Paper for the 89th TRB Annual Meeting, January 10-14, 2010,
Washington, DC

A Comparison of Two Algorithms for the Stochastic Time-Dependent Vehicle-Routing Problem

Submission date:
Word count: 5569 + 250 x 4 figures = 6569

Oren E. Nahum
Tel: +972-772004702, Fax: +972-772004702, e-mail: oren.nahum@live.biu.ac.il
The Interdisciplinary Department of Social Sciences, Faculty of Social Sciences,
Bar-Ilan University, Ramat Gan 52900, Israel

Yuval Hadas*
Tel: +972-49590565, Fax: +972-49597117, e-mail: hadasy@mail.biu.ac.il
The Interdisciplinary Department of Social Sciences, Faculty of Social Sciences,
Bar-Ilan University, Ramat Gan 52900, Israel

* Corresponding author

ABSTRACT

Vehicle-routing problems (VRP) 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 possible for a fleet of vehicles. This paper combines two important variants, the stochastic VRP and the time-dependent VRP, to form and define the Stochastic Time-Dependent VRP. Two algorithms for solving the stochastic time-depended VRP are compared, an efficient heuristic that is a new variant of the well-known saving algorithm, and a genetic algorithm. Both algorithms incorporate simulation that enables an estimate of each route's probability of being the quickest. The two algorithms yield results that are at most 10% off the optimal solutions (when possible to compare), yet are very different in their running times. Such results are similar to the performance of the saving algorithm when compared to the capacitated vehicle-routing problem.

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. The

objective function for the vehicle-routing problem is to minimize costs by finding optimal routes, which are usually the shortest routes. The classic VRP (also known as Capacitated Vehicle Routing Problem – CVRP) is defined on a graph $G=(V,E)$, where $V=\{v_0,v_1,\ldots,v_n\}$ is a set of vertices and $E=\{(v_i,v_j): i\neq j, v_i,v_j\in V\}$ is a set of edges. Vertex v_0 represents a depot, and the other vertex represents customers. A cost function, C_{ij}, is associated with each edge of E. Each customer has a non-negative demand, d_i. A fleet of m identical vehicles of capacity Q are based at the depot. VRP consists of designing a set of, at most, m delivery or collection routes, such that (1) each route starts and ends at the depot, (2) each customer is called at exactly once and by only one vehicle, (3) the total demand on each route does not exceed Q, and (4) the total routing cost is minimized. For the CVRP, the cost function, C_{ij}, represents edge distances, and therefore the optimal solution is a set of routes with the shortest length.

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

Two extensions are of interest to us, the stochastic VRP and the time-dependent VRP, which will be described in the following section.

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, we may obtain solutions that are far from optimal. The Time-Dependent VRP (TDVRP) was developed in order to avoid just such a mistake. 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 time-dependent vehicle routing compared to other VRP models [5].

The time dependent VRP was first formulated by Malandraki and Daskin ([6, 7]) using a mixed integer linear programming formulation. Malandraki and Daskin treated travel time as a function of both distance and the time of the day resulted in a piecewise constant distribution of the travel time. Although they only incorporated the temporal component of traffic-density variability, they acknowledged its importance. They developed two algorithms for solving the time-dependent vehicle-routing problem. The first algorithm was a greedy nearest-neighbor algorithm (three variants of the algorithm were introduced), and the second was a branch and bound-based algorithm that provided better solutions, but was suitable only for small problems.

Hill and Benton ([8]) considered a time dependent VRP (without time windows) and proposed a model based on time dependent traveling speeds that alleviates both the data collection and data storage problems inherent in time-dependent travel speed vehicle scheduling models. They also discussed the issue of developing algorithms to find near-optimal vehicle schedules with time-dependent travel speeds .Computational results for one vehicle and five customers were reported. Ahn and Shin ([9]) discussed modifications to the savings, insertion, and local improvement algorithms to better deal with TDVRP. In randomly generated instances, they reported computation time reductions as a percentage of "unmodified" savings, insertion, and local improvement algorithms. Malandraki and Dial ([10]) proposed a "restricted" dynamic programming algorithm for the time dependent traveling salesman problem, i.e. for a fleet of just one vehicle. A nearest-neighbor type heuristic was used to solve randomly generated problems. Although it is argued that many different types of travel time functions can be handled by this algorithm, results are only reported for step functions.

An important property for time dependent problems is the First In - First Out (FIFO) principal ([5, 9]). 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. This is an intuitive and desirable property though it is not present in all models. Earlier formulations and solutions methods ([6-8, 10]) do not guarantee the FIFO property.

Ichoua et al. ([5]) 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 tabu-search meta-heuristics, provided better solutions for most test scenarios.

Fleischmann et al. ([11]) utilized route construction methods already proposed in the literature, savings and insertion, to solve uncapacitated time dependent VRP with and without time windows. Fleischmann et al. 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 et al. tested their algorithms in instances created from Berlin travel time data. Jung and Haghani ([12, 13]) 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 et al. ([14]) 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. ([15]) proposed an

algorithm based on 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.

Stochastic VRP

A stochastic vehicle-routing problem arises when at least one problem variable is random [16]. A stochastic model is usually modeled in two stages [17]. In the first stage, a planned a-priori route is determined, followed by a realization of the random variables. In the second stage, corrective action, based on actual information, is applied to the solution of the first stage.

VRP with stochastic travel time is frequently encountered in pickup and delivery problems such as those arising in truckload operations. Wang and Regan ([18]) have proposed models for this class of problems under the presence of time windows.

