A dynamic programming approach to multi-objective time-dependent capacitated single vehicle routing problems with time windows

A Dynamic Programming Approach to Multi-Objective Time-Dependent Capacitated Single Vehicle Routing Problems with Time Windows

S. Dabia*, T. van Woensel, A.G. De Kok

Eindhoven University of Technology, School of Industrial Engineering, The Netherlands

Abstract

A single vehicle performs several tours to serve a set of geographically dispersed customers. The vehicle has a finite capacity and is only available for a limited amount of time. Moreover, tours' duration is restricted (*e.g.* due to quality or security issues). Because of road congestion, travel times are time-dependent: depending on the departure time at a customer, a different travel time is incurred. Furthermore, all customers need to get delivered in their specific time windows. Contrary to most of the literature, we consider a multi-objective cost function: simultaneously minimizing the total time traveled including waiting times at customers due to time windows, and maximizing the total demand fulfilled. Efficient dynamic programming algorithms are developed to compute the Pareto set of routes, assuming a specific structure for time windows and travel time profiles.

Keywords: Time-dependent travel times, Multiple objectives, Dynamic programming

1. Introduction

Consider the situation in which a vehicle is required to fill up ATMs located at different places from a central bank. For security reasons, it is not allowed to carry a large amount of money. Consequently, the vehicle is forced to make several short tours during its operating period (*e.g.* a working day) going back and forth to the central bank. Similarly, in the case of food home delivery, tours are relatively short as products are perishable (*e.g.* should remain warm) and thus need to be delivered as soon as possible to their destinations (Azi et al., 2007). This again leads to multiple tours of the same vehicle. Clearly, a vehicle can make several tours during its designated operating time, respecting the vehicle's capacity for each tour. A 3PL company we are currently working with, aims at scheduling its fleet such that a vehicle's total travel time is minimized, its capacity utilization is maximized and

*Corresponding author

Email addresses: s.dabia@tue.nl (S. Dabia), t.v.woensel@tue.nl (T. van Woensel), a.g.d.kok@tue.nl (A.G. De Kok)

URL: http://home.tm.tue.nl/tvwoense/ (T. van Woensel)

all customers get delivered in their specific time windows. This illustrates the importance to consider multiple dimensions for the objective function in the scheduling problem.

Formally, we consider a Multi-Objective Time-Dependent Capacitated Single Vehicle Routing Problem with Time Windows (denoted as MO-TD-CSVRP-TW). More specifically, a single vehicle departs from a central depot (denoted by 0), performs several tours to serve a set of geographically scattered customers and returns to the same depot. The vehicle routing problem at hand is capacitated on two dimensions: on the vehicle capacity (truckload) and on the time capacity. Time capacity means that the vehicle is only available for a limited amount of time, *e.g.* due to drivers' availability following the working regulations. Moreover, a tour is not allowed to last more than a certain amount of time. Because of these vehicle and time capacity limitations, it might not be possible to serve all customers. Therefore, we need to decide on a subset of customers to be actually served. Finally, because of road congestion, travel times are time-dependent: depending on the specific departure time at a customer, a different travel time is incurred. Customers have time windows and these are considered to be hard. However, since customers are not directly harmed by possible waiting times, the vehicle is allowed to wait if it arrives early at a customer. Rather than using a single objective function in which each of the to be optimized dimensions is weighted, we consider a multi-objective cost function. Here, we simultaneously minimize the total travel time including any waiting times (due to time windows), and maximize the total quantity delivered by the vehicle.

By imposing some structure on time windows and assuming travel time profiles that adhere to the FIFO assumption, we develop a pseudo-polynomial exact dynamic programming algorithm (denoted as DP) based on the non-dominance principle. By making one additional assumption regarding the structure of travel times, we develop an approximate time-dependent dynamic programming based algorithm (denoted as DP^ϵ), where ϵ is the approximation's worst case precision. The worst case performance of DP^ϵ is guaranteed and its running time is polynomially bounded.

The main contributions of this paper are twofold:

First, this paper tackles the vehicle routing problem from a multi-objective point of view. In real-life, decision makers might have numerous contradictory and equally important objectives they jointly want to optimize. Our approach determines the set of points representing the compromise solutions between the different conflicting objectives. Moreover, new objectives can easily be introduced in our proposed framework without losing the relevance of the initial ones. Even non-cost driven objectives (*i.e.* drivers' workload, customers' satisfaction, CO_2-emissions,...) can be considered. Furthermore, road congestion is captured by assuming time-dependent travel times. Moreover, transportation and time limitations (customers' time windows, the vehicles' time availability and the limitation on the tours' duration) are taken into account. As such, this paper deals with a rich VRP including many practical features.

Secondly, this work presents a time-dependent dynamic programming approach for the vehicle routing problem. By assuming some structure on time

windows and travel times, we present an exact DP algorithm based on the non-dominance principle. An approximate DP algorithm is formulated based on an additional assumption regarding the structure of travel times. DP^{ε} is based on the trimming method (Woeginger (2005)), and has a provable worst case performance guarantee.

The paper is organized as follows. Section 2 reviews the literature relevant to our problem. As a first building block, we analyze the situation where only one single tour can be performed. This is the case when *e.g.* the vehicle's time availability is short. Section 3 is devoted to the single tour MO-TD-CSVRP-TW. In a second step, when transportation capacity and the maximum tours's duration are binding, and the remaining time available after the first tour's execution is sufficient to perform more tours, multiple tours are scheduled. Section 4 explains how the results derived for the single tour MO-TD-CSVRP-TW can be applied, using the giant tour representation, to deal with the multiple tour situation. In Section 5, a numerical study is conducted. Finally, Section 6 concludes with a summary of the main results.

2. Selected Literature Review

This literature review intends to give the relevant literature on each of the important dimensions of the problem at hand.

Despite its practical importance, the **single vehicle routing problem** has received relatively little attention in the literature. In Azi et al. (2007), the single vehicle routing problem with time windows and multiple tours is treated. In Gribkovskaia et al. (2007) and Gribkovskaia et al. (2008), the single vehicle routing problem with pickups and deliveries and the single vehicle routing problem with deliveries and selective pickups, in which it is not necessary to meet all pickup demands, are respectively considered. Süral and Bookbinder (2003) consider a single vehicle routing with unrestricted back-hauls.

Contrary to most of the existing literature on single vehicle routing problems, we consider a **multi-objective cost function**. For an extensive literature review on multi-objective VRP models, we refer to Jozefowiez et al. (2008). In practice, managers might aim to minimize both the distance traveled and maximize the number of customer visited (Ribeiro and Lourenço, 2001), or to minimize both travel time and total customers' waiting time (Hong and Park, 1999). Multi-objective cost functions are very attractive for modeling practical situations in which contradictory objectives need to be optimized simultaneously. However, multi-objective cost functions are usually reduced to a composite single objective cost function by using a weighted sum of the various objectives (Rosenblatt and Sinuany-Stern, 1989). Ulungu and Teghem (1997) and Visée et al. (1996) argued that solutions obtained by the optimization of composite single objective function are only a small subset of the entire non-dominated set of solutions, and therefore could lead to suboptimal managerial decisions.

