

NEW APPROACHES FOR DETERMINING GREENEST PATHS AND EFFICIENT  
VEHICLE ROUTES ON TRANSPORTATION NETWORKS

by  
UMMAN MAHİR YILDIRIM

Submitted to the Graduate School of Engineering and Natural Sciences  
in partial fulfillment of  
the requirements for the degree of  
Doctor of Philosophy

Sabancı University  
Spring 2014


NEW APPROACHES FOR DETERMINING GREENEST PATHS AND EFFICIENT  
VEHICLE ROUTES ON TRANSPORTATION NETWORKS

APPROVED BY

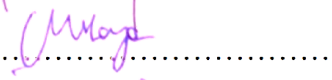
Assoc. Prof. Dr. Bülent Çatay  
(Thesis Supervisor)



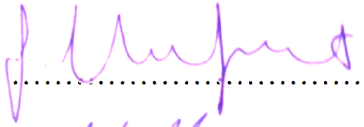
Assist. Prof. Dr. Hüsnu Yenigün



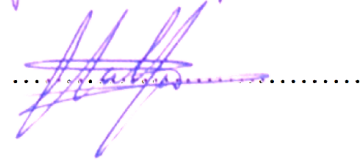
Assist. Prof. Dr. Maria Battarra



Assist. Prof. Dr. Murat Kaya



Assoc. Prof. Dr. Tonguç Ünlüyurt



DATE OF APPROVAL 06.08.2014

© Umman Mahir Yıldırım 2014

All Rights Reserved

# NEW APPROACHES FOR DETERMINING GREENEST PATHS AND EFFICIENT VEHICLE ROUTES ON TRANSPORTATION NETWORKS

Umman Mahir Yıldırım

PhD Thesis, 2014

Assoc. Prof. Bülent Çatay

Keywords: *vehicle routing, matheuristic, time-dependent, greenest path, network-consistent speeds*

## **Abstract**

Road transportation has hazardous and threatening impacts on the environment. However, the traditional logistics models and approaches used in transportation planning have mainly focused on minimizing the internal costs and lack the environmental aspect. Therefore, new planning techniques and approaches are needed in road transport by explicitly accounting for these negative impacts.

In this thesis, we address these issues by first concentrating on solution methods for the Greenest Path Problem (GPP) where fuel consumption and GHG emission objectives are incorporated to find the least GHG generating path, namely the greenest path, and propose a fast and effective heuristic. Taking the strong relation between the speed and the GHG emission into account, we also address the speed embedded minimum cost path problem in the most general case where the speed is also a decision variable as well as the departure time. Within this context, we develop a new network-consistent (which implies spatially and temporally consistent speeds) time-dependent speed and travel time layer generation scheme since real data is difficult to acquire. In the second part, we mainly focus on Vehicle Routing Problems (VRP). First, we propose an Ant Colony Optimization (ACO) approach for solving the Vehicle Routing



Problem with Time Windows (VRPTW). Then, we adapt this method to solve the environment friendly VRP, namely the Green VRP, where the greenest paths between all customer pairs are used as input. Finally, we extend the ACO algorithm to a parallel matheuristic approach for solving a class of VRP variants.

ULAŞIM AĞLARI ÜZERİNDE EN YEŞİL YOLUN VE ETKİN ARAÇ  
ROTALARININ BULUNMASI İÇİN YENİ YAKLAŞIMLAR

Umman Mahir Yıldırım

Doktor Tezi, 2014

Doç. Dr. Bülent Çatay

Anahtar Kelimeler: *araç rotalama, matesezgisel, zaman bağımlı, yeşil yol, ağ tutarlı hızlar*

**Özet**

Karayolu ulaşımının çevreyi tehdit eden tehlikeli etkileri bulunmaktadır. Ancak, ulaşım planlamasında kullanılagelen planlama yöntem ve yaklaşımları esas olarak içsel maliyetleri en aza indirmeye odaklanmakta ve çevresel bir bakış açısı barındırmamaktadır. Bu nedenle karayolu taşımacılığında, bu olumsuz etkileri dikkate alan yeni planlama teknikleri ve yaklaşımlarına ihtiyaç vardır.

Bu konuları ele almak için bu tezde, öncelikle en az sera gazı salımı (SGS) yaratan yolu bulmak amacıyla yakıt tüketimi ve SGS'yi göz önünde bulunduran En Yeşil Yol Problemi üzerinde yoğunlaşmış; hızlı ve etkin bir sezgisel yöntem sunulmuştur. Ayrıca, hız ve SGS arasındaki kuvvetli ilişki dikkate alınarak, en genel haliyle yola çıkış zamanı yanında hızın da bir karar değişkeni olduğu, hız içeren en az maliyetli yol problemi ele alınmıştır. Bu kapsamda, gerçek bir ağ için hız verisini elde etmek zor olduğundan, ağ-tutarlı (konum ve zaman bakımından tutarlı olan hızları içeren) zaman-bağımlı hız ve yolculuk zamanı katmanı yaratan yeni bir yöntem geliştirilmiştir. İkinci bölümde, esas olarak Araç Rotalama Problemi (ARP) üzerinde odaklanılmıştır. Öncelikle Zaman Pencereci Araç Rotalama Problemi (ZPARP) için bir Karınca Kolonisi Algoritması (KKA) sunulmuştur. Bu algoritma ayrıca, tüm müşteri çiftleri arasındaki en yeşil yolları girdi olarak kullanan Yeşil ARP'yi çözmek için

uyarlanmıřtır. Sunulan KKA son olarak, farklı ARP sınıflarını da çözecek řekilde, matematiksel model ile sezgisel metodu birleřtiren paralel bir matesezgisel yöntem olarak geniřletilmiřtir.

*To my family*

## Acknowledgements

I would like to express my gratitude to my advisor, Prof. Bülent Çatay, for his enthusiastic supervision, invaluable guidance and undeniable help during my graduate education, teaching assistantship and thesis process. He has always been complaisant and his support has been vital for the existence of this thesis.

I am also grateful for the help of Prof. Maria Battarra and Prof. Hüsni Yenigün who has always been so helpful and encouraging. I also owe thanks to the committee members Prof. Murat Kaya and Prof. Tonguç Ünlüyurt for their valuable comments. Moreover, I am thankful to Prof. Ş.İlker Birbil for his cordial attitude on all occasions and Prof. Kerem Bülbül from whom it was always a pleasure to learn.

Deepest thanks to Gülnur Kocapınar for her unique company, assistance and generous support. Without her help, this process would be much harder. She embellished my life with her entity. Moreover, I have always felt the support of my dear friends Nurşen Aydın with whom we have spent numerous sleepless nights coding, writing and chatting; Dr. L.Taner Tunç who always made my day with all his joy; and Dr. İbrahim Muter, a great friend and mentor. I would further like to thank to Sinem Taş for always being there when in need, Semih Atakan for all his projects, Birce Tezel for her warm greeting whenever we met, Halil Şen for his technological mentoring and always supporting me throughout all the office work, Mustafa Şahin for his modest comments, Eda Bilici for healthy snacks and manufacturing lab guys Utku Olgun, Alptunç Çomak and Recep Koca for all the laughter we shared. I will always be thankful to NC2 members Elif Özdemir Polat, Ezgi Yıldız, Gizem Kılıçaslan, Merve Şeker, Nimet Aksoy Tekmeci, Nükte Şahin and Dr. Semih Yalçındağ at every step of my life for their perfect personality and support. I give my sincere thanks to Anıl Can, Arda Şişbot, Belma Yelbay, Birgül Salıhoğlu, Burcu Atay, Ceyda Sol, A.Çetin Suyabatmaz, Emre Özlü, Esmâ Baytok, Fardin Dashty Saridarq, Figen Öztoprak, Gizem Çavuşlar, Merve Keskin, Ömer M. Özkırmımlı, Özge Arabacı, Özlem Çoban, Selma Yılmaz, Ümmühan Akbay, Volkan Aran, Yaşar Tüzel, and many other 1021 fellows.

Special thanks go to Serkan Çiftlikli, Ayfer Başar, Ece Erkol, Nihan Özşamlı and who made my days in Sabancı University so wonderful that I did not hesitate any minute to go on for a Ph.D., and Kubilay Küpeli who never lost his faith in me.

I am grateful also to my friends Çağrı Koç, Qazi Waheed-Uz-Zaman, Sarah Leidner, Jun Neoh, Philip Le, Joana C. Viana and Joe Zhao; “Team 4046” in short, and

Murat Oğuz. My 5.5 months stay at the Management School at the University of Southampton, UK, was in every respect wonderful. I also wish to thank Prof. Tolga Bektaş for his support and supervising and for his patience with me when I could not work on our project adequately because of the work involved in finishing this thesis.

Last but not least, I would like to express my dearest appreciation to my family for supporting me in every possible way. They have always made me to feel comfortable about my decisions.

I would like to also thank to the Scientific and Technological Research Council of Turkey (TÜBİTAK) for their financial support for my 5.5-month visit to UK as a researcher and İstanbul Transportation Communication and Security Technologies INC. (ISBAK) for providing speed data for research purposes.

## TABLE OF CONTENTS

Abstract.....	iv
Özet.....	vi
1 INTRODUCTION .....	1
2 A FAST ALGORITHM FOR FINDING THE GREENEST PATH.....	5
2.1 Introduction.....	6
2.2 Literature.....	7
2.3 Preliminaries .....	11
2.3.1 FIFO Property and Cost Consistency .....	11
2.3.2 Waiting/Non-waiting cases.....	14
2.3.3 Time-Space Network .....	15
2.4 Problem Description .....	16
2.5 Solution Methods on the TDMCP .....	21
2.5.1 Heuristics .....	21
2.5.1.1 Wen et al. (2014).....	21
2.5.1.2 The Greenest Path Algorithm.....	22
2.5.2 Exact methods under the discretization scheme .....	25
2.5.2.1 Decreasing Order of Time Algorithm of Chabini (1998).....	25
2.5.2.2 Greenest Path Algorithm.....	27
2.6 Computational Study.....	29
2.6.1 Synthetic and Real Networks Used in the Study .....	29
2.6.1.1 Synthetic Network Generation .....	29
2.6.1.2 Real Road Network .....	31
2.6.2 Cycles in TDMCP networks.....	31
2.6.3 Effect of Upper Bounds .....	34
2.6.4 Comparing Heuristics .....	36
2.7 Conclusion and Future Research.....	41
2.8 References.....	42
3 CREATING NETWORK-CONSISTENT SPEEDS ON TIME-DEPENDENT NETWORKS .....	45
3.1 Introduction.....	46
3.2 Implications on a Real Road Network and Congestion Circles.....	46

3.3	A New Network-Consistent Congestion Generation Scheme .....	51
3.4	Creating Time-Dependency on a Sample Large Scale Real Time-Independent Network.....	55
3.5	Conclusion and Future Research.....	57
3.6	References.....	57
4	EXACT AND HEURISTIC METHODS FOR THE TIME-DEPENDENT MINIMUM COST PATH PROBLEM WITH SPEED (TDMCP-S).....	58
4.1	Introduction.....	59
4.2	Relevant Literature.....	60
4.3	Problem Definition and Formulations for the TDSPP .....	63
4.4	Problem Definition and Formulation for the TDMCP-S .....	67
4.5	Time-space-speed expansion .....	71
4.6	Computational Study.....	72
4.6.1	Computational setup .....	73
4.6.2	Comparison of the proposed methods .....	73
4.7	Conclusion and Future Research.....	74
4.8	References.....	74
5	A TIME-BASED PHEROMONE APPROACH FOR THE ANT SYSTEM.....	76
6	GREEN VEHICLE ROUTING .....	97
6.1	Introduction.....	98
6.2	Literature.....	99
6.3	Computational Study.....	103
6.4	Conclusion and Future Research.....	105
6.5	References.....	105
7	A PARALLEL MATHEURISTIC FOR SOLVING THE VEHICLE ROUTING PROBLEMS .....	107
8	COMPUTATIONAL RESULTS ON A CLASS OF VRP USING THE PARALLEL MATHEURISTIC.....	122
8.1	Introduction.....	123
8.2	A brief description of the VRP variants considered .....	123
8.3	The proposed method.....	124
8.4	Computational Study.....	126
8.4.1	Comparing with the best known/optimal solutions .....	127
8.4.1.1	CVRP.....	127



8.4.1.2 OVRP .....	127
8.4.1.3 MDVRP.....	128
8.4.1.4 HVRP .....	128
8.4.1.5 VRPTW .....	131
8.4.2 Comparison of SP and SC formulations .....	133
8.5 Conclusion and Future Research.....	134
8.6 Appendix. New best solutions .....	135
8.7 References .....	136
9 CONCLUSION.....	138

# 1 INTRODUCTION

Transportation activities are a serious threat to the environment due to the pollution and Greenhouse Gas (GHG) emissions, especially carbon dioxide (CO<sub>2</sub>) emissions. Besides having differences from region to region, 21% of the global CO<sub>2</sub> emissions are caused by freight distribution and passenger transportation operations (Gorham, 2002). According to the European Environment Agency (EEA), transportation is the second-largest source of GHG particularly in the form of CO<sub>2</sub> and nitrous oxide. Among transportation activities, road transport is the largest emission source with a rate of 92% (EEA, 2003). In the EU, 44% of goods are transported by trucks. In addition, road transport is expected to grow 33% in the next 20 years. So, road transportation will be the fastest growing source of GHG (European Union, 2014). Despite all the technological advances, these numbers clearly reveal the need to take action against the adverse effects of road transportation on the environment.

Under the previously mentioned conditions where modern societies are more concerned with the environment and governments have started adopting new environmental regulations to reduce emissions and fuel consumptions, companies are obliged to take into account the external costs of their logistics activities caused by factors such as the climate change, GHG, environmental pollution, and noise. Therefore, new planning techniques and approaches for road transportation that take these negative impacts into consideration are needed. However, traditional logistics models and approaches used in transportation planning have mainly focused on minimizing the effect on the environment. The literature on these problems is scarce and the solution approaches lacking the environmental perspective fail to solve them.

Motivated by the aforementioned topics and related gaps in the literature, this thesis tackles these issues by first focusing on solution methods for the GPP (minimum cost path problem in the most general case). The results of the proposed solution method are used as data for the environment friendly Vehicle Routing Problem (VRP), namely the Green VRP. To solve the Green VRP, we use an adaptation of the Ant Colony Optimization (ACO) approach which we propose for solving the VRP with time windows. Finally, we extend the ACO algorithm to a parallel metaheuristic approach for solving a class of VRP variants. The thesis is organized as follows.

In Chapter 2, we address the GPP which is an extension of the well-known Time-Dependent Shortest Path Problem (TDSPP). In TDSPP, the costs of the edges on the

network vary with time, which may be predicted based on historical data. In general, travel time is considered as cost and the objective function becomes finding the fastest path in this case. However, in the GPP, the objective is to find the minimum fuel consuming/greenhouse gas (GHG) emitting path. Even though many concepts in TDSPP minimizing the travel time (fastest path problem) are also applicable to GPP, the peculiar nature of problem makes the traditional fastest/shortest path methods incapable of solving it. In this chapter, we first carry out a comprehensive review of the TDSPP literature and discuss some basic concepts that are also common for the GPP. After analysing the characteristics of GPP, we propose a fast and effective heuristic method for determining time-varying the greenest paths. The developed heuristic is applied to different scenarios to determine potential savings and sustainability benefits and compared with the current algorithms in the literature.

Chapter 3 presents a new network-consistent time-dependent speed and travel time layer generation scheme. In spite of the increasing interest in the literature on the time-dependent routing and path finding problems, the performances of the algorithms are usually evaluated on user-created instances due to the lack of a publicly available real-world road network with time-dependent arc costs and speeds. Most of the studies either modify the readily available time-independent road network to incorporate time-dependency by randomly generating travel time data that are not network-consistent or use limited real data, where available. Here, network consistency implies spatially and temporally consistent speeds. In addition, the increasing trend in green routing and path finding literature where the cost is also based on the speed of the travel, justifies the need for a network- consistent speed layer along with the travel time. Our proposed scheme can be used as a basis for what-if, scenario and vulnerability analyses using synthetic and real topological data as well as for routing and path finding problems on real road networks. We use the proposed method to generate test beds for the models and algorithms presented in Chapter 2, Chapter 4 and Chapter 6.

Chapter 4 extends our work in Chapter 2 to the Speed Embedded Time-Dependent Minimum Cost Path Problem where the departure times from the nodes and the speeds on the arcs are also decision variables. One of the main factors that affect the objective function in the GPP and the Green VRP is the speed of the vehicle during its journey. The carbon emission is a convex function of the speed of the vehicle, i.e. it decreases with the increasing speed to a certain minimum speed from which it increases exponentially. Yet, the most common assumption with the current time-dependent

routing and path finding algorithms is taking the speed as fixed during the corresponding time interval. The path is traversed with the maximum speed available and these values are taken as an input regardless of the emission created by the specific speed value. However, relaxing this assumption can reduce GHG emissions significantly. Qian and Eglese (2014) reported that about 6-7% savings in fuel emissions could be achieved by adjusting the speed values. In this chapter, we first develop mathematical models for the problem. We further propose a time-space-speed expansion and compare the performance of the models and the proposed expansion approach.

In Chapter 5, we introduce a new ACO approach for solving the VRP variants involving customers requiring service during specific time-windows. The proposed method, namely TbAS, is a time-based pheromone approach for the Ant System (AS) which is the first ACO approach developed for solving the Traveling Salesman Problem (TSP) (Dorigo et al., 1996). The novelty of the TbAS comes from introducing a new multi-layer pheromone network structure. The method is tested on the VRP with time windows (VRPTW), and high quality solutions are obtained. This chapter has been published as Yildirim and Çatay (2012).

In Chapter 6, we combine our work in the previous chapters to solve the Green VRP which aims to generate routes in such a way that the total GHG emission is minimized while using the 1-to-1 GHG minimizing path information. Here, the term “green” is independent of the type of the vehicle and hence, does not refer to any kind of low carbon emitting vehicles such as electric vehicles. We obtain the 1-to-1 minimum GHG generating paths using the algorithm presented in Chapter 2 as an input and adapt TbAS introduced in Chapter 5 to solve the problem. We also show the benefits of using time-dependent information on the network.

Chapter 7 proposes a matheuristic method, namely MathAnt, for solving the VRPs in a parallel manner. A matheuristic can be defined as any heuristic that utilizes mathematical programming in one of its solution steps (Bertazzi and Speranza, 2012). In our study, we combine the ACO metaheuristic (specifically TbAS) with the Set Partitioning formulation of the VRP and present our results. This chapter has been presented in the 16<sup>th</sup> meeting of the Association of European Operational Research Societies (EURO) Working Group on Transportation and published as Yildirim and Çatay (2014). Finally, Chapter 7 extends MathAnt to solve different VRP variants. The

approach is promising as it was able to improve some of the best-known solutions from the literature.

## **2 A FAST ALGORITHM FOR FINDING THE GREENEST PATH**

## 2.1 Introduction

Road movements are very important in the freight distribution (collection and delivery) and passenger transportation operations in both urban and rural areas and have significant economic and societal impacts. Transportation also has hazardous and threatening impacts on the environment: resource consumption, land use, toxic effects on ecosystems and humans, noise, and the effect induced by accidents and GHG emissions as such. Among these, GHG, especially CO<sub>2</sub> emissions are the most concerning since they have direct consequences on human health, such as pollution, and indirect ones, such as the depletion of the ozone layer and global warming. Consequently, new planning techniques and approaches are needed in road transport by explicitly accounting for these negative impacts because of the growing concerns about the hazardous effects of transportation on the environment. Research in this direction has been recently gaining momentum in the developed countries and modern societies.

The shortest path problem (SPP) has been extensively studied in the literature with numerous extensions and variations (interested reader is referred to Deo and Pang (1984) and Delling et al. (2009) for a detailed survey on SPP). The dynamic SPP is such an extension which gained momentum with the improvements in the GIS technology. There exist two main types of dynamic SPP's. In the first the network is subject to instantaneous and unpredictable partial changes; hence, the shortest paths need to be recomputed (Pallottino and Scutellà, 2003). In the second, namely the time-dependent SPP (TDSPP), the characteristics of the network may be predicted using past data. In this study, we focus on the latter case where the objective is to find the minimum cost paths on a network with time-dependent travel costs, that is, the cost of the travel depends on the time of the departure. These time-dependent cost functions are also referred to as delay functions.

Greenest path problem (GPP) further extends TDSPP such that the objective is to find the minimum fuel consuming/greenhouse gas (GHG) emitting path. Many concepts in the fastest path problem are also applicable to GPP. Yet, the solution methods for the fastest/shortest path problem cannot be used to solve the GPP due to the distinctive properties of the problem.

In a network where the travel speeds (times) are constant, the shortest path between two nodes can be easily found using the Dijkstra's Algorithm (DA) (Dijkstra, 1959), a label-setting algorithm which finds the shortest paths from a source node to all

other nodes in a network with nonnegative arc lengths. Using the same algorithm, it is also possible to find the fastest path between two nodes in a transportation network with variable travel speeds. The fastest path found in the morning rush hours will differ from the fastest path found in the noon when the traffic density is relatively low. So, the fuel consumption and GHG emissions will also be different in these two time intervals. However, it is not possible to find the least fuel consumption or GHG emission yielding path, namely the *greenest path*, using DA or any shortest/fastest path algorithm. Thus, a dynamic network structure and a method that finds the greenest route between the nodes in this network are required.

With this motivation, we propose a fast heuristic for finding the greenest path on a time-dependent road network with time varying speeds for which the traditional path finding algorithms do not work. The rest of the chapter is organized as follows; the next section provides a comprehensive review of the TDSPP and the GPP. Section 2.3 presents a general overview of some preliminary basic concepts on the time-dependent networks. After describing the GPP and analyzing the cost structure with different scenarios in Section 2.4, the implementation details of the current and the proposed heuristics and exact methods as well as the optimality conditions are provided in Section 2.5. The computational results are presented in Section 2.6. In Section 2.7, we finally give the concluding remarks and the future research directions.

## 2.2 Literature

In this section, we present a comprehensive review of the TDSPP literature and the relatively scarce GPP literature. Since the label-constrained (where the sequence of the nodes on the path are subject to constraints (Sherali et al., 2003), quasi-dynamic (where the arc costs are assumed to change quickly and then remain invariant until next changes (Tian et al., 2009) and multi-criteria (where pareto optimal solutions are sought (Disser et al., 2008; Nielsen et al., 2009)) versions of TDSPP are out of the scope of this study, the related literature is excluded.

Cooke and Halsey (1966) is the first study on TDSPP. They proposed a recursive formula to find the fastest paths from all nodes to a given destination by modifying Bellman's label-correcting shortest path algorithm presented in Bellman (1958). They used a discrete time framework  $S_M = \{t_0, t_0 + \delta, t_0 + 2\delta, \dots, t_0 + M\delta\}$  with  $\delta$  being a positive time unit and  $M$  being the longest travel time of any arc at any time interval.



The travel times on the arcs are defined as multiples of  $\delta$ . Note that most of the studies evaluating the time as discrete variables use the same approach. Dreyfus (1969) proposed a label-setting method by generalizing DA. The approach has the same complexity as the static case if the fastest path between two nodes (1-to-1) for a given departure time is sought. However, finding the paths from all nodes to a given destination (all-to-1) for all times requires the same complexity as Cooke and Halsey (1966). Halpern (1977) extended DA by addressing the issue of limited waiting at nodes whenever it decreases the total travel time. The delay functions used are nonnegative and piecewise continuous.

The *FIFO* property assures that commodities travel along arcs in a First-In-First-Out manner. A dynamic network is said to be a *FIFO network* if all the arcs satisfy FIFO property, and it is *non-FIFO* otherwise. A detailed analysis of the FIFO property and its variations are provided in Section 2.3.1.

In the previous studies, FIFO property was assumed to hold implicitly. However, Kaufman and Smith (1990) showed a counter-example where the method of Dreyfus (1969) fails to find the least cost path and emphasized the necessity of the FIFO property for the principle of optimality. Without introducing the term FIFO, they exemplified the situation where a later departing driver *passes* the first one, yielding the violation of the optimality principle. In their subsequent study they proved that if all the arcs in a network satisfy the *non-passing property* (NPP), then any classical shortest path algorithm can successfully be applied to solve the TDSPP in polynomial time (Kaufman and Smith, 1993). However, their method to transform the network to a FIFO network via data modification brings an extra computational burden.

Besides the FIFO term, Orda and Rom (1990) and Orda and Rom (1991) introduced the waiting concept in the time-dependent context and analyzed three different network settings: (i) waiting is allowed anywhere on the network; (ii) waiting is only allowed at the origin node; and (iii) waiting is prohibited. Their approach identifies the optimal waiting durations in the first two cases and yet fails to find the best path in the latter case. They also demonstrate on a continuous-time instance that a path might consist of infinite number of arcs. Finally, they show that the computation of the shortest path on a non-FIFO network is NP-hard. A discrete counterpart of their approach for the non-waiting case was proposed by Ziliaskopoulos and Mahmassani (1993) which uses a label-correcting shortest path algorithm. Cai et al. (1997) also analyzed three waiting conditions; an unrestricted waiting is allowed at all nodes,

waiting is prohibited at all nodes and there is a node-dependent upper bound on the waiting time. They proposed three algorithms and solved the variants to optimality assuming strictly positive travel times.

