International Journal of Computer System (ISSN: 2394 – 1065), Volume 01– Issue 02, November, 2014
Available at http://www.ijcsonline.com/

Multi-Objective Stochastic VRP – Fitness Calculation and Algorithm Converges Using a Generic Genetic Algorithm

Oren E. Nahum[A], Yuval Hadas[A], Uriel Spiegel[A,B] and Reuven Cohen[C]

[A] Department of Management, Faculty of Social Sciences, Bar-Ilan University, Ramat-Gan, Israel.
[B] Department of Economics, University of Pennsylvania, Philadelphia, PA, USA.
[C] Department of Mathematics, Faculty of Exact Sciences, Bar-Ilan University, Ramat-Gan, Israel.

Abstract

Vehicle-routing problems (VRP), which can be considered a generalization of TSP, have been studied in depth. Many variants of the problem exist, most of them trying to find a set of routes with the shortest distance or time possible for a fleet of vehicles. This paper combines two important variants, the stochastic time-dependent VRP and the multi-objective VRP. A genetic algorithm for solving the problem is introduced. A comparison of two fitness functions, with significant difference in computational time, is also presented. Finally, a comparison of solution selection based on TOPSIS method and the two fitness functions is also examined. Results show that a significant decrease in running time, minutes compared to hours, can be achieved, with no impact on the final results of the algorithm.

Keywords: *Multi-Objective, Vehicle Routing, Genetic Algorithms, Fitness, TOPSIS*

I. INTRODUCTION

The Vehicle-Routing Problem (VRP) 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. VRP, which can be considered a generalization of the "Traveling-Salesman Problem" [1], is a NP-Hard problem and, therefore, cannot be solved optimally within a reasonable running time. Since the problem was first introduced in 1959, a large number of algorithms for solving it, based on various heuristics and meta-heuristics, as well as extensions to the basic VRP, aiming to produce more realistic models, usually by adding more constraints to the original problem, were introduced. For a discussion about some of the most important algorithms developed so far, and various extensions see [2], [3] and [4].

This paper is based on an on going work, which aims to develop a model and an algorithm for solving the multi-objective real-time vehicle routing problem. In real-time vehicle routing problems, information, such as customers demands, travel time between two points and more, is not known to the algorithm at the beginning, and is reviled as the algorithm progress. If a given solution has to be updated, based on the new information, the changes to the solution are very small (if any) as long as the new information is processed as soon as it has been reviled. Genetic algorithms, a meta-heuristics for solving optimization problems, was chosen as a method for solving the multi-objective real-time vehicle routing problem. In genetic algorithms a set of solution is created in each iteration (the number of iteration is chosen by the user or is defined as a condition), based on the set of solution created in the previous iteration. It is easily possible to update a set of solution based on new information, and continue with this set as the base for the new set of solution. This is the main reason for choosing genetic algorithm. Several factors may affect the quality of a solution obtained from a genetic algorithm. One of them is the quality of accuracy of the fitness functions, on which the algorithm is based when creating the next generation of solutions in a given iteration. In this paper we address the problem of choosing the right fitness function for solving the multi-objective real-time vehicle routing problem, focusing on the accuracy and speed of calculation vs. the quality of the solution and the rate of conversion. To simplify the analysis, instead of using the multi-objective real-time vehicle routing problem, the multi-objective stochastic time-dependent vehicle routing problem, which is a combination of three known extensions, (1) stochastic VRP, (2) time-dependent VRP and (3) multi-objective VRP, was used.

The rest of this paper is as follows. Chapter 2 provides a literature review on Time Dependent VRP, stochastic VRP and multi-objective VRP. Chapter 3 provides a mathematical formulation of the multi-objective stochastic time-dependent vehicle routing problem. A genetic algorithm for solving the problem is described in chapter 3, and a discussion various aspects regarding the algorithm and its performance are presented in chapter 4. The conclusions are presented in chapter 5.

II. LITERATURE REVIEW

A. Time Dependent VRP

In the real world, especially in urban areas, the travel time is dependent on both the distance between two customers and the time of day. Ignoring the fact that for some routes the travel time changes throughout the day, may result in solutions that are far from optimal. For that reason, the Time-Dependent VRP (TDVRP) was developed. Whereas most VRP variants look for the shortest paths in terms of length, the TDVRP seeks the shortest paths in terms of travel time.

There has been limited research related to the time-dependent vehicle routing problem compared to other VRP models [5].

Time dependent VRP was first formulated by Malandraki and Daskin [6, 7] using a mixed integer linear programming formulation. Malandraki and Daskin developed two algorithms for solving the problem, a greedy nearest-neighbor algorithm, and a branch and bound-based algorithm that provided better solutions, but was suitable only for small problems. Hill and Benton [8] also studied the vehicle routing problem with time-dependent travel times and proposed a simple greedy heuristic for the problem.

Ahn and Shin [9] discussed modifications to the savings, insertion, and local improvement algorithms to better cope with TDVRP.

An important property for time dependent problems is the First In - First Out (FIFO) principal [9, 10]. A model with a FIFO property guarantees that if two vehicles left the same location for the same destination (and traveled along the same path), the one that left first would never arrive later than the other. While it is an intuitive and desirable property, it is not present in earlier work [6-8, 11], and therefore, the FIFO property is not guaranteed.

Ichoua, Gendreau and Potvin [10] introduced a model that guarantees the FIFO principle. This model is satisfied by working with step-like speed distributions and adjusting the travel speed whenever a vehicle crosses the boundary between two consecutive time periods. The algorithms that they developed, which were based on the tabu-search meta-heuristic, provided better solutions for most test scenarios.

Fleischmann, Gietz and Gnutzmann [12] utilized route construction methods already proposed in the literature, savings and insertion, to solve uncapacitated time dependent VRP with and without time windows. Fleischmann and Gietz assume travel times to be known between all pairs of interesting locations and constant within given time slots. Neighbor slots with similar travel times are joined to reduce memory requirements, and the transitions between slots are smoothed to ensure a FIFO property on travel times. Fleischmann and Gietz tested their algorithms in instances created from Berlin travel time data. Time dependent VRP with time windows was also addressed by Hashimoto, Yagiura and Ibaraki [13] who proposed an iterated local search algorithm.

Jung and Haghani [14, 15] proposed a genetic algorithm to solve time dependent problems. By formulating the problem as a mixed integer linear programming problem, they obtain lower bounds by relaxing most of the integer requirements. The lower bounds are compared with the primal solutions from the genetic algorithm to evaluate the quality of the solutions. Using randomly generated test problems, the performance of the genetic algorithm was evaluated by comparing its results with exact solutions.

Van Woensel, Kerbache, Peremans and Vandaele [16] used a tabu search to solve CVRP with time dependent travel times (with no time windows). Approximations based on queuing theory and the volumes of vehicles in a link were used to determine the travel speed. Donati, Montemanni, Casagrande, Rizzoli and Gambardella [17] proposed an algorithm based on an ant colony heuristic approach and a local search improvement approach. The algorithm was tested using a real life network in Padua, Italy, and some variations of the Solomon problem set.

Ji and Wu [5] proposed a revised scheme to the Artificial Bee Colony algorithm (a new population-based metaheuristic approach proposed by Karaboga [18], inspired by the intelligent foraging behavior of honeybee swarm), with improved performance for solving Capacitated VRP with Time-dependent Travel Times. Using a set of instances of different size, Ji and Wu showed that the ABC algorithm is improved in term of better solution achieved, greater robustness and higher computational efficiency.

B. Stochastic VRP

A stochastic vehicle-routing problem arises when at least one of the problem's variables is random [19]. Over the years, different solution frameworks have been developed for solving the problem [20]. A taxonomy of these frameworks classifies them into dynamic or static [21].

Stochastic VRP can be divided into the following classes [22]: (1) VRP with stochastic demand (VRPSD), in which the vehicles serve a set of customers with stochastic and uncertain demands [23-26]. (2) VRP with stochastic customers (VRPSC), in which each customer has a deterministic demand and a probability p of being present. (3) VRP with stochastic customers and demands, a combination of VRPSD and VRPSC. For a detailed survey of the SVRP, one may refer to [19].

A stochastic model is usually modeled in two stages [25]. In the first stage, a planned a-priori route is determined. In the second stage, corrective action, based on actual information, is applied to the solution of the first stage. Methods modeled in two stages include a branch-and-bound method based on the integer L-shaped algorithm for solving VRP with stochastic demands, proposed by Laporte, Louveaux and Van Hamme [27]. In a more recent work, Rei, Gendreau and Soriano [28] tackled the single VRPSD (SVRPSD), a variant where only one route is to be designed. Their method consists of using local branching to generate optimality cuts on an integer L-shaped algorithm. Although successful, these approaches are limited to solve instances of up to 100 customer nodes.

Stochastic travel times were introduced into the vehicle-routing problem by Laporte, Louveaux and Mercure [29], who presented a CCP model. Their aim was to find a set of paths that had a travel time that was no longer than a given constant value. The problem was solved optimally by means of an Integer L-shaped algorithm for $10 \leq n \leq 20$ and two to five travel time scenarios (each scenario corresponds to a different travel speed for the entire network).

In VRP with Stochastic Travel Times (VRPSTT). Vehicles follow their planned routes and may incur a penalty if the route duration exceeds a given deadline. It is natural to make this penalty proportional to the elapsed route duration in excess of the deadline [29]. Another possibility is to define a penalty proportional to the

uncollected demand within the time limit, as is the case in a money collection application studied by Lambert, Laporte and Louveaux [30]. Wang and Regan [31] have proposed models for this class of problems under the presence of time windows.

In a more recent study, Kenyon and Morton [32] have investigated properties of VRPSTT solutions and have developed bounds on the objective function value. They have developed two models for the stochastic VRP with random travel and service times and an unknown distribution. The first model minimizes the expected completion time, and the second model maximizes the probability that the operation is complete prior to a pre-set target time *T*. Both models are based on a heuristic that combines branch-and-cut and Monte-Carlo simulation which, if run to completion, terminates with a solution value within a preset percentage of the optimum. Using small instances (9-nodes and 28-nodes) Kenyon and Morton showed that using their models' solutions to VRPSTT can be significantly better than solutions obtained by solving the associated mean-value model.

