International Journal of Computer and Information Technology (ISSN: 2279 – 0764)
Volume 03 – Issue 01, January 2014

Multi-Objective Vehicle Routing Problems with Time Windows: A Vector Evaluated Artificial Bee Colony Approach

Oren Efraim Nahum
Department of Management
Bar-Ilan University
Ramat-Gan, Israel
Email: oren.nahum {at} live.biu.ac.il

Yuval Hadas
Department of Management
Bar-Ilan University
Ramat-Gan, Israel

Uriel Spiegel
Department of Management
Bar-Ilan University
Ramat-Gan, Israel
and visiting at Department of Economics,
University of Pennsylvania, USA.

Abstract— **The vehicle routing problem with time windows, widely used in practice, is an NP-hard problem. This paper presents a new meta-heuristic algorithm for solving the problem. Unlike traditional two-steps algorithms, the proposed optimization algorithm, based on the artificial bee colony algorithm combined with the vector evaluated technique, solves the problem as a multi-objective problem. As a result, the algorithm provides a simultaneous solution set. The approach was tested on standard Solomon's benchmark problems. The result shows that this algorithm was better than (in terms of number of vehicles) or equal to other existing heuristic algorithms.**

Keywords- Vehicle routing problem; artificial bee colony; multi-objective

I. INTRODUCTION

The Vehicle-Routing Problem (VRP), one of the most important and widely studied combinatorial optimization problems with many real-world applications in distribution and transportation logistics [1], is a common name for problems involving the construction of a set of routes for a fleet of vehicles. The vehicles start their routes at a depot, call at customers, to whom they deliver goods, and return to the depot. The objective function for the basic vehicle-routing problem (known as CVRP) is to minimize costs by finding optimal routes, which are usually the shortest routes. For the CVRP the following constraints must held (1) each route starts and ends at the depot, (2) each customer is called at exactly once and by only one vehicle and (3) the total demand on each route does not exceed the total capacity of one truck.

Since Dantzig and Ramser [2] first introduced the problem, several extensions to the problem were developed in which other types of "costs" and different constraints are used. A typical and often discussed problem is the basic VRP, expanded by time windows (VRPTW). In VRPTW, service at every customer *i* must start within a given time window $[a_i,b_i]$. A vehicle is allowed to arrive before a_i and wait until the customer becomes available. Arrivals after b_i are prohibited.

The objective of VRPTW is to design the shortest path for minimizing traveling costs, number of vehicles and waiting time at customer locations without violating either time windows or vehicle's loading capacity constraints [3]. Since minimizing one objective rarely minimizes all the other constraints, the optimization process needs to provide, not a single solution, but a range of solutions that represent trade-offs between the objectives [4-6]. Accordingly, in this paper the problem is considered as a multi-objective problem and an appropriate algorithm is provided.

VRPTW has numerous applications in distribution management such as in delivering beverages, food and newspapers and in commercial and industrial waste collection [7].

Optimal solutions for small instances of VRPTW, in which a single objective, minimizing the total travel distance, is considered, can be obtained using exact methods [8]. The current exact algorithms were proposed by Chabrier [9], Irnich and Villeneuve [10], Jepsen, Petersen [11], Jepsen, Petersen [12] and Kallehauge, Larsen [13]. To date, 45 instances out of 56 in Solomon's benchmarks have been solved to optimality [11] using exact methods.

Although exact methods can guarantee the optimality of the solution, they require considerable computer resources in terms of both computational time and memory. This is because VRPTW, which generalizes the CVRP by assuming a_i=0 and $b_i=\infty$ for every customer *i*, is NP-hard i.e., the problem cannot be solved optimally in polynomial time [14]. As a result, research on VRPTW has concentrated on heuristics and meta-heuristics. Solomon [15,16] was one of the first to research VRPTW. In his work, Solomon presented a number of extensions to existing heuristics, such as the savings heuristics, for solving VRPTW. For an extensive list of studies of different heuristics and meta-heuristics for solving VRP, as well as a comparison of the results obtained, the reader is referred to [17-20].

Current state-of-the-art heuristics for the VRPTW consist of evolution strategies [21, 22]; large neighborhood searches [23-25]; iterated local searches [26, 27]; and multi-start local searches [26, 28]. It should be noted that in all of these algorithms, the hierarchical objective is considered, and therefore, these heuristics are all based on a two-stage approach, where the number of routes is minimized in the first stage, and the total travel distance is then minimized in the second stage, allowing us to independently develop algorithms for minimizing the route and distance.