The solution methodology in Sung et al. (2000) is basically a modified version of DA. The novelty comes from the new travel time calculation model, namely Flow Speed Model (FSM), where the flow speed on each arc depends on the time interval. FSM preserves the FIFO property. Horn (2000) compared the performance of several exact and approximation methods that are adapted from the methods in the literature. Ahuja et al. (2002) handled the problem in a street network regulated by traffic lights that are subject to periodicity. The total travel and waiting times are minimized. Ahuja et al. (2003) further extended the previous work by proving new complexity results. Dean (2004a) surveyed different waiting policies and speed up techniques for dynamic programming to obtain practical algorithms.

The A\*(A-star) algorithm (Hart et al., 1968) is an extension of DA that uses a heuristic function to estimate the distance and direct the search towards the sink node. Kanoulas et al. (2006) gave an extension of the A\* algorithm and proposed a new lower bound estimator for the travel time. They clustered the network into non-overlapping cells and then used the nodes on the boundaries to compute a tighter lower bound on the heuristic function of the travel time. The new estimator is compared to the Euclidean distance divided by the maximum speed and proved to outperform from the search space point of view.

Delling and Wagner (2007) and Nannicini et al. (2008) also adapted ALT, a variation of A\* based on landmarks, to the time-dependent scenario. Emphasizing the complicated nature of the bi-directional approach on time-dependent networks, they implement the proposed method in a unidirectional way. They used 24 different transit times for time-dependency, each representing an hour of the day. The position and number of the landmarks on the graph greatly affect the size of the search space. Thus, different heuristics for landmark selection are compared taking into account the trade-off between the preprocessing time and the quality of the landmark choice.

Dell'Amico et al. (2008) extended Dreyfus (1969) to general travel times and proposed a new Dijkstra-like algorithm. They handled discontinuities by assigning a flag to each travel time. These flags are used to distinguish the travel times calculated by using right or left limits. In addition, by fixing the start times on the nodes and allowing unrestricted waiting at the intermediate nodes they generalized the

backtracking procedure of Orda and Rom (1990). Applying a unidirectional search Delling (2008) presented an exact time-dependent technique that extends their previously developed algorithm, namely SHARC (Bauer and Delling, 2008). The generalization allows fast exact shortest path queries on time-dependent networks.

Sanders et al. (2009) analyzed contraction hierarchies. These hierarchies are simply constructed by contracting the nodes in their importance order, thus creating a more condensed network. In addition to speeding up the algorithm, this makes bidirectional approach available at the cost of higher space consumption (Dehne et al., 2012).

The first approach to use random time-dependent travel times rather than a deterministic framework was proposed by Hall (1986). They first showed that the standard shortest path algorithms fail in this setting. Then they combined branch-and-bound and  $K$ -shortest paths techniques to determine the earliest expected arrival time. Miller et al. (1994) provided an alternative to this approach by focusing on the least possible time. Wellman et al. (1995) further gave an approximation algorithm which produces optimal paths under time-dependent uncertain costs.

Many of the above studies solve 1-to-all shortest paths. However, they can be applied for solving all-to-1 SPP by small modifications or reversing the time (Dean, 2004b). Moreover, Daganzo (2002) showed the reversibility of the TDSPP. Delling and Wagner (2009) reviewed TDSPP giving the focus to FIFO networks. In addition, we refer to Dean (2004b) for a starting point on the efficient implementation of TDSPP algorithms.

Chabini (1998) is the first study which is capable of minimizing a generic cost function rather than the travel time on a time-dependent network. The study proposes an exact method under a certain discretization scheme. The time intervals are discretized into time points and a static network is obtained by using a time-space expansion. To find the minimum cost all-to-1 paths for all departure times, a backward labeling algorithm is implemented. The algorithm visits the entire time-space network. Note that Pallottino and Scutellà (1998) also present a similar chronological algorithmic paradigm called Chrono-SPT which visits only the non-redundant portion of the network. However, the objective function is the minimization of the travel time rather than a generic cost minimization.

To the best of our knowledge, Wen et al. (2014) is the only heuristic approach to find the minimum cost path on a time-dependent network. They proposed two Dijkstra-

based heuristic methods namely Heuristic 1 and Heuristic 2. Heuristic 1 first applies DA but, on the contrary, keeps all the labels at the intermediate nodes. Next, the potential labels that are likely to improve the total cost are identified. Then, DA is run starting from each potential label and the final solution is obtained by selecting the best of the solutions. Heuristic 2 extends DA by dividing the time horizon into time intervals and keeping the minimum cost label within each interval. Then, all of the labels corresponding to each time interval are carried to the adjacent nodes increasing the total number of labels dramatically. They suggested putting a limit on the total number of labels at each node to circumvent this computational burden.

The details of the algorithm proposed in Chabini (1998) and Heuristic 2 in Wen et al. (2014) will be given in Section 2.5. In what follows, we discuss the basic concepts in TDSPP which are also applicable to GPP.

## 2.3 Preliminaries

### 2.3.1 FIFO Property and Cost Consistency

The FIFO property can be formally defined as follows: for each time pair  $t$  and  $t'$  if  $t < t'$  implies  $d_{ij}(t) + t < d_{ij}(t') + t'$ , where  $d_{ij}(t)$  is the travel time of reaching node  $j$  departing from node  $i$  at time  $t$ , then FIFO is said to be valid on arc  $(i,j)$ . If all the arcs on a network satisfy the FIFO property, then the network is said to be a *FIFO-network*; and *non-FIFO* otherwise. Note that alternative terms such as *non-passing property* (NPP), *non-overtaking property* or *time-consistency* are also used in the literature.

There is an ambiguity in the literature concerning the definition of the FIFO property. On a real transportation network, the FIFO property should hold when the arcs along a single path are taken into account individually. However, multiple paths may exist between the origin and destination; and a vehicle departing later from the origin may arrive earlier at the destination node through a different path. This situation is reasonable and does not violate the FIFO property on the individual arcs. Nevertheless Orda and Rom (1990) considers this situation described in the example in Table 2.1 as a violation of the FIFO property.

Table 2.1. Examples provided in Sung et al. (2000)

Start Time	Path	Arrival Time	Travel Time
35	$o - a - b$	60	25
40	$o - b$	55	15

In this example  $o$ ,  $a$ , and  $b$  correspond to the nodes. The vehicle departing from node  $o$  at time 40 arrives at node  $b$  earlier than the vehicle departing from the same node at time 35 but going to node  $b$  passing through node  $a$ . Although the two vehicles use different paths, Sung et al. (2000) takes it as a violation of NPP.

In the stochastic framework, Wellman (1990) introduced the concept of *stochastic consistency* (SC) inspired from the FIFO property in deterministic case. SC simply states that one cannot improve the probability of arriving at a given time by leaving later.

Table 2.2 groups the deterministic TDSPP literature with respect to the FIFO property and waiting, the details of which is given in the next section. Some of those non-FIFO studies did not pay any attention to satisfy the FIFO property whereas others implicitly violated FIFO by using a frozen link model (Orda and Rom, 1990) which assumes a constant travel speed during a time interval, thus a fixed cost at the beginning of the travel.

Table 2.2. Classification of TDSPP literature with respect to the FIFO property and waiting

	PW	SoW	SuW	UW
FIFO	(Cooke and Halsey, 1966), (Kaufman and Smith, 1990), (Dreyfus, 1969), (Kaufman and Smith, 1993) (Horn, 2000), (Kanoulas et al., 2006), (Delling, 2008), (Dean, 2004b), (Sung et al., 2000), (Dell'Amico et al., 2008), (Dehne et al., 2012), (Miller et al., 1994), (Chabini, 1998)		(Cai et al., 1997)	(Cai et al., 1997), (Ahuja et al., 2002), (Ahuja et al., 2003), (Dean, 2004a), (Dell'Amico et al., 2008), (Wellman et al., 1995)
Non-FIFO	(Orda and Rom, 1990), (Ziliaskopoulos and Mahmassani, 1993), (Halpern, 1977), (Daganzo, 2002), (Delling and Wagner, 2009), (Hall, 1986)	(Orda and Rom, 1990), (Orda and Rom, 1991)		(Orda and Rom, 1990), (Orda and Rom, 1991)

Kaufman and Smith (1990) mentioned the possible violation of FIFO in situations where the speed is a function of the number of vehicles on the arc or the network includes traffic lights. They pointed out that the effect of the latter case is relatively small as it is hard to evaluate the exact moment at which the driver arrives at the traffic lights. On the contrary, the elastic link (Orda and Rom, 1990) or the flow speed (Horn, 2000; Sung et al., 2000; Ahuja et al., 2002; Ahuja et al., 2003; Dean, 2004a; Kanoulas et al., 2006) models consider the speed variations during a time interval, which leads to a more realistic approach on real transportation networks.

The cost counterpart of FIFO is referred to as *cost consistency* (CC) which states that leaving a node earlier cannot cost more than leaving it later (Pallottino and Scutellà, 1998). Pallottino and Scutellà (1998) separated the CC calculation on FIFO and non-FIFO arcs. Let  $c_{ij}(t_h)$  be the cost of the travel when node  $i$  is departed at time  $t_h$ ,  $w_j(t_z)$  be the unit waiting cost at node  $j$  at time  $t_z$  and  $t_u = t_h + d_{ij}(t_h)$ ,  $t_v = t_k + d_{ij}(t_k)$ . A FIFO arc  $(i, j)$  is also a CC arc if for any time pair  $t_h < t_k$ :

$$c_{ij}(t_h) + \sum_{z=u}^{v-1} w_j(t_z)(t_{z+1} - t_z) \leq c_{ij}(t_k), \quad (1)$$

whereas for a non-FIFO arc  $(i, j)$ , the condition to be satisfied is given as:

$$c_{ij}(t_h) \leq c_{ij}(t_k) + \sum_{z=v}^{u-1} w_j(t_z)(t_{z+1} - t_z). \quad (2)$$

The inequalities (1) and (2) evaluate the costs at the time of latest arrival. That is, the early arriving vehicle is also subject to waiting with time-dependent cost until the arrival of the other vehicle. On the other hand, Ziliaskopoulos (1994) evaluates the costs according to the time of the earliest departure. After the departure of the first vehicle, the late departing vehicle is subject to waiting cost. The CC condition is given as:

$$c_{ij}(t_h) \leq c_{ij}(t_k) + \sum_{z=h}^{k-1} w_i(t_z)(t_{z+1} - t_z). \quad (3)$$

In our study the CC concept is independent of the FIFO condition. We solely take into account the cost at the time of the departure, disregarding any waiting if allowed. So, an arc is said to be CC if for any time pair  $t_h < t_k$ :

$$C_i(t_h) + c_{ij}(t_h) \leq C_i(t_k) + c_{ij}(t_k), \quad (4)$$

where  $C_i(t_h)$  refers to the cumulative cost at node  $i$  at time  $t_h$ .

Pallottino and Scutellà (1998) stated that domination rules can be used on a FIFO-network which is also CC in order to decrease the labels under consideration. Among two labels the one with higher time and higher cost is dominated by the other. No such domination rules hold if the network is either FIFO or CC but not both. However, if the network on hand is a FIFO-network, than it exhibits several properties that can be exploited to improve the solution algorithms (Dean, 1999).

Finding the minimum-time path on a non-FIFO network is NP-hard (Orda and Rom, 1990) whereas the Bellman's principle of optimality (Bellman, 1958) is valid on a FIFO network and the fastest path can be found in polynomial time (Kaufman and Smith, 1993). This optimality principle is also referred to as the concatenation of paths. A shortest path is concatenated if each of its sub-paths is also a shortest path between the source and the intermediate nodes for the same starting time (Orda and Rom, 1990). GPP satisfies the FIFO property but CC may not hold. Thus, we may end up with nonconcatenated paths. Different examples will be presented in Section 2.4.

The minimum-cost path problem can also be shown to be NP-hard by a reduction from the constrained SPP even if all arc costs are static but only one has time-dependent cost (Dean, 2004b). Ahuja et al. (2002) discussed some polynomially solvable realistic special cases.

### 2.3.2 Waiting/Non-waiting cases

Waiting (or *parking*) and delaying the start of the travel may be advantageous from the (time-dependent) cost point of view. If the cost represents the travel time, waiting may be preferable only on non-FIFO networks. Same is valid for a generic cost function (not including waiting time) on non-CC networks. If permissible, visit times (time windows) for each node are introduced. In that case, waiting can be observed on FIFO or CC networks as well.

Note that a node on the network may refer to a road junction or a landmark where the speed changes (including the departure and arrival nodes). Keeping this in mind, there are four different cases concerning the waiting concept:

- *Prohibited waiting (PW)*: waiting is not allowed.
- *Source waiting (SoW)*: waiting is allowed only at the source node.

- *Subset waiting (SuW)*: waiting is allowed only at a subset of the nodes.
- *Unrestricted waiting (UW)*: waiting is allowed at any node.

If waiting is assigned a cost then a solution method that models unrestricted waiting can be used to solve the other three cases by setting the cost to infinity where waiting is not allowed.

Along the above mentioned location dependent waiting, the waiting cases can be further divided into sub-groups according to the duration of the waiting. Waiting in forms of parking may be restricted to a limited amount of time or may be prohibited in certain time intervals. Also when modeling, if a parking facility is located on an arc between two nodes, then the facility is introduced as a new node creating two new arcs (Halpern, 1977). The classification of the relevant literature is given in Table 2.2.

### 2.3.3 Time-Space Network

One approach to study the discrete dynamic shortest paths is to use the so-called *time-space network*, where the network comprises a node for each node-time unit pair in the original graph. Arcs are included taking the travel times into account. However, it is also possible to work directly on the given network by holding multiple cost labels on nodes (Ahuja et al., 2002). The latter alternative arises from practical issues. The main drawback of time-space networks is the increase in the input size. The new network is  $M + 1$  times larger than the original where  $M + 1$  is the number of discrete time points. This makes the time-space network impractical for large scale road networks (Delling et al., 2009). A sample network and its corresponding time-space network are given in Figure 2.1.a and Figure 2.1.b. The travel times on the arcs (A-B), (A-C) and (B-C) are set to 30, 20 and 10 minutes respectively.

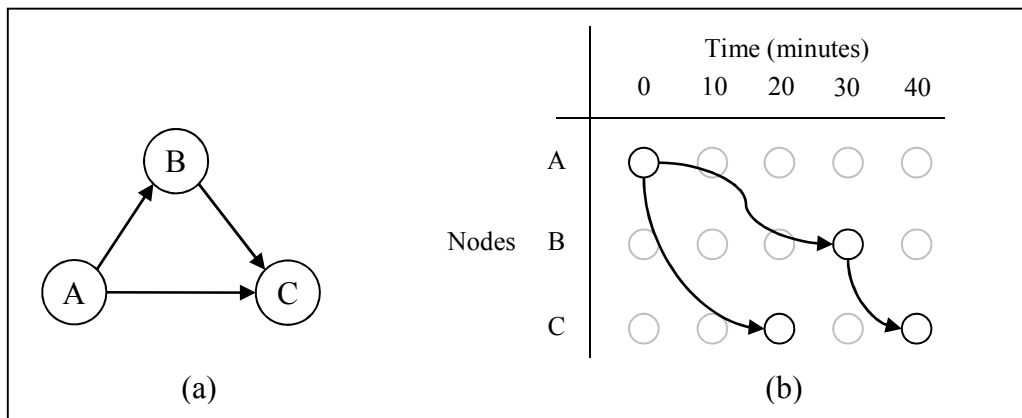


Figure 2.1. A sample network (a) and its corresponding time-space network (b)



## 2.4 Problem Description

The GPP problem can be defined on a time-dependent directed network with additional time-dependent cost  $G'(N, A, W, C)$ , where  $N = \{1, 2, \dots, n\}$  is the set of nodes,  $A \subseteq N \times N$  is the set of arcs,  $W$  and  $C$  are positive valued functions. Each arc  $(i, j) \in A$  is associated with travel time function  $w_{ij}(t) \in W$  and cost function  $c_{ij}(t) \in C$  which specify the travel time and the cost of the travel between  $i$  and  $j$  departing at time  $t$ , respectively, where  $t$  is a time variable in a time domain. Although we consider the greenhouse gas (GHG) emission of a vehicle as the cost of the travel within this context, the problem can be generalized for any non-negative cost function. Note that both  $w_{ij}(t)$  and  $c_{ij}(t)$  are equal to  $\infty$  (or a sufficiently large number)  $\forall (i, j) \notin A$  or  $\forall t \in (T, \infty)$  where  $T$  is an upper bound on the length of the planning horizon. This is a reasonable assumption as all the practical problems have an upper bound.

In addition, discontinuities concerning the cost function might be observed in real life applications. These might arise due to closure or opening of a section of the road in a certain time interval due to the regulations for the vehicle type or existence of traffic lights, especially in urban context. These finitely many points can also be modeled by setting the cost in the relevant non-overlapping intervals to infinity (Dell'Amico et al., 2008).

The European Environment Agency developed models to estimate the speed dependent fuel consumption (EMEP/CORINAIR, 2007). These models were used to obtain the fuel consumption curve for diesel-powered light commercial vehicles as depicted in Figure 2.2. Due to the structure of the cost function, the principle of optimality is not valid in GPP. So, the final greenest paths do not form a tree.

Let  $S$  denote the source node. A feasible path from  $S$  to  $k$  is an ordered set of nodes  $(S, n_1, n_2, \dots, k)$ . GPP seeks for the greenest path on  $G'$  from  $S$  to a particular node  $n \in N$ , assuming that we begin the trip at  $t \in [0, T]$ . Note that, by setting  $C$  equal to  $W$ , the GPP can be reduced to TDSPP which determines the fastest path between two predetermined nodes on  $G$ .

With every arc and time interval is an associated speed value. It may be advantageous to prefer a slower but more cost efficient speed rather than travelling at the maximum speed that the network topology permits. This will be further discussed in Chapter 4.

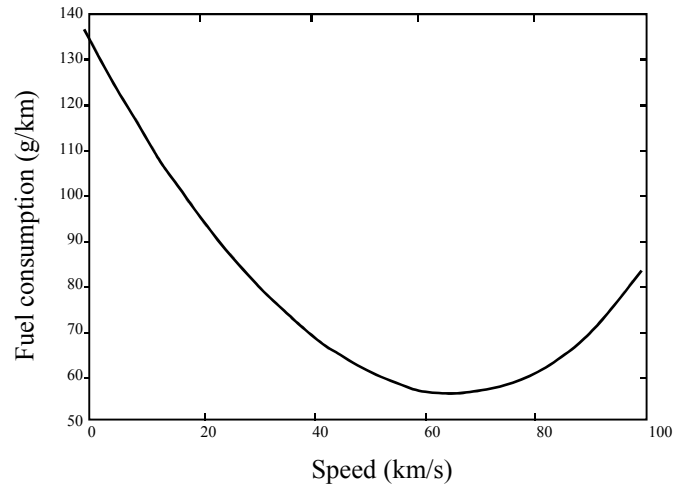


Figure 2.2. The relationship between speed and fuel consumption for a light duty diesel vehicle (Eglese and Black, 2010)

Another characteristic of the GPP that makes it hard to solve is that there is no direct correlation between the cumulative travel time and the GHG emission. In other words, no pattern exists for the greenest path. In the following, we will depict different scenarios to further analyze these characteristics of the problem. In each figure, the numbers in parentheses show the arc lengths whereas the numbers in brackets refer to the speed on an arc in the corresponding time interval. We assume that the GHG emission quantities at different speeds are as given in Table 2.3.

Table 2.3. GHG emissions at different speeds

Speed	20	30	40	50	60	70	80
GHG emission (g/km)	1.4	1.2	0.9	0.8	0.5	1.7	1.8

The planning horizon is divided into two equal intervals of length 1 time unit, namely  $Z_1 = [0,1]$ ,  $Z_2 = (1,2]$ . One can assume that  $Z_3 = (2, \infty)$  and the GHG emission corresponding to that time interval is equal to infinity.

Figure 2.3 and Table 2.4 give the details of scenario 1. Two alternative paths lead to internal node 2: path 1.1 visits node 1 whereas path 1.2 arrives at node 2 earlier but with a higher cost (GHG emission). However, as path 1.1 arrives later, the travel from intermediate node 2 to destination node 3 falls into the second time interval where the travel speed is less efficient. Thus, the path with higher cost and earlier time to node 2 results in less cost and earlier time when the arrival to node 3 is concerned.

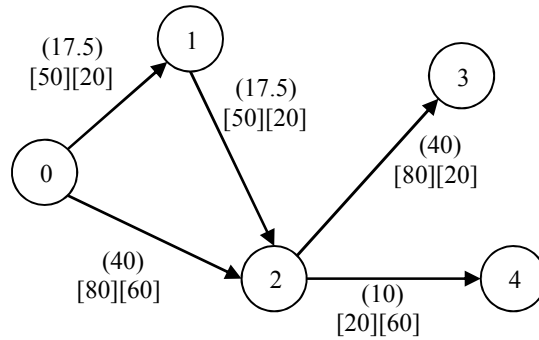


Figure 2.3. Sample network for scenario 1.

On the contrary, the travel speed in the second time interval on arc (2-4) is more efficient compared to that in the first interval. This makes the arrival within the second time interval more attractive. Accordingly, the final cost at the destination node 4 is lower on path 1.1.b. The path with lower cost yet later time in the previous stage yields lower cost and later arrival time at the destination.

Table 2.4. Calculations for scenario 1

Path	Detail	GHG emission	Time
1.1	0-1-2	28.0	0.70
1.2	0-2	32.0	0.50
1.1.a	0-1-2-3	69.6	1.80
1.2.a	0-2-3	64.0	1.00
1.1.b	0-1-2-4	38.4	1.07
1.2.b	0-2-4	46.0	1.00

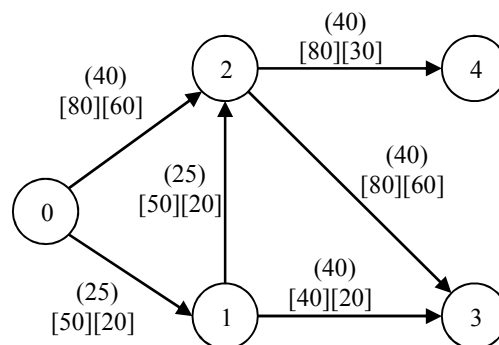


Figure 2.4. Sample network for scenario 2.

Scenario 2 is depicted in Figure 2.4 with calculations in Table 2.5. This scenario is similar to scenario 1 from the cost point of view. But in this case, path 2.1 arrives at the

intermediate node 2 later than path 2.2 due to its visit to node 1. As the speed in the second time interval is more efficient on arc (2,3), the cost at the destination node via path 2.1.a is less than the one via path 2.2.a. In summary, the path with higher cost and later time in the previous stage results in lower cost and later time at the destination.

Table 2.5. Calculations for scenario 2

Path	Detail	GHG emission	Time
2.1	0-1-2	40.0	1.00
2.2	0-2	32.0	0.50
2.1.a	0-1-2-3	60.0	1.67
2.2.a	0-2-3	64.0	1.00
2.1.b	0-1-2-4	88.0	1.75
2.2.b	0-2-4	64.0	1.00

Moreover, lower cost path at any intermediate node may still have lower cost at the destination. The destination node 4 has speed values of 80 and 30 in time intervals 1 and 2, respectively. This helps the early arriving path to preserve its lower cost relative to the other. In this case, the path with lower cost and earlier time in the previous stage gives lower cost and earlier time at the destination.

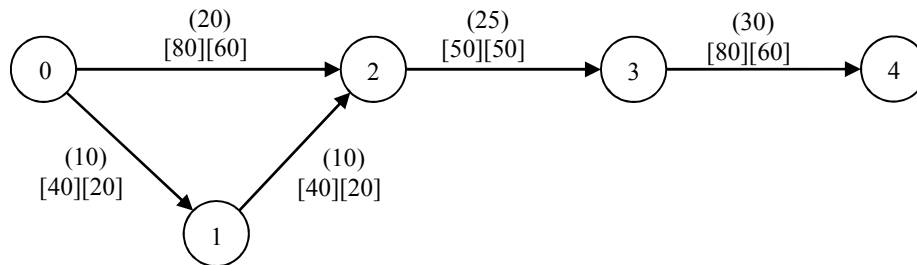


Figure 2.5. Sample network for scenario 3.

Scenario 3 in Figure 2.5 further extends the previous scenarios to illustrate that the gain may not be immediately observed on the following arc. Path 3.2 visits intermediate node 1 before visiting node 2. Then, both paths visit node 3 before reaching the destination node 4. The speed efficiency is the same in both intervals on the arc (2,3). The less efficient speed on the first interval of arc (3,4) causes path 3.1 to have higher cost, as given in Table 2.6. So, the path with higher cost and later time at two consecutive intermediate nodes results in lower cost and later time at the destination.