C. *Multi-objective VRP*

VRPs are frequently used to model real cases. However, they are often set up with the single objective of minimizing the cost of the solution, although the majority of the problems encountered in industry, particularly in logistics, are multi-objective in nature. As an example consider the work of Park and Koelling [33, 34], in which, the distance traveled must be minimized to avoid damaging the product being transported. Multi-objective optimization is one possible way to study other objectives other than the one initially defined, without changing the definition of the problem. The purpose of such extensions is often to enhance the practical applications of the model by recognizing that logistics problems are not only cost driven. An excellent survey of related multi-objective VRPs is given by Jozefowiez, Semet and Talbi [35]. This section will review some of the most recent research in this field.

Multi-objective routing problems are usually studied for a specific real-life situation, in which decision makers define several clear objectives that they would like to see optimized.

Gupta, Singh and Pandey [36] presented a case study with the overall goal of developing a plan for the Jain University bus service to be able to serve all customers in the most efficient way. In this study four objectives were considered: (1) the minimization of the total route length, (2) the minimization of the fleet size, (3) the maximization of average grade of customer satisfaction, and (4) the minimization of total waiting time over vehicles.

Motivated by the case of Lantmannen, a large distributor operating in Sweden, Wen, Cordeau, Laporte and Larsen [37] proposed model and solve the dynamic multi-period vehicle routing problem (DMPVRP). In the DMPVRP, customers place orders dynamically over a planning horizon consisting of several periods. Each request specifies a demand quantity, a delivery location and a set of consecutive periods during which delivery can take place. The distributor must plan its delivery routes over several days so as to (1) minimize the total travel time and (2) customer waiting, and to (3) balance the daily workload over the planning horizon.

Faccio, Persona and Zanin [38] studied the problem of municipality solid waste collection optimization considering real time input data, homogeneous and variable fleet size based in a single depot. In the study three objective functions where addressed: (1) the minimization of the number of vehicles, (2) the minimization of travel time and (3) the minimization of total distance covered.

Anbuudayasankar, Ganesh, Lenny Koh and Ducq [39] addressed the bi-objective vehicle routing problems with forced backhauls, in which the optimization of the process of replenishing money in ATMs is considered as a bi-objective problem which minimizes the total routing cost and the span of travel tour.

Most of the studies dealing with objectives related to node/arc activity involve time windows. In such studies, the time windows are replaced by an objective that minimizes either the number of violated constraints [40], the total customer and/or driver's wait time due to earliness or lateness [41-43], or both objectives at the same time [44].

One objective that often appears is the minimization of the number of vehicles. For VRP with time windows, the classic model has two objectives that are treated lexicographically (1) minimizing the number of vehicles and (2) minimizing then the length of the solution for that given number of vehicles. The existing research on multi-objective VRPs with time windows assigns the same level of priority for both objectives, rather than considering them lexicographically.

Some multi-objective routing problems do not share common objectives with classic routing problems at all. For example, Jozefowiez, Semet and Talbi [45] propose a meta-heuristic method based on an evolutionary algorithm for solving a bi-objective vehicle routing problem in which the total length of routes is minimized as well as the balance of routes, i.e. the difference between the maximal route length and the minimal route length.

Chitty and Hernandez [46] define a dynamic VRP in which the total mean transit time and the total variance in transit time are minimized simultaneously. Likewise, Murata and Itai [47, 48] define a bi-objective VRP which seeks to minimize both the number of vehicles and the maximum routing time of those vehicles (makespan).

III. The Multi-Objective Stochastic Time-Dependent Vehicle-Routing Problem

This chapter provides a mathematical formulation to the multi-objective stochastic time-dependent vehicle routing problem. Since VRP is a hard optimization problem [2, 3], the complexity of the problem will remain the same as CVRP, at least, because of the time dimension and the stochastic properties of the problem. Next a simple genetic algorithm designed for solving multi-objective optimization problems is described. This algorithm provides a set of routes that optimizes the following three objectives (all were addressed in previous litrature): (1) minimizing the total travel time [41-43, 49]; (2) minimize the number of vehicles in use [43, 47, 48, 50] and (3)

minimize the difference of travel times among the routes of the solution [46], considering the following properties: (1) for certain routes, the travel time varies during the day and (2) travel time is stochastic.

A. Mathematical Formulation

Let $G=(V,E)$ be a complete graph, where $V=\{0,\ldots,n\}$ is the vertex set and E is the edge set. Each vertex $i \in V \setminus \{0\}$ represents a customer, having a non-negative demand d_i, whereas vertex 0 corresponds to the depot. Each edge $e \in E=\{(i,j): i,j \in V, i<j\}$ is associated with a stochastic time-dependent nonnegative cost, $c_{ij}^{\,t}$, which represents the travel cost (equal to the travel time) spent to go from vertex i to vertex j starting at time t.

The use of the loop edges, (i,i), is not allowed (this is imposed by defining $c_{ii}^{\,t} = +\infty$ for all $i \in V$). A fixed fleet of M identical vehicles, each of capacity D, is available at the depot.

The Stochastic Time-Dependent VRP calls for the determination of a set of at most M routes whose total travel cost is minimized and such that: (1) for a given probability, α, the total travel cost will not be higher than c^*; (2) each customer is visited exactly once by one route; (3) each route starts and ends at the depot, (4) and the total demand of the customers served by a route does not exceed the vehicle capacity D.

It is possible to solve the stochastic time-dependent VRP using a mixed integer linear programming, however, an estimate, $\overline{c_{ij}^{\,t}}$, to the stochastic cost function, $c_{ij}^{\,t}$, has to be defined first.

The travel time cost function, $c_{ij}^{\,t}$, is stochastic in nature, meaning that it may vary from one day to another. Therefore, the cost function, $c_{ij}^{\,t}$, is associated with a mean, $\mathbb{E}\left(c_{ij}^{\,t}\right)$, which describes the average travel time from node i to node j at time t and a standard deviation, $\sigma\left(c_{ij}^{\,t}\right)$, which shows how much variation there is from the mean. As an estimation to $c_{ij}^{\,t}$ the mean $\mathbb{E}\left(c_{ij}^{\,t}\right)$ can be used. However, for a route based on this estimation, the total travel time of the route will not reflect the possibility of arriving to node earlier or later than expected, and the changes in travel time it may cause. For that reason a different estimation to the stochastic cost function, $c_{ij}^{\,t}$, is suggested. Let $\epsilon(t)$ be an impact factor, which defines how much the value of $c_{ij}^{\,t}$ is affected by possible changes in travel time (compared to the mean) in previous and future time intervals, and is defined as $\epsilon(t) = t \cdot \max_{t' \le t} \left| \frac{\sigma\left(c_{ij}^{\,t'}\right)}{\mathbb{E}\left(c_{ij}^{\,t'}\right)} \right|$. The estimation to the stochastic cost function, $\overline{c_{ij}^{\,t}}$, is defined as $\overline{c_{ij}^{\,t}} = \frac{1}{2\epsilon(t)+1} \sum_{t'=t-\lfloor \epsilon(t) \rfloor}^{t+\lfloor \epsilon(t) \rfloor} \mathbb{E}\left(c_{ij}^{\,t'}\right)$. In this definition $\overline{c_{ij}^{\,t}}$ equals to an average of expected values of $c_{ij}^{\,t}$, thereby taking into consideration the possibility of being early or late. Based on this definition, the objective functions are:

$$\min \quad Z_1 = \sum_{i=0}^{N}\sum_{j=0}^{N}\sum_{m=1}^{M}\sum_{t=0}^{T} \overline{c_{ij}^{\,t}}\, x_{ij}^{\,mt} \tag{1}$$

$$\min \quad Z_2 = M \tag{2}$$

$$\min \quad Z_3 = \left\{ \sqrt{\frac{\sum_{m=1}^{M} w_m^2}{M} - \left(\frac{\sum_{m=1}^{M} w_m}{M} \right)^2} \right\} \tag{3}$$

where

$$w_m = \sum_{i=0}^{N}\sum_{j=0}^{N}\sum_{t=0}^{T} x_{ij}^{t} c_{ij}^{t} \tag{4}$$

and the constraints are:

$$x_{ii}^{\,mt} = 0 \quad \forall i \in N,\ m \in M,\ t \in T \tag{5}$$

$$\sum_{j=1}^{N}\sum_{t=0}^{T} x_{0j}^{\,mt} \le 1 \quad \forall m \in M \tag{6}$$

$$\sum_{j=1}^{N}\sum_{t=0}^{T} x_{0j}^{\,mt} = \sum_{i=1}^{N}\sum_{t=0}^{T} x_{i0}^{\,mt} \quad \forall m \in M \tag{7}$$

$$\sum_{j=0}^{N}\sum_{m=1}^{M}\sum_{t=0}^{T} x_{ij}^{\,mt} = 1 \quad \forall i \in N,\ i \ne j \tag{8}$$

$$\sum_{i=0}^{N}\sum_{m=1}^{M}\sum_{t=0}^{T} x_{ij}^{\,mt} = 1 \quad \forall j \in N,\ i \ne j \tag{9}$$

$$\sum_{i=0}^{N}\sum_{t=0}^{T} x_{ip}^{\,mt} - \sum_{j=0}^{N}\sum_{t=0}^{T} x_{pj}^{\,mt} = 0 \quad \begin{matrix} \forall m \in M, \\ \forall p \in N,\ p \ne 0 \end{matrix} \tag{10}$$

$$\sum_{k=0}^{N}\sum_{t=0}^{T}\sum_{m=1}^{M}\left(t \times x_{jk}^{\,mt}\right) \ge \sum_{j=0}^{N}\sum_{t=0}^{T}\sum_{m=1}^{M}\left(\left(t + \overline{c_{ij}^{\,t}}\right) \times x_{ij}^{\,mt}\right) \quad \forall i > 0,\ j > 0 \tag{11}$$

$$\sum_{i=0}^{N} d_i \left(\sum_{j=0}^{N}\sum_{t=0}^{T} x_{ij}^{\,mt} \right) \le D \quad \forall m \in M \tag{12}$$

$$P\left(\sum_{i=0}^{N}\sum_{j=0}^{N}\sum_{m=1}^{M}\sum_{t=t_s}^{T} c_{ij}^{t} x_{ij}^{mt} \le c^* \right) \ge \alpha \tag{13}$$

$$x_{ij}^{\,mt} \in \{0,1\} \quad \forall m \in V,\ i,j \in N,\ t \in T \tag{14}$$

Three objective functions are considered in this model. The first objective function is minimizing the total travel time, and is defined in equation (1). The second objective function, defined in equation (2), is minimizing the number of vehicles in use, and the third objective function, minimize the difference of travel times among the routes of the solution, is expressed by minimizing the standard deviation of the traveling time of each route in a set of routes and defined in equations (4) and (3).