This article presents a new multi-objective optimization algorithm for a VRPTW application based on the behavior of bees during their search for nectar.

The rest of the paper is as follows. Section 2 provides a review of the basic artificial bee colony (ABC) algorithm, and the concepts of the vector evaluated technique. A combined vector evaluated ABC is described. Other issues, such as solution representation and neighborhood operators, are discussed as well. Experimental results showing the improvement in performance of the enhanced ABC are presented in Section 3. Finally, Section 4 concludes the paper.

II. Vector Evaluated Artificial Bee Colony (VE-ABC)

A. *Artificial Bee Colony*

The artificial bee colony (ABC) algorithm proposed by Karaboga [29] and later modified by Karaboga and Akay [30] is a new evolutionary meta-heuristic technique inspired by the intelligent behavior of natural honey bees in their search for nectar.

In the ABC algorithm, the colony of artificial bees consists of three groups of bees: (1) employed bees - bees that are currently exploiting a food source; (2) onlookers - bees that are waiting in the hive for the employed bees to share information about the food sources; and (3) scouts - bees that are searching for new food sources in the neighborhood of the hive.

The ABC algorithm is an iterative algorithm. It starts by assigning each employed bee to a randomly generated solution (known as a food source). Next, in each iteration, each employed bee, using a neighborhood operator, finds a new food source near its assigned food source. The nectar amount (defined as a fitness function) of the new food source is then evaluated. If the amount of nectar in the new food source is higher than the amount of nectar in the old one, then the older source is replaced by the newer one. Next, the nectar information of the food sources is shared with the onlookers (real bees do this by dancing in the dance area inside the hive). Usually, the number of onlookers is taken to be equal to the number of employed bees. The onlooker chooses a food source according to the probability proportional to the quality of that food source. Roulette wheel selection is the usual method. Therefore, good food sources, as opposed to bad ones, attract more onlooker bees. Subsequently, using a neighborhood operator, each onlooker finds a food source near its selected food source and calculates its nectar amount. Then, for each old food source, the best food source among all the food sources near the old one is determined. The employed bee associated with the old food source is assigned to the best food source and abandons the old one if the best food source is better than the old food source. A food source is also abandoned by an employed bee if the quality of the food source has not improved in the course of a predetermined and limited number of successive iterations. The employed bees then become scouts and randomly search for new food source. After a scout finds a new food source, it becomes an employed bee again. After each employed bee is assigned to a food source, another iteration of the ABC algorithm begins. The iterative process is repeated until a stopping condition is met.

Szeto, Wu [31] describe the steps of the ABC algorithm as follows:

1. Randomly generate a set S of i solutions as initial food sources, where i is the number of employed bees. Assign an employed bee to each food source.
2. Evaluate the nectar amount (fitness), $f(s_i)$, of each food source, for each objective function j.
3. Repeat until a stopping condition is met:
    a. For each food source $s_i \in S$, apply the neighborhood operator to generate a neighbor food source, s'_i. If the fitness of the neighbor food source is better than that of the original food source, i.e. $f(s_i) > f(s'_i)$, for maximization problems, then replace the original food source with this neighbor food source.
    b. Set $G_0 = \varnothing$, $G_1 = \varnothing, \ldots, G_i = \varnothing$, where i is the number of employed bees.
    c. For each onlooker, use the fitness-based roulette wheel selection method to select a food source, s_i. Apply a neighborhood operator to s_i to find a neighbor food source, say s'_i. Add s'_i to G_i, i.e. $G_i = G_i \cup s'_i$.
    d. For each food source $s_i \in S$, if $G_i \neq \varnothing$ then let s' be the source food with best fitness value in G_i. If the fitness of s' is better than that of s_i, then replace s_i with s'.
    e. Replace any food sources $s_i \in S$ whose fitness has not been improved for *limit* iterations with randomly generated solutions
4. Output the best food source (solution) found.

Since ABC algorithms were first introduced, a very limited number of ABC algorithms for solving VRPTW were developed [32-34].

B. *Vector Evaluated Technique*

Genetic algorithms (GA) [35], which constitute a class of evolutionary algorithms (EA), offer a randomized global search technique for finding exact or approximate solutions to optimization and search problems. Basically, a GA evolves a population of bit-strings, or chromosomes, where each chromosome encodes a solution to a particular instance. This evolution takes place through the application of operators that mimic natural phenomena observed in nature, such as inheritance, mutation, selection, and crossover. For more information about GAs, please refer to [36].

