *European Journal of Operational Research*, vol. 215, no. 1, pp. 126 – 135, 2011. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0377221711005121

[18] D. Karaboga, B. Gorkemli, C. Ozturk, and N. Karaboga, "A comprehensive survey: artificial bee colony (abc) algorithm and applications," *Artificial Intelligence Review*, pp. 1–37, 2012. [Online]. Available: http://dx.doi.org/10.1007/s10462-012-9328-0

[19] B. Akay and D. Karaboga, "A modified artificial bee colony algorithm for real-parameter optimization," *Information Sciences*, vol. 192, no. 0, pp. 120 – 142, 2012, ¡ce:title¿Swarm Intelligence and Its Applications¡/ce:title¿. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0020025510003336

[20] C. Prins, "A simple and effective evolutionary algorithm for the vehicle routing problem," *Computers & Operations Research*, vol. 31, no. 12, pp. 1985 – 2002, 2004. [Online]. Available: http://www.sciencedirect.com/science/article/B6VC5-48NX3CG-6/2/59c8437348cefff8e0ebfc72bf3f8dba

[21] R. K. Ahuja, zlem Ergun, J. B. Orlin, and A. P. Punnen, "A survey of very large-scale neighborhood search techniques," *Discrete Applied Mathematics*, vol. 123, no. 13, pp. 75 – 102, 2002. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0166218X01003389

[22] K. Mueller and S. P. Sgouridis, "Simulation-based analysis of personal rapid transit systems: service and energy performance assessment of the masdar city prt case," *Journal of Advanced*

Transportation, vol. 45, no. 4, pp. 252–270, 2011. [Online]. Available: http://dx.doi.org/10.1002/atr.158

[23] I. Kara, “Two indexed polonomyal size formulationsfor vehicle routing problems,” *Technical Report. BaskentUniversity, Ankara/Turkey*, 2008.

[24] A. E. Eiben, R. Hinterding, and Z. Michalewicz, “Parameter control in evolutionary algorithms,” *Evolutionary Computation, IEEE Transactions on*, vol. 3, no. 2, pp. 124–141, 1999.

[25] S. P. Coy, B. L. Golden, G. C. Runger, and E. A. Wasil, “Using experimental design to find effective parameter settings for heuristics,” *Journal of Heuristics*, vol. 7, no. 1, pp. 77–97, 2001.

[26] R. S. Barr, B. L. Golden, J. P. Kelly, M. G. Resende, and W. R. Stewart Jr, “Designing and reporting on computational experiments with heuristic methods,” *Journal of Heuristics*, vol. 1, no. 1, pp. 9–32, 1995.

[27] V.-P. Nguyen, C. Prins, and C. Prodhon, “A multi-start iterated local search with tabu list and path relinking for the two-echelon location-routing problem,” *Engineering Applications of Artificial Intelligence*, vol. 25, no. 1, pp. 56–71, 2012.

[28] D. Whitley, “A genetic algorithm tutorial,” *Statistics and computing*, vol. 4, no. 2, pp. 65–85, 1994.

[29] J. Genlin, “Survey on genetic algorithm,” *Computer Applications and Software*, vol. 21, no. 2, pp. 69–73, 2004.

[30] M. Srinivas and L. M. Patnaik, “Genetic algorithms: A survey,” *Computer*, vol. 27, no. 6, pp. 17–26, 1994.