Traditionally, total travel costs are calculated in terms of distances: the overall distance traveled is minimized. Routes obtained as such, do not guarantee a good solution when applied in real life. One major shortcoming is due to the difficulty of taking road **time-dependent** congestion into

account. Despite numerous publications dealing with efficient scheduling methods for vehicle routing, very few have addressed, due to its complexity, the inherent dynamic nature of travel times. Ichoua et al. (2003) present a time-dependent model with simple travel time profiles that satisfy the FIFO assumption. They adopt a parallel tabu search heuristic to obtain the schedules. Van Woensel et al. (2008) propose a more realistic model by applying queueing theory to better capture the congestion effects on travel times. In Van Woensel and Vandaele (2006), real-life data (simulation respectively) is used to validate their queuing approach for traffic flows.

In this paper, we present an exact time-dependent **dynamic programming (DP) algorithm**. Vehicle routing problems are NP-hard. Hence, reasonably sized instances are unlikely to be solved to optimality in polynomial time. Consequently, these types of problems are usually dealt with using (meta-)heuristics, which deliver good solutions in reasonable computation times (see *e.g.* Bräysy and Gendreau (2005a); Bräysy and Gendreau (2005b); Taillard et al. (1997) for some good reviews). Next to heuristics, many researchers have adopted exact approaches to handle VRPs (see *e.g.* Laporte and Nobert (1980) for a good review). In Malandraki and Dial (1996), a restricted DP heuristic is proposed to solve the time-dependent travelling salesman problem (TSP). In each iteration of the restricted DP, only a subset (hence "restricted") with a predefined size and consisting of the best solutions is kept and used to compute solutions in the next iteration. Time-dependency in dynamic programming was first introduced by Kostreva and Wiecek (1993) to solve a path-planning problem. In Klamroth and Wiecek (2000) and Klamroth and Wiecek (2001), time-dependent dynamic programming is used to deal with single machine scheduling and capital budgeting problems.

The SVRP is a generalization of the general TSP. It is known that the exact optimization of the general TSP is NP-complete (Karp (1972)). Moreover, approximating the general TSP within any constant factor in polynomial time is not possible unless $NP = P$ (Sahni and Gonzales (1976)). However, many TSP instances encountered in real life have special features that moderate their hardness. Christofides (1976) developed an approximation algorithm for the *metric* TSP that computes tours with a cost at most 1.5 times the optimal tour's cost. Sanjeev (1996) even succeeded to design a *Polynomial Time Approximation Scheme* (PTAS) for the *Euclidean* TSP.

The idea of the **giant tour** representation was previously adopted by many researchers to transform the multiple traveling salesmen problem (m-TSP) to the standard TSP (Laporte et al. (1988); GuoXing (1995); Bellmore and Hong (1974)). It is demonstrated that the multiple salesman problem can be solved by solving the standard travelling salesman problem on an expanded graph. The expanded graph is constructed by introducing imaginary depots that are used to link tours performed by the individual salesmen into a longer tour.

3. The Single Tour Problem

The notation used is summarized in Table 1.

Variable	Description
N	Number of customers
t	Time. Time origin is always taken to be 0
t_0	Vehicle's dispatch time at the depot, $t_0 \geq 0$
t_i	Departure time at customer i
s_i	Service time at customer i
Q	Vehicle's capacity
T	Vehicle's time availability. The vehicle is available during $[t_0, t_0 + T]$
t^{lim}	Restriction on tours' duration
$[c_{ij}(t)]$	Vector travel costs assigned to the link between customer i and customer j when leaving customer i at time t, $0 \leq i \leq N$ and $0 \leq j \leq N$
$tt_{ij}(t)$	Travel time from customer i to customer j when leaving customer i at time t
d_j	Customer's j demand
$[t_j^l, t_j^u]$	Time window imposed at customer j. t_j^l and t_j^u are the opening and closing times visited customers
$\{G_j^k(a_j)\}$	Set of vector costs of Pareto-optimal routes with state a_j, starting at the depot and ending at customer j
$\{G_j^k\}$	Set of vector costs of Pareto-optimal routes, starting at the depot and ending at customer j
$\|A\|$	Cardinality of a set A
$\lceil x \rceil$	Nearest integer larger or equal to the real number x
ϵ	Approximation's worst case precision

Table 1: Notation used in this paper

As mentioned earlier, the single tour MO-TD-CSVRP-TW is useful to analyze as a basic building block. In practice, the single tour model occurs when the vehicle's time availability is limiting. The aim for the Single Tour MO-TD-CSVRP-TW is to schedule one single tour such that the tour's execution time including possible waiting times at customers is minimized, the total demand fulfilled is maximized and time windows are respected. To each customer j, we associate time-dependent profits: a visited customer contributes with some travel time, waiting time and demand delivered to the objective function. The tour's execution time is the time elapsed between the departure from and the return to the depot. Furthermore, the tour should be completed within the vehicle's time availability, and the quantity delivered to customers should not exceed the vehicle's capacity. Because of time constraints and the vehicle's finite capacity, it is not necessary to serve all customers. Hence, from the entire set of customers, only a subset is selected. Dominique et al. (2004), with a TSP with profits, considers a similar situation where it is not necessary to visit all customers. All customers need to be served in their specific time windows. However, the vehicle is allowed to wait if it arrives early at a customer.

If the vehicle leaves customer i at time t_i towards customer j, the arrival time at customer j is $t_j = t_i + tt_{ij}(t_i)$. The vector travel costs related to the link from customer i to customer j, at time t_i is defined as follows:

$$[c_{ij}(t_i)] = \begin{bmatrix} tt^*_{ij}(t_i) = tt_{ij}(t_i) + max(0, t_j^l - t_i - tt_{ij}(t_i)) + s_j \\ d_j \end{bmatrix} \qquad (1)$$

The first element is the travel time consisting of the travel time itself, the possible waiting time and service time at customer j. The second element is the demand of customer j. If the vehicle arrives within customer's j time window, the waiting time is equal to zero. Otherwise, it is equal to the difference between the arrival time at customer j and the lower bound t_l^j of customer's j time window. Note that the quantity delivered to customer j is time-independent.

For route R_i, we define $[c_{ij}(a, t_i)]$ as the state-dependent vector travel costs related to the link between customer i and customer j when the vehicle leaves customer i at time t_i, and the state of route R_i is a, being the quantity delivered to customers contained by route R_i. In other words:

$$[c_{ij}(a,t_i)] = \begin{cases} [c_{ij}(t_i)] \quad \text{if} \quad \begin{cases} t_i + tt_{ij}^*(t_i) \leq t_j^u \\ t_i + tt_{ij}^*(t_i) + tt_{j0}^*(t_i + tt_{ij}^*(t_i)) \leq t_0 + T \\ a + d_j \leq Q \end{cases} \\ \begin{bmatrix} +\infty \\ -\infty \end{bmatrix} \text{otherwise} \end{cases} \tag{2}$$

The new vector travel costs (2) ensures that the vehicle, currently planned to serve customer i, may travel to customer j, only if the vehicle arrives before the closing time t_j^u at customer j, the trip to customer j and back to the depot can be done within the vehicle's time availability T, and if enough transportation capacity to serve customer j is available. In fact, extremely high costs are assigned to schedules that violate any of the constraints and hence such schedules will be infeasible.

In what follows, we define the *common band* (CB) as the overlap between two distinct customers' time windows. On each link, we assume the following structure on the common band:

Assumption 1. *For every two customers i and j such that $t_i^l \leq t_j^l$, it holds that for every t, $t_j^l \leq t \leq t_i^u$, we have:*

$$s_i + tt_{ji}(t) > t_i^u - t \tag{3}$$

in which $t_i^u - t_j^l$ is the size of the common band.