Table 2.6. Calculations for scenario 3

Path	Detail	GHG emission	Time
3.1	0-1	16.0	0.25
3.2	0-1-2	18.0	0.50
3.1	0-1-2	36.0	0.75
3.2	0-1-2-3	38.0	1.00
3.1	0-1-2-4	59.0	1.16
3.2	0-1-2-3-4	53.0	1.50

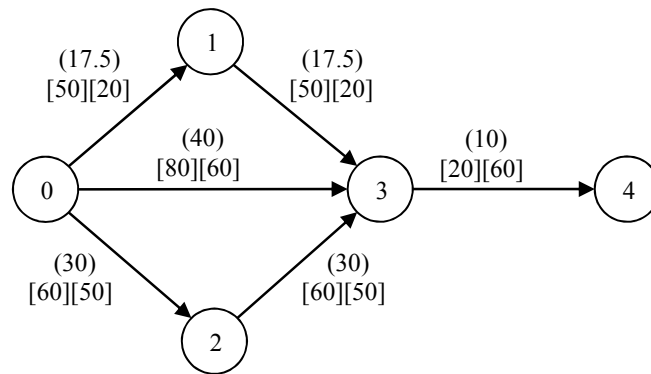


Figure 2.6. Sample network for scenario 4.

Finally, scenario 4 in Figure 2.6 and Table 2.7 shows that no pattern is valid in GPP and no dominance can be obtained between time-cost pairs. Among the three paths, the one whose cost is the second highest at intermediate node 3 gives the least cost path to destination node 4.

Table 2.7. Calculations for scenario 4

Path	Detail	GHG emission	Time
4.1	0-3	32.0	0.50
4.2	0-2-3	30.0	1.00
4.3	0-1-3	28.0	0.70
4.1	0-3-4	46.0	1.00
4.2	0-2-3-4	35.0	1.16
4.3	0-1-3-4	38.4	1.07

These scenarios also emphasize the fact that the optimal paths are not necessarily concatenated. In other words, the optimal paths do not form a tree. For example,

consider scenario 1.a where the optimal path to node 2 is (0-1-2) while the optimal path to node 4 is (0-2-4), not (0-1-2-4).

The scenarios are generated to exhibit the cost behavior of different cost-time combinations. For illustrative purpose, we utilized some extreme cases such as the speed jumping from 30 to 80 or vice versa from one time interval to another. Although such jumps can be observed in real life when the time intervals are long, smoother increments or decrements are more common. For a more realistic model, one may shorten the duration of the time intervals while using a discrete-time approach.

## **2.5 Solution Methods on the TDMCP**

In this section we first give a brief explanation of Wen et al. (2014). Then, we give implementation details of our proposed heuristic followed by the exact methods under the discretization scheme.

### **2.5.1 Heuristics**

#### **2.5.1.1 Wen et al. (2014)**

The Heuristic 2 method of Wen et al. (2014) is an extension of DA to the time-dependent case where the objective is to minimize a generic cost function. They keep the minimum cost label within each interval analogous to keeping the shortest distance label at each node in DA. When a node is selected from the queue, all the labels on all intervals are compared with the current labels on an adjacent node in the corresponding time interval. If the total cost of the current label carried to the adjacent node is less than the label at the adjacent node in the same time interval, the label on the adjacent is replaced. We refer this process as the evaluation of the labels.

Figure 2.7 gives the sample network in Wen et al. (2014) where the minimum cost path from node A to node E is sought. Evaluated labels are shown near the relevant nodes and the kept labels are given in bold. The numbers on the arcs represent the cost and the travel time (in minutes) in different time intervals. Each time interval spans a length of 60 minutes. That is, the first time interval limits are (0,60], the second time interval limits are (60,120] and so on. If a node is visited at different time intervals, then all the corresponding labels are kept at this node, unless the total number of labels has reached the maximum label limit. Accordingly, node C is visited from node A and B in intervals 1 and 2 respectively and thus, both of the labels C1 and C2 are kept and

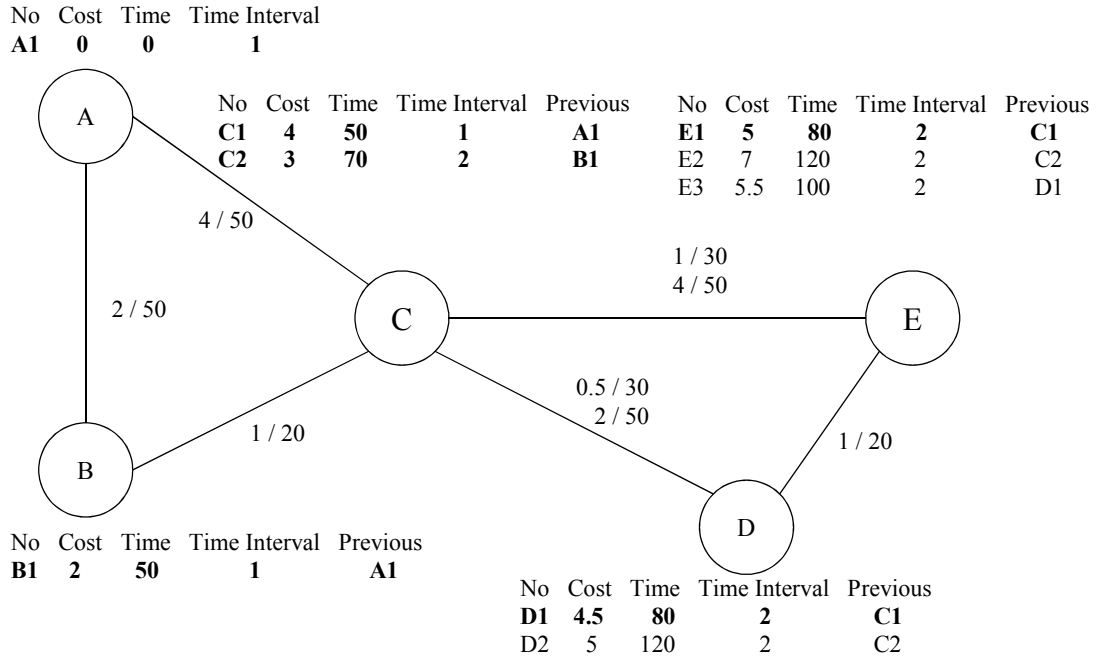


Figure 2.7. Example for Heuristic 2 in Wen et al. (2014).

carried to the adjacent node E. However, as the arrival times for both labels C1 and C2 at node E fall into the time interval 2, only the label with minimum cost, namely E1, is kept. Applying the same approach, only label D1 is kept at node D and carried to node E. The best solution can be obtained by comparing the costs of the labels on node E.

### 2.5.1.2 The Greenest Path Algorithm

The Greenest Path Algorithm (GPA), also built upon DA, finds minimum cost paths from the source node  $s$  to all other nodes in a network with nonnegative arc costs. Similar to Wen et al. (2014), the algorithm maintains a cost label  $c(i, t)$ , an upper bound on the minimum cost path length to node  $i$  arriving in time interval  $t$ , with each node  $i$  for every time interval in  $[0, T]$  where node  $i$  can be visited.

The main novelty with the algorithm comes from the usage of upper bounds to direct the search towards the sink node. We decrease the search space using upper bounds for the intermediate solutions. We implemented two kinds of bounds;

- i) On the actual cost (AC): If the cost of an intermediate solution is higher than the upper bound ( $UB^{AC}$ ), then the label is fathomed.
- ii) On the potential cost (PC): If the sum of the minimum possible cost of reaching from the current node to the sink node ( $LB^{PC}$ ) and the actual cost is higher than the upper bound, then the label is fathomed.

The upper bound can be found using a simple heuristic. In addition, to find the minimum possible cost at an intermediate node, a similar heuristic can be used. First the shortest path to the sink node is obtained. Then, the green cost of the path is calculated. As we must use estimation on the speed values, we assume that the entire path is traversed with the most efficient speed taking the GHG emissions into account.

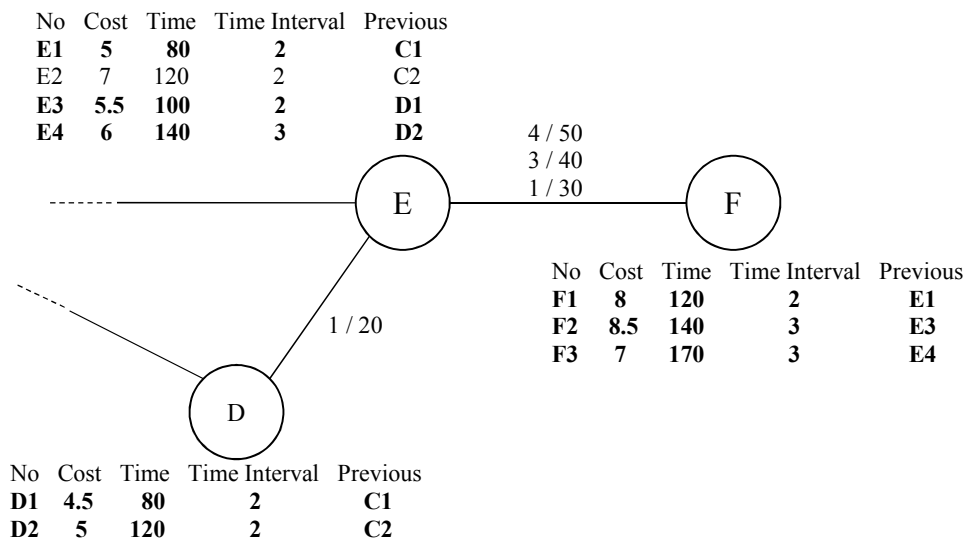


Figure 2.8. Example for multiple labels in a time interval

In Section 2.4, we show that no pattern exists for the greenest path problem. So, keeping only the minimum cost label may yield a suboptimal path whereas increasing the number of labels kept in a single time interval may improve the solution quality. We give an example in Figure 2.8 which shows a portion of the network in Figure 2.7 with the new added node F and the corresponding arc (E-F).

If we allow two labels at a node for the sample in Figure 2.8, also label D2 would be kept and carried to node E, arriving in time interval 3. Among the three labels in interval 2, label E2 that has the highest cost is eliminated. When we evaluate labels E1, E3 and E4 for node F, we observe that label E4 yields the optimal solution with cost 7. In case of keeping only a single label with the minimum cost, we would not obtain this solution but a suboptimal solution with cost 8. In this example, we adopt keeping the best two labels to deal with multiple labels in a single time interval. As an alternative, first two labels (in a FIFO manner) can be kept. In each case, a limit on the number of labels in a particular time interval can also be implemented to distribute the labels as smooth as possible over the planning horizon.



The pseudo code of the algorithm is given in Figure 2.9.  $L_i^t$  refers to the label at node  $i$  at time  $t$  where  $L_i^t.cost$  refers to the cost of that label.  $NL_i$  is the total number of labels at node  $i$ .  $Q$  is the queue where the labels that are to be evaluated are kept. When a label is pulled out of the queue, all of the labels on the corresponding node are evaluated. Thus, to prevent re-evaluation of the labels, we use an index for each node to keep record of the last evaluated label.

Also it is important to distinguish the terms “time bin” and “time interval”. Time bin is pertained to a time-dependent instance. It refers to a certain length of time which differs from others by a different speed value. On the other hand, being a parameter of the GPA, time interval is used to limit the number of labels that are kept in a certain length of time. Note that all the time bins and all the time intervals can be either equally distributed in length or otherwise among themselves.

---

**Algorithm 2.1.** Greenest Path

---

```

1   $L_i^t = \emptyset, NL_i = 0, \forall i \in N, \forall t \in T, Q = \{L_S^0\}$ ,
2  Solve Dijkstra to obtain  $UB^{AC}$ 
3  Solve Reverse Dijkstra to obtain  $LB^{PC}[i], i \in N$  array
4  Add  $L_S^0$  to  $Q$ 
5  while  $|Q| > 0$ 
6     $Q = Q - \{L_i^t\}$  where  $L_i^t.cost = \min\{L_j^t.cost : j \in N, t \in T\}$ 
7    for all  $(i, j) \in A$ 
8      if  $|NL_j| < M$ 
9        for every label  $L_i^t, \forall t \in T$ 
10         Calculate cost at  $j, C_j = L_i^t.cost + c_{ij}(t)$ 
11         if  $(C_j > UB^{AC}$  or  $C_j + LB^{PC}[j] > UB^{AC})$ 
12           continue
13         else if  $(L_j^{t+w_{ij}(t)})$  doesn't exist)
14           Set  $L_j^{t+w_{ij}(t)}$  as a new label with cost  $C_j$ 
15         else if  $(L_j^{t+w_{ij}(t)}.cost > C_j)$ 
16           Replace  $L_j^{t+w_{ij}(t)}$  with the new label with cost  $C_j$ 
17         endif
18       endfor
19     Endif
20   Endfor
21 Endwhile

```

---

Figure 2.9. Greenest path algorithm

## 2.5.2 Exact methods under the discretization scheme

To the best of our knowledge, no optimal solution methodology exists for GPA. In this section we briefly give details of two approaches that can be considered as exact methods only under a certain discretization scheme. The first method belongs to Chabini (1998). As the second method, we discuss optimality of GPA under certain assumptions.

### 2.5.2.1 Decreasing Order of Time Algorithm of Chabini (1998)

Chabini (1998) proposed a method based on backward (in time sense) labeling algorithm to minimize a generic cost function. In his Decreasing Order of Time Algorithm (DOT), to find the all-to-1 optimal paths for all departure times he implemented a discrete technique. Thus, to comply with this discrete nature of the algorithm, a *minimum time unit* (referred to as *MTU* from now on) should be defined and all time values should be a multiple of this unit. To prevent a travel time of zero (when the travel time is smaller than *MTU*), the values are rounded up such as;

$$t = \lceil t/MTU \rceil * MTU.$$

The method is mainly proposed for sufficiently small *MTU*. So, the optimal solution is valid only under this discretization scheme. There cannot be any time instant that is not a multiple of *MTU*. Thus, when we use bigger *MTU* values, the method becomes a heuristic. Note that the magnitude of the *MTU* directly affects the precision of the travel times hence the solution quality. However, we cannot claim that it is always inversely proportional to the solution quality. Following are 2 cases where a bigger *MTU* results in less and more cost respectively. Table 2.8 shows the calculations.

In the real scenario, the first time bin ends in the 55<sup>th</sup> second when the speed on the arc changes. The distance between the nodes is 2 kilometers. Two different *MTU* values that are tested are 5 and 50 seconds.

Case 1- Bigger *MTU*-lower cost: In this scenario the time bin limit remains the same for *MTU*=5 as rounding yields no change. Time bin limit is rounded up to 100 seconds for *MTU*=50. The speeds in the first and second time bin are 50 and 40 km/h respectively. When *MTU*=50, the travel in the first time bin (0-100 seconds) causes 267.53 g emission. The next time bin with a fuel less efficient speed causes 131.59 g emission

Table 2.8. Examining different *MTU* values

		Case 1		Case 2	
		<i>MTU</i> =50	<i>MTU</i> =5	<i>MTU</i> =50	<i>MTU</i> =5
Travel in the 1 <sup>st</sup> time bin	Start Time (s)	0.00	0.00	0.00	0.00
	Speed (km/h)	50	50	40	40
	End Time (s)	100.00	55.00	100.00	55.00
	Distance (km)	1.39	0.76	1.11	0.61
	Travel Time (s)	100.00	55.00	100.00	55.00
	Cost (gr)*	267.53	147.14	239.01	131.35
Travel in the 2 <sup>nd</sup> time bin	Start Time (s)	100.00	55.00	100.00	55.00
	Speed (km/h)	40	40	50	50
	End Time (s)	155.00	166.25	164.00	155.00
	Distance (km)	0.61	1.24	0.89	1.39
	Travel Time (s)	55.00	111.25	64.00	100.00
	Cost (gr)*	131.59	266.16	171.44	267.75
TOTAL	Travel Time (s)	155.00	166.25	164.00	155.00
	Cost (g)*	399.12	413.30	410.44	399.10

\* Emission costs for speed values of 40 and 50 km/h are 215.32 and 192.63 g/km respectively.

making a total of 399.12 g. On the other hand when *MTU*=5, first and second time bin costs sum up to 413.30 g, 14.18 g higher compared to *MTU*=50. Rounding the limit of the first time bin (with a more fuel-efficient speed) up to 100 seconds increases the travel time and so the distance traveled in the first time bin. Having traveled longer distance with a more fuel-efficient speed yields a lower cost.

Case 2 - Bigger *MTU*-higher cost: On the contrary to the first case, the first time bin has a less fuel-efficient speed with 50 km/h vice versa 40 km/h in the second time bin. As a result, rounding the time bin limit up to 100 seconds causes traveling longer with a less fuel-efficient speed. So the total cost when *MTU*=50 is 11.34 g higher than the total cost when *MTU*=5.

As we investigate only a single arc in these examples, we cannot make any comment on the performance on a path. However, as *MTU* decision affects the costs, it is clear that the next node to visit decision will also change. Despite the decreasing precision, one advantage of using a larger *MTU* is the less amount of computational burden. For a planning horizon of 8 hours (28,800 seconds) on a 100-node time-dependent network, *MTU*=5 seconds and *MTU*=1 minute result in 576,000 (28,800/5\*100) and 48,000 (28,800/5\*100) computations respectively.

Having all-to-1 paths for all departure times at the end of the algorithm may seem advantageous. Nevertheless, when only the minimum cost path starting from a single

node is sought, this information becomes redundant. In addition, on a real network where the number of nodes may reach up to millions these calculations become computationally intractable. Wen et al. (2014) showed that for large datasets, DOT is less accurate and less effective compared to their proposed heuristics. Besides, they observed that with the increasing size of the network and using a sufficiently small time unit for discretization, memory problems are inevitable when applying DOT.

### 2.5.2.2 Greenest Path Algorithm

When the limits on the total number of labels and the number of labels in time intervals are relaxed, that is, both are set to infinity, GPA becomes an optimal algorithm for the GPP.

We first define the notation used throughout the proof. A label is a pair  $(i, t)$  where  $i \in N$  and  $t = [0, T]$ . Other notation used is as follows;

$S$  : set of permanent labels

$\bar{S}$  : set of temporary labels

$c(i, t)$  : the cost of reaching node  $i$  at time  $t = [0, T]$ ,  $c(i, t) = \infty \forall t > T$

$A(i)$  : adjacency list of node  $i$

$d(i, j, t)$  : arrival time at node  $j$  when leaving node  $i$  at time  $t$

$F(i, k, t)$  : cost of traveling from node  $i$  to node  $k$  leaving node  $i$  at time  $t$

$pred(i, t)$  : predecessor of node  $i$  when reached at time  $t$

*Proposition:* GPA is optimal without any label limits

*Proof:* (following DA's optimality proof) We use inductive arguments to establish the validity of GPA. At any iteration, the algorithm has partitioned the labels into two sets,  $S$  and  $\bar{S}$ . Our induction hypotheses are as follows: (i) the cost value of each label in  $S$  is optimal, and (ii) the cost value of each label in  $\bar{S}$  is the cost of the minimum cost path from the source provided that each internal node in the path lies in  $S$ . We perform induction on the cardinality of the set  $S$ .

To prove the first inductive hypothesis, recall that at each iteration the algorithm transfers a label  $(i, t)$  in the set  $\bar{S}$  with smallest cost value to the set  $S$ . We need to show that the cost value  $c(i, t)$  of label  $(i, t)$  is optimal. Notice that by our induction hypothesis,  $c(i, t)$  is the cost of the minimum cost path to node  $i$  at time  $t$  among all paths that reach node  $i$  at time  $t$  and do not contain any label in  $\bar{S}$  as an internal label. We now show that the cost of any path from the source node  $s$  to  $i$  that reaches at time  $t$

and contains some labels in  $\bar{S}$  as an internal label will be at least  $c(i, t)$ . Consider any path  $P$  from the source to node  $i$  (reaching at time  $t$ ) that contains at least one label in  $\bar{S}$  as an internal label. The path  $P$  can be decomposed into two segments  $P_1$  and  $P_2$ : the path segment  $P_1$  does not contain any label in  $\bar{S}$  as an internal label, but terminates at a label  $(k, t')$ ,  $t' < t$  in  $\bar{S}$  (see Figure 2.10). By the induction hypothesis, the cost of the path  $P_1$  is at least  $c(k, t')$  and since label  $(i, t)$  is the label with the minimum cost in  $\bar{S}$ ,  $c(k, t') \geq c(i, t)$ . Therefore, the path segment  $P_1$  has cost at least  $c(i, t)$ . Furthermore, since all arc costs are nonnegative, the length of the path segment  $P_2$  is nonnegative. Consequently, cost of the path  $P$  is at least  $c(i, t)$ . This result establishes the fact that  $c(i, t)$  is the cost of the minimum cost path of reaching node  $i$  at time  $t$  from the source node.

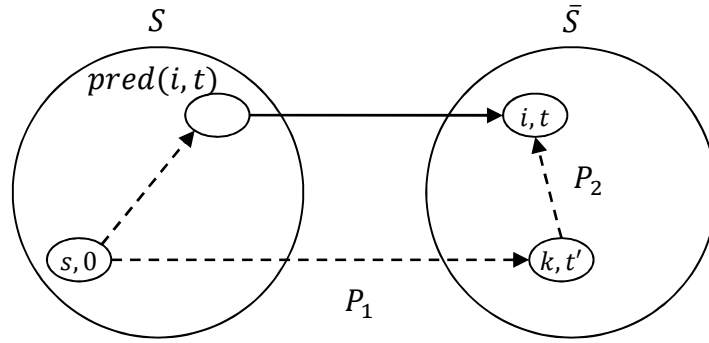


Figure 2.10. Proving GPA is optimal

We next show that the algorithm preserves the second induction hypothesis. After the algorithm has moved a label  $(i, t)$  to set  $S$ , the costs of some labels in  $\bar{S} - \{(i, t)\}$  might decrease, because label  $(i, t)$  could become an internal label in the tentative minimum cost paths to these nodes. But recall that after moving a label  $(i, t)$  to set  $S$ , the algorithm examines each node  $k \in A(i)$  and if  $c(k, d(i, k, t)) > c(i, t) + F(i, k, t)$ , then it sets  $c(k, d(i, k, t)) = c(i, t) + F(i, k, t)$  and  $pred(k, d(i, k, t)) = i$ . Therefore, after the cost update operation, by the induction hypothesis the path from node  $j$  to the source node defined by the predecessor indices satisfies the following property: a directed path  $P$  from the source node to node  $k$  reaching at time  $t$  is a minimum cost path if and only if  $c(j, d(i, j, t')) = c(i, t') + F(i, j, t') \forall (i, j) \in P$  where the vector  $d$  represents the optimal path costs. So the cost of each label in  $\bar{S} - \{(i, t)\}$  is the cost of

the path subject to the restriction that each internal node in the path must belong to  $S \cup \{(i, t)\}$ . □.

In the optimal version of GPA, there is a huge memory requirement as the number of labels a single node may have is not limited. Also the number of labels on a node increase as one advance along the path to the sink.

## 2.6 Computational Study

In this section, we first give details of the networks used in this study. Then, we analyze the effect of cycles and the proposed upper bounds. We next test the effect of keeping more than one label in a single time interval. Finally, we compare the performance of the proposed method with Wen et al. (2014). We prohibit waiting at all nodes.

The cost (emission rate in this study) is directly related with the speed. However, the method is not affected by the type of the cost function. In order to obtain the rate of emission ( $\varepsilon$ ) per kilometer with different speeds, we use the fuel consumption function of Hickman et al. (1999);

$$\varepsilon = 0.0617v^2 - 7.8227v + 429.51$$

where  $\varepsilon$  is the rate of emissions (g/km) for an unloaded goods vehicle on a road with a zero gradient and  $v$  is the average speed of the vehicle (km/h).

The algorithms are coded in C# programming language and executed on an Intel Xeon 3.30 GHz computer with 32.0 GB RAM and 64-bit operating system.

### 2.6.1 Synthetic and Real Networks Used in the Study

For our computational experiments, we use synthetic grid-type network instances as well as the real road network of Washington D.C. The details of congestion generation on both networks will be given in Chapter 3.

#### 2.6.1.1 Synthetic Network Generation

The main parameter in these instances is  $N$  which corresponds to the size of the grid. As a result, the total number of nodes is obtained as  $N^2$ . A sample unidirectional grid-type network is given in Figure 2.11.