Tillman [19] suggested a solution based on the saving algorithm for vehicle-routing problems with stochastic demands when there are a number of depots. Both Stewart and Golden [20] and Golden and Yee [21] presented a saving based on the CCP (constraints chance programming) model for the vehicle-routing problem with stochastic demands. Bertsimas [22] offered a number of algorithms for the solution of the vehicle-routing problem with stochastic customers.

In VRP with Stochastic Travel Times (VRPSTT) travel times on the edges and service times at the vertices are random variables. 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 ([23]). 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 [24].

Stochastic travel times were introduced into the vehicle-routing problem by Laporte, Louveaux, and Mercure [23], 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 a more recent study, Kenyon and Morton [25] 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.

THE STOCHASTIC TIME-DEPENDENT VEHICLE-ROUTING PROBLEM

The aim of this study was to develop a model for the Stochastic Time Dependent VRP (STDVRP). 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. Such complexity calls for the development of an efficient heuristic. This algorithm provides a set of routes that have the minimal total travel time, taking into consideration the following properties: (1) for certain routes, the travel time varies during the day; (2) travel time is stochastic.

Mathematical Formulation

VRP can be represented by a complete graph $G=(V,E)$, where $V=\{v_0,v_1,..,v_n\}$ is a set of nodes representing the depot (v_0) and the customers ($v_1,v_2,\ldots,v_n$), and $E=\{(v_i,v_j): i\neq j, v_i,v_j\in V\}$ is a set of directed edges. A fleet of R trucks ($\{r_1,r_2,\ldots r_R\}$) of capacity D is available. For each customer, a fixed non-negative demand d_i is given ($d_0=0$). A random cost function, $C_{ij}^{\ t}$, which denotes the cost (travel time) of traveling from customer i to customer j starting at time t, is also given, where t is the time interval index and T is the total number of time intervals. The aim is to find a set of routes with the shortest travel time such that for a given probability (α) it will not exceed C^*, and that C^* is minimal, in which the following constraints hold: (1) each route starts and ends at the depot, (2) every customer is called at exactly once by only one vehicle, (3) every vehicle route has a total demand not exceeding maximum vehicle capacity D.

This work assumes that the number of vehicles available is unlimited or equal to the number routes needed for an optimal solution under the given constraints.

Let $x_{ij}^{\ tr}$ donates a decision variable that is equal to 1 if vehicle R_r is assigned at time t to travel from customer i to customer j; otherwise, it is equal to 0. Since the cost function, $C_{ij}^{\ t}$, is stochastic, we can define the probability of having a traveling distance of C^* or less as $P\left(\sum_{i=0}^{n}\sum_{j=0}^{n}\sum_{t=0}^{T}\sum_{r=1}^{R} C_{ij}^{\ t} x_{ij}^{\ tr} < C^*\right)$.

It is now possible to define the formal stochastic time-dependent vehicle-routing problem The objective function is as follows:

$$\min \ Z = \sum_{i=0}^{n}\sum_{j=0}^{n}\sum_{t=0}^{T}\sum_{r=1}^{R} \bar{C}_{ij}^{\ t} x_{ij}^{\ tr} \tag{1}$$

under the following constraints:

$$x_{ii}^{\ tr} = 0 \quad \forall i \in \{0,1,...,n\}, r \in \{1,2,...,R\}, t \in \{0,1,...,T\} \tag{2}$$

$$\sum_{j=1}^{n}\sum_{t=0}^{T} {x_{0j}}^{tr} \le 1 \quad \forall r \in \{1,2,...,R\} \tag{3}$$

$$\sum_{i=1}^{n}\sum_{t=0}^{T} {x_{i0}}^{tr} \le 1 \quad \forall r \in \{1,2,...,R\} \tag{4}$$

$$\sum_{j=1}^{n}\sum_{t=0}^{T}\sum_{r=1}^{R} {x_{ij}}^{tr} = 1 \quad \forall i \in \{0,1,...,n\}, i \neq j \tag{5}$$

$$\sum_{i=1}^{n}\sum_{t=0}^{T}\sum_{r=1}^{R} {x_{ij}}^{tr} = 1 \quad \forall j \in \{0,1,...,n\}, j \neq i \tag{6}$$

$$\sum_{i=0}^{n}\sum_{t=0}^{T} {x_{ip}}^{tr} - \sum_{j=0}^{n}\sum_{t=0}^{T} {x_{pk}}^{tr} = 0 \quad \forall r \in \{1,2,...,R\}, \forall p \in \{0,1,...,n\} \tag{7}$$

$$\sum_{i=0}^{n} d_i \left(\sum_{j=0}^{n}\sum_{t=0}^{T} {x_{ij}}^{tr} \right) \le D \quad \forall r \in \{1,2,...,R\} \tag{8}$$

$$P\left(\sum_{i=0}^{n}\sum_{j=0}^{n}\sum_{t=0}^{T}\sum_{r=1}^{R} {C_{ij}}^{t} {x_{ij}}^{tr} < C^{*} \right) \ge \alpha \tag{9}$$

$${x_{ij}}^{tr} \in \{0,1\} \quad \forall i, j \in \{0,1,...,n\}, r \in \{1,2,...,R\}, t \in \{0,1,...,T\} \tag{10}$$

Objective function (1) is the total average travel time ($\overline{C}_{ij}^{\,t}$). Constraint (2) simply states that it is impossible to move from one customer to itself (in our graph, edges such as (*i*,*i*) do not exist). Constraints (3) and (4) state that no more than one vehicle leaves the depot and goes to each one of the customers, and no more than one vehicle returns from each one of the customers to the depot. Constraints (5) and (6) state that only one vehicle serves each one of the customers. Constraint (7) is added for route continuity. Constraint (8) states that the capacity of customers for each route does not exceed the maximum capacity of a single vehicle. Constraint (9) is a chance constraint, stating that we are looking for a set of routes whose travel time for a given probability (α) will not exceed C^*, and that C^* is minimal. This constraint makes the problem a stochastic rather than a deterministic problem. Constraint (10) states that the decision variables can accept values only of 0 or 1.