Assumption 1 implies that customer i, with opening time t_i^l, can only be visited before customers with later opening times. Note that this assumption is always true in the special case of non-overlapping time windows. Moreover, in practice, mainly when time windows are tight, travel times are relatively large compared to the size of time windows. Therefore, even when time windows are overlapping, it will probably be infeasible to serve a customer after customers with later opening times. Given assumption 1 is satisfied, customers can be re-ordered, according to their opening times such that, in the re-ordered graph, a link (v_l, v_k) between customers v_l and v_k, is only feasible if $k > l$.

Additionally, all links returning to the depot are possible. Clearly, because of this re-ordering, a customer will at most appear once in a route. Therefore, we can solve the problem by means of shortest path like dynamic programs. See Desrosiers et al. (1983) for dynamic program formulations for the time-independent incapacitated shortest path problem.

3.1. DP: The Exact Dynamic Programming Algorithm

In this section, an Exact Dynamic Programming formulation based on the non-dominance principle is constructed. The FIFO assumption is needed to guarantee the non-dominance principle and can be stated as follows:

Assumption 2. *For every two customers i and j, and times t_1 and t_2, $t_1 \leq t_2$ implies that:*

$$t_1 + tt_{ij}(t_1) \leq t_2 + tt_{ij}(t_2) \tag{4}$$

The FIFO assumption is needed to prevent constructing Pareto-optimal routes based on dominated routes. It guarantees that, for every link, a later departure will not result in an earlier arrival. Actually for the non-dominance principle to hold, the vector travel costs' first element, consisting of both travel time, possible waiting time and service time, should satisfy the FIFO assumption. Furthermore, the vector travel costs' second element need to be monotone non-decreasing in time. In our case, the second element represents demand which is assumed to be deterministic and stationary, and hence non-decreasing in time. For the vector travel costs's first element, we can show (see appendix) that the FIFO property is preserved. Therefore, we state the following lemma:

Lemma 1. *For every two customers i and j, and times t_1 and t_2, $t_1 \leq t_2$ implies that:*

$$t_1 + tt^*_{ij}(t_1) \leq t_2 + tt^*_{ij}(t_2) \tag{5}$$

Given the FIFO assumption, the principle of non-dominance is then stated as follows. Let R_i be a Pareto-optimal route starting at time t_0 at the depot and ending at customer i at time $t_i > t_0$. For each customer j lying on R_i, route $R_j \subseteq R_i$, starting at time t_0 at the depot and arriving at time $t_j, t_0 < t_j < t_i$, at customer j, is not dominated by any other route starting at the depot and ending at customer j.

DP works as follows: in the kth iteration, customer i is added to the current set of Pareto-optimal routes, with at most $k-1$ visited customers and state $a_i - d_i$, starting at the depot and ending at customer j. The newly generated routes, with at most k visited customers and state a_i, starting at the depot and ending at customer i are evaluated, and only Pareto-optimal ones, with distinct vector costs, are kept. This process is repeated for all customers. In the last iteration, when no more customers can be added, Pareto-optimal routes starting and ending at the depot are generated. Note that, because of customers' re-ordering, a customer will not appear more than once in a route.

In each iteration $|\{G^k_j\}|$ is bounded by $min(10^\alpha T, 10^\beta Q)$. α and β are the number of decimals allowed for time and demand. Therefore, the complexity of DP is pseudo-polynomial and proportional to:

$$\sum_k \sum_j |\{G^k_j\}| = O(N^2 min(10^\alpha T, 10^\beta Q)) \tag{6}$$

We formulate DP as follows:

$$\{G_i^k(a_i)\} = \underset{j<i}{vopt}\{\{G_j^{k-1}(a_i - d_i)\} + [c_{ji}(a_i - d_i, t_j)]\} \tag{7}$$

Initially, the cost of the first visit to a customer i is calculated. Then, we recursively compute the set of Pareto-optimal solutions $\{G_i^k(a_i)\}$ by using (7). The set of Pareto-optimal solutions $\{G_i^k\}$ is determined as:

$$\{G_i^k\} = \bigcup_{0 \leq a_i \leq Q} \{G_i^k(a_i)\} \tag{8}$$

DP is pseudo-polynomial. However, considering large instances might be computationally very exhaustive. In the next section, we develop an approximate DP based algorithm that deals with the curse of complexity.

3.2. DP^ϵ: The Approximate Dynamic Programming Algorithm

In order to reduce the complexity of DP, we impose extra structure to its execution. We observed from the DP that the set $\{G_i^k\}$ contains many solutions that are very close to each other and that the corresponding Pareto-optimal routes are very similar to each other. Therefore, we trim for each iteration k and for every customer i, the generated set of Pareto-optimal solutions $\{G_i^k\}$. More specifically, the set $\{G_i^k\}$ is trimmed by reducing the solutions that are very close to each other, and then this trimmed set $\{\widetilde{G}_i^k\}$ is used in the dynamic program to compute the untrimmed set $\{G_i^{k+1}\}$. This approach of adding structure to the execution of algorithms was first introduced by Ibarra and Kim (1975). Sahni (1976) applied it to a variety of scheduling problems. Woeginger (2005) applied the trimming method to the problem of scheduling two parallel machines.

Formally, the set $\{G_i^k\}$ can be represented by geometric points in the rectangle $[0, 10^\alpha T] \times [0, 10^\beta Q]$. The rectangle is cut into multiple boxes of exponentially increasing size. Points contained by the same box are considered to be very close to each other. In each box, and for each customer j, only one point is retained. The cuts on the travel time axis are executed at the coordinates $\Delta_1^m, m = 1, ..., L_1$, and the cuts on the quantity delivered axis are executed at the coordinates $\Delta_2^{-p}, p = 1, ..., L_2$, such that:

$$\Delta_1 = 1 + \frac{\epsilon}{2N} \quad \text{and} \quad \Delta_2 = 1 - \frac{\epsilon}{2N} \tag{9}$$

Where ϵ is a real number between 0 and 1. It represents the approximation's worst case precision.
The values of L_1 and L_2 are chosen such that $\Delta_1^{L_1} \leq 10^\alpha T$ and $\Delta_2^{-L_2} \leq 10^\beta Q$. We choose:

$$\begin{cases} L_1 = \lceil \frac{\ln(10^\alpha T)}{\ln \Delta_1} \rceil \leq \lceil (1 + \frac{2N}{\epsilon}) \ln(10^\alpha T) \rceil \\ L_2 = \lceil \frac{\ln(10^\beta Q)}{\ln \frac{1}{\Delta_2}} \rceil \leq \lceil \frac{2N}{\epsilon} \ln(10^\beta Q) \rceil \end{cases} \tag{10}$$

Figure 1 illustrates the reduction of the set of Pareto-optimal solutions. The trimmed set of Pareto-optimal solutions contains at most $L_1 \times L_2$ solutions. Similarly to (6), we can compute the complexity of DP^ϵ as being