In this model, constraint (5) states that a vehicle cannot travel from one node to itself. As stated before, all vehicles must start their routes at the depot (constraint (6)) and end their routes at the depot (constraint (7)). However, not all vehicles must leave the depot (implied by constraint (6)). Three constraints ensure that each customer is visited exactly once, when constraint (8) states that each visited customer, *j*, is visited by a vehicle arriving from either the depot (*i*=0) or another customer ($i \in N$, $i \neq 0$). Similarly, constraint (9) states that a vehicle serving customer *i*, must leave to either the depot (*j*=0) or another customer ($j \in N$, $j \neq 0$). Constraint (10) is a route continuity constraint. This constraint states that if there is a vehicle, visiting customer *i*, that leaves to customer *p*, and there also exists a vehicle, visiting customer *p*, that leaves to customer *j*, then, then the vehicle traveling from customer *i* to customer *p*, is the same vehicle traveling from customer *p* to customer *j*. If node *j* is visited after visiting node *i*, then the departure time, *t*, from node *j* is equal to or greater than the departure time from node *i* plus the travel time from node *i* to node *j* at time *t*. This is described by constraint (11). A demand constraint (constraint (12)) is also added. This constraint states that the total demand of all customers visited by the same vehicle must be less from or equal to the capacity of the vehicle. Constraint (13) is a chance constraint stating that the desired solution is a set of routes, that for a given probability, α, the traveling time will not be higher than c^*. A method for the determination of the value of c^* is described next in this chapter. The last constraint (constraint (14)) is added in order to verify that the decision variables, x_{ij}^{mt}, get values of either 0 or 1.

As mention earlier, constraint (13) is a chance constraint, which guarantees that the travel time of the set of routs, obtained by solving the mixed integer linear programming formulation, will not be higher than c^* for a given probability, α.

Let $c^{*}{}_{ij}^{t}$ be an instance of the stochastic cost function, c_{ij}^{t}. A relaxed deterministic version of the previous linear programming, can be defined by substituting $\overline{c_{ij}^{t}}$ with $c^{*}{}_{ij}^{t}$. A set of possible traveling times, Z^* can be created by solving the relaxed deterministic linear programming a large number of times (each time using a different instance of the cost function). From this set, $Z' = \min\left(Z \in Z^*\right)$ can be chosen, which satisfies $P\left(\left(Z \in Z^*\right) \leq Z'\right) \geq \alpha$ as c^*.

B. The Concept of "Best Solution"

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, as the optimum of each criterion do not usually point to the same alternative, a conflict exists. The notion of an "optimum solution" does not usually exist in the context of conflicting, multiple criteria. Optimal solution in multi-objective optimization problem is usually equivalent to choosing the best compromise solution. The "best solution" may be the "preferred (or best compromise) solution" or a "satisfying solution." In the absence of an optimal solution, the concepts of dominated and non-dominated solutions become relevant. In the multi-objective optimization literature, the terms "non-dominated solution," "Pareto optimal solution," and "efficient solution" are used interchangeably. In addition, concepts of "ideal solution" and "anti-ideal solution" are relevant in many multi-objective optimization methods.

A feasible solution that meets or exceeds the decision maker's minimum expected level of achievement (or outcomes) of criteria values is referred to as a *Satisfying Solution*.

A feasible solution (alternative) x_1 dominates another feasible solution (alternative) x_2 if x_1 is at least as good as (i.e., as preferred as) x_2 with respect to all objective functions and is better than (i.e., preferred to) 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. That is, for a non-dominated solution an increase in the value of any one objective function is impossible without some decrease in the value of at least one other objective function. Mathematically, a solution $x_1 \in X$ is non-dominated if there is no other $x \in X$ such that $C_i\left(x\right) \geq C_i\left(x_1\right)$, i = 1,2,...,k, and $C_i\left(x\right) \neq C_i\left(x_1\right)$.

IV. A Multi-Objective Genetic Algorithm

The first GA dealing with multiple objectives was the Vector Evaluated Genetic Algorithm (VEGA) proposed by Schaffer [51]. Being aware of the potential GAs have in multi-objective optimization, Schaffer proposed an extension of the simple GA to accommodate vector-valued fitness measures. In the VEGA Algorithm, the selection step was modified so that, at each generation, a number of sub-populations were 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 with the application of crossover and mutation in the usual way. However, as noted by Richardson, Palmer,

Liepins and Hilliard [52], shuffling all the individuals in the sub-populations together to obtain the new population is equivalent to linearly combining the fitness vector components to obtain a single-valued fitness function. The weighting coefficients, however, depend on the current population. This means that, in the general case, not only will two non-dominated individuals be sampled at different rates, but also, in the case of a concave trade-off surface, the population will tend to split into different species, each of them particularly strong in one of the objectives. Schaffer anticipated this property of VEGA and called it speciation. Speciation is undesirable in that it is opposed to the aim of finding a compromise solution.

This multi-objective optimization strategy has already been applied successfully for experimental medium optimization in many cases [53]. Although there are more recent published multi-objective GAs, such as the Non-Dominated Sorting Genetic Algorithm-II (NSGA-II) [54] and the Strength Pareto Evolutionary Algorithm (SPEA) [55], in this paper, the VEGA algorithm is used due to its simplicity.

A. An Improved VEGA Algorithm

The main disadvantage of the VEGA algorithm is the lack of elitism. In this section, an extended version of the VEGA algorithm, that uses elitism, is presented. In a regular GA, elitism means, that in every generation, the top ranked chromosomes are passed, without changes to the next generation. In this version of the algorithm, the set of non-dominated chromosomes is passed to the next generation.

The following is a description of the improved VEGA algorithm. For this problem, the structure of the chromosomes, crossover operation and mutation operations are the ones describes by [56].

PopSize – Population size

ProbCrossover – Probability of crossover

ProbMutation – Probability of mutation

IdealSolutionCount – Ideal Solution Counter

NumOfGeg – Number of generations

```
1.  IdealSolutionCount=0
2.  Population = Randomlly created population of size
    PopSize
3.  Calculate the fitnesses of each Chromosome in
    Population
4.  SaveNonDominated(Population,Elitism)
5.  PrevIdealSolution = GetIdealSolution(Elitism);
6.  Repeat NumOfGen times
7.    If |Elitism|>ElitismSize
8.         Population = CreateNewGeneration (Population,
           PopSize, ProbMutation, ProbCrossover)
9.    Else
10.        Population = CreateNewGeneration
           (Population,PopSize - |Elitism|,
           ProbMutation, ProbCrossover)
11.        Population = Population+Elitism
12.   Calculate the fitnesses of each Chromosome in
      Population
13.   SaveNonDominated(Population,Elitism
14.   CurrentIdealSolution=GetIdealSolution(Elitism)
15.   If CurrentIdealSolution=PrevIdealSolution
16.        IdealSolutionCount=IdealSolutionCount+1
17.        If IdealSolutionCount=10
18.             IdealSolutionCount=0
19.             ProbMutation=ProbMutation+0.05
20.             if ProbMutation>1
21.                  ProbMutation=1
22.        else
23.             IdealSolutionCount=0
24.             ProbMutation=0.1
```

The IVEGA algorithm starts with a creation of an initial population of size *PopSize*. Generally, the chromosomes of the initial population are randomly created, but in some cases, chromosomes can be created based on an initial solution (found by using any other heuristic algorithm). Next, for each chromosome, for each objective function, a fitness value is calculated. All non-dominated chromosomes found in the initial population are stored in the elitism set, using the `SaveNonDominated` procedure. The following is an iterative process, which is repeated *NumOfGen* times. The first step of the iterative process is the creation of a new population. If the size of the elitism set if bigger than *ElitisimSize*, the size of the new population is *PopSize* and the elitism set is added to the new population, otherwise it is *PopSize-ElitisimSize*. The new population is created using the `CreateNewPopulation` procedure. Again, for each chromosome, for each objective function, a fitness value is calculated. All non-dominated chromosomes are stored in the elitism set. If there is no change in the ideal solution of the previous solution and the current solution for the last 10 iteration (a parameter) *ProbMutation* is increased, the probability for mutation by 0.05 (a parameter), otherwise, *ProbMutation* is set it to its original value.

```
CREATENEWPOPULATION(POPULATION,POPSIZE,PROBMUTATION,PROBCROSSOVER)
1.  ObjSize= PopSize / NumOfObjectives
2.  ObjIndex = -1
3.  for i=1 to PopSize
4.     if i mod ObjSize = 0
5.        ObjIndex = ObjIndex + 1
6.     select Chromosome1 based on objective number
       ObjIndex
7.     select Chromosome2 based on objective number
       ObjIndex
8.     perform crossover operation with probability
       ProbCrossover
9.     perform mutation operations with probability
       ProbMutation
10.    add the new chromosomes to the new population
11. return new Population
```

The `CreateNewPopulation` procedure creates *PopSize* new chromosomes. Since multi objectives are optimized, a number of sub-populations, equal to the number of objectives, 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 *PopSize*/q each would be generated.

```
SAVENONDOMINATED(POPULATION,ELITISM)
1.  Elitism=∅
2.  For each Chromosome in Population do
3.     IsDominated=false
4.     For each Chromosome in Elitism do
5.        if Population's Chromosome is dominated by
          Elitism's Chromosome or Chromosomes are the
          same
6.           IsDominated=true
7.     If IsDominated=false
8.        For each Chromosome in Elitism
9.           If Elitism's Chromosome is dominated by
             Population's Chromosomes
10.             Remove Elitism's Chromosome from
                Elitism
11. Add Population's Chromosome to Elitism
```

Adding a chromosome to the set of elitism chromosome is a two steps process. First the chromosome is checked whether it is dominated by any of the chromosomes in the elitism set or not. If the chromosome is a non-dominated chromosome, it is added to the elitism set. Next, the new chromosome is checked whether it dominates any other chromosome in the elitism set. Such dominated chromosomes are removed from the elitism set.