For the same network structure, the direction of travel can be controlled by the adjacency lists. For a unidirectional travel between two nodes, say from node  $i$  to node

$j$ , node  $j$  should be included in the adjacency list of node  $i$  and not vice versa. Placing node  $i$  in the adjacency list of node  $j$  makes the arc  $i - j$  bidirectional also enabling travel from node  $j$  to node  $i$ . Note that this may lead to cycles. Also, networks with different density levels can be created using the same node set. Figure 2.12 shows a 4-node sample network with four different density levels used in this study. The first three density levels are set in a uni-directional network setting whereas density level 4

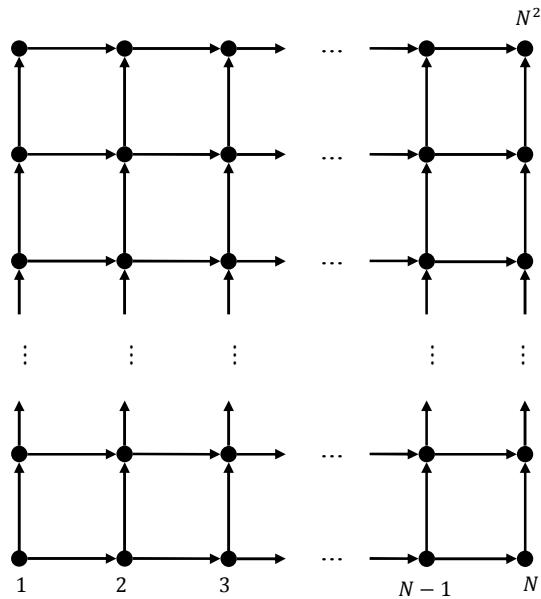


Figure 2.11. A sample unidirectional grid-type network

corresponds to a bi-directional network. The total number of arcs are  $2N(N - 1)$ ,  $(3N - 1)(N - 1)$ ,  $(4N - 2)(N - 1)$  and  $4N(N - 1)$  for density levels 1, 2, 3 and 4 respectively.

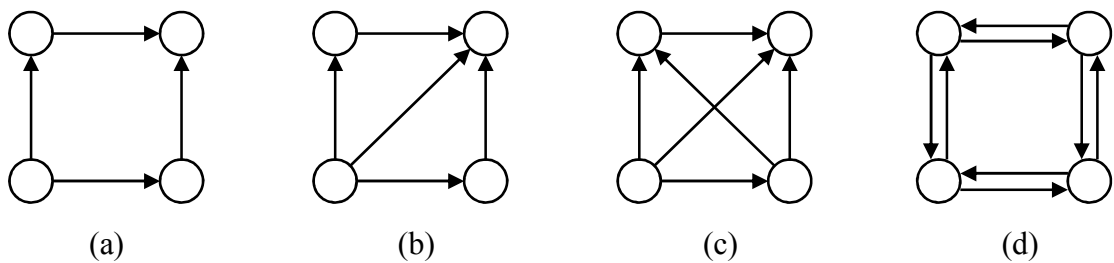


Figure 2.12. Different density levels on a sample 4-node sample network; level 1 (a), level 2 (b), level 3 (c) and level 4 (d).

The source and destination nodes are set as 1 and  $N^2$ . The length of each arc is randomly generated between 100.0 m and 2.0 km. As the size of the network changes, using a fixed planning horizon is unreasonable. Thus, in order to prevent infeasibility, the length of the planning horizon ( $T$ ) is taken as the time to traverse the shortest path between the source and destination nodes with a speed of 5 km/h.

### 2.6.1.2 Real Road Network

We also use the undirected Washington DC network data of Topologically Integrated Geographic Encoding and Referencing (TIGER) as test bed. The data includes 9,559 nodes and 14,909 arcs. The distribution of the nodes are shown in Figure 2.13.

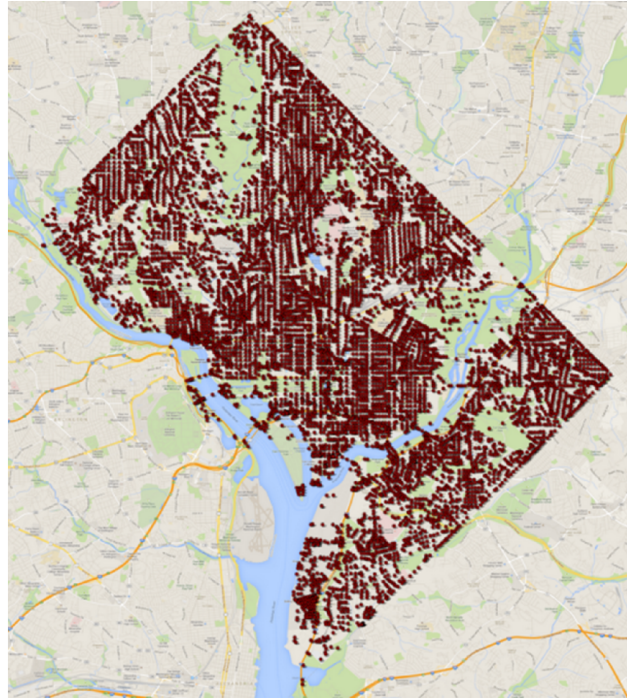


Figure 2.13. The distribution of the nodes of the Washington DC TIGER data

### 2.6.2 Cycles in TDMCP networks

A real road network has cycles due to the bidirectional arcs. Figure 2.14 shows the road network of the European side of İstanbul including only the highways. Even with the reduced network and prohibiting U-turns, we have nine cycles each identified with a number in parentheses. Note that we can also obtain cycles of larger sizes by taking a combination of these cycles.

In particular, we analyze the cycle that composes the nodes  $A$ ,  $B$  and  $C$ , starts from node  $A$  and the direction of travel is from node  $A$  to node  $B$ . The length of the



cycle is 12.2 km where the length of each arcs is shown in brackets in Figure 2.14. For ease of computation, we assume that we travel with a fixed speed of 61 km/h and our time interval length is 12 minutes (i.e. the interval limits are  $[0,12)$ ,  $[12,24)$ , and so on). As the total time to traverse the path  $A-B-C$  is also 12 minutes, a new label will be added for each node at each tour. Table 2.9 shows the details of the labels in the first two cycling tours.

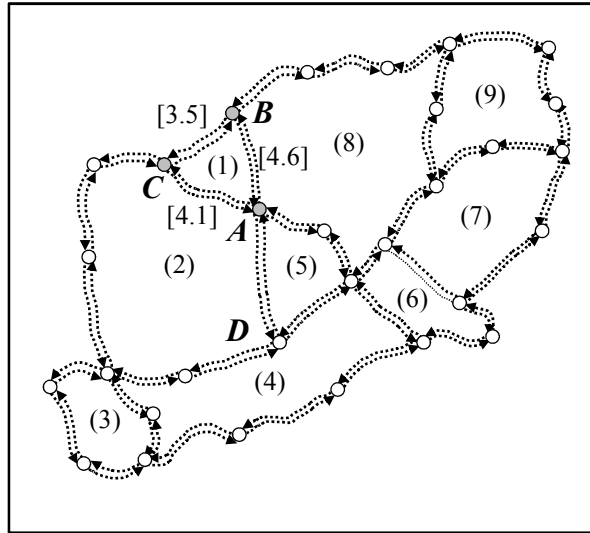


Figure 2.14. Sample network for cycle analysis

Table 2.9. Label details for cycle example

Label	CO <sub>2</sub> emission (kg)	Time	Time Interval	Previous Label
A1	0.00	0.00	1	-
B1	0.84	4.52	1	A1
C1	1.47	7.97	1	B1
A2	2.22	12.00	2	C1
B2	3.06	16.52	2	A2
C2	3.69	19.97	2	B2

Even if we keep only a single label in each time interval, all of these labels in Table 2.9 will be kept. Limiting the total number of labels as in Wen et al. (2014) or using upper bounds on the total cost as we propose will implicitly eliminate the cycles after a certain point where the total number of the labels has reached the limit or the total cost exceeds the upper bound respectively.

It is obvious that when waiting at nodes is allowed, cycling is never advantageous. Assuming no waiting, cycling may help to traverse the same arc with a more fuel efficient speed. As an example, suppose that the arc  $A-D$  is heavily congested between

7:00 and 9:00 due to the morning rush hour. Also suppose that the average speed is fixed at 30 km/h until the end of the rush hour exactly when the average speed starts to increase while the corresponding GHG emission decreases. A vehicle arriving at node  $A$  at 8:53 and traveling to node  $D$  can directly traverse the arc  $A-D$ . In addition, it can first traverse the cycle  $A-B-C-A$  in 7.97 minutes (from Table 2.9) and traverse arc  $A-D$  with a more fuel efficient speed. In fact, after every traversal of the cycle  $A-B-C-A$ , the speed on arc  $A-D$  increase to a more fuel efficient speed until it reaches the optimum speed after when the traversal cost of arc  $A-D$  starts to increase. Nevertheless, every traversal of the cycle also increases the total cost. So, it is questionable if cycling ever becomes advantageous taking the total cost into account.

In the best possible scenario that favors cycling in Figure 2.15, the speed on the cycle is  $v^*$  which is the least emission generating speed whereas the speed on the first road segment is  $v < v^*$ . After traversing the cycle, the speed on the road segment increases while the cost of the corresponding speed decreases by  $\Delta c$ . Thus, cycling is advantageous only when  $x C(v^*) < d \Delta c$  where  $C(v)$  refers to the per km cost of speed  $v$ . In other words, the total cost decrease on direct travel must be larger than the cost of cycling. Taking real speed and cost functions into account, we set  $v^*$  to 63.31 km/h which is the optimal speed for the convex cost function of Hickman et al. (1999). We set the distances  $x$  and  $d$  as 10 km and 5 km respectively. Analyzing the real data in Section 2.6.1, we observe that the largest speed increase rate is 5.94 (km/h)/min. Again for the best possible scenario, we take the speed of direct travel as 20 km/h whereas it increases to 48.14 km/h until the cycle is traversed. So, the final costs of cycling and direct travel is 2875.28 g and 2979.50g respectively. This shows that cycling, in theory, can reduce the total cost.

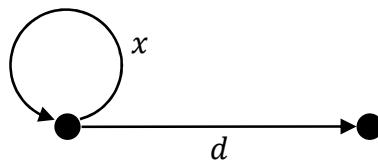


Figure 2.15. Effect of cycling on single arc

In spite of its theoretical gain, cycling might increase the number of labels kept without guaranteeing any improvement on the solution. Yet, in order to prevent the cycles, a specific mechanism should be implemented. One trivial solution is to check

the whole path each time when a node will be added. Taking the size of the real road networks into account where the number of nodes on a path can easily reach thousands, this approach is very time consuming and impractical. Instead, a predetermined number ( $\theta$ ) of the previous nodes can be checked to prevent cycles up to a size of  $\theta$ . We test the effect of the size of  $\theta$  on a sample network with grid size equal to 100 where  $\theta$  varies between 0 and 200. Here,  $\theta = 0$  means that no cycle check is implemented. The average computational time over 30 runs are summarized in Figure 2.16. The computational time increases with the increasing value of  $\theta$ . We also observed that only 4.14% of the labels include a cycle.

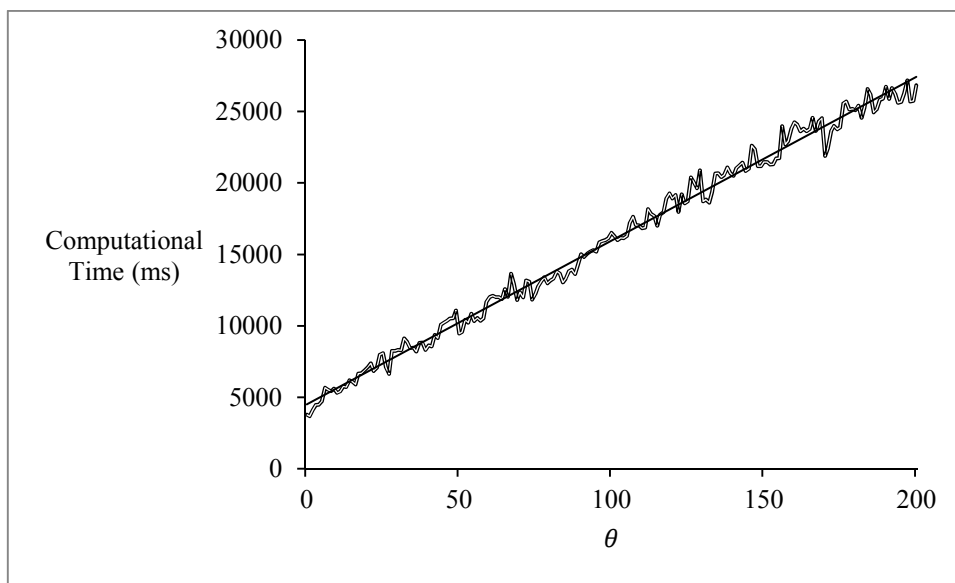


Figure 2.16. Effect of  $\theta$  on computational time

Due to the theoretical gain by cycling, the computational burden to check for the cycles and the low rate of the cycles that are observed, we do not implement a cycle check in our algorithm. Also, the usage of upper bounds further decreases the possibility of cycling indirectly with the help of decreasing number of labels as it will be noted in the next section.

### 2.6.3 Effect of Upper Bounds

In this section, we test the two bounds proposed in Section 2.4.  $N$  is set to 100 and the upper bound is calculated using DA. The minimum possible cost at an intermediate node is found by first solving a reverse DA, and then calculating the green cost of the path assuming that the entire path is traversed with the most efficient speed.

Table 2.10. Speeding-up the algorithm using upper bounds

	Upper Bound			
	None	AC	PC	AC & PC
Computational Time (s)	94.8	37.5	6.3	6.1
Emission (g)	14,790.7	14,790.7	14,790.7	14,790.7
Number of labels	7,592,858.3	2,222,351.7	283,153.4	283,153.4

Four different configurations are tested which are summarized in Table 2.10. The second column gives the average results of 5 runs for the algorithm without using any bound. Third and the fourth column show the results of the bounds on the actual cost (AC) and the potential cost (PC) respectively whereas the last column gives the result when using both of the bounds. To best analyze the effect of the upper bounds, the number of labels is not limited in this test. Also note that, all settings perform the same from emissions point of view as no limit exists on the number of labels.

Using AC reduces the computational time by 60.4% and the number of labels by 70.7% while keeping the cost at the same level. On the other hand, using PC dramatically decreases the number of labels, hence the computational time, by 96.3% and 93.3% respectively. In addition, using AC along with PC does not further contribute to the number of labels but slightly decreases the computational time. Thus, we use both of the bounds.

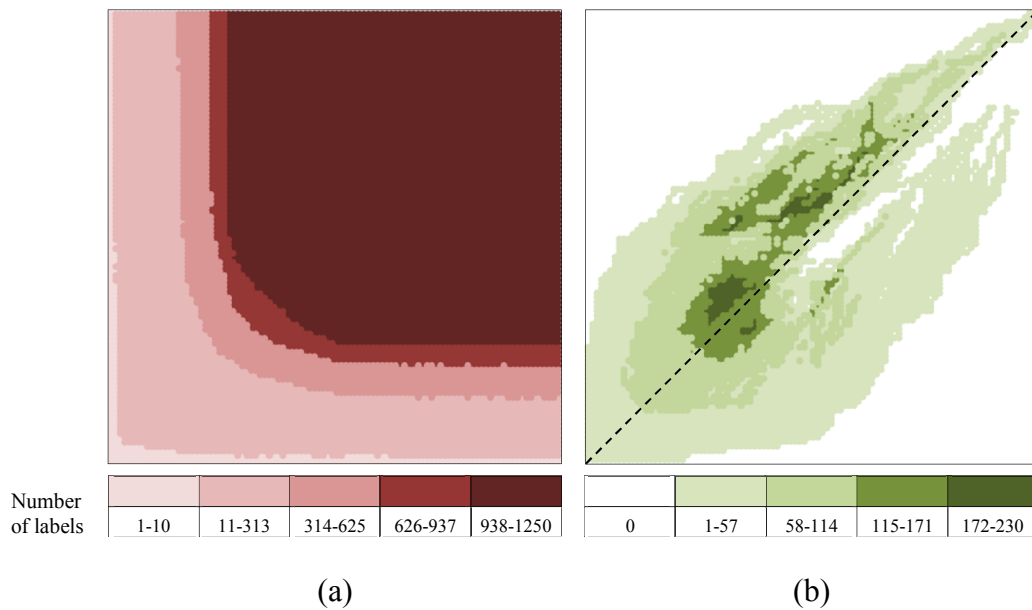


Figure 2.17. Distribution of the number of labels

Figure 2.17 shows the distribution of the number of labels on the network without using any bound and with using both bounds in (a) and (b) respectively. The minimum cost path from the node at the south-west corner to the node at the north-east corner is sought in each. Towards the north-east corner, the possible time intervals to visit increase as it is impossible to visit some nodes early in the horizon. Thus, the number of the labels increases towards the sink node in a nearly circular way centered at the sink node. But, when we analyze the distribution of the labels when using PC and AC bounds together, we see a pattern directed to the sink node. The number of the labels located near the dashed direct line from the source the sink node (Figure 2.17.b) is relatively high while labels kept decrease as we move away from the line. No labels are kept for the nodes in the north-east and south-west region.

It is clear that the usage of a bound decreases the number of labels and prevents decentralized exploration while directing the search towards the sink node. In addition, this decrease in the number of labels helps to decrease the computational effort. Thus, in the remaining experiments, both PC and AC bounds will be used.

#### 2.6.4 Comparing Heuristics

Chen et al. (2007) state that when the graph is sparse (i.e.,  $m = O(n)$ , which is the case for most of the road networks), implementing DA using a heap that supports only *Insert* and *Delete-Minimum* operations runs significantly faster than any implementation that also incorporates the *Decrease-Key* operation. They also note that this performance gap narrows as the graph becomes denser. Keeping this in mind, we used a basic heap implementation for our proposed method as well as Wen et al. (2014), both of which are based on Dijkstra's method.

We set  $N=100$  and test four different density schemes given in Section 2.6.1.1. In other words, all the instances have 10,000 nodes whereas the total numbers of arcs are 19,800, 29,601, 39,402 and 39,600 for density levels 1, 2, 3 and 4 respectively. For the number of labels, we use 1, 10, 100 and 1,000. The lengths of the time intervals are set to 30, 60 and 600 seconds. Although using multiple labels in a single time interval can yield better solutions in theory (Figure 2.8), we did not observe such a pattern in our preliminary tests. Thus, we keep a single label in each time interval.

Table 2.11. Comparison for instances with density = 1.

Time Interval Length		Label limit							
		1		10		100		1000	
		WÇE	GPA	WÇE	GPA	WÇE	GPA	WÇE	GPA
30	OF	26.890	25.393	26.451	25.261	25.871	24.736	24.818	24.637
	CT	3.4	3.2	4.5	3.5	14.8	8.8	78.5	13.1
60	OF	26.890	25.393	26.186	25.164	25.230	24.667	24.637	24.637
	CT	1.5	1.7	2.5	2.3	8.5	4.2	28.8	4.7
600	OF	26.890	25.393	24.940	24.659	24.637	24.637	24.637	24.637
	CT	0.2	0.2	0.5	0.3	1.1	0.3	1.0	0.3

Table 2.11, Table 2.12, Table 2.13 and Table 2.14 compare the objective function (OF) in kg and computational time (CT) in seconds of Wen et al. (2014) (WÇE) and GPA for instances with density 1, 2, 3 and 4 respectively. The objective function increases with the increasing shortest path distance, though they are not always directly proportional. The best average objective function value obtained for instances with density=1 is 24,637 whereas it is 14,820 for instances with density=2 where shorter paths can be obtained through the diagonal arcs.

Table 2.12. Comparison for instances with density = 2.

Time Interval Length		Label limit							
		1		10		100		1000	
		WÇE	GPA	WÇE	GPA	WÇE	GPA	WÇE	GPA
30	OF	16.596	15.129	16.406	15.129	15.867	14.931	15.083	14.820
	CT	2.0	1.8	2.8	2.0	9.9	5.7	64.6	7.6
60	OF	16.596	15.129	16.379	15.048	15.696	14.820	14.820	14.820
	CT	0.8	1.0	1.4	1.0	6.4	3.0	34.9	2.8
600	OF	16.596	15.129	15.095	14.820	14.820	14.820	14.820	14.820
	CT	0.1	0.1	0.5	0.3	2.1	0.2	2.0	0.2

In all instances, the computational time decreases with the increasing time interval length and decreasing label limit. When the label limit is set to 1, both algorithms become insensitive to the changing values of time interval length.

Table 2.13. Comparison for instances with density = 3.

Time Interval Length		Label limit							
		1		10		100		1000	
		WÇE	GPA	WÇE	GPA	WÇE	GPA	WÇE	GPA
30	OF	16.371	14.571	16.074	14.571	15.118	13.818	14.155	13.795
	CT	1.8	1.9	2.4	1.9	9.5	8.7	82.1	13.2
60	OF	16.371	14.571	15.797	14.527	14.535	13.795	13.795	13.795
	CT	0.9	0.8	1.2	1.2	6.6	3.9	52.9	4.2
600	OF	16.371	14.571	13.825	13.795	13.795	13.795	13.795	13.795
	CT	0.1	0.1	0.5	0.3	2.8	0.3	2.7	0.3

GPA performs better or matches the performance of WÇE in all four density settings taking the objective function into account except when density=1, time interval length=30, label limit=10; density=4, time interval length=30, label limit=1 and time interval length=600, label limit=10. From the computational effort point of view, GPA performs better with the increasing label limit. Nevertheless, WÇE can generate a solution in shorter time in some cases, especially when density=4. When time interval=30 and label limit=100, WÇE and GPA evaluates 1 million and 844,827 labels respectively. In this case, the gain obtained from the elimination of labels using upper bounds is surpassed by the computational effort used for checking the cost of the labels against the upper bounds. When the label limit is increased to 1,000, WÇE uses nearly all the label capacity (10 millions) whereas the total number of labels found by GPA increase only by 18%, hence the lower computational times.

Table 2.14. Comparison for instances with density = 4.

Time Interval Length		Label limit							
		1		10		100		1000	
		WÇE	GPA	WÇE	GPA	WÇE	GPA	WÇE	GPA
30	OF	27.257	27.303	27.206	27.206	26.382	25.514	25.670	25.165
	CT	3.3	3.9	4.6	4.5	17.0	24.8	180.0	82.7
60	OF	27.257	27.303	26.750	26.585	26.007	25.240	25.475	25.165
	CT	1.6	1.9	2.5	2.7	11.3	14.3	107.8	24.2
600	OF	27.257	27.303	25.166	25.192	25.165	25.165	25.165	25.165
	CT	0.2	0.2	0.6	0.9	6.2	1.3	8.2	1.0

The shortest path obtained by DA and the greenest path obtained by GPA for different source-sink pairs are depicted in Figure 2.18 and Figure 2.19. Three different congestion levels are shown with congestion level 3 being the highest level. The shortest path goes through the highly congested region. However, the path obtained by the GPA travels around the congested area to escape the arcs that are congested with 2 and 3 congestion levels. Nevertheless, in a setting where the free flow speed is set to 90 km/h, escaping congestion completely may also cause higher emission costs. Thus, the greenest paths in these figures are also exposed to congestion with level 1 where more fuel efficient speeds are used.

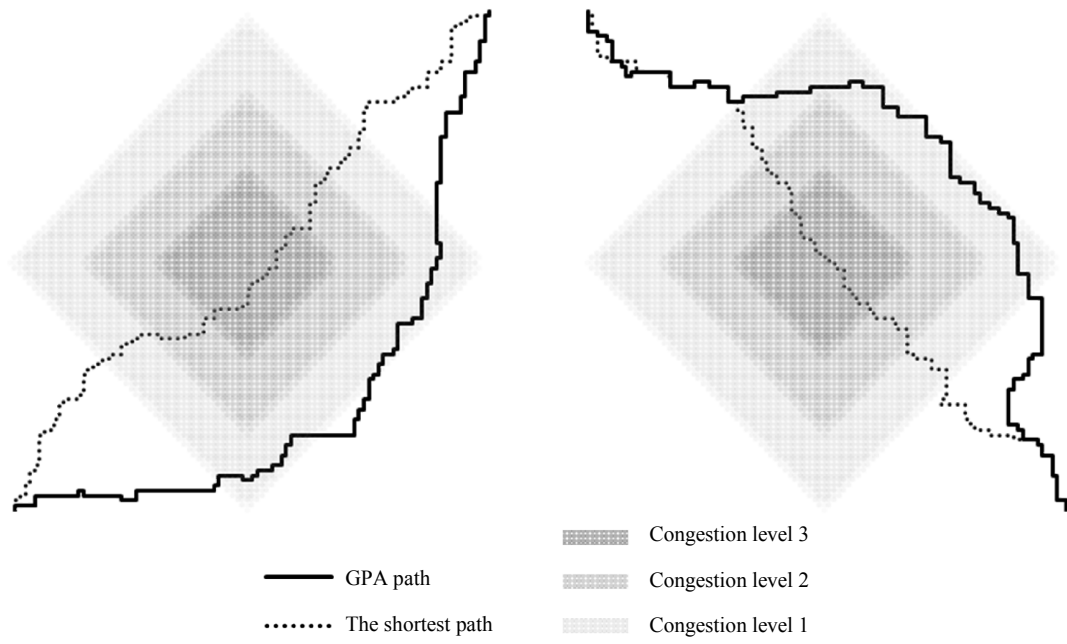


Figure 2.18. An illustration comparing GPA path and the shortest path on two synthetic data samples 1 and 2

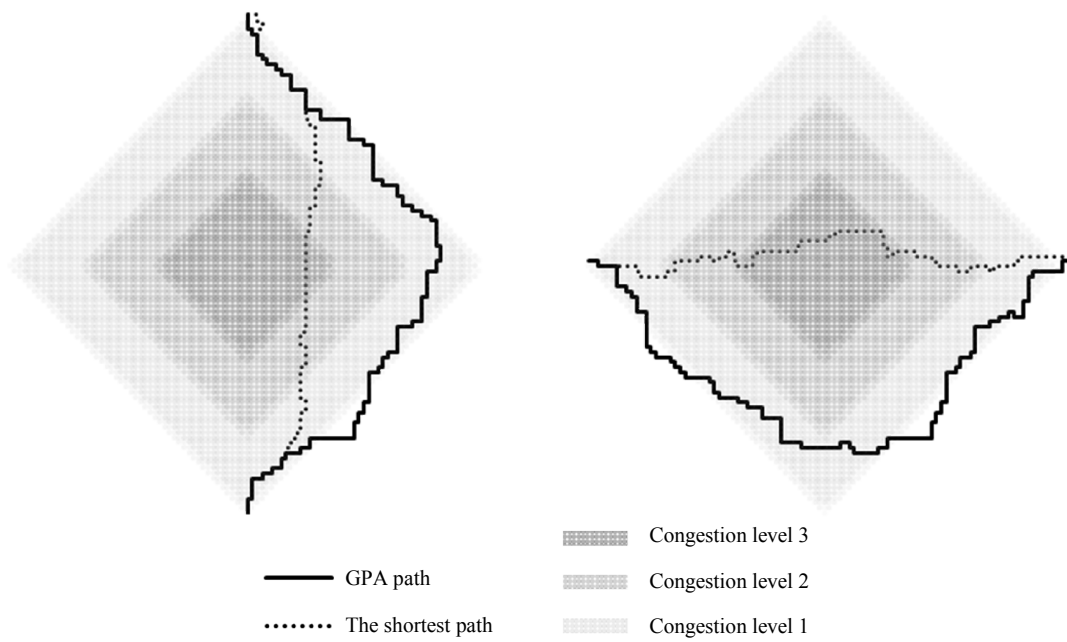


Figure 2.19. An illustration comparing GPA path and the shortest path on two synthetic data samples 3 and 4



We also conducted experiments on the real road network of Washington, DC. The results are summarized in Table 2.15. The performances of GPA and WÇE with the changing values of the time interval length and label limit are in parallel with their performances on the synthetic instances. Only when label limit is equal to 1 and the time interval length is equal to 30, WÇE finds a better result compared to GPA.