A Heuristic Algorithm for Solving STDVRP

In this chapter we present a heuristic algorithm, based on the saving algorithm [26], for solving STDVRP. The Savings heuristic is one of the best known and remains widely used in practice to this day, despite some of its shortcomings ([3]). This algorithm was designed for solving deterministic CVRP, and many heuristics are based on it [19, 21, 27]. It uses a simple, easy to understand and implement heuristic and therefore is commonly used. The saving algorithm yields fast, good results when compared with optimal solutions (although as of today, algorithms based on tabu-search are the best algorithms for solving VRP, for example the savings algorithm's average deviation from the best solution is 6.71% while the deviation of the Granular tabu search is 0.69% ([3])). For these reasons we choose the Savings algorithm as the base for our algorithm.

The presented heuristic algorithm, Savings-STDVRP, includes the following components in order to cope with stochastic and time-dependency: (1) transformation, (2) calculation, and (3) simulation. These components will be discussed next.

The STDVRP Algorithm

The STDVRP algorithm maintains two lists: (1) a solution list similar to the list used in the saving algorithm that is updated after each iteration; (2) a candidates list, which contains *m* routes that are picked according to their deterministic properties. The candidate list is then passed to a simulation that provides the route with the highest probability of being the quickest. This route is added to the solution list.

Following is a short description of the algorithm.

1. Create deterministic data based on the stochastic time-dependent data.
2. Algorithm initialization with the creation of an initial solution set. The initial solution set is a set of *n* routes, each of which starts at the depot, visits one customer, and returns to the depot. There are no two routes that call at the same customer.
3. Empty the candidates list.
4. For all pairs of routes do:
    a. Merge the two routes to create a new route (from the last customer of the first route continue to the first customer of the second route).
    b. If the new route does not violate the problem's constraints do:
        i. Calculate the savings value of the pair of routes using the deterministic data.
        ii. If the saving value is zero or higher, add the pair of routes to the candidates list. Eventually the candidates list contains *m* pair of routes that have the highest values of savings. The number of candidates, *m*, is defined by the user.
5. If the candidates list is empty the algorithm is finished, otherwise do:
    a. For each pair of routes stored in the candidates list, a simulation is performed *r* times in order to find the merged route with the highest probability of being the quickest.
    b. The merged route found in (a) is added to the solution list, and the original pair routes that were merged are removed from the list.
    c. Go to step 3
6. Goto step 1, and use a different method.

The algorithm is executed three times, each time using a different stochastic to deterministic encoder. The solution chosen is the best solution of the three runs.

Creating Deterministic Data

In order to cope with stochastic and time-dependent properties, the data is passed through "encoders", each of which produces different deterministic data. The deterministic data is used to estimate the savings value of each two pairs of paths. In each iteration, for *m* pairs

that have the highest estimated savings value, the true savings value is calculates using simulation, and the pair of paths that has the highest true saving value is merged to form a new path.

In our work the random cost function was defined as an empirical distribution function composed of a set of probability intervals. Such a definition is more flexible when estimating the travel time variance during a time period.

In this study, three "encoders" were used for transforming the stochastic and time-dependent data to deterministic data: (1) average value - the average time for each time period and probability intervals; (2) best value - the minimal time for all time periods, regardless of the probability; (3) worst value - the maximal time for all time periods, regardless of the probability.

Simulation

Use of the deterministic information of the problem results in estimates of only certain aspects of the original problem and does not describe the stochastic nature of the problem. Therefore, simulation [28] is used in order to calculate the implicit value of each saving of the paths merged and the probability of a route's being the quickest.

Simulation works by traveling paths, which is done in the following way: For every edge we have the edge's length, l_n, and a number of possible traveling speed, S_{np}^{t}, with their probabilities for each time interval. Assuming that our path is $(a_1,a_2,..,a_n)$, we start at time t_0=0 and calculate the traveling time from node a_1 to node a_2. This calculation is done by dividing the edge's length to a randomly picked possible speed from $S_{(a1,a2)p}^{0}$ base on its probability. If traveling an edge cannot be done in a single time interval, we use the information of both current and the following time intervals in order to calculate the traveling time of the edge. We continue and calculate the traveling time of the rest of the edges, each edge starting time equals to the sum of the traveling times of all previous edges. The traveling time of a path equals to the sum of traveling times of all edges of the path.

Each path is traveled w time, when w is 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 shorten than C.

It is possible to calculate the savings value of a given pair of routes by first finding the travel time of the first route, C_1, using simulation, then simulation is used again to find the traveling time, C_2, of the second route. The traveling time of the merged route, C_3, is also calculated. The saving value of the pair of routes is $C_1+C_2-C_3$. Again, assuming that α=0.95, this means that in 95% of all cases, the actual savings will be no higher than $C_1+C_2-C_3$.

Algorithm's Performance

The algorithm's performance should be analyzed in terms of (1) complexity and (2) accuracy of the algorithm's results compared to optimal solutions.

Complexity

Since VRPs belong to the NP-Hard set of problems, it is impossible to solve them within a reasonable amount of time. The Savings-STDVRP algorithm has a polynomial running time, which means that for problems involving a large number of customers, it is possible to arrive at a close to optimal solution in a reasonable running time.