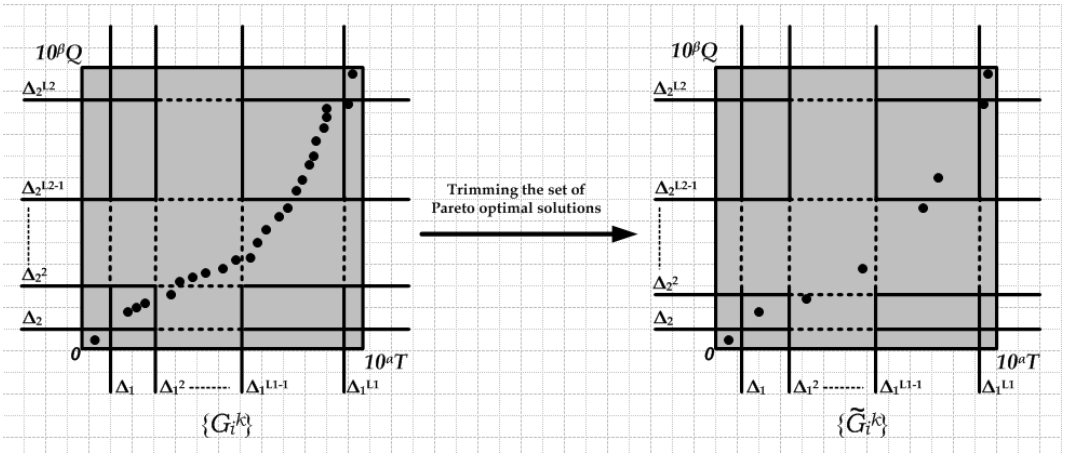

Figure 1: The reduction of the set of Pareto-optimal solutions.

proportional to:

$$\begin{aligned}\sum_k \sum_j |\{\widetilde{G}_j^k\}| &= O(N^2 L_1 L_2) \\ &= O\left(\frac{N^4}{\epsilon^2} \ln(10^{\alpha}T) \ln(10^{\beta}Q)\right)\end{aligned}$$

We conclude that DP^{ϵ} is polynomial in the input size and in $\frac{1}{\epsilon}$.

Worst case performance guarantee of DP^{ϵ}:

By making an additional assumption concerning travel times' structure, we can show that DP^{ϵ} has a provable worst case performance guarantee. In the case of a time-independent multi-objective optimization problem, a worst case performance guarantee is such that for every Pareto-optimal solution, there exists a Pareto-approximate solution that is not, in all objectives, worse by more than a known factor than the Pareto-optimal solution. In our case, travel times are time-dependent. Therefore, the worst case performance guarantee is such that for every solution given by DP, the total travel time including any waiting times is at most a factor $1 + \epsilon$ from that of a DP^{ϵ} solution plus some error depending on the vehicle's dispatch time, and the total quantity delivered is at least a factor $1 - \epsilon$ away from that of the DP solution. Of course, there is a trade-off between the worst case precision and the algorithm's computation time. On the one hand, small values of ϵ induce more accurate solutions at the expense of the computation time. In fact, for small values of ϵ, more solutions are kept during the execution of the dynamic program. As a result, the quality of the approximation improves. However, because more solutions need to be computed as we proceed with the execution of the dynamic program, the computation time increases. On the other hand, for large values of ϵ, many solutions are deleted, and hence both the computation and the quality of the approximation will decrease.

To guarantee a worst case precision for DP^{ϵ}, we make the following additional assumption regarding the structure of travel times:

Assumption 3. *For every two customers i and j, and every real number $1 \leq \alpha \leq 2$, it holds that for every time t:*

$$tt_{ij}(\alpha t) \leq \alpha tt_{ij}(t) \tag{11}$$

Assumption 3 means that leaving customer i towards customer j at a later time αt, instead of time t, will not multiply travel time by more than a coefficient α;. In other words, fast increases in travel time are not allowed. For instance, the travel time profiles $tt_{ij}(t) = t, tt_{ij}(t) = \sqrt{t}$ and $tt_{ij}(t) = lnt$ satisfy assumption (3). Note that the time origin is taken to be 0. We can show that if assumption 3 is satisfied, the following important lemma holds:

Lemma 2. *For every two customers i and j, and every real number $1 \leq \alpha \leq 2$, it holds that for every time t:*

$$tt^*_{ij}(\alpha t) \leq \alpha tt^*_{ij}(t) \tag{12}$$

Based on this, we formulate an important lemma. It states that in each iteration k of the DP, there is a solution whose first element is at most, and second element is at least, a known factor away from these of an approximate solution generated in DP^ϵ kth iteration. If lemmas 1 and 2 hold, we have the following:

Lemma 3. *Let k, $1 \leq k \leq N$, be an integer and i be a customer. Given assumptions 1, 2 and 3. For every solution* $\begin{bmatrix} x_{i,k} \\ y_{i,k} \end{bmatrix} \in \{G^k_i\}$, *there exists a solution* $\begin{bmatrix} \widetilde{x}_{i,k} \\ \widetilde{y}_{i,k} \end{bmatrix} \in \{\widetilde{G}^k_i\}$ *such that:*

$$\begin{cases} \widetilde{x}_{i,k} \leq \Delta^k_1 x_{i,k} + (\Delta^k_1 - 1)t_0 \\ \widetilde{y}_{i,k} \geq \Delta^k_2 y_{i,k} \end{cases} \tag{13}$$

Woeginger (2005) states a similar result for the problem of scheduling two parallel machines. However, in Woeginger (2005) costs are time-independent and only a single objective function is treated. Moreover, no prove is provided. In this paper, we prove the result in the case of a multi-objective optimization problem in which costs are time-dependent, and the cost of adding a customer to a solution depends on the previous visited customers. This makes the proof more technical. Therefore, a detailed proof (see appendix) is provided. Lemma 3 leads to the following theorem:

Theorem 1. *For every solution* $\begin{bmatrix} x_{0,k} \\ y_{0,k} \end{bmatrix} \in \{G^k_0\}$, *there exists a solution* $\begin{bmatrix} \widetilde{x}_{0,k} \\ \widetilde{y}_{0,k} \end{bmatrix} \in \{\widetilde{G}^k_0\}$ *such that:*

$$\begin{cases} \widetilde{x}_{0,k} \leq (1+\epsilon)x_{0,k} + \epsilon t_0 \\ \widetilde{y}_{0i,k} \geq (1-\epsilon)y_{0,k} \end{cases} \tag{14}$$

Proof. DP^{ϵ} and DP have at most N iterations. Let us assume they converge after k iterations ($k \leq N$). According to lemma 3, for every solution $\begin{bmatrix} x_{0,k} \\ y_{0,k} \end{bmatrix} \in \{G_0^k\}$, there is a solution $\begin{bmatrix} \tilde{x}_{0,k} \\ \tilde{y}_{0,k} \end{bmatrix} \in \{\tilde{G}_0^k\}$ such that:

$$\begin{cases} \tilde{x}_{0,k} \leq \Delta_1^k x_{0,k} + (\Delta_1^k - 1)t_0 \leq \Delta_1^N x_{0,k} + (\Delta_1^N - 1)t_0 \\ \tilde{y}_{0,k} \geq \Delta_2^k y_{0,k} \geq \Delta_2^N y_{0,k} \end{cases} \tag{15}$$

The sequence $\left(1 + \frac{\epsilon}{2N}\right)^N$ is increasing and converges to $e^{\frac{\epsilon}{2}}$ when N goes to infinity. Moreover, $\left(1 - \frac{\epsilon}{2N}\right)^N$ is decreasing and converges to $e^{-\frac{\epsilon}{2}}$ when N goes to infinity. Hence, for every $N \geq 1$:

$$\left(1 + \frac{\epsilon}{2N}\right)^N \leq e^{\frac{\epsilon}{2}} \quad \text{and} \quad \left(1 - \frac{\epsilon}{2N}\right)^N \geq e^{-\frac{\epsilon}{2}} \tag{16}$$

Furthermore, for $0 < \epsilon < 1$ we have:

$$e^{\frac{\epsilon}{2}} \leq 1 + \epsilon \quad \text{and} \quad e^{-\frac{\epsilon}{2}} \geq 1 - \epsilon \tag{17}$$

Therefore, we have the following important result:

$$\begin{cases} \tilde{x}_{0,k} \leq (1 + \epsilon)x_{0,k} + \epsilon t_0 \\ \tilde{y}_{0,k} \geq (1 - \epsilon)y_{0,k} \end{cases} \tag{18}$$

□

When the vehicle's dispatch time at the depot is 0, DP^{ϵ} belongs to the FPTAS family of algorithms. However, an additional error arises when the vehicle is dispatched at a later moment $t_0 > 0$. Therefore, although a later dispatch might result in a reduced total travel time (*e.g* congestion might be avoided), such a decision results in a additional error ϵt_0.