Table 2.15. Comparison for Washington DC data.

Time Interval Length		Label limit							
		1		10		100		1000	
		WÇE	GPA	WÇE	GPA	WÇE	GPA	WÇE	GPA
30	OF	6.277	6.259	6.243	6.251	6.232	6.232	6.232	6.232
	CT	0.8	0.8	1.3	1.2	6.7	3.1	37.1	2.8
60	OF	6.277	6.259	6.251	6.232	6.232	6.232	6.232	6.232
	CT	0.4	0.3	1.2	0.6	4.9	1.2	14.5	1.4
600	OF	6.277	6.259	6.232	6.232	6.232	6.232	6.232	6.232
	CT	0.1	0.1	0.5	0.1	1.2	0.1	1.2	0.1

In parallel with the observations on the synthetic data, the greenest paths obtained on the Washington DC data travels around the congested area whereas the shortest path goes through the congested center. Figure 2.20 visualizes the expansion of the congestion and the corresponding shortest and the greenest paths. Higher level of congestion in Figure 2.20 (b) and Figure 2.20 (c) causes the greenest path to change towards to the western side of the city.

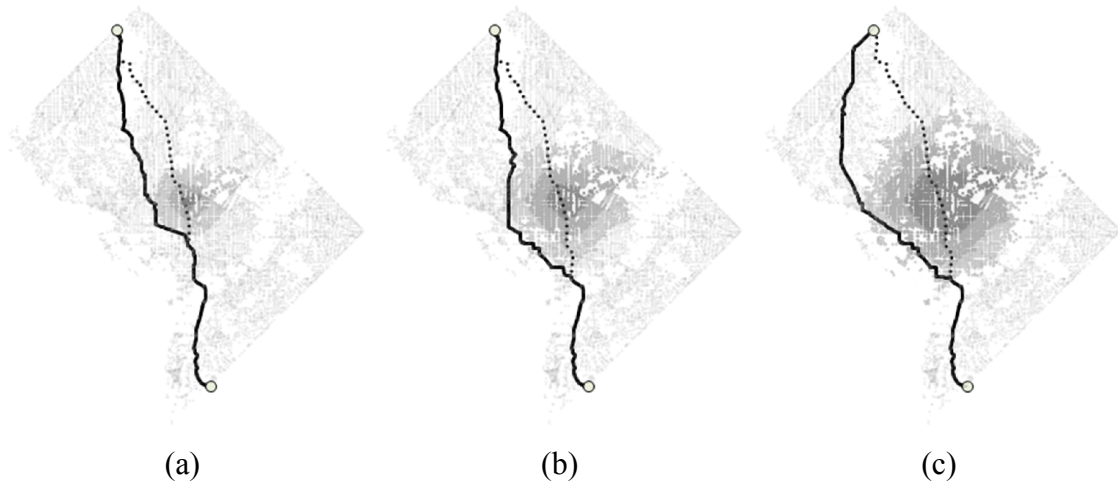


Figure 2.20. GPA path and the shortest path comparison on real data for three different congestion levels

Figure 2.21 depicts the greenest and the shortest paths for eight different geographically dispersed node pairs. The congestion levels for these networks are equal to the congestion level in Figure 2.20 (b).

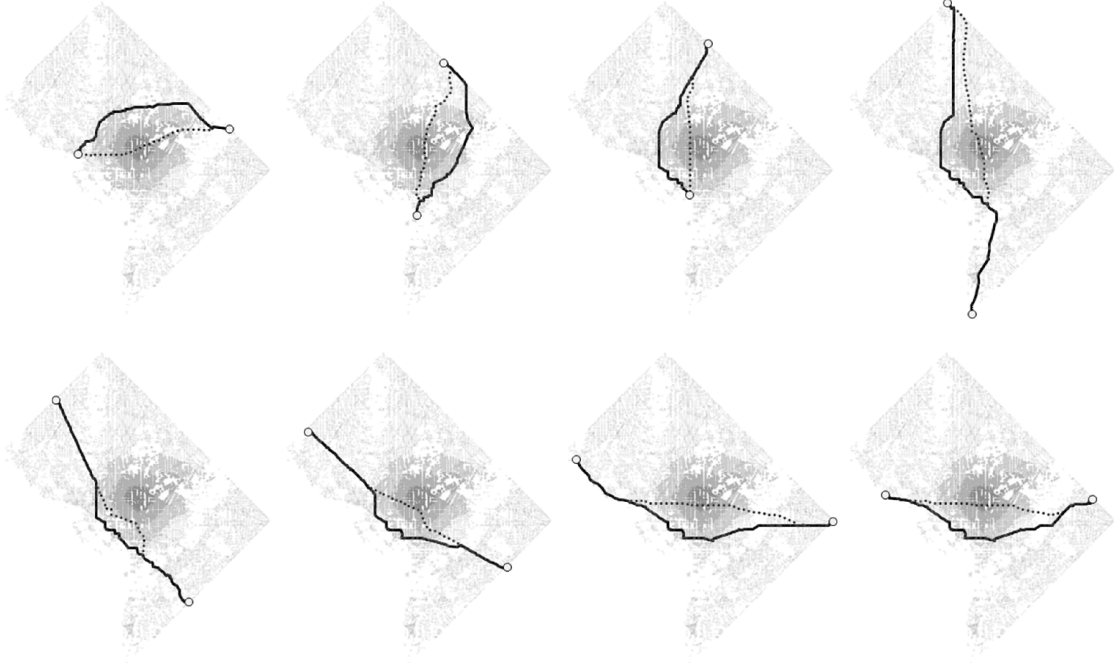


Figure 2.21. Illustration of greenest and shortest paths for 8 different node pairs

## 2.7 Conclusion and Future Research

With the growing concerns about the hazardous effects of transportation, sustainable logistics operations require new ways of doing business and planning approaches to decrease the negative impacts on the environment. Yet, finding the greenest path differs from the traditional path finding algorithms in having no pattern towards the optimal solution of the problem, which makes GPP a complicated optimization problem.

In this chapter, we discussed the properties of the GPP and showed the cases where traditional algorithms fail to find the greenest path after conducting a comprehensive literature survey. We next proposed a fast greenest path heuristic that makes use of bounds on the solution quality. Testing the proposed method on synthetic and real networks, we showed that GPA provides promising results. It achieved better average results in faster time compared to the only currently available heuristic method of Wen et al. (2014). We also showed visually how the green objective affects the routes to escape the congestion area whenever the network topology permits. Testing

the sensitivity of the algorithm to the changing values of time interval length and label limit, we reported the trade-off between the solution quality and the computational effort.

We did not observe for any setting on any instance that the upper bound path (shortest path) was equal to the greenest path. However, it is important to note that this depends on the type of the instance. For an instance where the speeds on the arcs are distributed between 50 km/h and 70 km/h and the optimal speed is 60 km/h, it is more likely that the shortest path will be equal to the greenest path.

In this study, we did not allow waiting at any node though we presented a classification of the literature from different waiting policies point of view. Further research will address incorporating these different waiting policies into the GPA. In spite of their theoretical gain, we did not observe any advantage of using cycles or rarely observed where a label with larger time yielded a better solution in our tests on the real network. However, to better test the effect of these concepts on the GHG emissions in real life, we will test GPA on other real networks.

## 2.8 References

- Ahuja, R. K., J. B. Orlin, S. Pallottino and M. G. Scutella (2002). "Minimum Time and Minimum Cost-Path Problems in Street Networks with Periodic Traffic Lights." Transportation Science **36**(3): 326-336.
- Ahuja, R. K., J. B. Orlin, S. Pallottino and M. G. Scutella (2003). "Dynamic Shortest Paths Minimizing Travel Times and Costs." Networks **41**(4): 197-205.
- Bauer, R. and D. Delling (2008). SHARC: Fast and Robust Unidirectional Routing. 2008 Proceedings of the Tenth Workshop on Algorithm Engineering and Experiments (ALENEX): 13-26.
- Bellman, R. (1958). "On a Routing Problem." Quarterly of Applied Mathematics **16**: 87-90.
- Cai, X., T. Kloks and C. K. Wong (1997). "Time-varying shortest path problems with constraints." Networks **29**(3): 141-150.
- Chabini, I. (1998). "Discrete dynamic shortest path problems in transportation applications: Complexity and algorithms with optimal run time." Transportation Research Record **1645**: 170-175.
- Chen, M., R. A. Chowdhury, V. Ramachandran, D. L. Roche and L. Tong (2007). Priority Queues and Dijkstra's Algorithm. Technical Report, University of Texas, Austin.
- Cooke, K. L. and E. Halsey (1966). "The shortest route through a network with time-dependent internodal transit times." Journal of Mathematical Analysis and Applications **14**(3): 493-498.
- Daganzo, C. F. (2002). "Reversibility of the time-dependent shortest path problem." Transportation Research Part B: Methodological **36**(7): 665-668.
- Dean, B. C. (1999). Continuous-Time Dynamic Shortest Path Algorithms. Technical Report, Massachusetts Institute of Technology, Cambridge.

- Dean, B. C. (2004a). "Algorithms for minimum-cost paths in time-dependent networks with waiting policies." Networks **44**(1): 41-46.
- Dean, B. C. (2004b). Shortest Paths in FIFO Time-Dependent Networks: Theory and Algorithms. Technical Report, Massachusetts Institute of Technology, Cambridge.
- Dehne, F., M. Omran and J.-R. Sack (2012). "Shortest Paths in Time-Dependent FIFO Networks." Algorithmica **62**(1-2): 416-435.
- Dell'Amico, M., M. Iori and D. Pretolani (2008). "Shortest paths in piecewise continuous time-dependent networks." Operations Research Letters **36**(6): 688-691.
- Delling, D. (2008). Time-Dependent SHARC-Routing. Algorithms - ESA 2008. D. Halperin and K. Mehlhorn, Springer Berlin Heidelberg. **5193**: 332-343.
- Delling, D., P. Sanders, D. Schultes and D. Wagner (2009). Engineering Route Planning Algorithms. Algorithmics of Large and Complex Networks. J. Lerner, D. Wagner and K. Zweig, Springer Berlin Heidelberg. **5515**: 117-139.
- Delling, D. and D. Wagner (2007). Landmark-Based Routing in Dynamic Graphs. Experimental Algorithms. C. Demetrescu, Springer Berlin Heidelberg. **4525**: 52-65.
- Delling, D. and D. Wagner (2009). Time-Dependent Route Planning. Robust and Online Large-Scale Optimization. R. Ahuja, R. Möhring and C. Zaroliagis, Springer Berlin Heidelberg. **5868**: 207-230.
- Deo, N. and C.-Y. Pang (1984). "Shortest-path algorithms: Taxonomy and annotation." Networks **14**(2): 275-323.
- Dijkstra, E. W. (1959). "A note on two problems in connexion with graphs." Numerische Mathematik **1**(1): 269-271.
- Disser, Y., M. Müller–Hannemann and M. Schnee (2008). Multi-criteria Shortest Paths in Time-Dependent Train Networks. Experimental Algorithms. C. McGeoch, Springer Berlin Heidelberg. **5038**: 347-361.
- Dreyfus, S. E. (1969). "An Appraisal of Some Shortest-Path Algorithms." Operations Research **17**(3): 395-412.
- Eglese, R. W. and D. Black (2010). Optimizing the Routing of Vehicles. Green Logistics: Improving the Environmental Sustainability of Logistics. A. McKinnon, S. Cullinane, M. Browne and S. Whiteing, London, Kogan Page: 215-228.
- EMEP/CORINAIR (2007). EMEP/CORINAIR Emission Inventory Guidebook: Group 7 road transport.
- Hall, R. (1986). "The fastest path through a network with random time-dependent travel times." Transportation Science **20**(3): 182-186.
- Halpern, J. (1977). "Shortest route with time dependent length of edges and limited delay possibilities in nodes." Zeitschrift für Operations Research **21**(3): 117-124.
- Hart, P. E., N. J. Nilsson and B. Raphael (1968). "A Formal Basis for the Heuristic Determination of Minimum Cost Paths." Systems Science and Cybernetics, IEEE Transactions on **4**(2): 100-107.
- Hickman, J., C. Hassel, R. Joumard, Z. Samaras and S. Sorenson (1999). MEET Methodology for Calculating Transport Emissions and Energy Consumption. Technical Report.
- Horn, M. E. T. (2000). "Efficient modeling of travel in networks with time-varying link speeds." Networks **36**(2): 80-90.
- Kanoulas, E., D. Yang, X. Tian and Z. Donghui (2006). Finding Fastest Paths on A Road Network with Speed Patterns. Data Engineering, 2006. ICDE '06. Proceedings of the 22nd International Conference on.
- Kaufman, D. E. and R. Smith (1990). Minimum travel time paths in dynamic networks with application to intelligent vehicle-highway systems. Ann Arbor, Mich., University of Michigan, Transportation Research Institute.

- Kaufman, D. E. and R. Smith (1993). "Fastest Paths in Time-Dependent Networks for Intelligent Vehicle-Highway Systems Application." Journal of Intelligent Transportation Systems **1**(1): 1-11.
- Miller, E. D., H. S. Mahmassani and A. Ziliaskopoulos (1994). Path search techniques for transportation networks with time-dependent, stochastic arc costs IEEE International Conference on Systems, Man and Cybernetics.
- Nannicini, G., D. Delling, L. Liberti and D. Schultes (2008). Bidirectional A \* Search for Time-Dependent Fast Paths. Experimental Algorithms. C. McGeoch, Springer Berlin Heidelberg. **5038**: 334-346.
- Nielsen, L., D. Pretolani and K. Andersen (2009). Bicriterion Shortest Paths in Stochastic Time-Dependent Networks. Multiobjective Programming and Goal Programming. V. Barichard, M. Ehrgott, X. Gandibleux and V. T'Kindt, Springer Berlin Heidelberg. **618**: 57-67.
- Orda, A. and R. Rom (1990). "Shortest-path and minimum-delay algorithms in networks with time-dependent edge-length." J. ACM **37**(3): 607-625.
- Orda, A. and R. Rom (1991). "Minimum weight paths in time-dependent networks." Networks **21**(3): 295-319.
- Pallottino, S. and M. G. Scutellà (1998). Shortest Path Algorithms In Transportation Models: Classical and Innovative Aspects. Equilibrium and Advanced Transportation Modelling. P. Marcotte and S. Nguyen, Springer US: 245-281.
- Pallottino, S. and M. G. Scutellà (2003). "A new algorithm for reoptimizing shortest paths when the arc costs change." Operations Research Letters **31**(2): 149-160.
- Sanders, P., C. Vetter, D. Delling and G. V. Batz (2009). Time-Dependent Contraction Hierarchies. 2009 Proceedings of the Eleventh Workshop on Algorithm Engineering and Experiments (ALENEX): 97-105.
- Sherali, H. D., A. G. Hobeika and S. Kangwalklai (2003). "Time-Dependent, Label-Constrained Shortest Path Problems with Applications." Transportation Science **37**(3): 278-293.
- Sung, K., M. G. H. Bell, M. Seong and S. Park (2000). "Shortest paths in a network with time-dependent flow speeds." European Journal of Operational Research **121**(1): 32-39.
- Tian, Y., K. C. K. Lee and W.-c. Lee (2009). Monitoring Minimum Cost Paths on Road Networks With Time-Dependent Edge Availabilities. Proceedings of the 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems GIS '09.
- Wellman, M. P. (1990). "Fundamental concepts of qualitative probabilistic networks." Artificial Intelligence **44**(3): 257-303.
- Wellman, M. P., M. Ford and K. Larson (1995). Path planning under time-dependent uncertainty. Proceedings of the Eleventh conference on Uncertainty in artificial intelligence. Montrel, Quebec, Canada, Morgan Kaufmann Publishers Inc.: 532-539.
- Wen, L., B. Çatay and R. Eglese (2014). "Finding a minimum cost path between a pair of nodes in a time-varying road network with a congestion charge." European Journal of Operational Research **236**(3): 915-923.
- Ziliaskopoulos, A. (1994). Optimum path algorithms on multidimensional networks: Analysis, design, implementation and computational experience, Technical Report, University of Texas, Austin.
- Ziliaskopoulos, A. K. and H. S. Mahmassani (1993). "Time-Dependent , Shortest-Path Algorithm for Real-Time Intelligent Vehicle Uighway System Applications." Transportation Research Record **1408**: 94-100.

### **3 CREATING NETWORK-CONSISTENT SPEEDS ON TIME-DEPENDENT NETWORKS**

### **3.1 Introduction**

To the best of our knowledge, only Lecluyse et al. (2013) proposed a systematic approach for creating network-consistent time-dependent travel time. Their method is best fitted for synthetic network data for routing optimization problems. However, there is still a gap in the literature to generate network-consistent travel time data on real road networks that takes into account the realistic features such as the connectivity of the arcs, travel direction and side road and main road differentiation. The main contribution of this section is to fill this gap.

In Section 3.2 we analyze a real road network with time-dependent speeds in order to deduce some existing patterns taking the expansion of the congestion into account. We also discuss the practicability of Lecluyse et al. (2013) on real road networks. Note that Lecluyse et al. (2013) propose their method for creating time-dependency on VRP instances and do not claim to be applicable for real networks after all. In Section 3.3, we propose our new network-consistent congestion generation scheme and the implementation details followed by a sample implementation on a real network in Section 3.4. Conclusions and future research directions follow in Section 3.5.

### **3.2 Implications on a Real Road Network and Congestion Circles**

To achieve a much larger degree of realism, we first analyze a real road network and the corresponding speed data. The data in consideration belongs to the European side of İstanbul and is obtained from İstanbul Transportation Communication and Security Technologies INC. (ISBAK). A total number of 159 sensors are used to collect data that covers 4 weeks with intervals of 2 or 5 minutes depending on the type of the sensor. They also differ in measuring capability. Some sensors can gather bidirectional data whereas some can only measure a single direction on a specific road segment. The locations of the sensors are shown in Figure 3.1.

İstanbul is a transcontinental city with a distinctive congestion structure. During the rush hours, there is relatively high congestion on the roads that are connected directly to the bridges between the European and Asian sides. Nevertheless, we can still observe some patterns.

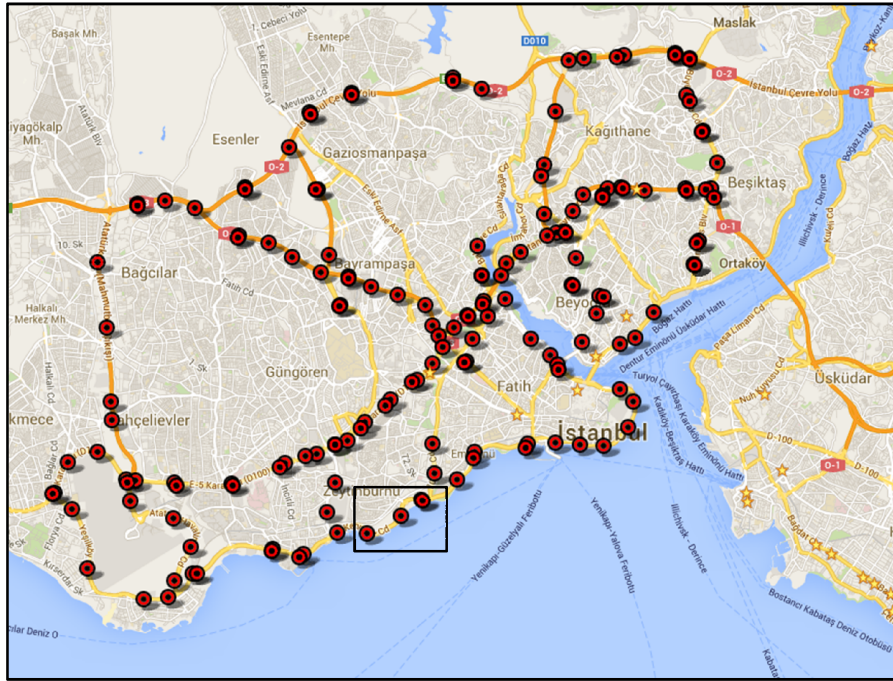


Figure 3.1. The locations of the sensors used to gather data

The first observation is on the expansion of the congestion. Figure 3.2 gives the distribution of normalized speed levels for a sample set of three sensors that are enclosed with a rectangle in Figure 3.1. The sensor at the east end is labeled as sensor 1 whereas the sensor on the west end is labeled as sensor 3. Congestion sets on the road segment corresponding to the sensor 1 after 6:30. Next road segment of sensor 2 is affected by the congestion after 7:00. The congestion finally reaches the road segment of sensor 3 after 8:30.

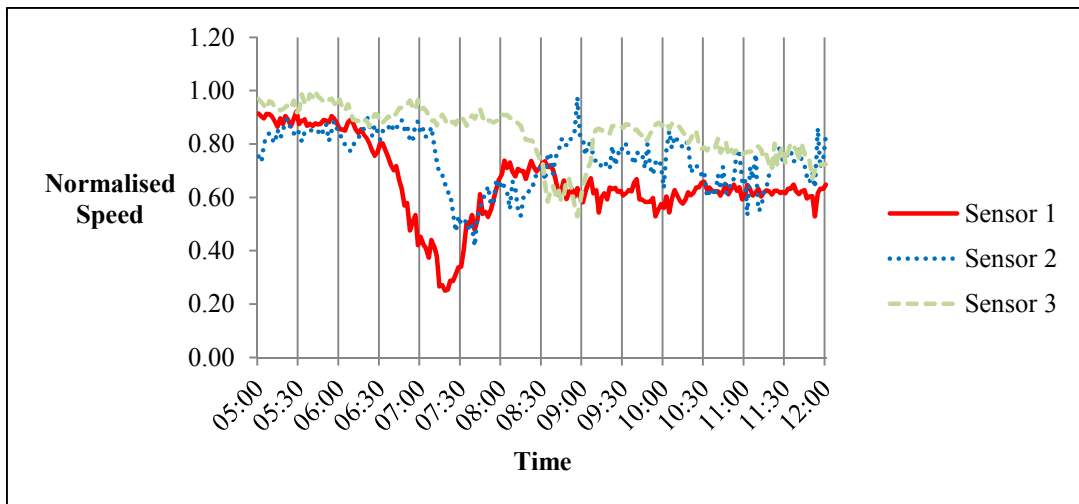


Figure 3.2. Distribution of speed on a small segment of İstanbul



The second observation is on the rate of the decrease in speed. Figure 3.3 shows the decreasing speed values of the sensor 1 between 06:00 and 07:26 when the congestion reaches its peak value. At 06:00, the measured speed is 73.39 km/h. It takes 64 minutes for the average speed to decrease to 68.46 km/h. After 14 minutes, the speed decreases to 62.04 km/h and it takes only 5 minutes to further decrease to 56.24 km/h. That is, the rate of speed decrease increases after congestion onset. Although a uniform decrease pattern also exists for some road segments, we find it more convenient to follow the first pattern for congestion generation.