B. Algorithm's Performance

An important part of any genetic algorithm, whether it solves a single objective problem or a multi objectives problem, is the proper selection of chromosomes from which the next generation is created. In the literature, several methods for the selection process can be found, all of them relay on a ranking mechanism, known as fitness function, which can rank the chromosomes based on the objective function(s).

Three objective functions are addressed in this paper: (1) minimizing the total travel time; (2) minimize the number of vehicles in use and (3) minimize the difference of travel times among the routes of the solution. For this reason, three fitness functions have to be defined, each for each objective function. The first fitness function should return the total traveling time of a set of routes. The second fitness function should return the size of a set of routes (the number of routes in the set). The third fitness function should return the standard deviation of the traveling time of each route in a set of routes. Due to the stochastic nature of the problem, the first and last fitness functions have to use simulation in order to get accurate values.

Simulation works by traveling paths. Each path is traveled w time, when w is pre-determined by the user. The traveling times are stored in an array, and are sorted. The returned traveling time, C, returned by the simulation is defined as the traveling time stored in entry $w \cdot \alpha$ of the array. Assuming that α=0.95, this means that in 95% of all cases, the actual traveling time will be shorter than C. A higher value of w will, usually, increase the accuracy of the result obtained from the simulation.

A high value of w usually results in accurate results of the simulation; however, it increases dramatically the running time of the algorithm. For example, in this paper, values of w=1 and w=1000 were used. Using the IVEGA algorithm, several problems where solved. The average running time when w=1 was about 20 minutes, and when w=1000, about 8 hours. Next it will be shown that w=1 can be used without affecting the algorithm performance (meaning that for different values of w the algorithm converges to the same results).

1) Converges

To validate this approach, a methodology normally adopted in the evolutionary multi-objective optimization literature was used.

The use of performance measures (or metrics) allows the assessment (in a quantitative way) of algorithm's performance. For multi objective optimization problems, measures tend to focus on the objective domain as to the accuracy of the results. For this comparative study, the two following metrics were implemented:

Two Set Coverage (SC): This metric was proposed by Zitzler, Deb and Thiele [55], and it can be termed *relative coverage comparison of two sets*. Consider X' and X'' as two competing sets of phenotype decision vectors. SC is defined as the mapping of the order pair (X', X'') to the interval $[0,1]$, which reflects the percentage of individuals in one set (X'') dominated by the individuals of the other set (X'). The mathematical definition of this metric is shown in equation (15):

$$SC(X', X'') = \frac{\left|a'' \in X''; \forall a' \in X' : a' \succ a''\right|}{\left|X''\right|} \qquad (15)$$

This definition implies that CS=1 when all points in X' dominate or equal to all points in X''. SC=0 implies the opposite. In general, $SC(X', X'')$ and $SC(X'', X')$ both have to be considered due to set intersections not being empty. Of course, this metric can be used for both spaces (objective function or decision variable space), but in this case it was applied to objective function space. It should be noted, that knowledge of the PF_{true} is not required for this metric. This important property is the main reason for choosing this metric.

Error Ratio (ER): This metric was proposed by Veldhuizen [57] to indicate the percentage of solutions in the known Pareto front, PF_{known}, that are not members of the true Pareto front, PF_{true}. In order to use this metric, it is essential that that the researcher knows the PF_{true}. The mathematical representation of this metric is shown in equation (16):

$$ER = \frac{\sum_{i=0}^{n} e_i}{n} \qquad (16)$$

where n is the number of vectors in PF_{known} and e_i is a 0 when the i vector is an element of PF_{true} or 1 if i is not an element. It should then be clear that ER=0 indicates an ideal behavior, meaning that the PF_{known} is the same as PF_{true}; but when ER=1 indicates that none of the points in PF_{known} are in PF_{true}.

However, since know PF_{true} is not known, a slightly different definition to the error ratio metric is presented. Given a set of non-dominated solutions, ND, (obtained from the last iteration of the genetic algorithm) and a known Pareto front, PF_{known}, the error ratio metric is defined as the percentage of vectors in ND, that are not members of PF_{known}. Using the formulation presented in (16), the new ER can be calculated, where n is the number of vectors in ND and e_i is a 0 when the i vector is an element of PF_{known} or 1 if i is not an element.

A genetic algorithm usually starts with a randomly generated first generation, however, if the first generation is smartly generated, for example, by using results obtained from an heuristic algorithm, then the genetic algorithm will converge to the optimal solution much faster, and the result, assuming that the same parameters, such as the number of generations, are kept, will be more accurate.

As stated before, the fitness evaluation procedure uses simulation, which works by traveling paths. Each path is traveled w time, when a high value of w usually results in accurate results of the simulation, and therefore, a more accurate fitness value is obtained. It will be shown that w=1 can be use without affecting the algorithm performance (meaning that for different values of w the algorithm converges to the same results). In order to show that, 30 test problems were randomly generated, 10 with 50 customers, another 10 with 100 customers, and the last 10 problems with 150 customers. In all test problems, the number of time intervals is 24, and in each time internal the speed is in the range of Each problem was solved 4 times, twice with w=1 ("approximated" fitness evaluation) and twice with w=1000 ("exact" fitness evaluation).

2) Metrics Comparison Results

In this section the results of the metrics comparison, using paired-samples t-tests are reported. Throughout this section X' refers to a solution obtained when w=1 and X'' to a solution obtained when w=1000. In genetic algorithm, since the way the first generation was generated may change the results of the algorithm, each comparison is done twice, once using a randomly generated first population and the second using a savings based first population.

Eight paired-samples t-tests were conducted to compare the results of the two set coverage metric ($CS(X',X'')$ vs. $CS(X'',X')$). The results are listed in TABLE I.

The results show that for problems with 50 and 100 customers, when the first generation was randomly generated or Savings based, there is no significant difference in the scores for $CS(X',X'')$ and $CS(X'',X')$. However, for problems with 150 customers, there is a significant difference in the scores for $CS(X',X'')$ and $CS(X'',X')$. This indicates that by average, 59% of the non-dominated solutions, when the first generation was randomly generated, and 29% of the non-dominated solutions, when the first generation was Savings based, obtained from the last iteration of the genetic algorithm, when w=1, are dominated by the non-dominated solutions obtained when w=1000. In addition, 26% of the non-dominated solutions, when the first generation was randomly generated, and 54% of the non-dominated solutions, when the first generation was savings based, obtained when w=1000, are dominated by the non-dominated solutions obtained when w=1. From the above, it can be concluded that for problems with 150, the results obtained when w=1000 are better than the results obtained when w=1 when using a randomly generated first generation. However, if the first generation is Savings based, then results obtained when w=1 are better than the results obtained when w=1000. Two paired-samples t-tests, in which all groups of problems are combined to a single sample, shows that there is no significant difference in the scores for $CS(X',X'')$ and $CS(X'',X')$.

TABLE I. A COMPARISON OF CS(X',X'') AND CS(X'',X') USING PARIED-SAMPLES T-TESTS

Size	$CS(X',X'')$		$CS(X'',X')$				
	M	*SD*	*M*	*SD*	*t*	*df*	*Sig.*
Randomly generated fist population							
50	0.479	0.411	0.493	0.459	-0.112	39	0.911
100	0.349	0.447	0.397	0.447	-0.382	39	0.705
150	0.59	0.458	0.258	0.411	2.654	39	0.011
All	0.411	0.424	0.416	0.411	-0.064	119	0.949
Savings based first population							
50	0.535	0.41	0.452	0.376	0.690	39	0.494
100	0.408	0.435	0.254	0.365	1.427	39	0.162
150	0.29	0.399	0.541	0.446	-2.199	39	0.034
All	0.473	0.447	0.389	0.446	1.235	119	0.219

As with the results of the two set coverage metric, paired-samples t-tests were used to check if there are any differences in the results of the error ratio metric for w=1 and for w=1000,. The results are listed in TABLE II.

The results show that for problems with 50, when the first generation was randomly generated or Savings based, there is no significant difference in the scores for $ER(X')$ and $ER(X'')$. This is also the case for problems with 100 customers, when the first generation was randomly generated and for problems with 150 customers, when the first generation is Savings based.

However, for problems with 150 customers, when the first generation was randomly generated and for problems with 100 customers, when the first generation is Savings based, there is a significant difference in the scores for $ER(X')$ and $ER(X'')$. This means that for problems with 150 customers, when the first generation was randomly generated, by average 55% of the non-dominated solutions do not belong to PF_{known} when w=1, while, when w=1000, 80% of the non-dominated solutions do not belong to PF_{known}. Similarly, for problems with 100 customers, when the first generation is Savings based, by average 49% of the non-dominated solutions do not belong to PF_{known} when w=1 while, when w=1000, 73% of the non-dominated solutions do not belong to PF_{known}.

This means, that for problems with 150 customers, when the first generation was randomly generated and for problems with 100 customers, when the first generation was Savings based, the chance for a non-dominated solution belongs to PF_{known} is twice higher when w=1 than when w=1000.

The results of the paired-samples t-tests, in which all groups of problems are combined to a single sample, show that there is no significant difference in the scores for $ER(X')$ and $ER(X'')$.

TABLE II. A COMPARISON OF ER(X') AND ER(X'') USING PARIED-SAMPLES T-TESTS

Size	$ER(X')$		$ER(X'')$				
	M	*SD*	*M*	*SD*	*T*	*df*	*Sig.*
Randomly generated fist population							
50	0.852	0.248	0.759	0.308	1.402	39	0.169
100	0.734	0.383	0.589	0.458	1.416	39	0.165
150	0.552	0.481	0.803	0.353	-2.535	39	0.015
All	0.712	0.399	0.717	0.389	-0.088	119	0.93
Savings based first population							
50	0.699	0.307	0.74	0.306	-0.532	39	0.598
100	0.496	0.404	0.731	0.342	-2.592	39	0.013
150	0.695	0.343	0.585	0.427	1.229	39	0.227
All	0.63	0.364	0.686	0.366	-1.092	119	0.277

From the results, it can be concluded that the results obtained from the genetic algorithm, whether using w=1 or w=1000 are the same, regardless of the problem size and the method for the generation of the first population.