GAs are widely used for solving VRPTWs. Homberger and Gehring [37] described two GAs for VRPTW. In the first algorithm, new individuals are generated directly through mutations, and no recombination takes place. In the second algorithm, offsprings are generated through a two-step recombination procedure in which three individuals are involved. In both algorithms, the fitness of an individual depends first on the number of vehicles used and second on the total distance traveled.

In a later work, Gehring and Homberger [38] proposed a two-phase meta-heuristic in which the first phase uses a GA to minimize the number of vehicles, while the second one minimizes the total distance through a tabu search.

Berger and Barkaoui [39] developed a GA that concurrently evolves two distinct populations pursuing different objectives under partial constraint relaxation. The first population aims to minimize the total distance traveled while the second one focuses on minimizing the violations of the time window constraints. This approach has proven to be rather efficient in minimizing the number of vehicles used.

Since ABC algorithms share common characteristics with GAs, simple modifications made to the basic GAs, can be adopted and applied to ABC algorithms in order to solve multi-objective VRPTWs.

The vector evaluated genetic algorithm (VEGA) proposed by Schaffer [40], was the first GA dealing with multiple objectives. Aware of the potential that GAs have in multi-objective optimization, Schaffer proposed an extension of the simple GA to accommodate vector-valued fitness measures. With the VEGA, the selection step is modified so that at each generation, a number of sub-populations are generated by performing proportional selection according to each objective function in turn. Thus, for a problem with q objectives, q sub-populations of size N/q each would be generated, assuming a population size of N. These would then be shuffled together to obtain a new population of size N, in order for the algorithm to proceed by applying crossover and mutation in the usual way. This multi-objective optimization strategy has already been successfully applied in many cases for experimental medium optimization [41].

Elitism guarantees that the best solutions found in each iteration are passed on to the next iteration and not lost. The original ABC algorithm does not use elitism. Conventionally, elitism is achieved by simply copying the solutions directly into the new generation; for multi-objective optimization problems, it can be used in a slightly different way.

In order to describe how the preservation of high-performance solutions can be done for multi-objective optimization problems, the concepts of dominated and non-dominated solution have to be defined first. In single objective optimization problems, the "best" solution is defined in terms of an "optimum solution" for which the objective function value is optimized when compared to any other alternative in the set of all feasible alternatives. In multi-objective optimization problems, however, the notion of an "optimum solution" does not usually exist since the optimum of each criterion does not usually point to the same alternative. The optimal solution in a multi-objective optimization problem is usually equivalent to choosing the best compromise solution. In the absence of an optimal solution, the concepts of dominated and non-dominated solutions become relevant.

A feasible solution, x_1, dominates another feasible solution, x_2, if x_1 is at least as good as x_2 with respect to all objective functions and is better than x_2 with respect to at least one objective function. A non-dominated solution is a feasible solution that is not dominated by any other feasible solution. Hence the solution of a multi-objective problem is a set of non-dominated feasible solutions.

Using the definition above, the set of high-performance solutions can be defined as the set of non-dominated solutions obtained in all iterations of the algorithm. This set of non-dominated solutions, denoted as E, can be obtained if, in each iteration, any newly obtained solution is added to the set E if it is not dominated by any solution already in E. Moreover, if a newly obtained solution should be added to the set E, then any solution already in E that is dominated by the newly obtained solution is removed from E. After the last iteration, the result of the algorithm is the set E, which is the set of non-dominated solutions obtained in all of the algorithm's iterations.

Based on the structure of the ABC algorithm, the vector evaluated technique and the use of elitism, which is the process of preserving previous high-performance solutions from one generation to the next using the concept of non-dominated solutions, the new combined VE-ABC algorithm that we propose is defined as follows:

1. Set $E=\varnothing$
2. Randomly generate a set S of i solutions as initial food sources, where i is the number of employed bees. Assign an employed bee to each food source.
3. Evaluate the nectar amount (fitness), $f_j(s_i)$, of each food source, for each objective function j.
4. For each $s_i \in S$, if s_i is a non-dominated solution add s_i to E.
5. Repeat until a stopping condition is met
   a. For each food source $s_i \in S$, apply the neighborhood operator to generate a neighbor food source, s'_i. If the fitness of the neighbor food source is better than that of the original food source, based on objective function j, i.e. $f_j(s_i)>f_j(s'_i)$ for maximization problems, when $j=\lfloor i/\lfloor NEB/NOBJ \rfloor \rfloor \ mod\ NOBJ$ (when NEB is the number of employed bees and $NOBJ$ if the

number of objective functions), then replace the original food source with this neighbor food source

b. Set $G_0=\emptyset$, $G_1=\emptyset,\ldots, G_i=\emptyset$, where i is the number of employed bees.