Keeping these findings in mind, we show why the congestion circles of Lecluyse et al. (2013) are not suitable for real road networks. We will first visually show how congestion circles work. Figure 3.4 shows a small part of the İstanbul network. The road on the north with orange color is a toll road where the road on the south with yellow color is a toll-free road. These roads are not connected to each other at any point in the given map section.

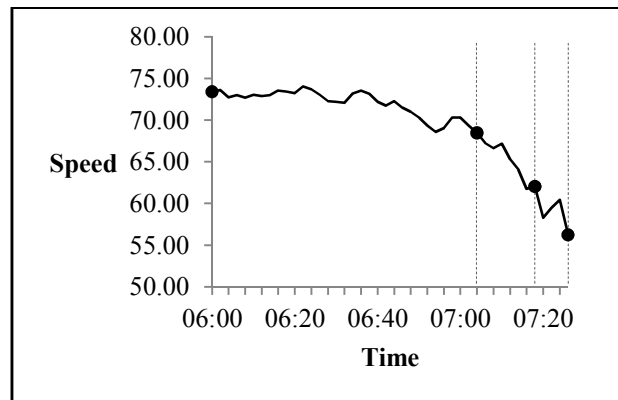


Figure 3.3. Distribution of speed on a small segment of İstanbul

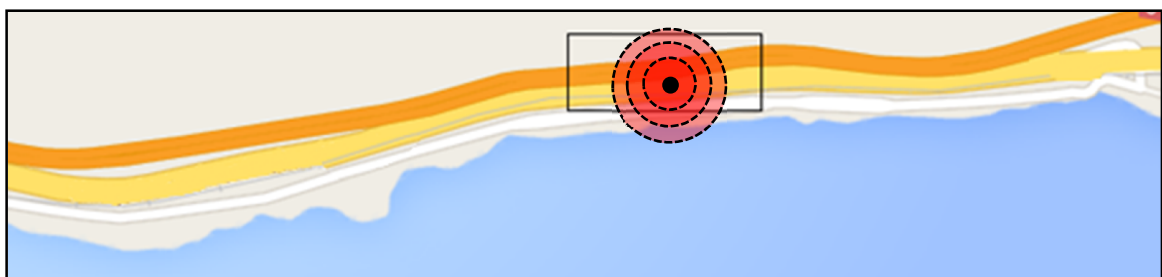


Figure 3.4. A sample congestion circle

Let there be congestion on the toll-free road. The exact place of the congestion is shown with a black dot. The congestion expands by a circle namely a congestion circle. Each circle corresponds to an instant in time and shows the affected regions at that time. The circles are darker where the level of congestion is higher. Congestion circles expand, preserve their maximum radius for a while, shrink and disappear. Figure 3.5 zooms to the region in black rectangle in Figure 3.4.

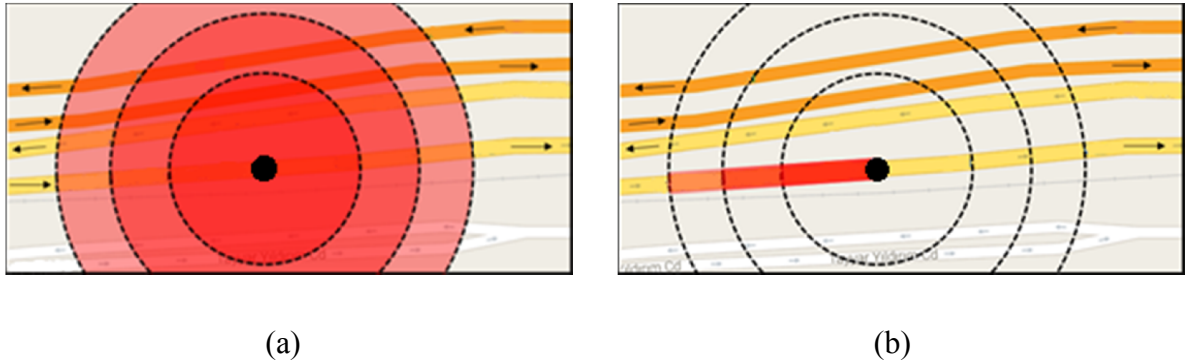


Figure 3.5. A sample congestion circle in detail.

The first shortcoming of Lecluyse et al. (2013) is that it does not take direct connectivity between arcs into consideration. So, the congestion in toll-free road affects the toll-road in the model although these roads are not connected. But in reality, congestion will not affect a nearby road if there is not a linking road between them. Similarly, congestion on this part of toll-free road where the direction of travel is from left to right may not create congestion on the opposite travel direction (neglecting those drivers who slow down to have a look in case of an accident).

Secondly, the direction of travel is not taken into account in Lecluyse et al. (2013). As mentioned above with the real data, no congestion is created beyond the exact spot of the congestion. So the expansion of the congestion should be as given in Figure 3.5.b; in the reverse direction of the traffic flow and on the connected road segments only.

A strong assumption in Lecluyse et al. (2013) is that the rate of travel time increase or decrease is fixed which is not the case in real networks as we observed in Istanbul data. To ensure a fixed increase or decrease rate in travel time, they implicitly assume a single jump in speed values as shown in Figure 3.6.a. However, the travel time

can be modelled more accurately as the speed increase or decrease is represented in more detail as shown in Figure 3.6.b-c.

Last but not the least, Lecluyse et al. (2013) uses a projection of the original graph onto the Euclidean space. The geometric congestion circle calculations are based on the assumption that the link between each node pair consists of a single straight line. They operate directly on inter-node distances and do not consider the underlying road network. They point out this as an advantage as only inter-node distances are given for most of the VRP instances. Nevertheless, this structure prevents it from directly being applied to the real road networks successfully. An extreme case is depicted in Figure 3.7. The congestion circle centered at point  $c$  intersect with the direct link between nodes  $i$  and  $j$ , thus affecting the travel time. However, the congestion circle does not intersect with the real link.

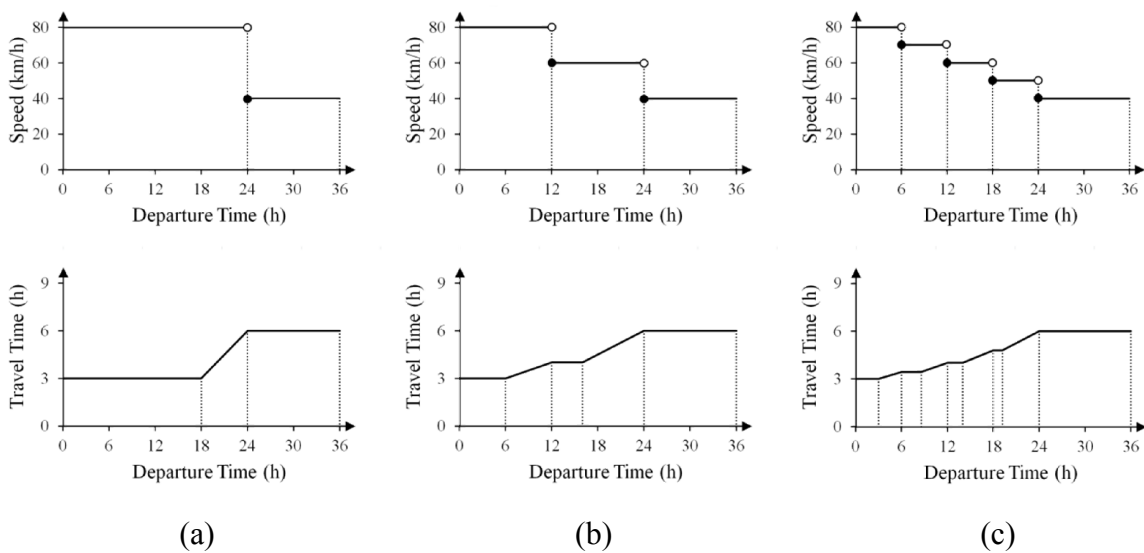


Figure 3.6. Speed and travel time functions with different accuracy levels

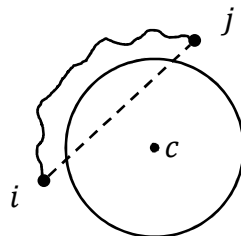


Figure 3.7. Congestion circles do not consider the underlying road network

The proposed method in the following section utilizes the implications of this section as a basis.

### 3.3 A New Network-Consistent Congestion Generation Scheme

In our method, we handle the temporal and spatial congestion using *congestion arcs* which correspond to the road segments where the congestion sets first. Taking the direction of the travel on this specific arc into account, the congestion expands through the arcs that are directly connected to the congestion arc. This prevents any arc that is geographically close but not connected to the congestion arc from being congested.

Figure 3.8 shows a sample congestion generation where the width of the arc represents the current congestion level. As congestion increases, the arc becomes thicker. Congestion is created on the arc (A-B) from south to north in Figure 3.8.a. In Figure 3.8.b, congestion is only seen onsets on an arc from south to north. After a predetermined time those arcs that are connected directly are affected taking the direction of the travel on arcs into consideration. The time between the states Figure 3.8.b and Figure 3.8.c is longer compared to the time between the states Figure 3.8.c and Figure 3.8.d. In other words, the speed is decreased in an increasing rate in accordance with the previous observations on real data.

The assumption in this approach is that the whole arc is affected by the congestion uniformly. However, this assumption can be justified when the arc lengths are small. Also, artificial nodes can be introduced to divide the arc into smaller arc segments in case of arcs with long distances.

The main input of the proposed method is  $\mathcal{C}$ , an array of congestion arcs where congestions set. Associated with each congestion arc  $c_k \in \mathcal{C}$  is a list of parameters that are summarized in Table 3.1. For ease of notation and completeness, we follow the same parameter names in Lecluyse et al. (2013) for congestion generation where available. The pseudo-code of the algorithm is given in Algorithm 3.1 and Algorithm 3.2.

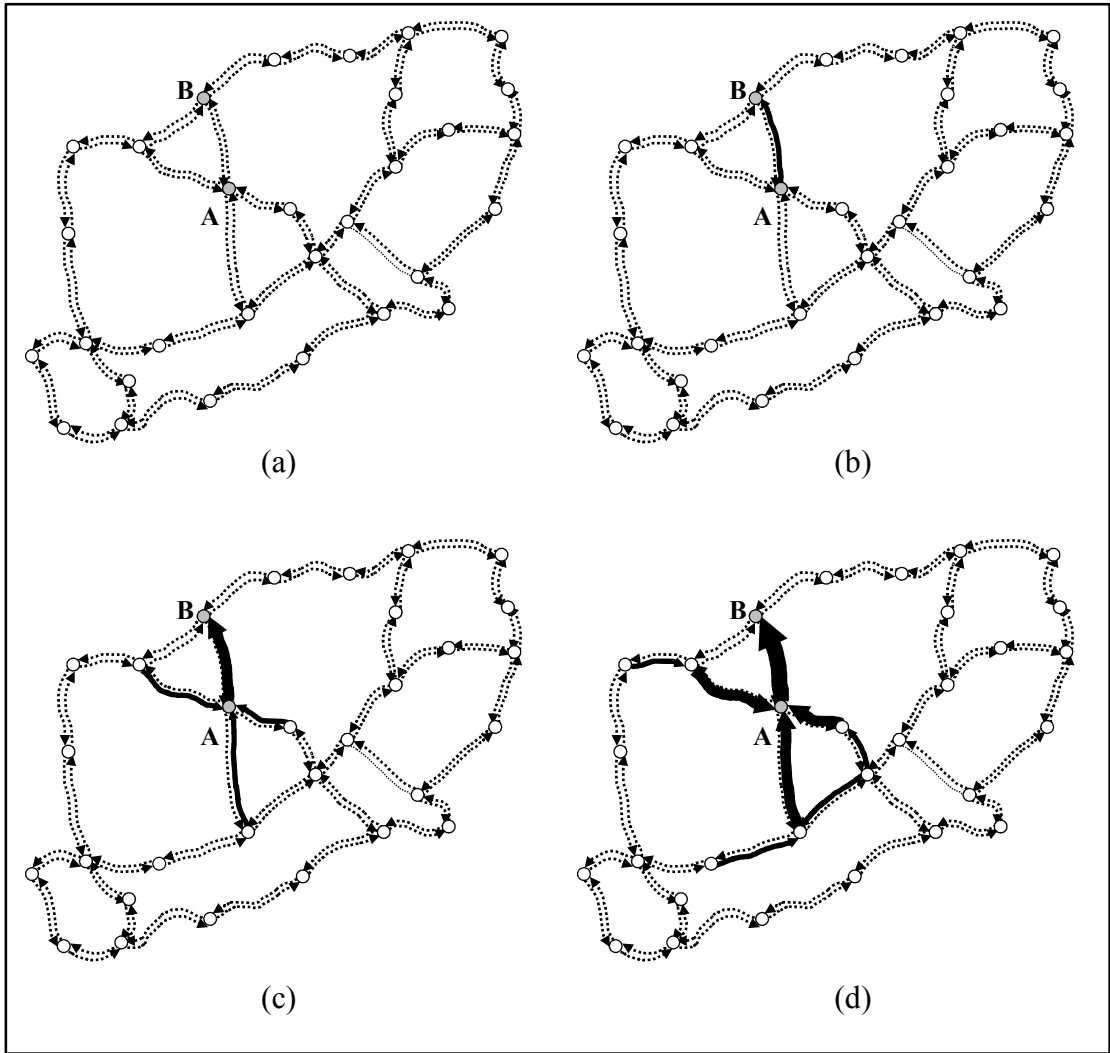


Figure 3.8. Sample congestion generation.

Table 3.1. Parameters of a congestion arc

Parameter Name	Description
<i>From</i>	Start node of arc $c_k$
<i>To</i>	End node of arc $c_k$
<i>Con</i>	Time of congestion onset
<i>Fc</i>	Time of full congestion
<i>Cof</i>	Time of congestion offset
<i>Ff</i>	Time of free flow conditions
<i>Noe</i>	Number of expansions
<i>Sd</i>	Speed decrease percentage per expansion
<i>Td</i>	Time decrease between expansions

---

**Algorithm 3.1.** Main

---

*Input:* forward star and backward star lists ( $FS$  and  $BS$ ), congestion arc array ( $C$ )

```
1  for all  $c_k \in C$ 
2    Calculate time between expansions ( $t_e$ ) and contractions ( $t_c$ )
3    numberOfExpansions[ $c_k$ ] =  $c_k$ .numberOfExpansions
4    Create congestion( $c_k, c_k, t_e, t_c, \text{numberOfExpansions}[c_k]$ )
5    FifoArcList = FifoArcList  $\cup c_k$ 
6    while |FifoArcList| > 0
7      parentArc = FifoArcList[0]
8      FifoArcList = FifoArcList / parentArc
9      for all Node  $i \in BS(\text{parentArc.From})$ 
10       childArc.From =  $i$ 
11       childArc.To = parentArc.From
12       numberOfExpansions[childArc] = numberOfExpansions[parentArc] - 1
13       Create congestion(childArc,  $c_k, t_e, t_c,$ 
14         numberOfExpansions[childArc], speedDecreasePercentage)
15       if FifoArcList does not contain childArc
16         FifoArcList = FifoArcList  $\cup$  childArc
17       endif
18     endfor
19   endwhile
20 Finalize
```

---

Figure 3.9. Main congestion generation algorithm

In the *main* algorithm (Figure 3.9), time between expansions and contractions are calculated first. The number of expansions (and contractions) refers to the maximum level of arcs that will be affected by the current arc in a tree fashion. Thus, when the number of expansions is 1, the congestion will only affect the congestion arc. To exemplify, Figure 3.8.a, b and c may refer to the cases where the number of expansions are 1, 2 and 3 respectively. After creating congestion on the congestion arc and adding it to the arc list which works in a FIFO basis, congestion is created on the child arcs that are connected to the parent arc, the arc that is pulled from the arc list. The number of expansions for a child arc is 1 less than that of the parent arc.

The *create congestion* algorithm (Figure 3.10) works arc by arc and adds an instant in time and the corresponding speed multiplier for every expansion and contraction. After generating all speed and time information for all affected arcs, a final step is required to convert the speed multipliers to the real speed values.

Table 3.2 gives a sample speed multiplier and time information and the final output for a congestion arc where the horizon starts at 6:00 and ends at 11:00.

---

**Algorithm 3.2.** Create Congestion (arc, congestionArc, timeBetweenExpansion (TBE), timeBetweenContraction (TBC), numberOfExpansions (NOE), speedDecreasePercentage (SDP))

---

```

1  SpeedTimeInfo[arc].Add(time=congestionArc.con, speedCoefficient=1)
2  for  $i = 1$  to NOE
3    SpeedTimeInfo[arc].Add(time=congestionArc.con +  $i$ * TBE,
4      speedCoefficient=(1- SDP)
5  endfor
6  SpeedTimeInfo[arc].Add(time=congestionArc.cof, speedCoefficient=1)
7  for  $i = 1$  to NOE
8    SpeedTimeInfo[arc].Add(time=congestionArc.cof +  $i$ * TBC,
9      speedCoefficient=1/(1- SDP))
10 endfor

```

---

Figure 3.10. Congestion generation algorithm on a single arc

Congestion onset and offset are realised at 7:00 and 9:00 respectively. The number of expansions is 3, time between expansions is 20 minutes and time between contractions is 25 minutes. Speed decrease percentage is 10% and the free flow speed is 90 km/h.

Table 3.2. A sample speed generation

Time	6:00	7:00	7:20	7:40	8:00	9:00	9:25	9:50	10:15	11:00
Speed Multiplier	-	1.00	0.90	0.90	0.90	1.00	1.11	1.11	1.11	1.00
Speed	90.0	90.0	81.0	72.9	65.6	65.6	72.9	81.0	90.0	90.0

The time-dependent routing or path finding literature includes speed-based calculations. The stepwise speed model of Ichoua et al. (2003) which guarantees FIFO property is widely used for this purpose. They stated that, although the travel speeds also change continuously over time, it is more reasonable to use a step function for the travel speeds rather than for the travel time itself. Thus, as a post-process, we include a finalization step to create a stepwise speed model using predetermined time duration. Figure 3.11.a shows the corresponding speeds for the example in Table 3.2 assuming a linear increase or decrease whereas Figure 3.11.b shows a stepwise model where the step lengths are 15 minutes.

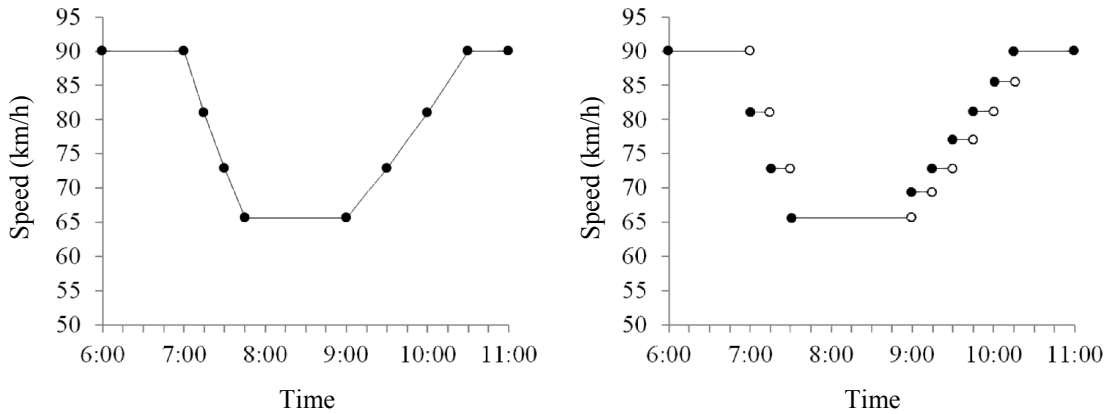


Figure 3.11. Continuous (a) and corresponding step-wise (b) speed functions

### 3.4 Creating Time-Dependency on a Sample Large Scale Real Time-Independent Network

We test our approach for creating network-consistent speeds on the real data of Washington D.C. which includes 9,559 nodes and 14,909 arcs. Taking Summary of State Speed Laws Report of NHTSA (2012) as a basis, we define four types of roads that are primary highway with limited access/freeway rural (e.g. interstates), primary road without limited access/freeway urban (e.g. us highways), secondary and connecting road/undivided rural (e.g. state highways) and local, neighborhood, and rural road/residential. The underlying road structure is given in Figure 3.12. The corresponding speed values and representative colors are 70 mi/h (112.65 km/h) and blue, 60 mi/h (96.56 km/h) and brown, 55 mi/h (88.51km/h) and black and 35 mi/h (56.33 km/h) and gray respectively. Please note that this is not restrictive as one can change the speed limits or introduce new road types.

In Figure 3.13, we create congestion in the center of the city. The number of expansions is 30. The green color refers to the freeflow speed. Dark red colored links have the highest level of congestion. Figure 3.13.a shows the early phase of the congestion where only the close vicinity of the center is affected by congestion. After expanding through the connected arcs (Figure 3.13.b), the network reaches its maximum congestion level (Figure 3.13.c). Note that, any number of congestions with any parameters can be defined on the network. Figure 3.14 shows a two congestion arc example.



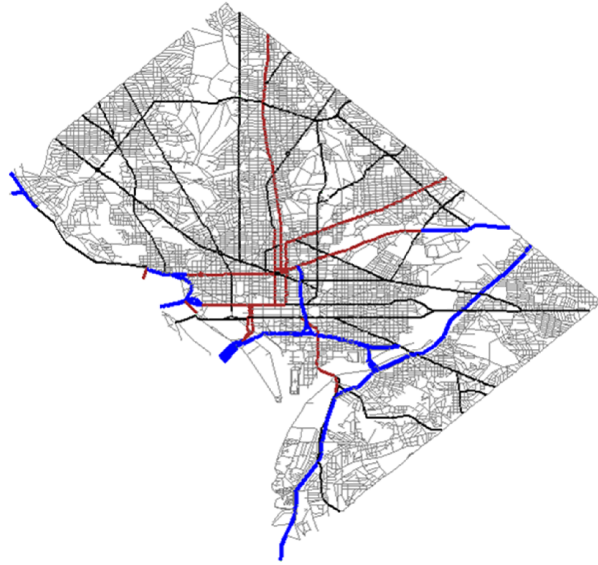


Figure 3.12. Different road types for real data

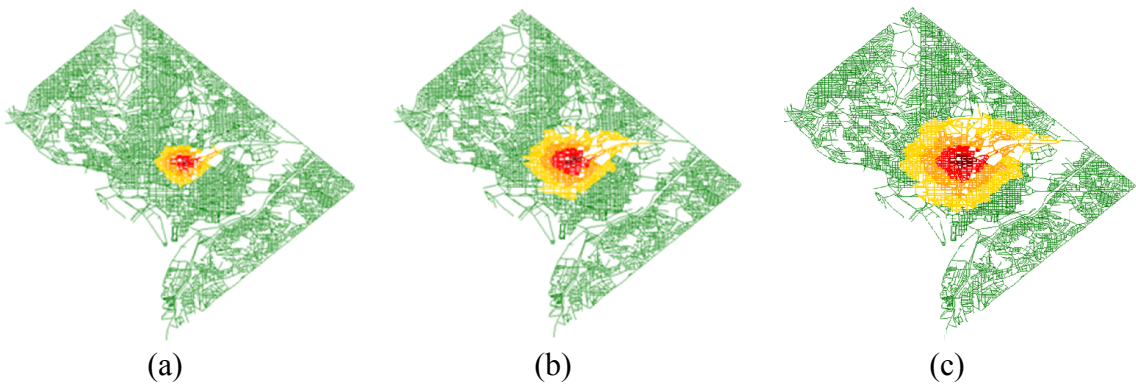


Figure 3.13. Three phases of a single congestion area on real data

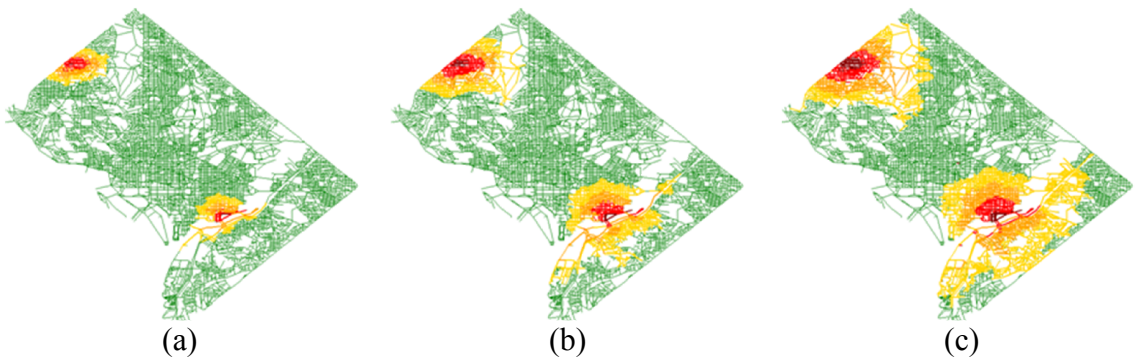


Figure 3.14. Three phases of multiple congestion areas on real data

### 3.5 Conclusion and Future Research

In this chapter we proposed a new method for generating network-consistent time-dependent speed and travel time layer on a given time-dependent network, either real or synthetic. To the best of our knowledge, this is the first such study proposed for real networks that also takes the realistic features such as the connectivity of the arcs, direction of the travel or side road and main road differentiation into account.