3) TOPSIS Comparison

In most cases, when solving a multi-objective optimization problem, the result is a set of non-dominated solution, from which, the decision maker has to choose his preferred alternative. In an automated enviourment, a mechanism for choosing a preferred solution from a set of non-dominated solution has to be implemented. A number of techniques for automating the process of choosing have been developed. Among the various methods, one can find the Max-Min method, Min-Max method, Compromise Programming, ELECTRE Method and more [58]. In this paper, the TOPSIS method is used as a mean for choosing a preferred alternative.

In the previous section, it has been shown that a set of non-dominated solution obtained when w=1 is as good as a set of non-dominated solution obtained when w=1000. However, this does not mean that the same results exist in both sets, and therefore, it is not guaranteed that the TOPSIS method selects similar results from both sets. In this section, a comparison of the results of TOPSIS method applied on the solution sets obtained from the 30 test cases is presented.

A solution is a set of three results, each for every objective function. The analysis begins with correlation analysis. Correlation analysis is used to check whether there exists a correlation between the three values of a result or not. As with the metrics comparison, each comparison is done twice, once using a randomly generated first population and the second using a savings based first population.

Eight Pearson product-moment correlation coefficients were computed to assess the relationship between the results of the first objective function and the second objective function. The results are listed in TABLE III.

TABLE III. PEARSON PRODUCT-MOMENT CORRELATION COEFFICIENTS BETWEEN THE FIRST AND SECONDS OBJECTIVES, FOR W=1 AND W=1000

Problem Size	w=1		w=1000	
		Obj. 2		Obj. 2
Randomly generated fist population				
50	Obj. 1	r=0.94, n=40, p=0	Obj. 1	r=0.954, n=40, p=0
100	Obj. 1	r=0.973, n=40, p=0	Obj. 1	r=0.986, n=40, p=0
150	Obj. 1	r=0.909, n=40, p=0	Obj. 1	r=0.944, n=40, p=0
All	Obj. 1	r=0.81, n=120, p=0	Obj. 1	r=0.785, n=120, p=0
Savings based first population				
50	Obj. 1	r=0.835, n=40, p=0	Obj. 1	r=0.814, n=40, p=0
100	Obj. 1	r=0.82, n=40, p=0	Obj. 1	r=0.865, n=40, p=0
150	Obj. 1	r=0.281, n=40, p=0.079	Obj. 1	r=0.174, n=40, p=0.282
All	Obj. 1	r=0.716, n=120, p=0	Obj. 1	r=0.716, n=120, p=0

For problems with 50, 100 and 150 customers, whether the first generation was randomly created or Savings based, a positive correlation between the two variables was found for solutions obtained when w=1 and for solutions obtained when w=1000. Since a strong positive correlation, exist between the two variables, a correlation analysis, using the results of all problems as a single result was performed. The results show a positive correlation when using w=1 and when using w=1000.

A second set of eight Pearson product-moment correlation coefficients was computed to assess the relationship between the results of the first objective function and the third objective function. All tests, whether the first generation was randomly created or Savings based, show a negative correlation. However, since the third objective minimizes the difference of travel times among the routes of the solution, and is defined in means of standard deviation, and since the maximum value obtained for this objective is 0.05 when w=1 and 0.009 when w=1000, it can be assumed that the value of the third objective is always 0, and therefore, ignore it in the analysis.

Since it has been shown that a correlation exists between the first and second objectives, and that the third objective can be ignored, since it can be treated as zero, a paired t-test can be used to compare the results obtained by using the TOPSIS method.

Eight paired-samples t-tests were conducted to compare the results obtained by using the TOPSIS method when w=1 and when w=1000. The results are listed in TABLE IV.

All paired-samples t-tests show that there is no significant difference in the scores for w=1 and for w=1000.

TABLE IV. A COMPARISON OF TOPSIS RESULTS FOR W=1 AND W=1000 USING PARIED-SAMPLES T-TESTS

Problem Size	W=1		w=1000				
	M	SD	M	SD	t	df	Sig.
Randomly generated fist population							
50	31.2	10.6	30.9	10.2	0.429	39	0.67
100	72.1	28.8	70.9	26.3	0.899	39	0.374
150	76.3	15.8	79.1	14.2	-1.967	39	0.056
All	59.8	28.2	60.3	27.8	-0.676	119	0.5
Savings based first population							
50	24.9	12.3	24.8	11.9	0.02	39	0.984
100	52.7	31.7	51.9	30.6	0.244	39	0.808
150	42.2	27.0	40.6	26.4	0.34	39	0.736
All	39.9	27.4	39.1	26.6	0.417	119	0.678

C. Travel time characteristics

The previous analysis shows that it is possible to increase the running time of the algorithm by use an "approximated" fitness function, without influencing the accuracy of the algorithm. The analysis was done using 30 randomly generated test problems, with 50, 100 and 150 customers, all having 24 time intervals, when for each time interval the travel speed ranges from 80-120 $^{KM}/_{H}$, with empiric probability. However, travel time is more likely to be lognormally distributed because (1) the positive skew shape (i.e., right skewed) is more suitable for travel time description; that is, a higher probability exists for long

travel time than for short travel time, and (2) the range [0, ∞) of the distribution is more natural than a truncated normal distribution (because negative travel times are impossible) [59].

For that reason, a second set of tests was done, this time using Solomon's instances. Since Solomon's instances were designed for TWVRP, a simple modification had to be done. Solomon's instances provide the location of each customer, assuming that the travel speed is constant. Since this is not the case in this problem, time intervals were added (24 of them) and for each time interval a lognormal random travel time functions was assigned for which σ=0.03 and μ=4.1 (theses values may slightly change between time interval) and therefore the average traveling speed is 60 $^{KM}/_{H}$. In order to decrease running time, Solomon's instances were solved using the IVEGA algorithm, using 500 generations and population size of 200, once when *w*=1 and next when *w*=100. The results of the test are presented in TABLE V.

TABLE V. A COMPARISON OF TOPSIS RESULTS FOR *W*=1 AND *W*=100 USING PARIED-SAMPLES T-TESTS

		***w*=1**		***w*=100**				
Size	**Type**	**M**	**SD**	**M**	**SD**	**t**	**df**	**Sig.**
Randomly generated fist population								
25	C1	3.35	0.03	3.34	0.04	0.641	35	0.526
	C2	2.47	0.09	2.45	0.1	0.561	31	0.579
	R1	4.14	0.86	3.63	0.22	3.912	47	0
	R2	3.15	0.28	3.15	0.22	-0.059	43	0.953
	RC1	3.96	0.24	3.84	0.21	2.492	31	0.018
	RC2	2.57	0.13	2.59	0.15	-0.69	31	0.495
50	C1	6.6	0.22	6.61	0.44	-0.133	35	0.895
	C2	5.4	0.55	4.94	0.1	4.616	31	0
	R1	11.83	1.21	11.4	1.45	1.466	47	0.149
	R2	11.63	0.67	11.54	0.94	0.538	43	0.593
	RC1	9.46	0.69	9.27	0.78	0.931	31	0.395
	RC2	8.6	0.72	8.46	0.41	1.027	31	0.312
100	C1	16.44	1.01	16.35	0.94	0.467	35	0.643
	C2	14.4	1.00	13.85	0.94	2.292	31	0.029
	R1	38.94	2.38	37.43	1.85	3.462	47	0.001
	R2	35.74	3.01	32.37	2.71	5.224	43	0
	RC1	32.88	2.45	30.99	0.29	3.514	31	0.001
	RC2	28.41	3.29	24.76	1.96	5.604	31	0
Savings based first population								
25	C1	3.38	0.07	3.37	0.07	0.453	35	0.653
	C2	2.38	0.09	2.35	0.08	1.544	31	0.133
	R1	4.36	1.07	3.66	0.28	4.413	47	0
	R2	3.32	0.3	3.23	0.31	1.427	43	0.161
	RC1	4.11	0.26	4	0.19	1.971	31	0.058
	RC2	2.51	0.1	2.54	0.1	-1.275	31	0.212
50	C1	6.4	0.08	6.34	0.01	4.128	35	0
	C2	5.34	0.85	4.96	0.26	2.42	31	0.022
	R1	12.56	1.26	11.21	0.8	6.353	47	0
	R2	11.91	0.6	11.27	0.63	5.16	43	0
	RC1	9.17	0.63	8.89	0.22	2.412	31	0.022
	RC2	8.61	0.67	8.42	0.53	1.258	31	0.218
100	C1	14.92	0.63	14.53	0.08	4.253	35	0
	C2	12.33	0.57	11.45	0.54	6.486	31	0
	R1	38.88	2.42	37.81	1.78	2.466	47	0.017
	R2	35.76	2.19	32.16	2.27	7.837	43	0
	RC1	29.82	2.17	28.54	1.34	2.894	31	0.007
	RC2	28.21	4.07	24.22	1.74	4.836	31	0

As seen from the results, for problems with 25 and 50 customers, when using randomly generated first generation, and for problems with 25 customers when using a savings based first generation, these is no difference in the TOPSIS results when using *w*=1 and *w*=100. However, for problems with 100 customers, when using randomly generated first generation, and for problems with 50 and 100 customers when using a savings based first generation, a better solution is obtained when *w*=100 compared to the solution obtained when *w*=1.

As stated before, in order to decrease running time, Solomon's instances were solved using the IVEGA algorithm, using 500 generations and population size of 200. It is known that the number of generations used by a genetic algorithm may affect its results. Generally, a high number of generations gives the algorithm more chance to converge towards the optimal solution than a low number of generations. However, in real-time applications, the number of generations is bounded by the time given to the algorithm to come with a solution. For that reason, the algorithm was tested again, this time the stopping condition was 30 minutes of running time, instead of the 500 generations. Results are reported (TABLE VI) for problems with 100 customers.