c. For each onlooker, use the fitness-based roulette wheel selection method to select a food source, s_i, using objective function j, where $j=\lfloor i/\lfloor NEB/NOBJ\rfloor\rfloor \bmod NOBJ$. Apply a neighborhood operator to s_i to find a neighbor food source, say s'_i. Add s'_i to G_i, i.e. $G_i=G_i\cup s'_i$.

d. For each food source $s_i\in S$, if $G_i\neq\emptyset$ then let s' be the source food with best fitness value in G_i, when the fitness is evaluated regarding objective j, when $j=\lfloor i/\lfloor NEB/NOBJ\rfloor\rfloor$. If the fitness of s' is better than that of s_i, then replace s_i with s'.

e. For each $s_i\in S$, if s_i is not dominated by any solution $e_j\in E$, add s_i to E and check each solution $e_j\in E$. If e_j is dominated by s_i, remove e_j from E.

f. Replace any food sources $s_i\in S$ whose fitness has not been improved for *limit* iterations with randomly generated solutions

6. Output the best food source (solution) found (meaning the set E).

C. *VRPTW Implementation*

A candidate solution to an instance of the VRPTW must specify the number of vehicles required, the distribution of the demands imposed upon these vehicles and the delivery order for each route. In order to maintain the simplicity of the VE-ABC algorithm, a simple and straightforward solution representation scheme is adopted. Suppose n customers are visited by m vehicle routes. The representation has the form of a vector of length (n+m-1). In the vector, there are n integers between 1 and n inclusively representing the identities of customers. There are also m-1 0s in the vector acting as separators between vehicle routes from the depot. The sequence between two 0s is the sequence of customers to be visited by a vehicle.

4	7	0	1	3	2	0	6	5

Figure 1. – An example of a chromosome with 7 customers and 3 routes

This kind of representation was used by Pereira, Tavares [42] for their GA and by Szeto, Wu [31] for their ABC algorithm.

Any solution used by the VE-ABC algorithm must be feasible. In order to simplify the process of creating a feasible solution, a solution in which every customer is assigned to a different vehicle was chosen. This approach is similar to the initial solution of Clarke and Wright [43] savings algorithm. In order to increase the randomness of the solutions, and increase the effectiveness of the neighborhood operators, the order of the customers in the initial solutions is randomly selected.

A neighborhood operator is used to obtain a new solution x' from the current solution x in step 5 of the VE-ABC heuristic. For every solution, a number of randomly chosen neighborhood operators are applied. The order of the neighborhood operation, is also randomly chosen. The possible operators include:

1. **Random swaps** – After two positions in the solution vector are randomly selected, the customers located at these two points are swapped.

2. **Random swaps of subsequences** – This is an extension of the previous operation. Two subsequences of random lengths of customers and depot are selected and swapped.

3. **Random insertions** - This operator consists of randomly selecting two positions, i and j, and relocating the customer from position i to position j.

4. **Random insertions of subsequences** – This is an extension of the previous operation. A subsequence of random length of customers and depot starting from position i is relocated to position j.

5. **Reversing a subsequence** – A subsequence of random length that includes consecutive customers and a depot is selected and then the order of the corresponding customers and depot is reversed.

6. **Random swaps of reversed subsequences** – This operation is a combination of (a) random swap of subsequences and (b) reversing a subsequence operators. Two subsequences of random lengths of customers and depot are chosen and swapped. Then each of the swapped subsequences is reversed.

7. **Random insertions of reversed subsequences** - A subsequence of random length of customers and depot starting from position i is relocated to position j. Then the relocated subsequence is reversed.

8. **Merge routes** – Two following routes in a solution are merged into a single route.

9. **Split routes** – A single route in a solution is split into two different routes.

Neighborhood operation may cause a feasible solution to become not feasible. Accordingly, before applying a neighborhood operation, the original solution is saved, and then the new solution is checked against the problem's constraints. If the new solution does not violate the problem's constraints, the new solution is kept. Otherwise, the solution is returned to its original state.