The slope on the road also affects real emissions and fuel consumption. As a future research direction, we plan to incorporate the road slope into the algorithm in a similar manner to the congestion generation. Instead of selecting arcs to create congestion, nodes to increase elevation will be selected this time. Similar to congestion generation, after setting an elevation level for a selected node, the elevation level of the nodes that are directly connected will be increased. One can either select single node to create a hill (Figure 3.15.a) or a group of nodes to create a high but flat surface (Figure 3.15.b).

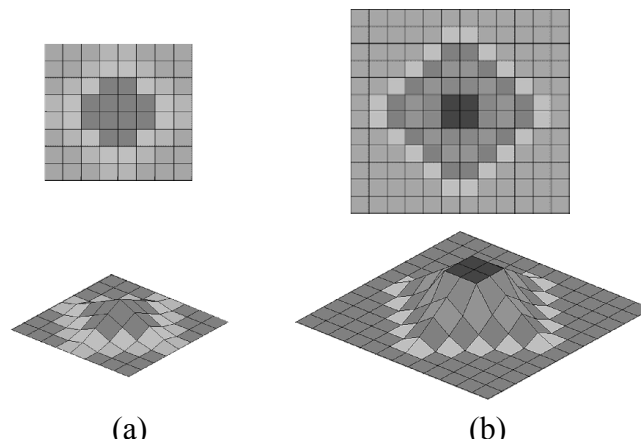


Figure 3.15. Sample slope generation.

### 3.6 References

- Ichoua, S., M. Gendreau and J.-Y. Potvin (2003). "Vehicle dispatching with time-dependent travel times." *European Journal of Operational Research* **144**(2): 379-396.
- Lecluyse, C., K. Sorensen and H. Peremans (2013). "A network-consistent time-dependent travel time layer for routing optimization problems." *European Journal of Operational Research* **226**(3): 395-413.
- Yıldırım, U. M. (2014). "Speed Generator." from <http://myweb.sabanciuniv.edu/mahiryldrm/speedGenerator>.

## **4 EXACT AND HEURISTIC METHODS FOR THE TIME-DEPENDENT MINIMUM COST PATH PROBLEM WITH SPEED (TDMCP-S)**

## 4.1 Introduction

Most of the studies in the Time-Dependent Shortest Path Problem (TDSPP) literature assume that the speed is fixed during any time interval. The corresponding emission value of a speed is often disregarded and the path is traversed with the maximum available speed. However, a significant amount of GHG emission reduction can be obtained by altering the speed as the carbon emission function is strongly related with the speed of the vehicle. Figliozzi (2011) pointed out that the emissions could be reduced indirectly even by reducing the speed limits to a speed that is optimal from an emissions perspective. The speed values are fixed and are not a decision variable in this case. Moreover, Qian and Eglese (2014) reported that about 6-7% savings in fuel emissions could be achieved by adjusting the speed values. Note that, this reduction is on top of the 7%, reported by Maden et al. (2010), that can be achieved by taking the time-dependent speeds into account.

Taking the human behaviour into account, it may be questionable whether a driver can be motivated to reduce the speed of the vehicle to a certain value to decrease the total emission cost. Nevertheless, self-driving cars are not a far future. Google has already been testing their developed technology and its software powered autonomous cars along with their Google Self-Driving Car project (Wikipedia, 2014). In United States, three states, namely Nevada, Florida and California, have legalized the use of self-driven cars for testing purposes. Germany, Netherlands and Spain have also allowed testing robotic cars in traffic and Finland is planning on passing a law before 2015. With the use of self-driven cars, it would be legitimate to take the speed as a decision variable.

In this chapter, we develop a mathematical model where the speeds on the arcs are also decision variables as well as the departure times and further propose a time-space-speed expansion. The remainder of this chapter is organized as follows: Section 4.1 provides a brief review of the relevant literature. Section 4.2 gives the problem definition and mathematical formulations for the TDSPP and its extension to the Time-Dependent Minimum Cost Path Problem. The problem definition and formulation for the Speed Embedded Time-Dependent Minimum Cost Path Problem (hereafter called TDMCP-S for brevity) is given in Section 4.3. Section 4.4 briefly introduces the time-space-speed expansion as an alternative modelling technique for the TDMCP-S. Section 4.5 presents a computational study to test the performance of the models and the time-

space-speed expansion and reports the results. Finally, concluding remarks and future research directions are given in Section 4.6.

## 4.2 Relevant Literature

It has become more common lately in the routing and path finding literature to adjust the speed values. Most of these studies that take the speed as a decision variable use a post-processing step to decide the speeds after the routes/paths are generated. The speed optimization procedure in most of these studies are built upon Norstad et al. (2011) and Hvattum et al. (2013) which are specifically designed speed-optimization procedures (SOP). They calculate the optimal speeds for a given path with a single time window and zero service time at each node.

In spite of the increasing momentum in the speed optimization literature, there are only a few studies that try to optimize speed in a time-dependent environment. In the first study, Figliozzi (2010) proposed a model for an extension of the vehicle routing problem (VRP), namely the emissions VRP. Yet, they did not attempt to solve the model directly. Their proposed method is composed of two stages. They first minimize the number of vehicles using a time-dependent VRP algorithm. Taking the fleet size into account, they then optimize the departure times and further improve the emission costs by changing the routes.

Jabali et al. (2012) used Tabu Search to solve the emissions-based time-dependent VRP where the speed is also a decision variable. They modelled the travel times by introducing two different regions in the planning horizon. The first is a peak period with congestion where the vehicle speed is fixed. They assumed that this fixed speed value is imposed by traffic conditions. In the second period, the free flow speed is a decision variable. They applied local search procedures to update the free flow speed.

Franceschetti et al. (2013) extended SOP and proposed a departure time and speed optimization procedure by considering the departure time in addition to speed in the time-dependent pollution routing problem context. They considered the special case with only a single vehicle and a fixed sequence of customers. Thus, similar to Figliozzi (2010), the proposed procedure of Franceschetti et al. (2013) is also not fitted for deciding the path/route and the speeds on each arc simultaneously. They also proposed an integer linear programming formulation for an environment with three time intervals which are composed of all congestion, transient and free-flow regions. They presented the analytical results on a single-arc time-dependent pollution routing problem instance.

In the most current study of Qian and Eglese (2014), a two-stage time-increment based dynamic programming approach is proposed. After computing the optimal fuel emissions between every node pair on a given sequence of customers with all combinations of starting time and finishing times, they calculate the optimal fuel emissions for the complete route. Pointing out the irrelevant calculations of the DP, they proposed an approach called the adaptive searching method. By setting different artificial speed restrictions in a similar way to the claim of Figliozzi (2011) and assuming that the vehicle travels at the artificial speed limits, they generated promising fastest routes with each artificial speed limit setting. Next, they developed an approximate dynamic programming algorithm in the speed adjustment process to modify the speeds along the candidate routes and reach the final minimum fuel emission generating route and speeds. Please note that even that they treat the speed as a decision variable as well as the arcs to be traversed, they evaluate them sequentially rather than simultaneously.

None of the previous studies guarantee optimality. Figliozzi (2010) and Franceschetti et al. (2013) sequentially find the route/path and optimize the speed values selection whereas Jabali et al. (2012) use a metaheuristic. Although the proposed algorithm in Qian and Eglese (2014) is shown to generate promising routes, there is also no optimality guarantee.

In Figure 4.1, we give a sample network on which Qian and Eglese (2014) misses the optimal route. The numbers in parentheses and brackets refer to the distance and the speed of the corresponding arc, respectively. The objective is to find the least emission generating path that starts from node 1 and ends at node 4. It can easily be seen that there exist three paths that are 1-3-4, 1-2-4 and 1-2-3-4. The costs of these paths are

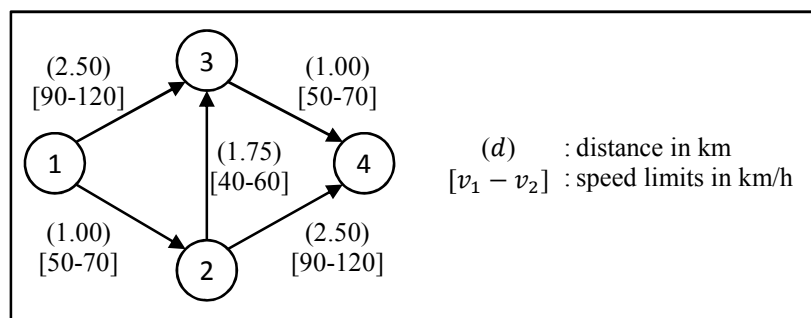


Figure 4.1. A sample network to test post speed optimization process

1132.42 gr, 1132.42 gr and 687.47 gr respectively. As for the cost calculation, we use the cost function of Hickman et al. (1999).

Table 4.1. Adjusted speeds and travel times

Artificial Speed Limit (km/h)	Adjusted Speed (km/h)					Travel Time (min)				
	(1-2)	(1-3)	(2-3)	(2-4)	(3-4)	(1-2)	(1-3)	(2-3)	(2-4)	(3-4)
120	70	120	70	70	70	0.86	1.25	1.75	1.25	0.86
110	70	110	70	70	70	0.86	1.36	1.75	1.36	0.86
100	70	100	70	70	70	0.86	1.50	1.75	1.50	0.86
90	70	90	70	70	70	0.86	1.67	1.75	1.67	0.86
80	70	80	70	70	70	0.86	1.88	1.75	1.88	0.86
70	70	70	70	70	70	0.86	2.14	1.75	2.14	0.86
60	60	60	60	60	60	1.00	2.50	1.75	2.50	1.00
50	50	50	50	50	50	1.20	3.00	2.10	3.00	1.20
40	40	40	40	40	40	1.50	3.75	2.63	3.75	1.50
30	30	30	30	30	30	2.00	5.00	3.50	5.00	2.00
20	20	20	20	20	20	3.00	7.50	5.25	7.50	3.00
10	10	10	10	10	10	6.00	15.00	10.50	15.00	6.00

Table 4.1 summarizes the adjusted speed values and the corresponding travel times. The first column gives the artificial speed limits in km/h. We take the same artificial limits that are used in Qian and Eglese (2014). Columns [2-6] give the adjusted speed in km/h whereas the following columns give the travel time in minutes for each arc separately. The AS method of Qian and Eglese (2014) calculates the fastest path for each artificial speed setting in order to generate suitable potential candidates for the original problem.

Table 4.2. Travel time of alternative paths under different artificial speed limits

Artificial Speed Limit (km/h)	Travel Time (min)		
	(1-3-4)	(1-2-4)	(1-2-3-4)
120	2.52	3.46	2.52
110	2.52	3.46	2.52
100	2.52	3.46	2.52
90	2.52	3.46	2.52
80	2.73	3.46	2.73
70	3.00	3.46	3.00
60	3.50	3.75	3.50
50	4.20	4.50	4.20
40	5.25	5.63	5.25
30	7.00	7.50	7.00
20	10.50	11.25	10.50
10	21.00	22.50	21.00

Table 4.2 gives the travel time of each path for each artificial speed limit. Note that path 1-2-3-4, the minimum cost path with 687.47 g, is also the slowest path for each

speed setting. Thus, it is never selected as a candidate route. Even after the speed adjustment process, the cost of the paths 1-3-4 and 1-2-4 will be higher compared to the optimal path 1-2-3-4.

With this motivation, we first develop and solve the mathematical formulation of the TDMCP-S in a more general context. Our model is more generic than Franceschetti et al. (2013) who state their modelling framework to best fit for routing problems which must be executed in the first half of the day. Next, we propose a discrete time-space-speed expansion model for solving TDMCP-S on a discrete network.

### 4.3 Problem Definition and Formulations for the TDSPP

A time dependent directed network is defined as  $G(N, A, W)$  where  $N = \{1, 2, \dots, n\}$  is the set of nodes,  $A \subseteq N \times N$  is the set of arcs and  $W$  is a positive valued function. For every arc  $(i, j) \in A$ , there is a function  $w_{ij}(t) \in W$  where  $t$  is a time variable in a time domain  $T$ . A travel time function  $w_{ij}(t)$  specifies the travel time of the travel between  $i$  and  $j$  departing at time  $t$ . The total travel time of a path  $P$  visiting nodes  $i \in \tilde{N} \subseteq N$  is given by  $\sum_{i=0}^{|\tilde{N}|-1} w_{p_i p_{i+1}}(\alpha_i)$  where  $p_i$  and  $\alpha_i$  denote the  $i^{\text{th}}$  node in the path and the departure time from that node respectively. The departure times are calculated as  $\alpha_{i+1} = \alpha_i + w_{p_i p_{i+1}}(\alpha_j)$ . In this study, we assume that the first node  $p_0$  is left at time 0. In other words, we set  $\alpha_0 = 0$ .

The TDSPP seeks for the fastest path between two predetermined nodes on  $G$ . In this section, we first give alternative mathematical programming models for solving the TDSPP. After giving the definition of the more generic time-dependent minimum cost problem, we show that, prior to some modifications, the models for the TDSPP can be used to solve the generic model.

Table 4.3. Notation used in this section.

Variable	Description
$S, D$	: Source and destination nodes
$x_{ij}$	: Binary flow variable for arc $(i, j)$
$x_{ij}^t$	: Binary flow variable for arc $(i, j)$ departing at time $t$
$\tau_i$	: Arrival time at node $i$
$FS(i)$	: $j   (i, j) \in A$
$BS(i)$	: $j   (j, i) \in A$
$brp_{ij}^k$	: Time of the $k^{\text{th}}$ breakpoint for arc $(i, j)$
$brvw_{ij}^k$	: Value of the $k^{\text{th}}$ breakpoint for arc $(i, j)$
$\delta_{ij}^k$	: Binary variables to select which piece of the piecewise linear function is active for a given departure time $\tau_i$ from node $i$



The set of common variables and parameters that will be used in this section in addition to these in the problem definition are summarized in Table 3. We will use index  $t$  to represent time and so define the source and destination pair with symbols  $S$  and  $D$  unlike the traditional symbols  $s$  and  $t$  to prevent disambiguation.

To the best of our knowledge, Nannicini (2009) is the only work that *solves* TDSPP using mixed integer linear programming (MILP). Stating that it is hard to solve large problems to optimality, they define their aim as to compare the traditional branching rules with the proposed methods rather than solving network problems. Their model for minimizing time on a time-dependent network is given as follows;

*Model 1:*

$$\begin{aligned}
& \text{minimize} && \tau_D && (1) \\
& \text{s.t.} && && \\
& && \sum_{j \in BS(i)} x_{ji} - \sum_{j \in FS(i)} x_{ij} = \begin{cases} -1, & i = S \\ 0, & i \in V \setminus \{S, D\} \\ 1, & i = D \end{cases} && (2) \\
& && \tau_i \leq \alpha_i && \forall i \in N && (3) \\
& && x_{ij} \left( \alpha_i + \sum_{k=1}^{|H|-1} \delta_{ij}^k \left( \frac{\alpha_i - brp_{ij}^k}{brp_{ij}^{k+1} - brp_{ij}^k} (brvw_{ij}^{k+1} - brvw_{ij}^k) + brvw_{ij}^k \right) \right) \leq \tau_j && \forall (i, j) \in A && (4) \\
& && \sum_{k=1}^{|H|-1} \delta_{ij}^k brp_{ij}^k + \delta_{ij}^{|H|} M \geq \alpha_i && \forall (i, j) \in A && (5) \\
& && \sum_{k=2}^{|H|} \delta_{ij}^k brp_{ij}^{k-1} \leq \alpha_i && \forall (i, j) \in A && (6) \\
& && \sum_{k=1}^{|H|} \delta_{ij}^k = 1 && \forall (i, j) \in A && (7) \\
& && x_{ij} \in \{0, 1\} && \forall (i, j) \in A && (8) \\
& && \delta_{ij}^k \in \{0, 1\} && \forall (i, j) \in A, k \in H && (9) \\
& && \tau_i \geq 0 && \forall i \in N && (10)
\end{aligned}$$

where  $H$  refers to the set of breakpoints related with the arc  $(i, j)$ . Note that although different time limits and hence breakpoints can be associated with each arc (introducing  $h_{ij}$ ), we assume all of the breakpoints in time to be equal for all arcs without loss of generality.

Although Nannicini (2009) define a generic model where waiting times are allowed, we currently prohibit waiting (i.e.  $\alpha_i = \tau_i \forall i \in N$ ). As a result, (3) becomes  $\tau_i \leq \tau_i \forall i \in N$  and thus can be neglected. We also omitted the fixed static cost in (4) without loss of generality.

A time-dependent directed network with additional time-dependent cost is defined as  $G'(N, A, W, C)$  where positive valued cost functions  $c_{ij}(t) \in C$  are introduced for

every arc  $(i, j) \in A$  and  $t \in T$ . The cost function  $c_{ij}(t)$  specifies the cost of traveling from  $i$  to  $j$  departing at time  $t$ . The total cost of a path  $P$  visiting nodes  $n_i \in \tilde{N} \subseteq N$  is given by  $\sum_{i=0}^{|\tilde{N}|-1} c_{p_i p_j}(\alpha_i)$ .

The time-dependent minimum cost path problem (TDMCP) seeks for the minimum cost path between two predetermined nodes on  $G'$  where the objective is different than the travel time minimization. Note that, by setting  $C$  equal to  $W$ , the TDMCP can be reduced to TDSPP.

In order to minimize a generic cost function rather than the travel time, we define a new variable  $C_i$  which corresponds to the total (cumulative) cost of traveling from the source node  $S$  to  $i$ . We replace  $\tau_D$  in (1) with  $C_D$  and add the following constraint;

$$x_{ij} \left( C_i + \sum_{k=1}^{|H|-1} \delta_{ij}^k \left( \frac{\tau_i - brp_{ij}^k}{brp_{ij}^{k+1} - brp_{ij}^k} (brvc_{ij}^{k+1} - brvc_{ij}^k) + brvc_{ij}^k \right) \right) \leq C_j \quad \forall (i, j) \in A \quad (11)$$

where  $brvc_{ij}^k$  refers to the value of the  $k^{\text{th}}$  cost breakpoint for the cost function between  $i$  and  $j$ . Note that the breakpoints of  $c_{ij}(t)$  and  $w_{ij}(t)$  overlap with each other. An illustration is given in Figure 4.2 for a single arc with a length of 60 km. Therefore, we do not define a new breakpoint position variable for cost function. We refer the modified model as Model 1.

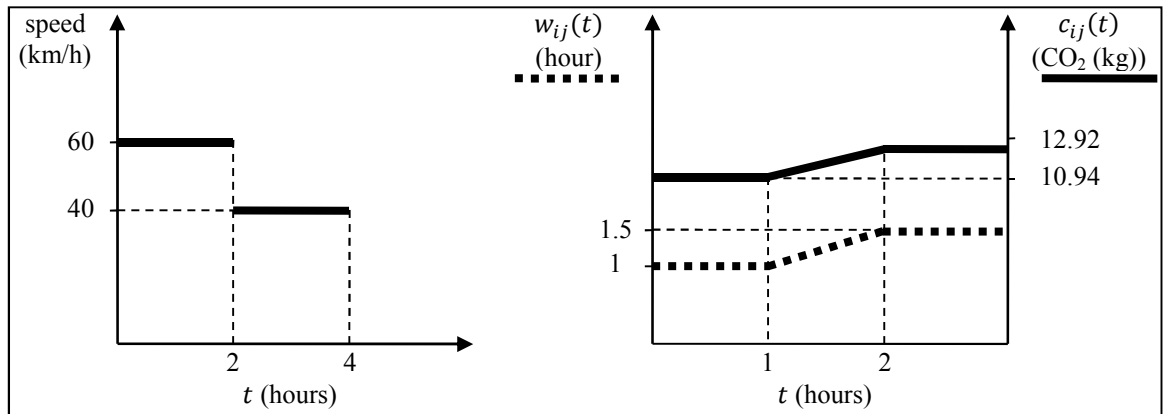


Figure 4.2. Overlapping of travel time and cost functions breakpoints. The arc length is 60 km. Cost values for the corresponding speeds are calculated using (Hickman et al., 1999).

Alternatively, we give the following model (Model 2) where the time domain  $T$  is discretized into time intervals  $T_m = m\mu$  where  $T_m \in T$  refers to the  $m^{\text{th}}$  time interval and  $\mu$  is the smallest time unit used for discretization. In other words, the travel can only start at a time that is a multiple of  $\mu$ . Also  $w_{ij}(t) \in W$  are rounded up to the nearest integer that is multiple of  $\mu$ . The total number of intervals is denoted with  $|T|$  in this context.

*Model 2:*

$$\text{minimize } \sum_{(i,j) \in A} \sum_{t \in T} x_{ij}^t c_{ij}(t) \quad (12)$$

$$\text{s. t. } \sum_{j \in BS(i)} \sum_{t \in T} x_{ji}^t - \sum_{j \in FS(i)} \sum_{t \in T} x_{ij}^t = \begin{cases} -1, & i = S \\ 0, & i \in N \setminus \{S, D\} \\ 1, & i = D \end{cases} \quad (13)$$

$$\sum_{i \in BS(j)} \sum_{t \in T} (t + w_{ij}(t)) x_{ij}^t = \tau_j \quad \forall j \in N \setminus \{S\} \quad (14)$$

$$\sum_{t \in T} (\tau_i - t) x_{ij}^t = 0 \quad \forall (i,j) \in A \quad (15)$$

$$x_{ij}^t \in \{0,1\} \quad \forall (i,j) \in A, t \in T \quad (16)$$

$$\tau_i \geq 0 \quad \forall i \in N \quad (17)$$

Both Model 1 and Model 2 have nonlinear terms. In Model 1, (4) involves a product between a binary variable and a continuous variable, and between two binary variables and a continuous variable. In Model 2, (15) violates linearity involving a product between a binary variable and a continuous variable. Next, we show the linearization of (15) following Liberti et al. (2009). The nonlinearities in other models can be handled in a similar manner. We introduce a new continuous variable  $q_{ij}^t, \forall (i,j) \in A, t \in T$  and replace (15) with the following set of constraints to obtain

*Model 3:*

$$\sum_{t \in T} q_{ij}^t = \sum_{t \in T} t x_{ij}^t \quad \forall (i,j) \in A \quad (18)$$

$$q_{ij}^t = |T| x_{ij}^t \quad \forall (i,j) \in A, t \in T \quad (19)$$

$$q_{ij}^t \geq 0 \quad \forall (i,j) \in A, t \in T \quad (20)$$

$$q_{ij}^t \leq \tau_i \quad \forall (i,j) \in A, t \in T \quad (21)$$

$$q_{ij}^t \geq \tau_i - (1 - x_{ij}^t) |T| \quad \forall (i,j) \in A, t \in T \quad (22)$$

In their computational study Fleischmann et al. (2004) conduct tests with travel time data obtained from a traffic information system in the city of Berlin. They divide planning horizon into time slots and reports that with only five time slots, a rather good

approximation of the true travel times is reached, which is not significantly improved with 10 time slots. Taking this observation into account, Model 1 is better where the travel time function is modelled using breakpoints. The number of time periods in time discretization is very high compared to using breakpoints. This also justifies Model 1. Yet, we think that Model 2, Model 3 and variations can be used for evaluating the performance of heuristic and/or dynamic programming approaches where the time is also discretized or can somehow be integrated with any of these models.

Note that in TDSPP model, the objective function will prohibit any cycles as it is never advantageous in a time minimization setting. However, as shown in Section 2.6.4, cycling can reduce the total cost in TDMCP networks. As a result, the solution of a TDMCP model or its variation can include cycles.

#### 4.4 Problem Definition and Formulation for the TDMCP-S

The network  $G'(N, A, W, C)$  with the time-dependent travel time function  $w_{ij}(t) \in W$  inherently comprises distance and time-dependent speed information although they are not involved in the network definition explicitly. However, only a single speed value is applicable at each time instant  $t \in T$ . When lower and upper speed limits are also introduced, we obtain a new time-dependent directed network with time-dependent speed limits which is defined as  $G''(N, A, D, \bar{V}, \underline{V}, F)$  where  $D$ ,  $\bar{V}$ ,  $\underline{V}$ , and  $F$  are positive valued functions. Functions  $d_{ij} \in D$ ,  $\bar{v}_{ij}(k) \in \bar{V}$  and  $\underline{v}_{ij}(k) \in \underline{V}$  where  $k \in H'$ , with  $H'$  being the set of speed breakpoints in time for arc  $(i, j)$ , are defined for every arc  $(i, j) \in A$ . A speed lower limit function  $\underline{v}_{ij}(k)$  and a speed upper limit function  $\bar{v}_{ij}(k)$  specify the lower and upper speed limits respectively for traveling from  $i$  to  $j$  departing in time interval  $k$ . We assume equal time interval lengths again without loss of generality. The distance of an arc  $(i, j)$  is given by a distance function  $d_{ij}$  whereas the travel time is calculated implicitly using  $d_{ij}$  and the selected travel speed  $v_{ij}$ . A cost function  $F(v_{ij})$  gives the per km cost of traveling from  $i$  to  $j$  with speed  $v_{ij}$ , regardless of the departure time. The total cost of the travel between  $i$  and  $j$  is simply given by  $d_{ij}F(v_{ij})$ .