TABLE VI. A COMPARISON OF TOPSIS RESULTS FOR *W*=1 AND *W*=100 USING PARIED-SAMPLES T-TESTS

		***w*=1**		***w*=100**				
Size	**Type**	**M**	**SD**	**M**	**SD**	**t**	**df**	**Sig.**
Randomly generated fist generation								
100	C1	14.85	0.67	16.43	0.81	-8.883	35	0
100	C2	12.64	1.58	14.19	0.74	-4.814	31	0
100	R1	33.67	2.08	39.61	1.69	-14.488	47	0
100	R2	32.77	4.57	38.62	2.94	-7.073	43	0
100	RC1	30.14	2.23	33.22	1.07	-7.905	31	0
100	RC2	27.13	4.42	28.14	2.49	-0.979	31	0.335
Savings based first generation								
100	C1	14.56	0.4	14.5	0.45	-0.411	35	0.684
100	C2	11.72	1.31	12.15	2.03	0.985	31	0.332
100	R1	34.44	2.56	40.09	1.53	-13.098	47	0
100	R2	31.25	2.79	39.58	2.31	-16.071	43	0
100	RC1	27.29	1.12	30.16	2.11	-6.568	31	0
100	RC2	24.92	4.21	26.27	1.4	-1.841	31	0.075

As can be seen from TABLE VI, when using a randomly generated first generation, the results obtained by the algorithm, when *w*=1 were better than the results obtained when *w*=100, except for RC2, in which no significant difference were found between the results. When using a savings based first generation, the results obtained by the algorithm, when *w*=1 were better than the results obtained when *w*=100, for problems R1, R2 and RC1, while for problems C1, C2 and RC2 no significant difference were found between the results.

More over, an analysis of the converges of the algorithm shows, that when *w*=100, the best solution is reached after almost 30 minutes of running, while the same solution is found much earlier, when *w*=1. To illustrate these finding, the analysis of problems C101, C201, R101, R201, RC101 and RC201 is given for both randomly generated and savings based first generation.

For problem C101, when *w=100* the algorithm, using TOPSIS, reached its best solution in which objective one

equals 16.52, objective two equals 10 and objective three equals 0, after 422 generations (see Figure 1). Since the algorithms, when *w*=100, was able to generate 486 generations in 30 minutes, this means that the algorithms best solution was reached after 26 minutes and 10 seconds. For the same problem, C101, when *w*=1, the algorithm reached the best solution after 326 generations out of 7916 generation that were generated during 30 minutes, meaning, after one minute and ten seconds. More over, when using *w*=1, the algorithm was able to reach a better solution than the best solution, in which objective one equals 19.95, objective two equals 10 and objective three equals 0, after 7394 generations, meaning after 28 minutes and one second.

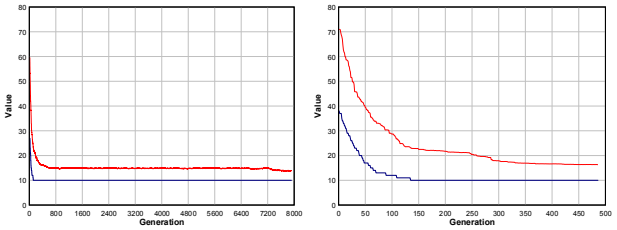

Figure 1. Algorithm convergence when *w*=1 (top) and *w*=100 (buttom) for problem C101 during the first 30 minutes

For problem C201, when *w=100* the algorithm, using TOPSIS, reached its best solution in which objective one equals 13.23, objective two equals 3 and objective three equals 0, after 422 generations (see Figure 2). Since the algorithms, when *w*=100, was able to generate 464 generations in 30 minutes, this means that the algorithms best solution was reached after 27 minutes and 17 seconds. For the same problem, C201, when *w*=1, the algorithm reached the best solution after 733 generations out of 7397 generation that were generated during 30 minutes, meaning, after one 9 minutes and ten seconds. More over, when using *w*=1, the algorithm was able to reach a better solution than the best solution, in which objective one equals 10.54, objective two equals 3 and objective three equals 0, after 1799 generations, meaning after 22 minutes and 30 seconds.

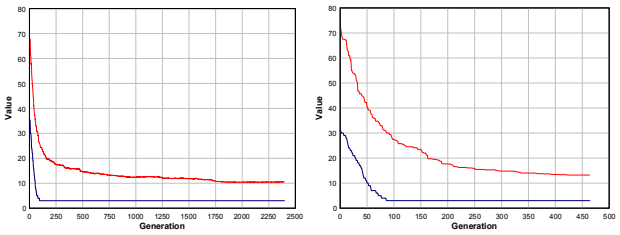

Figure 2. Algorithm convergence when *w*=1 (left) and *w*=100 (right) for problem C201 during the first 30 minutes

For problem R101, when *w=100* the algorithm, using TOPSIS, reached its best solution in which objective one equals 38.72, objective two equals 8 and objective three equals 0, after 345 generations (see Figure 3). Since the algorithms, when *w*=100, was able to generate 365 generations in 30 minutes, this means that the algorithms best solution was reached after 28 minutes and 21 seconds. For the same problem, R101, when *w*=1, the algorithm reached the best solution after 309 generations out of 1692 generation that were generated during 30 minutes, meaning, after 5 minutes and 28 seconds. More over, when using *w*=1, the algorithm was able to reach a better solution than the best solution, in which objective one equals 32.07, objective two equals 8 and objective three equals 0, after 1052 generations, meaning after 18 minutes and 39 seconds.

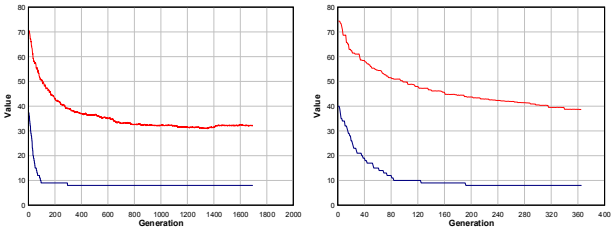

Figure 3. Algorithm convergence when *w*=1 (left) and *w*=100 (right) for problem R101 during the first 30 minutes

For problem R201, when *w=100* the algorithm, using TOPSIS, reached its best solution in which objective one equals 39.15, objective two equals 2 and objective three equals 0.00047, after 207 generations (see Figure 4). Since the algorithms, when *w*=100, was able to generate 215 generations in 30 minutes, this means that the algorithms best solution was reached after 28 minutes and 53 seconds. For the same problem, R201, when *w*=1, the algorithm reached the best solution after 154 generations out of 1246 generation that were generated during 30 minutes, meaning, after 3 minutes and 42 seconds. More over, when using *w*=1, the algorithm was able to reach a better solution than the best solution, in which objective one equals 32.91, objective two equals 2 and objective three equals 0.00008, after 785 generations, meaning after 18 minutes and 54 seconds.

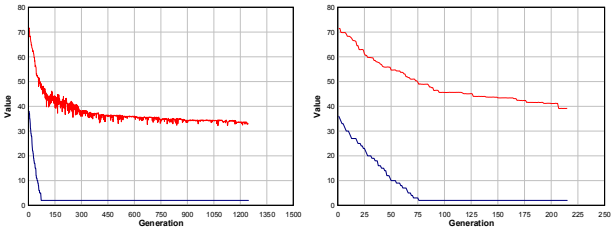

Figure 4. Algorithm convergence when *w*=1 (left) and *w*=100 (right) for problem R201 during the first 30 minutes

For problem RC101, when *w=100* the algorithm, using TOPSIS, reached its best solution in which objective one equals 30.53, objective two equals 10 and objective three equals 0, after 321 generations (see Figure 5). Since the algorithms, when *w*=100, was able to generate 327 generations in 30 minutes, this means that the algorithms best solution was reached after 29 minutes and 26 seconds. For the same problem, RC101, when *w*=1, the algorithm reached the best solution after 432 generations out of 806 generation that were generated during 30 minutes, meaning, after 16 minutes and 4 seconds. More over, when using *w*=1, the algorithm was able to reach a better solution than the best solution, in which objective one equals 28.52, objective two equals 10 and objective three equals 0, after 704 generations, meaning after 26 minutes and 12 seconds.

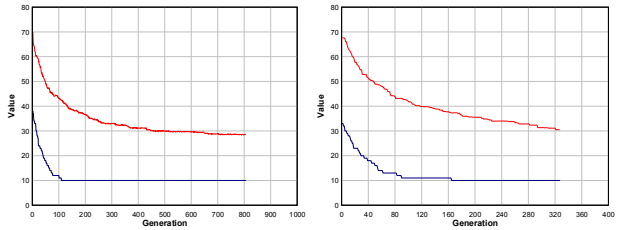

Figure 5. Algorithm convergence when *w*=1 (left) and *w*=100 (right) for problem RC101 during the first 30 minutes

For problem RC201, when *w=100* the algorithm, using TOPSIS, reached its best solution in which objective one equals 26.99, objective two equals 2 and objective three equals 0.00001, after 307 generations (see Figure 6). Since the algorithms, when *w*=100, was able to generate 323 generations in 30 minutes, this means that the algorithms best solution was reached after 28 minutes and 30 seconds. For the same problem, RC201, when *w*=1, the algorithm reached the best solution after 425 generations out of 979 generation that were generated during 30 minutes, meaning, after 13 minutes and one second. More over, when using *w*=1, the algorithm was able to reach a better solution than the best solution, in which objective one equals 24.22, objective two equals 2 and objective three equals 0, after 967 generations, meaning after 29 minutes and 37 seconds.

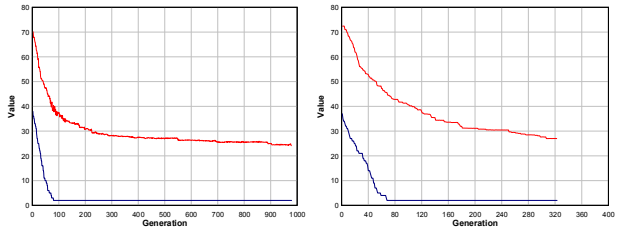

Figure 6. Algorithm convergence when *w*=1 (left) and *w*=100 (right) for problem RC201 during the first 30 minutes

For problem C101, when *w=100* the algorithm, using TOPSIS, reached its best solution in which objective one equals 13.92, objective two equals 10 and objective three equals 0, after one generations (see Figure 7). Since the algorithms, when *w*=100, was able to generate 513 generations in 30 minutes, this means that the algorithms best solution was reached after 0 minutes and 0 seconds. For the same problem, C101, when *w*=1, the algorithm reached the best solution after 1 generations out of 7747 generation that were generated during 30 minutes, meaning, after 0 minutes and 0 seconds.