Assuming that we use the simplest data structures, the algorithm's complexity can easily be calculated. The algorithm begins with creating an initial solutions set, which is an $\boldsymbol{O}(n)$ operation. Then an iterative process begins, the first step of which is the initialization of the candidates list, which is an $\boldsymbol{O}(1)$ operation. Next, every pair of routes that can be merged is added to the candidates list. The operation of adding a merged route to the candidates list has the complexity of $\boldsymbol{O}(m)$, where m is the size of the candidates list. Since we have at most (n-1) by (n-1) pair of routes that can be added to the candidates list, the total complexity of adding all pairs of routes that can be merged to that list is $\boldsymbol{O}(n^2m)$. The last operation of the iterative process is to search the candidates list for the pair of routes with the highest probability. For each pair of routes stored in the candidates list (m pairs of routes), the probability of its being the best is calculated by simulation. The simulation is carried out w times for each route (the two routes composing the pair and the merged route), making the total complexity equal to $\boldsymbol{O}(3wnm)$. The entire iterative process can be repeated, at most, n times and 3 times for each filter, making the total complexity of the entire algorithm equal to $\boldsymbol{O}(\mathrm{n})+3\mathrm{n}(\boldsymbol{O}(1)+\boldsymbol{O}(n^2m)+\boldsymbol{O}(3wnm))$, which is equal to $\boldsymbol{O}(3n^3m+9wn^2m+4n)$, usually referred to as $\boldsymbol{O}(n^3m)$.

From the algorithm complexity analysis we learn that the most influential factors on the algorithm's running time are the number of nodes (n) in the graph and the size of the candidates list (m). Experimental result (given in Table 1) show that an increase in the number of customers (n) does increase the running time of the algorithm, but not as expected, for example, from the algorithm complexity analysis we expect that the running time of a problem with 150 customers will be 27 times slower than a problem with 50 customers, in reality it is only 8.5 (candidates list set to 1) to 3 (candidates list set to 1) times slower. Also, an increase in the size of the candidates list (m) increases the running time of the algorithm. From the algorithm complexity analysis it is expected that the running time of a problem with candidates list set to 15 will be 15 times slower than a problem where the candidates list is set to 1. In reality it is only 8.72 (50 customers) to 3 times slower. These results can easily be explained by the fact that in the algorithm's analysis be ignored constraints satisfaction, which can dramatically decrease the number of iterations preformed by the algorithm.

Table 1 – Running time as a function of number of nodes and candidates list size

Number of Customers	*Problem No.*	*Candidates list size*									
		1		*3*		*5*		*10*		*15*	
		Running time	*Relative running time*	*Running time*	*Relative running time*	*Running time*	*Relative running time*	*Running time*	*Relative running time*	*Running time*	*Relative running time*
50	1	3.438	1	6.984	2.031	10.828	3.15	19.641	5.713	27.625	8.035
	2	3.406	1	8.031	2.358	11.953	3.509	23.437	6.881	32.5	9.542
	3	3.125	1	8.109	2.595	12.359	3.955	21.594	6.91	33.578	10.745
	4	2.422	1	4.656	1.922	6.609	2.729	11.938	4.929	16.234	6.703
	5	3.25	1	6.781	2.086	10.594	3.260	18.516	5.697	26.375	8.115
	Avrg.	3.128	1	6.912	2.21	10.469	3.347	19.025	6.082	27.262	8.716
75	6	4.578	1	7.078	1.546	9.687	2.116	15.844	3.461	22	4.806
	7	5.031	1	8.188	1.628	11.828	2.351	19.297	3.836	27.625	5.491
	8	3.766	1	5.516	1.465	7.125	1.892	11.063	2.938	14.672	3.896
	9	6.016	1	10.422	1.732	15.359	2.553	28.859	4.797	36.844	6.124
	10	5.781	1	10.875	1.881	16.078	2.781	27.406	4.741	38.75	6.703
	Avrg.	5.034	1	8.416	1.672	12.015	2.387	20.494	4.071	27.978	5.557
100	11	11.423	1	20.219	1.770	29.016	2.540	49	4.290	72.391	6.337
	12	8.438	1	11.797	1.398	14.938	1.770	23.656	2.804	31.875	3.778
	13	10.156	1	16.109	1.586	21.313	2.099	34.938	3.440	46.75	4.603
	14	7.203	1	9.906	1.375	12.406	1.722	17.313	2.404	23.922	3.321
	15	10.453	1	16.766	1.604	22.609	2.163	38.656	3.698	54.016	5.168
	Avrg.	9.535	1	14.959	1.569	20.056	2.104	32.713	3.431	45.791	4.803
150	16	28.484	1	40.594	1.425	51.313	1.801	79.438	2.789	105.844	3.716
	17	23.375	1	28.203	1.207	33.781	1.445	45.953	1.966	56.344	2.410
	18	28.641	1	39.313	1.373	48.969	1.710	74.031	2.585	96.234	3.360
	19	26.141	1	32.813	1.255	40.078	1.533	57.423	2.197	72.438	2.771
	20	26.203	1	33.391	1.274	39.813	1.519	56	2.137	72.438	2.764
	Avrg.	26.569	1.000	34.863	1.312	42.791	1.611	62.569	2.355	80.660	3.036

Accuracy

The algorithm's results were tested under a number of conditions:

1. Data is deterministic.
2. Data is stochastic, with the following factors:
   a. Influence of the percentage of edges acting stochastically.
   b. Influence of the range of travel times.
   c. Influence of the number of probability intervals on travel time for each edge in a time unit.
3. Data is time dependent and stochastic.