4. The Multiple Tour Problem

When the vehicle capacity Q and the maximum tours' duration t^{lim} are binding, the time used up for the first tour might be less than the vehicle's total time availability T. Therefore, we can still use the vehicle for its remaining time to schedule a second tour (or more if time allows). In what follows, we explain how we adapt the *giant tour* representation to our problem. Using the *giant tour* representation introduced by Funke et al. (2005), we can reduce the multiple tour model to a single tour model. As such, all results derived for the single tour MO-TD-CSVRP-TW in the previous section can be applied.

A formal description of the *giant tour* representation can be found in Funke et al. (2005). Many other researchers adopted the idea of the *giant tour* representation to transform the m-TSP to a standard TSP (Laporte et al. (1988); GuoXing (1995)). Therefore, the result from the standard TSP can be applied on an extended graph describing the *giant tour* representation of the m-TSP. In our case, the case of a multiple tour single VRP, the solution is a set of Pareto-optimal schedules. Each schedule is a collection of routes all starting and ending at the depot. The modelling idea is to add dummy depots representing duplications of the real depot to the existing graph. Routes of

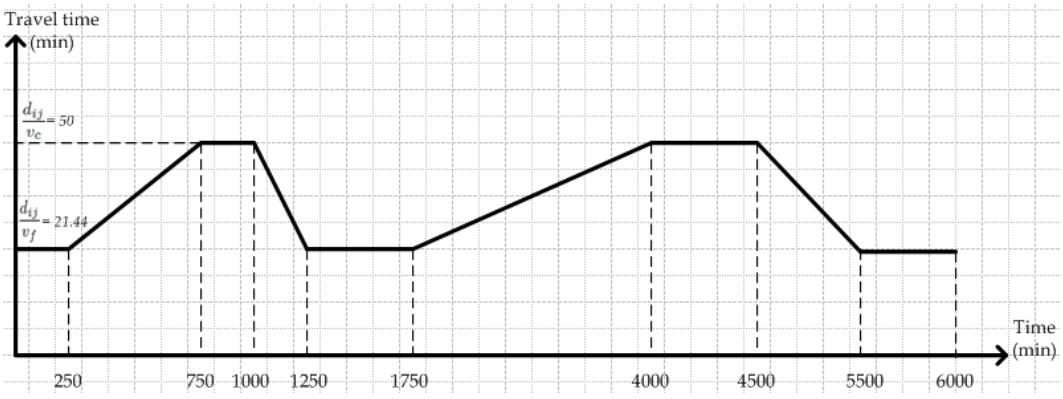

Figure 2: Travel time profile.

a given schedule can then be linked via the dummy nodes into a long (*giant*) tour. Hence, a route is such that customers are visited only once and the depot is visited multiple times. Moreover, Each time the depot is visited, the resources consumption is reset.

Funke et al. (2005) showed that the order of routes is not important. However, because of time-dependency, the order of routes is very important in our case. Since a schedule is equivalent to its *giant tour* representation, the results derived for the one tour SVRP can be directly applied to find the set of Pareto-optimal giant tours. The maximum dummy nodes that can be added to the existing graph is $N-1$. This is the case when all customers are served and during every tour only one customer is served. Moreover, when a schedule consists of $N-1$ tour, the total quantity delivered is bounded by $(N-1)Q$. Therefore, the Multiple Tour Model is more complex than the single tour model.

5. Computational Results

For our numerical study, customers' location as well as their demand are taken from Solomon's data sets (Solomon (1987)). Time windows are modified in such a way that the common band assumption is satisfied. Furthermore, congestion is taken into account. This is achieved by assuming that the speed on each link is time-dependent and derive the travel time profile by using the relation $tt_{ij}(t) = \frac{d_{ij}}{v_{ij}(t)}$ where d_{ij}, distance between customers i and j, is computed based on Solomon's data sets. $v_{ij}(t)$ is the time-dependent speed bij which the vehicle traverse the link between customers i and j. The congested speed is taken to be $v_c = 30km/hr$, and the free speed is taken to be $v_f = 70km/hr$. The resulting travel times satisfy both assumptions 2 and 3. Figure 2 illustrates the travel time between customers 24 and 66 in case of C instances.

Furthermore, we take $t^{lim} = \{500, 2000\}$, $Q = \{100, 200\}$ and $\epsilon = \{0.01, 0.05, 0.1, 0.3\}$. Large values of t^{lim} and Q allow including more customers in a tour.

We consider instances with $N = 100$ customers and a planning horizon of $T = 6000$ minutes. To compare the different Pareto fronts generated, we define two measures introduced by Zitzler et al. (2000): the two-set coverage

metric and the average distance metric. The choice of the metrics is motivated by their intuitive explanation. The *two-set coverage* metric is defined as:

$$C(X \succ Y) = \frac{|\{y \in Y; \exists x \in X : x \text{ dominates } y\}|}{|Y|} \tag{19}$$

in which X and Y are two Pareto curves. The two-set coverage metric calculates the fraction of solutions in Y that are dominated by a solution in X.
The *average distance* metric is defined as:

$$M(X,Y) = \frac{1}{|X|} \sum_{x \in X} min\{\|x - y\|; y \in Y\} \tag{20}$$

The metric M reflects how distant two Pareto curves are from each other.

The algorithms DP and DP^ϵ are implemented on a Intel(R) Core(TM)2 CPU, 2.13 GHz, 3 GB of RAM computer, in a Matlab R2008b environment. All instances and software is available from the authors upon request.

5.1. Comparing DP and DP^ϵ

Based on the results from Table 3, we conclude that, as expected, the accuracy of DP^ϵ is excellent. For all instances, on average no more than 15% of the solutions generated by $DP^{0.01}$ are dominated by any of DP's solutions. Moreover, no more 35% of the solutions of $DP^{0.05}$, $DP^{0.1}$ and $DP^{0.3}$ are dominated by a DP solution . Furthermore, the distance of the 0.01-Pareto front from the optimal one is negligible (less than 2 in most cases). Moreover, even the distance of the 0.05-Pareto front, 0.1-Pareto and the 0.3-Pareto front from the optimal one can be considered as very small. We also observe that the fraction of DP^ϵ solutions dominated by a DP solution increases slightly when randomness is added to the location of customers.