III. Experimental Results

In order to evaluate the results of the VE-ABC algorithm, the algorithm was tested using the well-known Solomon's instances, as did Iqbal and Rahman [34], Häckel and Dippold [32] and Shi, Meng [33] in their researches. Solomon's instances are three sets of test problems, each with a different number of customers (25, 50 and 100). Each set is composed of 56 test problems that are grouped into six problem sets according to geographical data; percent of time-constrained customers; and tightness and positioning of the time windows. The geographical data is randomly generated in problem sets R1 and R2 and clustered in problem sets C1 and C2. In problem sets RC1 and RC2 there is a mix of random and clustered structures. Problem sets R1, C1 and RC1 have a short scheduling horizon and only a few customers per route can be serviced. R2, C2 and RC2 have a long scheduling horizon permitting many customers to be serviced by the same vehicle. Detailed data about the test problems, as well as best-known solutions, can be found in *http://web.cba.neu.edu/~msolomon/problems.htm*.

All three problem sets were tested using VE-ABC algorithm. All experiments were performed on a 2.8 Giga-Hertz AMD Athlon 64 X2 5600 computer running Windows XP with 2GB RAM. The heuristics were coded in C# and Dot Net 4.0 (Visual studio 2010).

A setup phase was conducted, in which it was found that the algorithm performs best when *limit* is set to 25 for problems with 25 customers, 50 for problems with 50 customers and 300 for problems with 100 customers. Using curve fitting, the size of *limit* can be defined as 0.053*n*2-3*n*+66.667, where *n* is the number of customers. It was also found that the number of iterations, number of employed bees as well as the number of onlookers bees are dependent on the size of the problem. For problems with 25 customers, by average, 125 iterations were needed in order to converge to the optimal solution. For problems with 50 customers, 500 iterations were needed and for problems with 100 customers, 5000 iterations were needed in order to converge to the optimal solution. Using curve fitting, the numbers of iterations can be defined as n^2-60n-1000. It was also found that the number of employed bees should be 5 times the number of customers, and that the number of onlookers bees should be 10 times the numbers of employed bees.

Szeto, Wu [31] showed that each neighborhood operator had a different effect on the algorithm's result. In order to overcome this effect whenever a neighborhood operator has to be selected and applied, roulette wheel selection is used. Each neighborhood operation has been assigned with a counter. Each time a neighborhood operator is applied on a given solution, its result is compared with the original solutions, and if the new solution is better than the original solution, the counter associated with the specific neighborhood operator is increased. Next time a neighborhood operator has to be used, roulette wheel selection, based on the neighborhood operator's counter is used for selecting a neighborhood operator. Whenever a counter reaches a pre-determined value (20 in the described test) all counters are set back to 1. This is done to avoid the possibility of choosing the same neighborhood operator every time due to a high value of a specific counter.

As stated above, the objective of VRPTW is to compute the shortest path in order to obtain minimum traveling costs and the least number of vehicles [3]. The results show that for all problems, the VE-ABC algorithm was unable to reach a solution with a distance lower than the distance of the best-known solution. Nevertheless, the results are very close to the result of the best-known solutions. As it can be seen from the results, for all problems with 25 customers, the algorithm was able to find a solution in which the distance of the tour is not higher than 2.08% (0.3% in average) from the best-known solution, with average running time of 10.3 seconds. For problems with 50 customers, the best-known solutions are provided for 26 out of the 55 problems (47%). In this case, the algorithm was able to find a solution in which the distance of the tour is no higher than 2.5% (0.9% on average), when the average deviation of all solutions from the best-known solution is 3.4%, with average running time of 179.6 seconds. As for problems with 100 customers, the best-known solutions are provided for 38 out of the 55 problems. Using 5000 generations and setting the limit to 300, the algorithm was able to find a solution in which the distance of the route is no higher than 2.4% (1.3% on average) from the best-known solution, when the average deviation of all solutions from the best-known solution is 4.7%, with average running time to 4304 seconds.

As for the second objective, minimizing the number of vehicles, for some of the test problems the algorithm was able to find a solution in which the number of vehicles is smaller than the number of vehicles of the best-known solutions. For problems with 25 customers, there was an improvement of one vehicle in 15 out of 56 problems; in two additional problems, there was an improvement of two vehicles. For problems with 50 customers, there was an improvement of one vehicle in 11 out of 55 problems; an improvement of two vehicles was found in two more problems; and in one additional problem, there was an improvement of three vehicles. No improvement was found for problems with 100 customers.