To our knowledge, no previous work involving stochastic and time dependency exists to which we could compare the results. Accordingly we created our own test scenarios, which were solved optimally (using brute-force approach). This made it possible to compare the Savings-STDVRP algorithm results to the optimal solution. Because of the complexity of finding an optimal solution, all our test scenarios included seven customers and a depot. All test scenarios used in this work are available online ([29]).

Table 2 summarizes the scenarios used to test the Savings-STDVRP algorithm results versus the optimal solutions and the original saving algorithm.

Table 2 - Scenarios (7 customers and a depote) and properties

Scenario Set	*Number of time periods*	*Number of probability intervals*	*Percentage of edges with stochastic properties*	*Speed range (Km/H)*

Scenario Set	*Number of time periods*	*Number of probability intervals*	*Percentage of edges with stochastic properties*	*Speed range (Km/H)*
1	2	1	0	80-120
2	6	1	0	80-120
3	12	1	0	80-120
4	24	1	0	80-120
5	1	3	25	50-120
6	1	3	25	80-120
7	1	5	25	50-120
8	1	5	25	80-120
9	1	3	50	50-120
10	1	3	50	80-120
11	1	5	50	50-120
12	1	5	50	80-120
13	1	3	100	50-120
14	1	3	100	80-120
15	1	5	100	50-120
16	1	5	100	80-120
17	1	100	100	50-120
18	24	5	100	50-120

Each scenario set contains 10 problems. Scenario sets 1-4 were designed to test the Savings-STDVRP algorithm when only a time dependency exists. Four groups of problems were created, with 2, 6, 12, and 24 time periods. Scenarios 5-17 were designed to test the Savings-STDVRP algorithm with only stochastic data. These test scenarios are constructed of 150 problems, which can be divided into three sub-groups: (1) problems that test the influence of a number of probability intervals on the algorithm's results; (2) problems that test the influence of the number of edges with stochastic properties on the algorithm's results; (3) problems that test the influence of the speed range on the algorithm's results. Scenario set 18 was designed to test the algorithm with both stochastic and time-dependent data.

The average results of the test problems deviating from the optimal solution are summarized in Table 3. In this table, "Range" represents the range of gap to optimality of the various scenarios in the scenario set. "Average", and "Standard Deviation" also refer to the gap to optimality of the various scenarios in the scenario set.

Table 3 - Results of the saving algorithm and the STDVRP algorithm as deviations from the optimal solution (all test scenarios are of 7 customers and a depot)

Scenario Set	*Saving Algorithm*			*STDVRP Algorithm*		
	Range	*Average*	*Standard Deviation*	*Range*	*Average*	*Standard Deviation*
The influence of time dependency						
1	0-22.2	7.7	6.3	0-15.4	5.9	5.1
2	0-15.3	5.1	5.4	0-15.3	4.3	5.7
3	0-26.4	9.7	9.2	0-23.8	7.8	8.0
4	0-21.1	11.3	8.7	0.5-18.7	11.4	6.4
The influence of a number of probability intervals						
5	4.6-20.6	10.7	5.3	0-9.2	2.6	3.4
7	7.2-40.2	21.7	11.1	0-15.6	6.0	5.0
6	0-17.1	7.2	5.4	0-6.6	1.9	2.5

Scenario Set	Saving Algorithm			STDVRP Algorithm		
	Range	Average	Standard Deviation	Range	Average	Standard Deviation
8	2-22.1	7.9	6.2	0-8.9	3.2	2.7
9	4.6-35.9	10.7	5.3	0-31.3	7.4	9.9
11	5.7-27.5	18.9	7.3	0-15.4	4.9	4.4
10	0.8-19.3	8.9	5.7	0-15.8	4.3	4.5
12	0.2-18.6	9.3	5.0	0-15.3	3.3	4.5
13	8.4-34.7	14.5	7.6	0-17.6	6.4	5.1
15	6.0-55.2	21.7	15.5	0-12.1	4.3	4.6
17	8-20.6	14.2	4.5	1.8-19.4	11.3	5.3
14	0-18.2	7.7	6.5	0-14.3	5	5
16	2.7-14.7	8	4.5	0-10	3.2	3.3
The influence of the speed range						
5	4.6-20.6	10.7	5.3	0-9.2	2.6	3.4
6	0-17.1	7.2	5.4	0-6.6	1.9	2.5
7	7.2-40.2	21.7	11.1	0-15.6	6.0	5.0
8	2-22.1	7.9	6.2	0-8.9	3.2	2.7
9	4.6-35.9	10.7	5.3	0-31.3	7.4	9.9
10	0.8-19.3	8.9	5.7	0-15.8	4.3	4.5
11	5.7-27.5	18.9	7.3	0-15.4	4.9	4.4
12	0.2-18.6	9.3	5.0	0-15.3	3.3	4.5
13	8.4-34.7	14.5	7.6	0-17.6	6.4	5.1
14	0-18.2	7.7	6.5	0-14.3	5	5
15	6.0-55.2	21.7	15.5	0-12.1	4.3	4.6
16	2.7-14.7	8	4.5	0-10	3.2	3.3
The influence of the percentage of edges with stochastic properties						
5	4.6-20.6	10.7	5.3	0-9.2	2.6	3.4
9	4.6-35.9	10.7	5.3	0-31.3	7.4	9.9
13	8.4-34.7	14.5	7.6	0-17.6	6.4	5.1
6	0-17.1	7.2	5.4	0-6.6	1.9	2.5
10	0.8-19.3	8.9	5.7	0-15.8	4.3	4.5
14	0-18.2	7.7	6.5	0-14.3	5	5
7	7.2-40.2	21.7	11.1	0-15.6	6.0	5.0
11	5.7-27.5	18.9	7.3	0-15.4	4.9	4.4
15	6.0-55.2	21.7	15.5	0-12.1	4.3	4.6
8	2-22.1	7.9	6.2	0-8.9	3.2	2.7
12	0.2-18.6	9.3	5.0	0-15.3	3.3	4.5
16	2.7-14.7	8	4.5	0-10	3.2	3.3

In addition, the impact of the candidates' list size was also tested, using 20 problems. Each problem was tested 5 times, each time with a different list size. The results are given in Table 4.