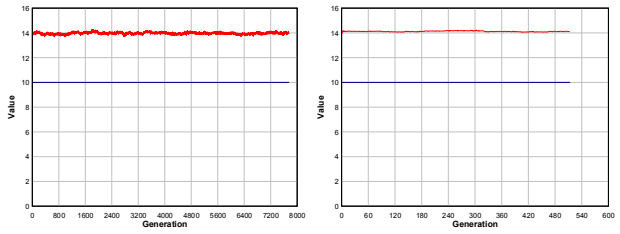

Figure 7. Algorithm convergence when *w*=1 (left) and *w*=100 (right) for problem C101 during the first 30 minutes

For problem C201, when *w=100* the algorithm, using TOPSIS, reached its best solution in which objective one equals 10.89, objective two equals 3 and objective three equals 0, after 494 generations (see Figure 8). Since the algorithms, when *w*=100, was able to generate 500 generations in 30 minutes, this means that the algorithms best solution was reached after 29 minutes and 38 seconds. For the same problem, C201, when *w*=1, the algorithm didn't reach the best solution after 2569 generations that were generated during 30 minutes.

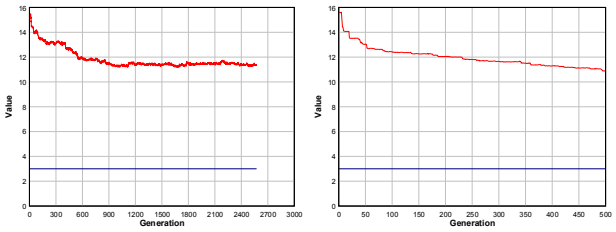

Figure 8. Algorithm convergence when *w*=1 (left) and *w*=100 (right) for problem C201 during the first 30 minutes

For problem R101, when *w=100* the algorithm, using TOPSIS, reached its best solution in which objective one equals 38.07, objective two equals 8 and objective three equals 0, after 371 generations (see Figure 9). Since the algorithms, when *w*=100, was able to generate 380 generations in 30 minutes, this means that the algorithms best solution was reached after 29 minutes and 17 seconds. For the same problem, R101, when *w*=1, the algorithm reached the best solution after 362 generations out of 1812 generation that were generated during 30 minutes, meaning, after 5 minutes and 59 seconds. More over, when using *w*=1, the algorithm was able to reach a better solution than the best solution, in which objective one equals 30.04, objective two equals 8 and objective three equals 0, after 1706 generations, meaning after 28 minutes and 14 seconds.

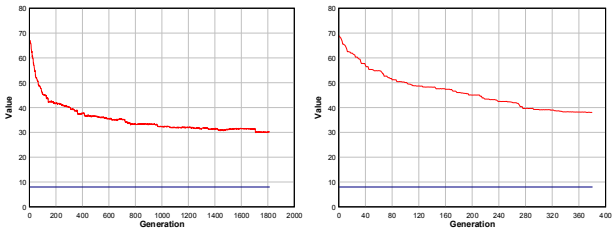

Figure 9. Algorithm convergence when *w*=1 (left) and *w*=100 (right) for problem R101 during the first 30 minutes

For problem R201, when *w=100* the algorithm, using TOPSIS, reached its best solution in which objective one equals 37.9, objective two equals 2 and objective three equals 0.00007, after 175 generations (see Figure 10). Since the algorithms, when *w*=100, was able to generate 177 generations in 30 minutes, this means that the algorithms best solution was reached after 29 minutes and 14 seconds. For the same problem, R201, when *w*=1, the algorithm reached the best solution after 162 generations out of 1308 generation that were generated during 30 minutes, meaning, after 3 minutes and 51 seconds. More over, when using *w*=1, the algorithm was able to reach a better solution than the best solution, in which objective one equals 33.61, objective two equals 2 and objective three equals 0.00009, after 874 generations, meaning after 20 minutes and two seconds.

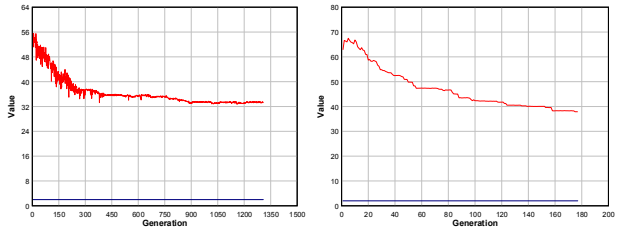

Figure 10. Algorithm convergence when w=1 (left) and w=100 (right) for problem R201 during the first 30 minutes

For problem RC101, when *w=100* the algorithm, using TOPSIS, reached its best solution in which objective one equals 28.47, objective two equals 9 and objective three equals 0, after 358 generations (see Figure 11). Since the algorithms, when w=100, was able to generate 362 generations in 30 minutes, this means that the algorithms best solution was reached after 29 minutes and 40 seconds. For the same problem, RC201, when w=1, the algorithm reached the best solution after 798 generations out of 1164 generation that were generated during 30 minutes, meaning, after 20 minutes and 34 seconds. More over, when using w=1, the algorithm was able to reach a better solution than the best solution, in which objective one equals 25.93, objective two equals 9 and objective three equals 0, after 1099 generations, meaning after 28 minutes and 19 seconds.

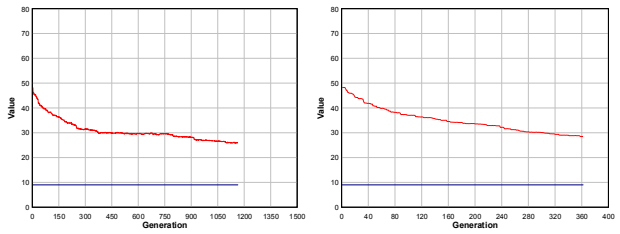

Figure 11. Algorithm convergence when w=1 (left) and w=100 (right) for problem RC101 during the first 30 minutes

For problem RC201, when *w=100* the algorithm, using TOPSIS, reached its best solution in which objective one equals 26.95, objective two equals 2 and objective three equals 0.00002, after 343 generations (see Figure 12). Since the algorithms, when w=100, was able to generate 363 generations in 30 minutes, this means that the algorithms best solution was reached after 28 minutes and 20 seconds. For the same problem, RC201, when w=1, the algorithm reached the best solution after 314 generations out of 1057 generation that were generated during 30 minutes, meaning, after 8 minutes and 54 seconds. More over, when using w=1, the algorithm was able to reach a better solution than the best solution, in which objective one equals 21.69, objective two equals 2 and objective three equals 0, after 930 generations, meaning after 26 minutes and 23 seconds.

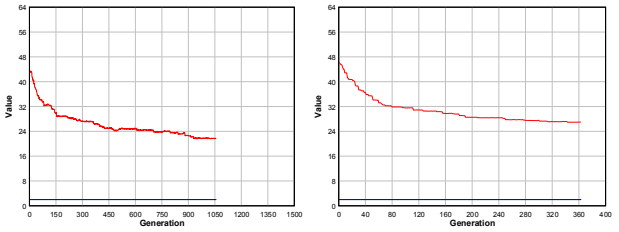

Figure 12. Algorithm convergence when w=1 (left) and w=100 (right) for problem RC201 during the first 30 minutes

V. CONCLUSIONS

This paper presented the multi-objective stochastic time-dependent vehicle routing problem. An improved version of the VEGA genetic algorithm was also presented. The main problem faced was the fitness functions, which, in order to be accurate, uses simulation, a time consuming operation.

It was shown that it is possible to increase the running time of the algorithm by use an "approximated" fitness function, without influencing the accuracy of the algorithm. A fast algorithm is necessary when coping with real-time problems, which is the final goal.

Two metrics, two set coverage and error ration, were used to compare results obtained from 30 test cases when using an "accurate" fitness function (meaning simulation) and "approximated" fitness function. The results show that there is no difference in the quality of the results obtained using the two methods.

Usually, when solving a multi-objective optimization problem, the result is a set of non-dominated solution, from which, the decision maker has to choose his preferred alternative. Since the final goal is to create an automated algorithm for solving a real-time multi-objective vehicle routing problem, the TOPSIS method, a mechanism for choosing a preferred solution from a set of non-dominated solution has been implemented. It was shown that there is no difference in the quality of the results obtained using the "approximated" or "accurate" methods, however, this does not mean that the same results exist in both sets, and therefore it is not guarantied that the TOPSIS method selects similar results from both sets. It was shown, by means of correlation testing and paired-samples t-tests, that the solutions selected by the TOPSIS methods are similar regardless of the method used for calculating the fitness functions.

Since travel time is more likely to be lognormally distributed a second set of tests was done, using Solomon's instances. Using 500 generation and a population of 200 chromosomes, the result of the IVEGA algorithm showed that for problems with large number of chromosomes (50 and 100 customers) using w=100 results with a better solution the when using w=1, while for problems with small number of customers (25 and 50) no significant difference was found. Since it is known that the number of generations used by a genetic algorithm may affect its results, and since in real-time applications, the number of generations is bounded by the time given to the algorithm to come with a solution. The algorithm was tested again, this time the stopping condition was 30 minutes of running time, instead of the 500 generations. This time the result

showed that in all cases, the result obtained by the algorithm when w=1 are better then the results obtained when w=100. Moreover, when w=1, the algorithm converges to the best solution much faster than when w=100.

A future research should test more cases, in order to broaden the analysis. More metrics can be used for the comparison of the results obtained by using the different methods for calculating the fitness functions.

Furthermore, it is possible to add more objectives to the problem, and check whether the results remain the same, or whether the results shown in the paper are specific to the selected objectives.

REFERENCES

[1] G. B. Dantzig and J. H. Ramser, "The Truck Dispatching Problem," Management Science, vol. 6(1), pp. 80-91, 1959.

[2] R. H. Ballou and Y. K. Agarwal, "A Performance Comparison of Several Popular Algorithms for Vehicle Routing and Scheduling," Journal of Business Logistics, vol. 9, pp. 51-65, 1988.

[3] J. F. Cordeau, M. Gendreau, G. Laporte, J. Y. Potvin, and F. Semet, "A Guide to Vehicle Routing Heuristics," Journal of the Operational Research Society, vol. 53, pp. 512-522, 2002.

[4] P. Toth and D. Vigo, "The Vehicle Routing Problem," Philadelphia: Siam, 2001.