The results described above are summarized in Table 1. In this table for each test problem, $\mathbf{Opt_V}$ denotes the minimum number of vehicles needed as obtained by the VE-ABC algorithm, while $\mathbf{Best_V}$ denotes the minimum number of vehicles known so far. Similarly, $\mathbf{Opt_L}$ denotes the shortest tour length obtained by the VE-ABC algorithm, while $\mathbf{Best_L}$ denotes the shortest tour length known so far. Using these notations, $\mathbf{Opt_V}<\mathbf{Best_V}$ denotes that the number of vehicles found by the VE-ABC algorithm is smaller than the number of vehicles used by the best-known solution. $\mathbf{Opt_V}=\mathbf{Best_V}$ denotes that the number of vehicles found by the VE-ABC algorithm is equal to the number of vehicles used by the best-known solution, and $\mathbf{Opt_V}>\mathbf{Best_V}$ denotes that the number of vehicles found by the VE-ABC algorithm is higher than the number of vehicles used by the best-known solution. $\mathbf{Opt_L}<\mathbf{Best_L}$ denotes that the tour length found by the VE-ABC algorithm is smaller than the tour length of the best-known solution. (the difference is higher than 0.025%). $\mathbf{Opt_L}=\mathbf{Best_L}$ denotes that the tour found by the VE-ABC algorithm is equal to the tour length of the best-known solution (the difference is less than 0.025%), and $\mathbf{Opt_L}>\mathbf{Best_L}$ denotes that the tour length found by the VE-ABC algorithm is higher than tour length of the best-known solution (the difference is higher than 0.025%).

Num. of Customers	**Length Vehicles**	$\mathbf{Opt_L>Best_L}$	$\mathbf{Opt_L=Best_L}$	$\mathbf{Opt_L<Best_L}$

Num. of Customers	Length Vehicles	$Opt_L>Best_L$	$Opt_L=Best_L$	$Opt_L<Best_L$
25	$Opt_v>Best_v$	0	4	0
	$Opt_v=Best_v$	2	54	0
	$Opt_v<Best_v$	15	4	0
50	$Opt_v>Best_v$	15	4	0
	$Opt_v=Best_v$	25	19	0
	$Opt_v<Best_v$	13	4	0
150	$Opt_v>Best_v$	33	2	0
	$Opt_v=Best_v$	8	7	0
	$Opt_v<Best_v$	0	0	0

An analysis of the best-known solutions shows that the best-known solutions for the set of problems with 25 customers was found using at least four different algorithms. For problems with 25 customers, since the VE-ABC algorithm was able to arrive at the best-known solution for all 56 different problems, it provides a good alternative to the four other algorithms. For the set of problems with 50 customers, the best-known solutions were found using at least six different algorithms. For the set of problems with 100 customers, the best-known solutions were found using at least four different algorithms. For problems with 50 and 100 customers, the VE-ABC algorithm was able to find a solution close to the best-known solution for all problems. There is no other algorithm known to the authors that is capable of achieving this result.

As it can be seen from the results, the main problem of the algorithm is that the running time of the algorithm increases dramatically as the problem size increases. This can be solved by combining the VE-ABC algorithms with other heuristics as can be seen from the next examples.

For problem C101 with 100 customers, when divided to 4 groups each of 25 customers using a Sweep like heuristics, a result with 14 vehicles and total length of 1034.6 was obtained. For problem C102 with 100 customers, a result with 13 vehicles and total length of 964.2 was obtained and for problem C103 with 100 customers, the result obtained was a result with 12 vehicles and total length 957.9. The average running time for all problems was 63.7 seconds.

IV. Conclusions

In this paper, an ABC algorithm which implements the vector evaluation approach was developed to solve a multi-objective vehicle routing problem with time windows. Most heuristic and meta-heuristic algorithms for solving VRPTW, solve the problem in two stages. In the first stage, the primary objective, minimizing the number of vehicles, is optimized. In the second stage, the total route length is optimized based on the results of the previous stage.

In this paper, the VRPTW problem was constructed as a multi-objective problem, therefore, the two objectives were simultaneously optimized. Using Solomon's instances, it was shown that in cases with 25 or 50 customers, the algorithm was able to find solutions better than the best-known solutions regarding the number of vehicles used. For problems with 100 customers, no improvement was found compared to the best-known solutions. It was also shown that for problems with 100 customers, an increase in the number of iterations and a higher value of limit can bring the solution closer to the best-known solutions.

The proposed algorithm also has the advantage of being able to solve a wide range of problems in contrast to the common development of a problem specific algorithm.

Further research directions might include evaluating the number of iterations and of customers and new neighborhood operators, etc., as well as combining the VE-ABC algorithms with other heuristics.