Table 4 - Problem results as a function of the number of customers and the number of path pairs stored in the candidates list

Number of Customers	Problem No.	Candidates list size				
		1	3	5	10	15
50	1	293.0	289.2	288.4	287.2	288.3
	2	286.3	273.2	270.9	270.4	269.2
	3	302.4	291.7	286.0	285.0	279.3
	4	329.6	323.1	324.4	325.6	326.2
	5	292.9	281.3	284.3	281.1	278.0

	6	516.3	507.3	509.3	502.3	505.5
	7	493.7	486.9	485.1	479.3	480.7
75	8	616.1	604.5	604.6	601.0	599.6
	9	445.0	442.5	443.2	443.4	442.4
	10	450.7	447.5	440.3	431.8	440.1
	11	565.6	560.7	557.1	548.3	541.0
	12	676.5	681.4	676.3	676.3	660.9
100	13	585.0	582.2	582.4	581.3	574.4
	14	803.5	782.7	772.8	774.0	774.8
	15	597.4	580.7	578.5	569.3	569.5
	16	851.8	846.4	829.0	832.5	833.4
	17	1058.6	1051.4	1058.2	1059.6	1056.9
150	18	876.1	850.4	848.8	842.2	841.1
	19	968.9	964.2	960.3	951.9	956.4
	20	957.5	952.3	935.1	942.2	934.5

Finally, since we compared our Savings-STDVRP algorithm's result to both the optimal solution and the Savings algorithm's results for problems with 7 customers, it was important to observe if the ratio between the results of the original savings algorithm and the results of the Savings-STDVRP algorithm remains the same as the number of customers increase. What was found is that for problems with 10 customers, the results of the Savings-STDVRP were 13% better than results of the savings algorithm. For 200 customers the results of the Savings-STDVRP were 2% better than the Saving algorithm.

Comparison to Other Algorithms

Genetic algorithms (GAs) have seen widespread use amongst modern meta-heuristics, and several applications to VRPs have been reported ([30-34]). Due to our interest in genetic algorithms, we implemented a simple genetic algorithm for solving STDVRP. The GA-STDVRP algorithm is based on the work of [35], who presented a genetic vehicle representation, designed to deal in an effective way with all the information that candidate solutions must encode. Experimental results show that this method is both effective and robust.

For a discussion about the genetic algorithm, chromosome representation and the crossover and mutations operations, see [35]. The fitness function, which is different from the one proposed by [35], due to the nature of the problem, will be discussed next.

Fitness function

Generally speaking, the fitness function should return the total traveling time of a set of routes. Due to the stochastic nature of the problem, and the time dependency, this can be done using simulation in the same manner that it was used in the *Savings STDVRP* algorithm. Since using simulation increases dramatically the running time, a different approach for calculating the fitness function is used. The fitness function can be calculated using averages or simulation. For each generation, the fitness function of each chromosome is calculated using averages values. This is done to all chromosomes except to the chromosome that has the best fitness value (using averages), that its fitness value is

calculated using simulation. Since using averages for calculating the fitness values results is a biased fitness values and may cause the algorithm to ignore relevant chromosomes, we start calculating the fitness values of random chromosomes if there is no improvement over 20 generations. The number of chromosomes that their fitness values are calculated by simulation increases over time. If there is no improvement over 1000 generations or more, all fitness values are calculated using simulation.

Algorithm Performance

We first examined our implementation using a group of 10 instances, each containing 200 customers, with no time-dependency and no stochastic data. Each instance was solved twice, once by using the Savings algorithm and one by using the genetic algorithm. Using the results of the savings algorithm as a seed for the first generation of the genetic algorithm and running the genetic algorithm for 5000 generations, we managed to receive results that are by average 5% better than the results of the savings algorithm.

To evaluate the *GA STDVRP* algorithm two sets of problems, each containing 10 problems, were randomly generated. The first set of problems (numbered 1 to 10) contained problems with 100 customers, and the second set (numbered 11 to 20) contained problems with 200 customers.

The settings of the *GA STDVRP* algorithm are: Number of generations: 5000 or 1050 generations without an improvement and CPU time is more than 3 hours (whatever comes first); Population size: 1000; Tournament selection with tourney size: 5; Elitist strategy (in each generation the 10 best chromosomes are automatically passed to the next generation); Crossover rate: 0.75; Mutation rates: swap: 0.05; inversion: 0.1; insertion: 0.05; displacement: 0.15. These settings are the same setting used by [35] (other settings as well as different elitism size and different selection methods were tested during the development stage, but no significant difference in the algorithm's result was found). For all test problems, 50% of the chromosomes of the initial populations were randomly generated and fixed, so it would match the stochastic time-dependent VRP's constraints. For the other 50% of the chromosomes, the results of the Savings algorithm were used as a seed, on which the mutation operation was used to create the new chromosomes.

Each problem was solved by *GA STDVRP* and by *Savings STDVRP* algorithm ([36]). In table 1 we present the result of each algorithm and each problem. For both algorithms we provide the running time. For the *GA STDVRP* we provide the best result and average result, and for the *Savings STDVRP* we provide the algorithm's result as well.