[5] P. Ji and Y. Wu, "An Improved Artificial Bee Colony Algorithm for the Capacitated Vehicle Routing Problem with Time-Dependent Travel Times," 2011.

[6] C. Malandraki, "Time Dependent Vehicle Routing Problems: Formulations, Solution Algorithms and Computational Experiments", Northwestern University. 1989.

[7] C. Malandraki and M. S. Daskin, "Time Dependent Vehicle Routing Problems: Formulations, Properties and Heuristic Algorithms," Transportation Science, vol. 26(3), pp. 185-200, 1992.

[8] A. V. Hill and W. C. Benton, "Modelling intra-city time-dependent travel speeds for vehicle scheduling problems," The journal of the Operational Research Society, vol. 43(4), pp. 343-351, 1992.

[9] B. H. Ahn and J. Y. Shin, "Vehicle-routeing with time windows and time-varying congestion," The journal of the Operational Research Society, vol. 42(5), pp. 393-400, 1991.

[10] S. Ichoua, M. Gendreau, and J. Y. Potvin, "Vehicle Routing with Time-Dependent Travel Times," European Journal of Operational Research, vol. 144, pp. 379-396, 2003.

[11] C. Malandraki and R. B. Dial, "A restricted dynamic programming heuristic algorithm for the time dependent traveling salesman problem," European Journal of Operational Research, vol. 90(1), pp. 45-55, 1996.

[12] B. Fleischmann, M. Gietz, and S. Gnutzmann, "Time-varying travel times in vehicle routing," Transportation Science, vol. 38(2), p. 160, 2004.

[13] H. Hashimoto, M. Yagiura, and T. Ibaraki, "An iterated local search algorithm for the time-dependent vehicle routing problem with time windows," Discrete Optimization, vol. 5(2), pp. 434-456, 2008.

[14] S. Jung and A. Haghani, "Genetic algorithm for the time-dependent vehicle routing problem," Transportation Research Record: Journal of the Transportation Research Board, vol. 1771(-1), pp. 164-171, 2001.

[15] A. Haghani and S. Jung, "A dynamic vehicle routing problem with time-dependent travel times," Computers and Operations Research, vol. 32(11), pp. 2959-2986, 2005.

[16] T. Van Woensel, L. Kerbache, H. Peremans, and N. Vandaele, "Vehicle routing with dynamic travel times: A queueing approach," European Journal of Operational Research, vol. 186(3), pp. 990-1007, 2008.

[17] A. V. Donati, R. Montemanni, N. Casagrande, A. E. Rizzoli, and L. M. Gambardella, "Time dependent vehicle routing problem with a multi ant colony system," European Journal of Operational Research, vol. 185(3), pp. 1174-1191, 2008.

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

[19] M. Gendreau, G. Laporte, and R. Seguin, "Stochastic Vehicle Routing," European Journal of Operational Research, vol. 88, pp. 3-12, 1996.

[20] Z. Shen, F. Ordónez, and M. Dessouky, "The minimum unmet demand stochastic vehicle routing problem", Working paper 2006-07, University of Southern California. 2006.

[21] N. Secomandi and F. Margot, "Reoptimization approaches for the vehicle-routing problem with stochastic demands," Operations Research, vol. 57(1), pp. 214-230, 2009.

[22] X. Li, P. Tian, and S. C. H. Leung, "Vehicle routing problems with time windows and stochastic travel and service times: Models and algorithm," International Journal of Production Economics, vol. 125(1), pp. 137-145, 2010.

[23] F. A. Tillman, "The Multiple Terminal Delivery Problem with Probabilistic Demands," Transportation Science, vol. 3(3), pp. 192-204, 1969.

[24] W. R. Stewart, "Stochastic vehicle routing: A comprehensive approach," European Journal of Operational Research, vol. 14(4), pp. 371-385, 1983.

[25] D. J. Bertsimas, "A Vehicle Routing Problem With Stochastic Demand," Operations Research, vol. 40(3), pp. 574-585, 1992.

[26] M. Gendreau, G. Laporte, and R. Seguin, "An exact algorithm for the vehicle routing problem with stochastic demands and customers," Transportation Science, vol. 29(2), p. 143, 1995.

[27] G. Laporte, F. V. Louveaux, and L. Van Hamme, "An integer L-shaped algorithm for the capacitated vehicle routing problem with stochastic demands," Operations Research, vol., pp. 415-423, 2002.

[28] W. Rei, M. Gendreau, and P. Soriano, "Local branching cuts for the 0-1 integer L-Shaped algorithm", Technical Report CIRRELT-2007. 2007.

[29] G. Laporte, F. V. Louveaux, and H. Mercure, "The Vehicle Routing Problem With Stochastic Travel Times," Transportation Science, vol. 26(3), pp. 161-170, 1992.

[30] V. Lambert, G. Laporte, and F. Louveaux, "Designing collection routes through bank branches," Computers and Operations Research, vol. 20(7), pp. 783-791, 1993.

[31] X. Wang and A. C. Regan, "Assignment models for local truckload trucking problems with stochastic service times and time window constraints," Transportation Research Record, vol. 1171, pp. 61-68, 2001.

[32] A. S. Kenyon and D. P. Morton, "Stochastic Vehicle Routing with Random Travel Times," Transportation Science, vol. 37(1), pp. 69-82, 2003.

[33] Y. B. Park and C. P. Koelling, "A solution of vehicle routing problems in multiple objective environment," Engineering Costs and Production Economics, vol. 10, pp. 121-132, 1986.

[34] Y. B. Park and C. P. Koelling, "An interactive computerized algorithm for multicriteria vehicle routing problems," Computers and Industrial Engineering, vol. 16(4), pp. 477-490, 1989.

[35] N. Jozefowiez, F. Semet, and E. G. Talbi, "Multi-objective vehicle routing problems," European Journal of Operational Research, vol. 189(2), pp. 293-309, 2008.

[36] R. Gupta, B. Singh, and D. Pandey, "Multi-Objective Fuzzy Vehicle Routing Problem: A Case Study," Int. J. Contemp. Math. Sciences, vol. 5(29), pp. 1439-1454, 2010.

[37] M. Wen, J. F. Cordeau, G. Laporte, and J. Larsen, "The dynamic multi-period vehicle routing problem," Computers & Operations Research, vol. 37(9), pp. 1615-1623, 2010.

[38] M. Faccio, A. Persona, and G. Zanin, "Waste collection multi objective model with real time traceability data," Waste Management, vol. 31(12), pp. 2391-405, 2011.

[39] S. P. Anbuudayasankar, K. Ganesh, S. C. Lenny Koh, and Y. Ducq, "Modified savings heuristics and genetic algorithm for bi-objective vehicle routing problem with forced backhauls," Expert Systems With Applications, vol., 2011.

[40] M. Rahoual, B. Kitoun, M. H. Mabed, V. Bachelet, and F. Benameur. "Multicriteria genetic algorithms for the vehicle routing problem with time windows," 2001.

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

[42] N. El-Sherbeny, "Resolution of a vehicle routing problem with multi-objective simulated annealing method", Ph. D. Dissertation. Faculte Polytechnique de Mons. 2001.

[43] B. Baran and M. Schaerer. "A multiobjective ant colony system for vehicle routing problem with time windows," 2003.

[44] M. J. Geiger, "Genetic algorithms for multiple objective vehicle routing," Arxiv preprint arXiv:0809.0416, vol., 2008.

[45] N. Jozefowiez, F. Semet, and E. G. Talbi, "An evolutionary algorithm for the vehicle routing problem with route balancing," European Journal of Operational Research, vol. 195(3), pp. 761-769, 2009.

[46] D. M. Chitty and M. L. Hernandez, "A hybrid ant colony optimisation technique for dynamic vehicle routing," Lecture Notes In Compter Science, vol., pp. 48-59, 2004.

[47] T. Murata and R. Itai, "Local search in two-fold EMO algorithm to enhance solution similarity for multi-objective vehicle routing problems," Lecture Notes In Compter Science, vol. 4403, p. 201, 2007.

[48] T. Murata and R. Itai. "*Multi-objective vehicle routing problems using two-fold EMO algorithms to enhance solution similarity on non-dominated solutions*," Springer, 2005.

[49] K. G. Zografos and K. N. Androutsopoulos, "A heuristic algorithm for solving hazardous materials distribution problems," European Journal of Operational Research, vol. 152(2), pp. 507-519, 2004.

[50] K. C. Tan, Y. H. Chew, and L. H. Lee, "A hybrid multi-objective evolutionary algorithm for solving truck and trailer vehicle routing problems," European Journal of Operational Research, vol. 172(3), pp. 855-885, 2006.

[51] J. D. Schaffer. "Multi-Objective Optimization with Vector Evaluated Genetic Algorithms." in 1st International Conference on Genetic Algorithms. 1985.

[52] J. T. Richardson, M. R. Palmer, G. E. Liepins, and M. R. Hilliard. "Some guidelines for genetic algorithms with penalty functions," Morgan Kaufmann Publishers Inc, 1989.

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

[54] K. Deb, et al., "A fast and elitist multi-objective genetic algorithm: NSGA-II," IEEE transactions on evolutionary computation, vol. 6(2), 2002.

[55] E. Zitzler, K. Deb, and L. Thiele, "Comparison of multiobjective evolutionary algorithms: Empirical results," Evolutionary Computation, vol. 8(2), p. 195, 2000.

[56] F. B. Pereira, J. Tavares, P. Machado, and E. Costa, "GVR: A New Genetic Representation for the Vehicle Routing Problem," Lecture Notes in Computer Science, vol. 2464, pp. 95-102, 2002.

[57] D. Veldhuizen, "Multi-objective evolutionary algorithms: Classifications, analyses, and new innovation", in Department of Electrical Engineering and Computer Engineering, Air-force Institute of Technology, Ohio. 1999.

[58] A. S. M. Masud and A. R. Ravindran, "Multiple Criteria Decision Making," in Operations Research and management Science Handbook, A.R. Ravindran, Editor, Taylor & Francis Group, LLC, 2008.

[59] Y. Hadas and A. A. Ceder, "Improving Bus Passenger Transfers on Road Segments Through Online Operational Tactics," Transportation Research Record: Journal of the Transportation Research Board, vol. 2072(-1), pp. 101-109, 2008.