References

[1] Toth, P. and D. Vigo, *The Vehicle Routing Problem*, ed. P. Toth and D. Vigo. 2001, Philadelphia: Siam.

[2] Dantzig, G.B. and J.H. Ramser, *The Truck Dispatching Problem.* Management Science, 1959. 6(1): p. 80-91.

[3] Cordeau, J.-F.c., et al., *Vehicle Routing.* 2005.

[4] Bowerman, R., B. Hall, and P. Calamai, *A Multi-Objective Optimization Approach to Urban School Bus Routing: Formulation and Solution Method.* Transportation Research Part A, 1995. 29(2): p. 107-123.

[5] Hong, S.C. and Y.B. Park, *A Heuristic for Bi-Objective Vehicle Routing with Time Window Constraints.* International Journal of Production Economics, 1999. 62(3): p. 249-258.

[6] Coello, C.A.C., G.B. Lamont, and D.A. Van Veldhuizen, *Evolutionary Algorithms for Solving Multi-Objective Problems*. 2007, New-York, USA: Springer-Verlag New-York Inc.

[7] Golden, B.L., A.A. Assad, and E.A. Wasil, *Routing Vehicles in the Real World: Applications in the Solid Waste, Beverage, Food, Dairy, and Newspaper Industries.* The Vehicle Routing Problem, 2002: p. 245-286.

[8] Desrochers, M., J. Desrosiers, and M.M. Solomon, *A New Optimization Algorithm for the Vehicle Routing Problem with Time Windows.* Operations Research, 1992. 40(2): p. 324-354.

[9] Chabrier, A., *Vehicle Routing Problem With Elementary Shortest Path Based Column Generation.* Computers & Operations Research, 2006. 33(10): p. 2972-2990.

[10] Irnich, S. and D. Villeneuve, *The Shortest Path Problem with Resource Constraints and K-Cycle Elimination for K>=3.* INFORMS Journal on Computing, 2003. 18(3): p. 391-406.

[11] Jepsen, M., et al., *A Non-Robust Branch-and-Cut-and-Price Algorithm for the Vehicle Routing Problem with Time Windows.* Oper. Res., 2006.

[12] Jepsen, M., et al., *Subset-Row Inequalities Applied to the Vehicle-Routing Problem with Time Windows.* Operations Research, 2008. 56(2): p. 497.

[13] Kallehauge, B., J. Larsen, and O.B.G. Madsen, *Lagrangian Duality Applied to The Vehicle Routing Problem with Time Windows.* Computers & Operations Research, 2006. 33(5): p. 1464-1487.

[14] Lenstra, J.K. and A. Kan, *Complexity of Vehicle Routing and Scheduling Problems.* Networks, 1981. 11(2): p. 221-227.

[15] Solomon, M.M., *Algorithms for the Vehicle Routing and Scheduling Problems with Time Window Constraints.* Operations Research, 1987. 35(2): p. 254-265.

[16] Abbass, H.A., *An Economical Cognitive Approach for Bi-Objective Optimization Using Bliss Points, Visualization, and Interaction.* Soft Computing-A Fusion of Foundations, Methodologies and Applications, 2006. 10(8): p. 687-698.

[17] Bräysy, O. and M. Gendreau, *Vehicle Routing Problem With Time Windows, Part I: Route Construction and Local Search Algorithms.* Transportation Science, 2005. 39(1): p. 104-118.

[18] Bräysy, O. and M. Gendreau, *Vehicle Routing Problem With Time Windows, Part II: Metaheuristics.* Transportation Science, 2005. 39(1): p. 119-139.

[19] Cordeau, J.F., et al., *The Vehicle Routing Problem With Time Windows*, in *The Vehicle Routing Problem*, P. Toth and D. Vigo, Editors. 2001, SIAM Monographs on Discrete Mathematics and Applications: Philadelphia. p. 157–193.

[20] Golden, B.L., S. Raghavan, and E.A. Wasil, *The Vehicle Routing Problem: Latest Advances and New Challenges*. Vol. 43. 2008, New-York, USA: Springer Verlag.

[21] Mester, D. and O. Bräysy, *Active Guided Evolution Strategies for Large-Scale Vehicle Routing Problems with Time Windows.* Computers & Operations Research, 2005. 32(6): p. 1593-1614.

[22] Homberger, J. and H. Gehring, *A Two-Phase Hybrid Metaheuristic for the Vehicle Routing Problem with Time Windows.* European Journal of Operational Research, 2005. 162(1): p. 220-238.