To illustrate the different Pareto fronts, Figure 3 depicts these corresponding to an instance with a relatively bad accuracy, namely the **R100** instance with the input parameters $t^{lim} = 500$ and $Q = 100$. The horizontal axis represents the total demand delivered and the vertical axis represents the total travel time. As expected, smaller values of the worst case precision ϵ result in a better ϵ-Pareto front, *i.e.* closer to the optimal one. Furthermore, the deviation of a ϵ-Pareto front from the optimal one increases in later iterations of DP (The size of boxes in Figure 1 increases). However, the deviation stays clearly within the worst case precision ϵ.

5.2. Impact of the Worst Case Precision ϵ for DP^ϵ

The complexity of the DP^ϵ algorithm increases with $\frac{1}{\epsilon}$. Hence, choosing smaller worst case precisions results in higher computation times. Table 2 shows the impact of ϵ on the computation times of DP^ϵ. We observe that with an $\epsilon = 0.01$, there is a small increase of about 0.1% in computation times with regard to DP. In fact, for small values of ϵ not many solutions are deleted during the execution of DP^ϵ and therefore it is not compensated for the trimming time. Furthermore, for $\epsilon = 0.05$, $\epsilon = 0.1$ and $\epsilon = 0.3$, CPU times are remarkably low, respectively 18%, 35% and 65% on average with regard to DP. Furthermore, the number of solutions generated remarkably decreases when the value of ϵ increases.

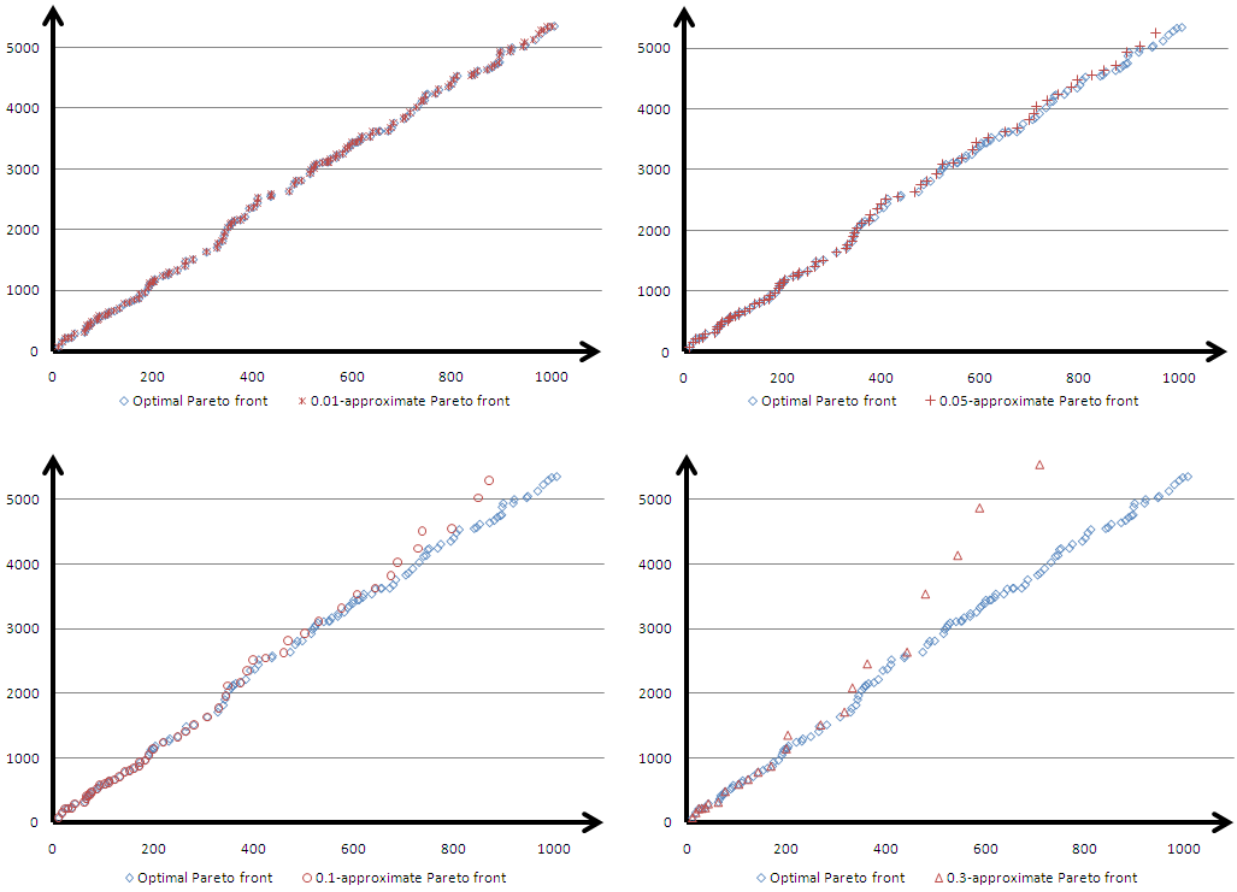

Figure 3: ϵ-Pareto fronts.

6. Conclusions and Future Research

Vehicle routing problems with time windows are NP-hard. However, some VRPs encountered in practice have special features that moderate their hardness. Although, modeling time windows in general increases complexity as even finding feasible solutions is NP-hard, including time windows that adhere to the common band assumption turns out to remarkably reduce the complexity of the problem. Therefore, efficient exact and approximate algorithms can be developed. The dynamic program algorithms proposed in this paper are very flexible as many practical features can easily be included. In fact, contrary to most of the literature dealing with vehicle routing problems, we take road congestion into account by assuming time-dependent travel time profiles. Furthermore, rather than assuming a single objective cost function, multiple objectives are considered. In this way, we are provided with a complete set of Pareto-optimal solutions instead of one optimal solution. Therefore, managers will have more flexibility choosing the most beneficial schedule that fits their specific situation. Additionally, in this paper, contrary to most of the literature, we considered a single vehicle routing problem where a vehicle is allowed to make multiple tours. The proposed DP based approximation has provable worst case precision. In other words, a decision maker can control the error of the solutions. However, we showed that

small precisions require more computation time. Furthermore, the number of solution is reduced when applying the approximation which facilitate the selection of a solution.

Acknowledgment

The research of Said Dabia has been funded by TRANSUMO, project number 10004927.

Appendix

Proof of lemma 1:

Let t_1 and t_2 be two moments, such that $t_1 \leq t_2$. We have:

$$\begin{aligned} t_1 + tt^*_{ij}(t_1) - t_2 - tt^*_{ij}(t_2) &= t_1 + tt_{ij}(t_1) + max(0, t^l_j - t_1 - tt_{ij}(t_1)) \\ &\quad -t_2 - tt_{ij}(t_2) - max(0, t^l_j - t_2 - tt_{ij}(t_2)) \end{aligned} \quad (21)$$

Observing that for any two real numbers a and b, the following equality is always true:

$$a + max(0, b - a) = max(a, b) \quad (22)$$

We write:

$$t_1 + tt^*_{ij}(t_1) - t_2 - tt^*_{ij}(t_2) = max(t^l_j, t_1 + tt_{ij}(t_1)) - max(t^l_j, t_1 + tt_{ij}(t_1))$$

Because of the FIFO assumption, we have:

$$t_1 + tt_{ij}(t_1) \leq t_2 + tt_{ij}(t_2)$$

Hence:

$$max(t^l_j, t_1 + tt_{ij}(t_1)) \leq max(t^l_j, t_2 + tt_{ij}(t_2))$$

And therefore:

$$t_1 + tt^*_{ij}(t_1) \leq t_2 + tt^*_{ij}(t_2)$$

Proof of lemma 2:

Let $\alpha \geq 1$ be a real number. For time t, we have:

$$tt^*_{ij}(\alpha t) = tt_{ij}(\alpha t) + max(0, t^l_j - \alpha t - tt_{ij}(\alpha t))$$

Again, using equality 22, we obtain:

$$\begin{aligned} tt^*_{ij}(\alpha t) &= max(t^l_j - \alpha t, tt_{ij}(\alpha t)) \\ &\leq max(\alpha t^l_j - \alpha t, \alpha tt_{ij}(t)) && \text{(using assumption 3)} \\ &= \alpha tt_{ij}(t) + max(0, \alpha t^l_j - \alpha t - \alpha tt_{ij}(t)) && \text{(using equality (22))} \\ &= \alpha tt^*_{ij}(t) \end{aligned}$$

Proof of lemma 3:

If $\begin{bmatrix} \alpha \\ \beta \end{bmatrix}$ and $\begin{bmatrix} \tilde{\alpha} \\ \tilde{\beta} \end{bmatrix}$ are two points in the same box, then:

$$\frac{\alpha}{\Delta_1} \leq \tilde{\alpha} \leq \Delta_1 \alpha \quad \text{and} \quad \Delta_2 \beta \leq \tilde{\beta} \leq \frac{\beta}{\Delta_2} \tag{23}$$

To prove lemma 3, we use induction on k, $1 \leq k \leq N$.
Let i be a customer. From (23), we conclude that lemma 3 is true for $k = 1$.
Let us assume lemma 3 is true for $k-1$.
Let $\begin{bmatrix} x_{i,k} \\ y_{i,k} \end{bmatrix} \in \{G_i^k\}$. Per definition of the set $\{G_i^k\}$, $\begin{bmatrix} x_{i,k} \\ y_{i,k} \end{bmatrix}$ is feasible.
Hence, there exists a feasible point $\begin{bmatrix} x_{j,k-1} \\ y_{j,k-1} \end{bmatrix} \in \{G_j^{k-1}\}$ such that:

$$\begin{cases} x_{i,k} = x_{j,k-1} + tt_{ji}^*(t_0 + x_{j,k-1}) \\ y_{i,k} = y_{j,k-1} + d_i \end{cases} \tag{24}$$

On the other hand, because of the induction assumption, there exists $\begin{bmatrix} \tilde{x}_{j,k-1} \\ \tilde{y}_{j,k-1} \end{bmatrix} \in \{\tilde{G}_j^{k-1}\}$ such that:

$$\begin{cases} \tilde{x}_{j,k-1} \leq \Delta_1^{k-1} x_{j,k-1} + (\Delta_1^{k-1} - 1)t_0 \\ \tilde{y}_{j,k-1} \geq \Delta_2^{k-1} y_{j,k-1} \end{cases} \tag{25}$$

Furthermore, $DP^{\varepsilon}_{Approx}$ generates the solution $\begin{bmatrix} \tilde{x}_{j,k-1} + tt_{ji}^*(t_0 + \tilde{x}_{j,k-1}) \\ \tilde{y}_{j,k-1} + d_i \end{bmatrix}$ in the kth step. Note that the addition of customer i to $\begin{bmatrix} \tilde{x}_{j,k-1} \\ \tilde{y}_{j,k-1} \end{bmatrix}$ is possible thanks to the re-ordering of customers allowed by assumption 1.
The point $\begin{bmatrix} \tilde{x}_{j,k-1} + tt_{ji}^*(t_0 + \tilde{x}_{j,k-1}) \\ \tilde{y}_{j,k-1} + d_i \end{bmatrix}$ might be removed after trimming. However some vector $\begin{bmatrix} \tilde{x}_{i,k} \\ \tilde{y}_{i,k} \end{bmatrix}$, located in the same box as $\begin{bmatrix} \tilde{x}_{j,k-1} + tt_{ji}^*(t_0 + \tilde{x}_{j,k-1}) \\ \tilde{y}_{j,k-1} + d_i \end{bmatrix}$ should be left.
From (23), we obtain:

$$\begin{cases} t_0 + \tilde{x}_{i,k} \leq \Delta_1(t_0 + \tilde{x}_{j,k-1} + tt_{ji}^*(t_0 + \tilde{x}_{j,k-1})) \\ \tilde{y}_{i,k} \geq \Delta_2(\tilde{y}_{j,k-1} + d_i) \end{cases} \tag{26}$$

Because of assumption 1 and the induction assumption, we have:

$$\begin{cases} t_0 + \tilde{x}_{i,k} \leq \Delta_1(\Delta_1^{k-1}(t_0 + x_{j,k-1}) + tt_{ji}(\Delta_1^{k-1}(t_0 + x_{j,k-1})) \\ \tilde{y}_{i,k} \geq \Delta_2(\Delta_2^{k-1} y_{j,k-1} + d_i) \end{cases} \tag{27}$$

Using lemma 2, we obtain:

$$\begin{cases} t_0 + \tilde{x}_{i,k} \leq \Delta_1(\Delta_1^{k-1}(t_0 + x_{j,k-1}) + \Delta_1^{k-1} tt_{ji}(t_0 + x_{j,k-1})) \\ \tilde{y}_{i,k} \geq \Delta_2(\Delta_2^{k-1} y_{j,k-1} + \Delta_2^{k-1} d_i) \end{cases} \tag{28}$$

Hence,

$$\begin{cases} \tilde{x}_{i,k} \leq \Delta_1^k x_{i,k} + (\Delta_1^{k-1} - 1)t_0 \\ \tilde{y}_{i,k} \geq \Delta_2^k y_{i,k} \end{cases} \tag{29}$$

References

Azi, N., Gendreau, M., Potvin, J.Y., 2007. An exact algorithm for a single-vehicle routing problem with time windows and multiple routes. European Journal of Operational Research 178, 755–766.

Bellmore, M., Hong, S., 1974. Transformation of multisalesmen problem to the standard traveling salesman problem. Journal of the Association for Computing Machinery 21, 500–504.

Bräysy, O., Gendreau, M., 2005a. Vehicle routing problem with time windows, part i: Route construction and local search algorithms. Transportation Sci 39, 104–118.

Bräysy, O., Gendreau, M., 2005b. Vehicle routing problem with time windows, part ii: Metaheuristics. Transportation Sci 39, 119–139.

Christofides, N., 1976. Worst-case analysis of a new heuristic for the traveling salesman problem. Report 388, Grad School of Industrial Administration, CMU 9, 417–430.

Desrosiers, J., Pelletier, P., Soumis, F., 1983. Plus court chemin avec constraints d'horaires. RAIRO 17, 357–377.

Dominique, F., Dejax, P., Gendreau, M., 2004. Traveling salesman problems with profits. Transportation Sci 39, 188–205.

Funke, B., Grünert, T., Irnich, S., 2005. Local search for vehicle routing and scheduling problems: Review and conceptual integration. Journal of heuristics 11, 267–306.

Gribkovskaia, I., sr. Halskau, O., Laporte, G., Vlĉek, M., 2007. General solutions to the single vehicle routing problem with pickups and deliveries. European Journal of Operational Research 180, 568–584.