Table 5 – GA STDVRP and Savings STDVRP results

Problem No.		*GA STDVRP*		*Savings STDVRP*	
		(Best) Result (traveling time)	*Running Time (in seconds)*	*Result (traveling time)*	*Running Time (in seconds)*
Cu sto ‹	1	17.39	9,286.53	18.91	17.98
	2	83.10	10,121.44	85.73	8.75

Problem No.		GA STDVRP		Savings STDVRP	
		(Best) Result (traveling time)	*Running Time (in seconds)*	*Result (traveling time)*	*Running Time (in seconds)*
	3	16.39	9,390.44	17.09	18.80
	4	16.55	12,858.64	17.15	23.55
	5	16.08	6,520.59	17.67	20.83
	6	30.23	5,985.73	31.76	12.89
	7	26.51	6,099.67	27.18	13.13
	8	16.95	6,530.75	18.44	16.09
	9	26.51	18,374.08	27.23	60.41
	10	43.87	8,807.23	47.34	9.48
200 Customers	11	31.23	34,586.95	33.87	59.06
	12	27.54	16,481.63	27.96	63.20
	13	24.80	13,374.63	25.36	63.47
	14	27.98	25,951.44	28.68	113.20
	15	48.11	26,224.38	47.86	53.94
	16	37.97	11,595.56	38.55	51.95
	17	205.01	12,090.61	212.96	55.75
	18	38.54	19,906.13	39.06	96.28
	19	26.38	9,783.16	27.06	62.08
	20	62.01	10,771.47	67.20	110.06
Average		41.16	13,737.05	42.85	46.55

As we can see from the results presented in Table 5, the results of the GA-STDVRP algorithm are better than the result of the Savings-STDVRP algorithm. In average for problems with 100 customers, the results of the GA-STDVRP are by average 5% better than the results of the Savings-STDVRP. For problems with 200 customers, the results of GA-STDVRP are by average 3% better than the results of Savings-STDVRP.

Yet, the running time of GA-STDVRP is significantly longer than the running time of Savings-STDVRP. In average, for problems with 100 customers, the running time of GA-STDVRP is 560 times longer than the running time of Savings-STDVRP. For problems with 200 customers, the running time of *GA STDVRP* is 457 times longer than the running time of *Savings STDVRP*.

CONCLUSIONS

This paper presented the Stochastic Time-Dependent VRP. A saving-based algorithm was also presented. Various problems, each with seven customers, were tested, and it seems that the algorithm results for the stochastic time-dependent VRP are similar to the results of the saving algorithm for CVRPs. Because the saving algorithm is well known and popular, the STDVRP algorithm is believed to demonstrate similar properties concerning the stochastic time-dependent vehicle-routing problem.

From the results of these 20 problems, it seems that the STDVRP algorithm on average deviates from the optimal solution by about 6%, while the original saving algorithm deviates from the optimal solution by 17%. Cordeau et al. ([3]) compared a few

well-known algorithms and showed, among others, that when the saving algorithm is used for solving CVRP (the original problem that this algorithm was designed to solve), the deviation of the algorithm's results from the optimal (or best-known solution) was on average about 6%. Based on our findings for stochastic time-dependent vehicle-routing problems, the results of the STDVRP are similar to the results of the saving algorithm for CVRP.

The algorithm's complexity was calculated and found to be $\boldsymbol{O}(n^3 m)$, where n is the number of customers and m is the size of the candidates list. Based on a complexity analysis, we had expected that the running time of the algorithms would increase in linear proportion to the number of route pairs stored in the candidates list (m). In reality, we found, based on observations, that the running-time increase was moderate.

It was also found that a candidates list of 3 to 5 yields the most significant improvement for the algorithm results. The use of a larger number of candidates does improve the results; however, the time devoted to search for the results increases linearly to the number of candidates and yields only an insignificant improvement when using a candidate list larger than 5.

Genetic algorithms (GAs) have been widespread use amongst modern meta-heuristics, and several applications to VRPs have been reported ([30-34]). The algorithm was also tested in comparison to a generic genetic algorithm. The GA-STDVRP algorithms is bases on the work of [35], who presented a genetic vehicle representation, designed to deal in an effective way with all the information that candidate solutions must encode. Experimental results show that this method is both effective and robust. The GA-STDVRP algorithm presented results that are about 1.05 times better than the result of the Savings-STDVRP (for problems with 100 and 200 customers), yet with running time is at least 200 times slower than the Savings-STDVRP. This means that in order to achieve a result that about 5% better than the results of the Savings-STDVRP algorithm using the GA-STDVRP algorithm more than two hours (for problems with 100 customers, or five hours for problems with 200 customers) has to be spent compared to the 20 seconds spent by the Savings-STDVRP algorithm (73 seconds for problems with 200 customers). It is important to note that the GA-STDVRP was implemented mainly for accuracy comparison rather than performance and efficiency. It is believed that the GA algorithm can be implemented more efficiently.

REFERENCES

1. Dantzig, G.B. and J.H. Ramser, *The Truck Dispatching Problem.* Management Science, 1959. **6**(1): p. 80-91.
2. Ballou, R.H. and Y.K. Agarwal, *A Performance Comparison of Several Popular Algorithms for Vehicle Routing and Scheduling.* Journal of Business Logistics, 1988. **9**: p. 51-65.
3. Cordeau, J.F., et al., *A Guide to Vehicle Routing Heuristics.* Journal of the Operational Research society, 2002. **53**: p. 512-522.