[23] Bent, R. and P. Van Hentenryck, *A Two-Stage Hybrid Local Search for the Vehicle Routing Problem With Time Windows.* Transportation Science, 2004. 38(4): p. 515-530.

[24] Pisinger, D. and S. Ropke, *A General Heuristic for Vehicle Routing Problems.* Computers & Operations Research, 2007. 34(8): p. 2403-2435.

[25] E., P.G., D. G., and R.L. M., *A Branch and Price Based Large Neighborhood Search Algorithm for the Vehicle Routing Problem with Time Windows.* Networks, 2009. 54(4): p. 190-204.

[26] Ibaraki, T., et al. *Effective Local Search Algorithms for the Vehicle Routing Problem with General Time Window Constraints*. in *MIC'2001 - 4th Metaheuristics International Conference*. 2001. Citeseer.

[27] Ibaraki, T., et al., *An Iterated Local Search Algorithm for the Vehicle Routing Problem with Convex Time Penalty Functions.* Discrete Applied Mathematics, 2008. 156(11): p. 2050-2069.

[28] Lim, A. and X. Zhang, *A Two-Stage Heuristic with Ejection Pools and Generalized Ejection Chains for The Vehicle Routing Problem with Time Windows.* INFORMS Journal on Computing, 2007. 19(3): p. 443-457.

[29] Karaboga, D., *An Idea Based on Honey Bee Swarm for Numerical Optimization*, in *Technical Report TR06*. 2005, Computer Engineering Department, Erciyes University, Turkey.

[30] Karaboga, D. and B. Akay, *A Modified Artificial Bee Colony (ABC) Algorithm for Constrained Optimization Problems.* Applied Soft Computing, 2011. 11(3): p. 3021-3031.

[31] Szeto, W.Y., Y. Wu, and S.C. Ho, *An Artificial Bee Colony Algorithm for the Capacitated Vehicle Routing Problem.* European Journal of Operational Research, 2011. 215: p. 126–135.

[32] Häckel, S. and P. Dippold. *The Bee Colony-Inspired Algorithm (BCiA): A Two-Stage Approach for Solving the Vehicle Routing Problem With Time Windows*. in *11th Annual Conference on Genetic and Evolutionary Computation*. 2009. Montréal, Québec, Canada: ACM.

[33] Shi, Y.J., F.W. Meng, and G.J. Shen, *A Modified Artificial Bee Colony Algorithm for Vehicle Routing Problems with Time Windows.* Information Technology Journal, 2012. 11: p. 1490-1495.

[34] Iqbal, S., and Rahman, R.M., *Vehicle Routing Problems With Soft Time Windows*. in *2012 7th International Conference on Electrical & Computer Engineering (ICECE)*. 2012. IEEE.

[35] Holland, J.H., *Adaptation in Natural and Artificial Systems*. 1975, Michigan, USA: University of Michigan Press Ann Arbor.

[36] Mitchell, M., *An Introduction to Genetic Algorithms*. 1996, Cambridge, Massachusetts, USA: MIT Press.

[37] Homberger, J. and H. Gehring, *Two Evolutionary Metaheuristics for the Vehicle Routing Problem with Time Windows.* Infor-Information Systems and Operational Research, 1999. 37(3): p. 297-318.

[38] Gehring, H. and J. Homberger, *Parallelization of a Two-Phase Metaheuristic for Routing Problems with Time Windows.* Journal of heuristics, 2002. 8(3): p. 251-276.

[39] Berger, J. and M. Barkaoui, *A New Hybrid Genetic Algorithm for the Capacitated Vehicle Routing Problem.* Journal of the Operational Research Society, 2003. 54: p. 1254-1262.

[40] Schaffer, J. *Multi-Objective Optimization with Vector Evaluated Genetic Algorithms*. in *1st International Conference on Genetic Algorithms*. 1985.

[41] Weuster-Botz, D., *Experimental Design for Fermentation Media Development: Statistical Design or Global Random Search?* Journal of Bioscience and Bioengineering, 2000. 90(5): p. 473-483.

[42] Pereira, F.B., et al., *GVR: A New Genetic Representation for the Vehicle Routing Problem.* Lecture Notes in Computer Science, 2002. 2464: p. 95-102.

[43] Clarke, G. and J. Wright, *Scheduling of Vehicles from a Central Depot to a Number of Delivery Points.* Operations Research, 1964. 12: p. 568-581.