Gribkovskaia, I., Laporte, G., Aliaksandr, S., 2008. The single vehicle routing problem with deliveries and selective pickups. Computers and Operations Research 35, 2908–2924.

GuoXing, Y., 1995. Transformation of multidepot multisalesmen problem to the standard travelling salesman problem. European Journal of Operational Research 81, 557–560.

Hong, S.C., Park, Y.B., 1999. A heuristic for bi-objective vehicle routing with time window constraints. International Journal of Production Economics 62, 249–258.

Ibarra, O.H., Kim, C.E., 1975. Fast approximation algorithms for the knapsack and sum of subset problems. Journal of the ACM 22, 463–468.

Ichoua, S., Gendreau, M., Potvin, J.Y., 2003. Vehicle dispatching with time-dependent travel times. European Journal of Operational Research 144, 379–396.

Jozefowiez, N., Semet, F., Talbi, E., 2008. Multi-objective vehicle routing problems. European Journal of Operational Research 189, 293–309.

Karp, R.M., 1972. Reducibility among combinatorial problems, in: In R.E. Miller and J. W. Thatcher, editors, Complexity of Computer Computations, Advances in Computing Research, pp. 85–103.

Klamroth, K., Wiecek, M.M., 2000. Dynamic programming approaches to the multiple criteria knapsack problem. Naval Research Logistics 47, 57–76.

Klamroth, K., Wiecek, M.M., 2001. A time-dependent multiple criteria single-machine scheduling problem. European Journal of Operational Research 135, 17–26.

Kostreva, M.M., Wiecek, M.M., 1993. Time dependency in multiple objective dynamic programming. Journal of Mathematical Analysis and Applications 173, 289–307.

Laporte, G., Nobert, Y., 1980. A cutting planes algorithm for the m-salesmen problem. Journal of the Operational Research Society 31, 1017–1023.

Laporte, G., Nobert, Y., Taillefer, S., 1988. Solving a family of multi-depot vehicle routing and location-routing problems. Transportation Sci 22, 161–172.

Malandraki, C., Dial, R.B., 1996. A restricted dynamic programming heuristic algorithm for the time dependent traveling salesman problem. European Journal of Operational Research 90, 45–55.

Ribeiro, R., Lourenço, H.R., 2001. A multi-objective model for a multi-period distribution management problem. Meta-heuristics International Conference MIC'2001, 97–101.

Rosenblatt, M.J., Sinuany-Stern, Z., 1989. Generating the discrete efficient frontier to the capital budgeting problem. Operations Research 37, 384–394.

Sahni, K.S., 1976. Algorithms for scheduling independent tasks. Journal of the ACM 23, 116–127.

Sahni, K.S., Gonzales, T., 1976. P-complete approximation problems. Journal of the ACM 23, 555–565.

Sanjeev, A., 1996. Polynomial-time approximations schemes for the euclidean tsp and other geometric problems, in: Proceedings of 37th IEEE Symp. on Foundations of Computer Science, pp. 2–12.

Solomon, M.M., 1987. Algorithms for the vehicle routing and scheduling problems with time window constraints. Operations Research 35, 254–265.

Süral, H., Bookbinder, J.H., 2003. The single-vehicle routing problem with unrestricted backhauls. Networks 41, 127136.

Taillard, E., Badeau, P., Gendreau, M., Geurtin, F., Potvin, J.Y., 1997. A tabu search heuristic for the vehicle routing problem with soft time windows. Transportation Sci 31, 170–186.

Ulungu, E.L., Teghem, J., 1997. Solving multi-objective knapsack problems by a branch and bound procedures to solve the bi-objective knapsack problem. Multicriteria analysis, J. N. Climaco(Editor), Springer-Verlag, New york , 269–278.

Van Woensel, T., Kerbache, L., Peremans, H., Vandaele, N., 2008. Vehicle routing with dynamic travel times: a queueing approach. European Journal of Operational Research 186, 990–1007.

Van Woensel, T., Vandaele, N., 2006. Empirical validation of a queueing approach to uninterrupted traffic flows. 4OR, A Quarterly Journal of Operations Research 4, 59–72.

Visée, M., Teghem, J., Pirlot, M., Ulungu, E.L., 1996. Two-phases method and branch and bound procedures to solve the bi-objective knapsack problem. Technical report, Departement of Mathematics and Operational Research, Facult polythecnique de Mons, Mons, Belgium .

Woeginger, G.J., 2005. A comment on schedulling two parallel machines with capacity constraints. Dicrete Optimization 2, 269–275.

Zitzler, E., Deb, K., Thiele, L., 2000. Comparaison of multiobjective evolutionary algorithms: Empirical results. Evolutionary Computation 8, 173–195.

Inst	t^{lim}	Q	DP		$DP^{0.01}$		$DP^{0.05}$		$DP^{0.1}$		$DP^{0.3}$	
			CPU	**Nb sol**	**CPU**	**Nb sol**	**CPU**	**Nb sol**	**CPU**	**Nb sol**	**CPU**	**Nb sol**
C100	**500**	**100**	1079	97	1091	97	947	68	775	46	640	23
		200	1182	97	1184	97	1080	62	814	44	670	23
	2000	**100**	1217	101	1232	102	1054	65	863	46	596	22
		200	1245	102	1270	102	1087	68	871	47	610	21
R100	**500**	**100**	1370	122	1376	119	1136	77	925	53	643	22
		200	1521	128	1453	122	1167	77	947	54	654	22
	2000	**100**	1762	143	1737	130	1273	74	999	53	596	22
		200	1555	136	1552	127	1210	76	944	53	638	22
RC100	**500**	**100**	1200	99	1195	97	935	55	748	42	541	19
		200	1300	103	1283	100	1023	56	831	40	587	19
	2000	**100**	1270	105	1261	98	969	56	757	40	510	20
		200	1335	107	1328	99	1036	60	809	40	575	19

Table 2: CPU and number of solutions generated

Inst	t^{lim}	Q	$C(DP \succ DP^{0.01})$	$M(DP, DP^{0.01})$	$C(DP \succ DP^{0.05})$	$M(DP, DP^{0.05})$	$C(DP \succ DP^{0.1})$	$M(DP, DP^{0.1})$	$C(DP \succ DP^{0.3})$	$M(DP, DP^{0.3})$
C100	**500**	**100**	0.01	0.04	0.34	29.62	0.37	76.17	0.30	358.36
		200	0.01	0.04	0.29	28.89	0.30	90.69	0.28	401.62
	2000	**100**	0.05	0.10	0.32	21.7	0.39	69.91	0.32	321.84
		200	0.04	0.59	0.34	21.88	0.38	60.85	0.35	315.69
R100	**500**	**100**	0.43	4.61	0.40	21.81	0.38	45.23	0.42	276.37
		200	0.45	1.43	0.39	16.29	0.42	45.77	0.39	266.63
	2000	**100**	0.45	0.67	0.39	27.47	0.43	54.63	0.41	281.85
		200	0.22	0.67	0.45	27.47	0.28	54.63	0.42	285.70
RC100	**500**	**100**	0.06	0.65	0.30	27.47	0.33	54.63	0.30	286.38
		200	0.17	0.88	0.30	24.81	0.25	59.38	0.21	290.64
	2000	**100**	0.08	1.40	0.34	24.27	0.30	62.48	0.24	321.86
		200	0.11	0.73	0.20	18.85	0.30	62.48	0.33	315.71

Table 3: The metrics C and M