4. Toth, P. and D. Vigo, *The Vehicle Routing Problem*, ed. P. Toth and D. Vigo. 2001, Philadelphia: Siam.
5. Ichoua, S., M. Gendreau, and J.-Y. Potvin, *Vehicle Routing with Time-Dependent Travel Times.* European Journal of Operational Research, 2003. **144**: p. 379-396.
6. Malandraki, C., *Time dependent vehicle routing problems: Formulations, solution algorithms and computational experiments*. 1989, Northwestern University.
7. Malandraki, C. and M.S. Daskin, *Time Dependent Vehicle Routing Problems: Formulations, Properties and Heuristic Algorithms.* Transportation Science, 1992. **26**(3): p. 185-200.
8. Hill, A.V. and W.C. Benton, *Modelling intra-city time-dependent travel speeds for vehicle scheduling problems.* The journal of the Operational Research Society, 1992. **43**(4): p. 343-351.
9. Ahn, B.H. and J.Y. Shin, *Vehicle-routeing with time windows and time-varying congestion.* The journal of the Operational Research Society, 1991. **42**(5): p. 393-400.
10. Malandraki, C. and R.B. Dial, *A restricted dynamic programming heuristic algorithm for the time dependent traveling salesman problem.* European Journal of Operational Research, 1996. **90**(1): p. 45-55.
11. Fleischmann, B., M. Gietz, and S. Gnutzmann, *Time-varying travel times in vehicle routing.* Transportation Science, 2004. **38**(2): p. 160.
12. Jung, S. and A. Haghani, *Genetic algorithm for the time-dependent vehicle routing problem.* Transportation Research Record: Journal of the Transportation Research Board, 2001. **1771**(-1): p. 164-171.
13. Haghani, A. and S. Jung, *A dynamic vehicle routing problem with time-dependent travel times.* Computers and Operations Research, 2005. **32**(11): p. 2959-2986.
14. Van Woensel, T., et al., *Vehicle routing with dynamic travel times: A queueing approach.* European Journal of Operational Research, 2008. **186**(3): p. 990-1007.
15. Donati, A.V., et al., *Time dependent vehicle routing problem with a multi ant colony system.* European Journal of Operational Research, 2008. **185**(3): p. 1174-1191.
16. Gendreau, M., G. Laporte, and R. Seguin, *Stochastic Vehicle Routing.* European Journal of Operational Research, 1996. **88**: p. 3-12.
17. Bertsimas, D.J., *A Vehicle Routing Problem With Stochastic Demand.* Operations Research, 1992. **40**(3): p. 574-585.
18. Wang, X. and A.C. Regan, *Assignment models for local truckload trucking problems with stochastic service times and time window constraints.* Transportation Research Record, 2001. **1171**: p. 61-68.
19. Tillman, F.A., *The Multiple Terminal Delivery Problem with Probabilistic Demands.* Transportation Science, 1969. **3**(3): p. 192-204.
20. Stewart, W.R. and B.L. Golden. *A Chance-Constrained Approach to the Stochastic Vehicle Routing Problem*. in *1980 Northeast AIDS Conference*. 1980. Philadelphia.
21. Golden, B.L. and J.R. Yee, *A Freamework for Probalistic Vehicle Routing.* IIE Transactions, 1979. **11**(2): p. 109-112.
22. Bertsimas, D.J., *Probabilistic Combinatorial Optimization Problems*. 1988, Massachusetts Institute of Technology.

23. Laporte, G., F. Louveaux, and H. Mercure, *The Vehicle Routing Problem With Stochastic Travel Times.* Transportation Science, 1992. **26**(3): p. 161-170.
24. Lambert, V., G. Laporte, and F. Louveaux, *Designing collection routes through bank branches.* Computers and Operations Research, 1993. **20**(7): p. 783-791.
25. Kenyon, A.S. and D.P. Morton, *Stochastic Vehicle Routing with Random Travel Times.* Transportation Science, 2003. **37**(1): p. 69-82.
26. 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.
27. Solomon, M.M., *Algorithms for the Vehicle Routing and Scheduling Problems with Time Window Constraints.* Operations Research, 1987. **35**(2): p. 254-265.
28. Banks, J., *Handbook of simulation : principles, methodology, advances, applications, and practice*. 1998, New York: Wiley. xii, 849 p.
29. Nahum, O.E. *Oren Nahum's Homepage, Benchmark Problems*. 2009 [cited 07-Oct-2009]; Available from: http://orennahum.dyndns.org/Problems%20and%20Instances.html.
30. Thangiah, S.R., K.E. Nygard, and P.L. Juell. *GIDEON: a genetic algorithm system for vehicle routing with timewindows*. 1991.
31. Blanton Jr, J.L. and R.L. Wainwright. *Multiple vehicle routing with time and capacity constraints using genetic algorithms*. 1993: Morgan Kaufmann Publishers Inc. San Francisco, CA, USA.
32. Thangiah, S.R. and A.V. Gubbi. *Effect of genetic sectoring on vehicle routing problems with timewindows*. 1993.
33. Schmitt, L.J., *II. An empirical computational study of genetic algorithms to solve order based problems: An emphasis on TSP and VRPTC*, PhD thesis, The University of Memphis, 1994.* DAI.
34. Potvin, J.Y., C. Duhamel, and F. Guertin, *A genetic algorithm for vehicle routing with backhauling.* Applied Intelligence, 1996. **6**(4): p. 345-355.
35. 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.
36. Nahum, O.E. and Y. Hadas. *Developing a Model for the Stochastic Time-Dependent Vehicle-Routing Problem*. in *International Conference on Computers & Industrial Engineering (CIE39)*. 2009. The University of Technology of Troyes, Troyes, France.