Note that the so called speed limits in this study do not correspond only to the limit of speed allowed by law but also those incurred by congestion. Therefore, the lower and upper speed limits on a congested highway can be 20 km/h and 60km/h respectively whereas the uncongested lower and upper limits speed limits are 40 km/h

and 90 km/h. The lower limit in the latter uncongested case is set to maintain the reasonable flow of traffic. We take these limits as given and do not try to optimise them for cost minimization.

The time-dependent minimum cost path problem with speed as a decision variable (TDMCP-S) seeks for the minimum cost path between two predetermined nodes on  $G''$  while deciding which arcs to traverse as well as the speed to traverse these arcs.

When the speed upper limit is higher than the optimal speed (in terms of emission), one greedy approach is to select the optimal speed. Although the latter seems promising from cost point of view, it is myopic and may yield higher cost. Figure 4.3 illustrates such an example. The cost information for the corresponding network is given in Table 4.4.

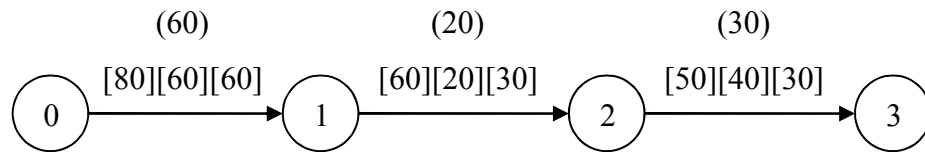


Figure 4.3. Speed decision example

Table 4.4. GHG emissions for varying speeds

Speed (km/s)	80	70	60	50	40	30	20
GHG (unit/km)	0.8	0.7	0.5	0.8	0.9	1.2	1.4

The planning horizon is divided into three equal intervals of length 1, namely  $Z_1 = [0,1)$ ,  $Z_2 = [1,2)$  and  $Z_3 = [2,3)$ . On the first arc (0-1), a vehicle (V1) departing at  $t = 0$  and traveling with the maximum speed of 80 km/h reaches node 1 at time 0.75 with a GHG value of 48 whereas another vehicle (V2) traveling with the most efficient speed of 60 km/h reaches node 1 at time 1.00 with a GHG value of 30. On the second arc (1-2), V1 travels with 60 km/h and 20 km/h for 0.25 hours for each and reaches a GHG value of 62.5. Traveling in the second time interval V2 reaches node 3 at time 2.00 with a cumulative GHG of 58. On the last arc (2-3), V1 finishes its travel in the second time interval with a total GHG value of 89.5, yet V2 starts and finishes its travel in the third time interval and yields GHG value of 94, 4.5 more than V1. In other words, traveling with the optimal speed on the current arc yielded a higher cost on the

following arcs. Keeping this in mind, we will develop a mathematical model to solve the TDMCP-S to optimality.

One approach for the formulation of the TDMCP-S is to add a new index for the flow variables (e.g.  $x_{ij}^{tv}$ ). A discrete speed set  $\hat{V}_{ij}(t)$  is associated with every arc  $(i, j)$  and every discrete point  $t \in T$  to specify the potential speeds available. Here,  $\hat{V}_{ij}(t) = \{\underline{v}_{ij}(t), \underline{v}_{ij}(t) + \theta, \underline{v}_{ij}(t) + 2\theta, \dots, \bar{v}_{ij}(t)\}$  where  $\theta$  represents the speed increment to create discrete speeds for the corresponding arc and travel time. Note that the difference between the speeds is assumed to be equal for the notational brevity. As the speed limits are embedded in the discrete speed set  $\hat{V}_{ij}(t)$ , the model does not include additional constraints for the speed limits. Though trivial, this approach makes the model harder to solve as the model already has a large number of variables. We refer to this trivial model as Model 4.

*Model 4:*

$$\min \sum_{(i,j) \in A} \sum_{t \in T} \sum_{v \in \hat{V}_{ij}(t)} x_{ij}^{tv} c_{ij}(t) \quad (23)$$

$$\text{s.t.} \quad \sum_{j \in BS(i)} \sum_{t \in T} \sum_{v \in \hat{V}_{ji}(t)} x_{ji}^{tv} - \sum_{j \in FS(i)} \sum_{t \in T} \sum_{v \in \hat{V}_{ij}(t)} x_{ij}^{tv} = \begin{cases} -1, & i = S \\ 0, & i \in N \setminus \{S, D\} \\ 1, & i = D \end{cases} \quad (24)$$

$$\sum_{i \in BS(j)} \sum_{t \in T} \sum_{v \in \hat{V}_{ij}(t)} \left( t + \frac{d_{ij}}{v} \right) x_{ij}^{tv} = \tau_j \quad \forall j \in N \setminus \{S\} \quad (25)$$

$$\sum_{t \in T} \sum_{v \in \hat{V}_{ij}(t)} (\tau_i - t) x_{ij}^{tv} = 0 \quad \forall (i, j) \in A \quad (26)$$

$$x_{ij}^{tv} \in \{0, 1\} \quad \forall (i, j) \in A, t \in T, v \in \hat{V}_{ij}(t) \quad (27)$$

$$\tau_i \geq 0 \quad \forall i \in N \quad (28)$$

We now propose a new formulation (Model 5) for the TDMCP-S as follows which does not require the discretization of the speed and thus, needs additional lower and upper speed limit constraints that are given by the inequalities (32) and (33) respectively.

Model 5:

$$\min \sum_{(i,j) \in A} x_{ij} d_{ij} F(v_{ij}) \quad (29)$$

$$\text{s. t.} \quad \sum_{j \in BS(i)} x_{ji} - \sum_{j \in FS(i)} x_{ij} = \begin{cases} -1, & i = S \\ 0, & i \in N \setminus \{S, D\} \\ 1, & i = D \end{cases} \quad (30)$$

$$\sum_{i \in BS(j)} \left( \tau_i + \frac{d_{ij}}{v_{ij}} \right) x_{ij} = \tau_j \quad \forall j \in N \quad (31)$$

$$\sum_{k=1}^{|H|} \delta_{ij}^k \bar{v}_{ij}(k) \geq v_{ij} \quad \forall (i,j) \in A \quad (32)$$

$$\sum_{k=1}^{|H|} \delta_{ij}^k \underline{v}_{ij}(k) \leq v_{ij} \quad \forall (i,j) \in A \quad (33)$$

$$\sum_{k=1}^{|H|-1} \delta_{ij}^k brp_{ij}^k + \delta_{ij}^{|H|} M \geq \tau_i \quad \forall (i,j) \in A \quad (34)$$

$$\sum_{k=2}^{|H|} \delta_{ij}^k brp_{ij}^{k-1} \leq \tau_i \quad \forall (i,j) \in A \quad (35)$$

$$\sum_{k=1}^{|H|} \delta_{ij}^k = 1 \quad \forall (i,j) \in A \quad (36)$$

$$x_{ij} \in \{0,1\} \quad \forall (i,j) \in A \quad (37)$$

$$\delta_{ij}^k \in \{0,1\} \quad \forall (i,j) \in A, \forall k \in H \quad (38)$$

$$v_{ij} \geq 0 \quad \forall (i,j) \in A \quad (39)$$

$$\tau_i \geq 0 \quad \forall i \in N \quad (40)$$

where  $M$  refers to a large number. As  $M$  is used as an upper bound on the arrival times  $\tau_i$ ,  $i \in N$ , it can be set to the horizon length. Similar to the time-increment based dynamic programming model of Qian and Eglese (2014), we assume that each vehicle travels at a constant speed along each arc. On the contrary, the starting time of the travel on the arc is considered as the determining factor.

Constraints (31) ensure that the arrival times at nodes are consistent with each other taking the selected speeds into account. Alternatively, a departure time  $q_{ij}$  can be defined for the travel from node  $i$  to node  $j$ . We replace the constraints (31), (34) and (35) for the modified model and refer it as Model 6, which is given as follows;

Model 6:

$$\min \sum_{(i,j) \in A} d_{ij} F(v_{ij}) \quad (41)$$

$$\text{s. t. } \sum_{j \in BS(i)} x_{ji} - \sum_{j \in FS(i)} x_{ij} = \begin{cases} -1, & i = S \\ 0, & i \in N \setminus \{S, D\} \\ 1, & i = D \end{cases} \quad (42)$$

$$\sum_{i \in FS(j)} q_{ji} - \sum_{i \in BS(j)} q_{ij} = \sum_{i \in BS(j)} \left( \frac{d_{ij}}{v_{ij}} \right) x_{ij} \quad \forall j \in N \setminus \{S, D\} \quad (43)$$

$$L - \sum_{i \in BS(j)} q_{ij} \geq \sum_{i \in BS(j)} \left( \frac{d_{ij}}{v_{ij}} \right) x_{ij} \quad j = D \quad (44)$$

$$q_{Sj} = 0 \quad \forall j \in FS(S) \quad (45)$$

$$q_{ij} \leq Lx_{ij} \quad \forall (i,j) \in A, i \neq S \quad (46)$$

$$v_{ij} \leq Lx_{ij} \quad \forall (i,j) \in A \quad (47)$$

$$\sum_{k=1}^{|H|} \delta_{ij}^k \bar{v}_{ij}(k) \geq v_{ij} \quad \forall (i,j) \in A \quad (48)$$

$$\sum_{k=1}^{|H|} \delta_{ij}^k \underline{v}_{ij}(k) \leq v_{ij} \quad \forall (i,j) \in A \quad (49)$$

$$\sum_{k=1}^{|H|-1} \delta_{ij}^k brp_{ij}^k + \delta_{ij}^{|H|} M \geq \tau_i \quad \forall (i,j) \in A, i \neq S \quad (50)$$

$$\sum_{k=2}^{|H|} \delta_{ij}^k brp_{ij}^{k-1} \leq \tau_i \quad \forall (i,j) \in A, i \neq S \quad (51)$$

$$\sum_{k=1}^{|H|} \delta_{ij}^k = 1 \quad \forall (i,j) \in A \quad (52)$$

$$x_{ij} \in \{0,1\} \quad \forall (i,j) \in A \quad (53)$$

$$\delta_{ij}^k \in \{0,1\} \quad \forall (i,j) \in A, \forall k \in H \quad (54)$$

$$v_{ij}, q_{ij} \geq 0 \quad \forall (i,j) \in A \quad (55)$$

$$\tau_i \geq 0 \quad \forall i \in N \quad (56)$$

where  $L$  refers to the length of the horizon.

#### 4.5 Time-space-speed expansion

This section briefly introduces the concept of time-space-speed network which is based on the time-space network with embedded speed. Note that the speed optimization can be embedded without increasing the number of nodes but introducing a new arc for each different speed value.

We discretize the speed as well as the travel time values due to the inherent structure of the time-space expansion. Then, the set of arcs in the time-space-speed is defined as  $A_d = \{(i_t, j_{t'}) \mid (i,j) \in A, t + w_{ij}^{tv} = t', t \leq t', t, t' \in T, v \in V_{ij}\}$  where  $T$  and  $V_{ij}, (i,j) \in A$  refer to the discrete time and speed domains respectively. The time and



speed dependent variable  $w_{ij}^{tv}$  refers to the travel time between nodes  $i$  and  $j$  departing node  $i$  at time  $t$  and traveling with speed  $v$ .

The speed and travel time information for a sample arc is given Table 4.5. The direction of the travel is from node A to node B and the distance of the arc is 2 km. The lower and the upper speed limits are 10 km/h and 40 km/h respectively.

Table 4.5. Speed and travel time information for a sample arc

Speed (km/h)	Travel time (minute)
10	12
20	6
30	4
40	3

The corresponding time-space-speed network of the sample arc is given in Figure 4.4. These networks are used either explicitly or implicitly in time-dependent context. As the shortest path problem (SPP) can be solved very efficiently even for large instances, a maximum number of  $|T| \times |T|$  SPP's can be solved to solve the TDMCP\_S (much less in the sample network as many departure times are eliminated beforehand for illustrative purposes).

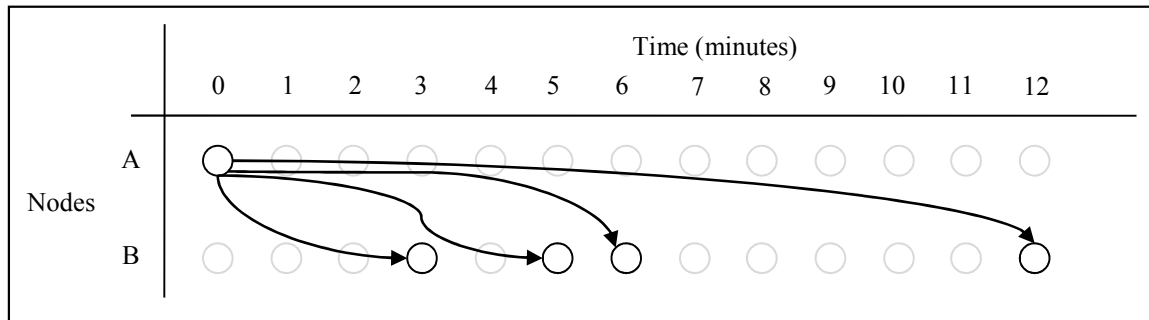


Figure 4.4. Time-space-speed network of the sample arc.

#### 4.6 Computational Study

In this section, we test the performance of Model 2 and Model 4 and the time-space-speed network approach by testing them on different sized networks. Note that, Model 2 and the time-space-speed network approach uses a discretized network whereas Model 4 works on the continuous network.

#### 4.6.1 Computational setup

The models are tested on unidirectional grid-type networks. Three different base speed values are created randomly on each arc; 40 km/h, 60km/h, and 80 km/h. And then, adding 50% decreased and increased speeds, the discrete available speeds are obtained for each arc. Note that the discretization may prevent the discrete methods to travel with the optimal speed. Also, three different density schemes namely density=1, density=2 and density=4, given in Section 2.6.1.1, are used.

The MILP models are solved using IBM CPLEX 12.6 on an Intel Core2 Quad 2.33 GHz computer with 8.0 GB RAM and 64-bit operating system. For the MINLP models, GAMS 24.2.3 is used as the modelling environment and the models are solved on the Network-Enabled Optimization System (NEOS) Server.

#### 4.6.2 Comparison of the proposed methods

The performances of the proposed approaches are compared in Table 4.6. OF corresponds to the objective function value, CO<sub>2</sub> emission in grams, whereas CT represents the computational time in milliseconds.

Table 4.6. Performance comparison for the proposed approaches

Network Size	Density Level	Time-space-speed network approach		MILP (Model 2)		MINLP (Model 4)	
		OF (gr)	CT (ms)	OF (gr)	CT (ms)	OF (gr)	CT (ms)
4	1	3936.9	~0	3936.9	51	3922.6	~0
	2	1312.3	~0	1312.3	70	1307.5	~0
	4	6561.6	~0	6561.6	90	6542.7	1
9	1	5601.5	5	5601.5	1437	5230.1	1
	2	9538.5	7	9538.5	4198	9152.0	*
	4	15149.6	5	15149.6	2536	14388.0	1
16	1	9538.5	18	9538.5	11808	9183.3	~0
	2	12636.0	24	12636.0	18059	-	**
	4	19802.4	15	19802.3	12889	19613.0	*
25	1	17060.3	46	17060.3	27652	-	**
	2	10733.4	45	10733.4	46163	-	**
	4	24688.1	38	24688.1	135468	-	**

\* Best possible, \*\* Node limit exceeded.

With the increasing size of the network and the density, the computational effort for solving the MILP model increase dramatically. Nevertheless, the time-space-speed approach can obtain the same solution in under a second whereas the computational time for solving MILP can reach up to 135.5 seconds. Although the MINLP model can also generate solutions in short time, we could not obtain any solution when the network size of the instance is 25. Note that, the objective function of the MINLP model is lower than the two other methods due to the discretization of speed used in time-space-speed approach and MILP model.

#### **4.7 Conclusion and Future Research**

Many of the time-dependent methods in the routing and path finding literature assume a constant travel speed which is equal to the speed limit of the corresponding road section. In this chapter, we relaxed this assumption and showed that traveling at a different speed than the speed limit can generate better solutions from the GHG point of view.

We showed that, in spite of being developed by taking the speed as a decision variable, the currently available heuristic methods may fail to obtain the optimal solution. Thus, we first discussed the currently available mathematical formulations for the TDSPP and developed a mathematical model for the TDMCP-S where the speeds on the arcs are also decision variables as well as the departure times.

We further proposed a time-space-speed expansion method which, under a certain speed and time discretization scheme, can obtain the optimal solution. Yet, as the computational study proves, this discretization has an important effect on the solution quality and may cause to miss the optimal solution.

For future research, we will test the relative performance of the proposed approaches compared to Qian and Eglese (2014) who also uses a discretized speed scheme. Also we plan to apply a Lagrangian relaxation and analyze the strength of the bounds that will be proposed. Also the performance of the relaxation as well as the solution quality and computational performance will be tested.

#### **4.8 References**

Figliozzi, M. A. (2010). "Vehicle Routing Problem for Emissions Minimization." Transportation Research Record: Journal of the Transportation Research Board **2197**: 1-7.

- Figliozzi, M. A. (2011). "The impacts of congestion on time-definitive urban freight distribution networks CO2 emission levels: Results from a case study in Portland, Oregon." Transportation Research Part C: Emerging Technologies **19**(5): 766-778.
- Fleischmann, B., M. Gietz and S. Gnutzmann (2004). "Time-Varying Travel Times in Vehicle Routing." Transportation Science **38**(2): 160-173.
- Franceschetti, A., D. Honhon, T. Van Woensel, T. Bektaş and G. Laporte (2013). "The time-dependent pollution-routing problem." Transportation Research Part B: Methodological **56**(0): 265-293.
- Hickman, J., C. Hassel, R. Joumard, Z. Samaras and S. Sorenson (1999). MEET Methodology for Calculating Transport Emissions and Energy Consumption. Technical Report.
- Hvattum, L. M., I. Norstad, K. Fagerholt and G. Laporte (2013). "Analysis of an exact algorithm for the vessel speed optimization problem." Networks **62**(2): 132-135.
- Jabali, O., T. Van Woensel and A. G. de Kok (2012). "Analysis of Travel Times and CO2 Emissions in Time-Dependent Vehicle Routing." Production and Operations Management **21**(6): 1060-1074.
- Liberti, L., S. Cafieri and F. Tarissan (2009). Reformulations in Mathematical Programming: A Computational Approach. Foundations of Computational Intelligence Volume 3. A. Abraham, A.-E. Hassanien, P. Siarry and A. Engelbrecht, Springer Berlin Heidelberg. **203**: 153-234.
- Maden, W., R. Eglese and D. Black (2010). "Vehicle Routing and Scheduling with Time Varying Data: A Case Study." Journal of the Operational Research Society **61**(3): 515-522.
- Nannicini, G. (2009). Point-to-Point Shortest Paths on Dynamic Time-Dependent Road Networks. PhD, Ecole Polytechnique, Palaiseau.
- Norstad, I., K. Fagerholt and G. Laporte (2011). "Tramp ship routing and scheduling with speed optimization." Transportation Research Part C: Emerging Technologies **19**(5): 853-865.
- Qian, J. and R. Eglese (2014). "Finding least fuel emission paths in a network with time-varying speeds." Networks **63**(1): 96-106.
- Wikipedia. (2014). "Google driverless car." Retrieved June 01, 2014, from [http://en.wikipedia.org/wiki/Google\\_driverless\\_car](http://en.wikipedia.org/wiki/Google_driverless_car).

## **5 A TIME-BASED PHEROMONE APPROACH FOR THE ANT SYSTEM**

# A Time-based Pheromone Approach for the Ant System

Umman Mahir Yıldırım, Bülent Çatay

## Abstract

The ant system (AS) is a metaheuristic approach originally developed for solving the traveling salesman problem. AS has been successfully applied to various hard combinatorial optimization problems and different variants have been proposed in the literature. In this study, we introduce a time-based pheromone approach for AS (TbAS). Due to this nature, TbAS is applicable to routing problems involving time-windows. The novelty in TbAS is the multi-layer pheromone network structure which implicitly utilizes the service time information associated with the customers as a heuristic information. To investigate the performance of TbAS, we use the well-known vehicle routing problem with time-windows as our test bed and we conduct an extensive computational study using the Solomon (1987) instances. Our results reveal that the proposed time-based pheromone approach is effective in obtaining good quality solutions.

**Keywords** Ant systems, Vehicle routing, Time windows, Metaheuristics, Ant colony optimization

## 1. Introduction

Ant colony optimization (ACO) is a population-based metaheuristic inspired from the foraging behaviour of ants. It simulates this natural behaviour of real ants to solve combinatorial optimization problems by using artificial ants. To apply ACO, the optimization problem is transformed into the problem of finding the best path on a weighted graph. The artificial ants incrementally build solutions by moving on the graph using a stochastic construction process guided by artificial pheromone and heuristic information known as visibility (Dorigo, 2010). The ant system (AS) is the first ACO approach developed for solving the traveling salesman problem (TSP) (Dorigo et al., 1996). Some early applications include the elitist strategy for AS (EAS) (Dorigo et al., 1996), rank-based version of AS (ASrank) (Bullnheimer et al., 1999a; Bullnheimer et al., 1999b), MAX-MIN AS (MMAS) (Stützle and Hoos, 1997), and ant colony system (ACS) (Dorigo and Gambardella, 1997). Although the ant algorithms are designed as constructive methods, Zufferey (2011) has recently shown that the ants can also be efficiently used in a local search setting.

In this chapter, we propose a new AS which uses a time-based pheromone approach, namely TbAS. TbAS takes into account the time-window nature of the routing problem in the visibility mechanism of AS. Basically, it may not be time-wise

desirable to travel from one customer to the next after at a certain point in the scheduling horizon. However, the traditional AS assigns the same probability for visiting a customer after another at any time, which may lead to unnecessary waiting times and poorer solutions. Using this motivation, we introduce a multi-layer pheromone network structure in an attempt to distinguish the pheromone levels belonging to different time intervals and utilize the timing of the visit as an implicit heuristic information in the route construction phase. To the best of our knowledge, this is the first AS approach that utilizes “time” as the heuristic information.

Due to its time-based pheromone nature, TbAS is applicable to vehicle routing problems involving time-windows. In this study, we investigate the performance of TbAS using the vehicle routing problem with time windows (VRPTW). VRPTW is a well-known vehicle routing problem (VRP) which has been extensively studied in the literature. It determines a set of routes that belongs to a homogeneous fleet of  $M$  vehicles with capacity  $Q$ , to serve  $n$  geographically dispersed customers. Each customer  $i$  has a demand ( $q_i$ ), a service time ( $s_i$ ) and a service time window ( $[e_i, l_i]$ ) in which the customer must be served. Although some models allow early or late servicing with a penalty cost (soft time windows), the time window is assumed to be strict in this study (hard time windows). Thus, any vehicle arriving at customer  $i$  before  $e_i$  must wait until  $e_i$ . The vehicles reside in a depot denoted with 0 which also has its own time windows;  $[e_0, l_0]$ . This time window implies that any vehicle must leave the depot after  $e_0$  and must return to the depot before  $l_0$ . The demands of the customers must be satisfied such that each customer is serviced exactly once by exactly one vehicle, each route originates and terminates at the depot, and total demand of the customers assigned to each route must not exceed the vehicle capacity. The interested reader is referred to Toth and Vigo (2002) for more details and the formulation of the problem.

VRPTW has been extensively studied over the last three decades and different exact and heuristic methods have been proposed in the literature. While the exact methods aim at minimizing the total travel distance almost all heuristic methods employ the hierarchical objective approach where minimizing the number of vehicles (routes) is the primary objective and minimizing total travel distance is the secondary. However, research on heuristic methods focusing on the second objective has recently gained momentum. Alvarenga et al. (2007) emphasized the lack of heuristic approaches using only the second objective and justified the need for research effort in this direction by providing different real-life examples from Brazil where the minimization of the travel

distance is appropriate. We have faced similar situations in different business environments in our country as well. Many companies outsource the collection of the raw materials and components and the delivery of the products to third party logistics (3PL) service providers or contractors. They either make annual or longer term contracts usually requiring a fleet dedicated to their operations or hire trucks from small businesses or the spot market where individually owned trucks are available. In the former case, they are charged with a fixed cost depending on the fleet size and type and a variable cost per kilometer basis. Then, the minimization of the total distance is the appropriate objective when the total capacity of the fleet is larger than the items to be collected or delivered. In the latter case, hired trucks are paid on a kilometer basis. In any case, minimizing the total travel distance arises as the primary and sole objective, which constitutes the motivation of this study.

The remainder of the chapter is organized as follows: Section 2 provides a brief review of the relevant literature. Section 3 describes the mechanisms of TbAS. Section 4 presents an extensive computational study to test its performance and reports detailed results. Finally, concluding remarks and future research directions are given in Section 5.

## **2. Relevant Literature**

Since the hierarchical objective approach is out of the scope of this study, we review the literature related with the distance minimization objective and refer the interested reader to Toth and Vigo (2002), Bräysy and Gendreau (2005a), and Bräysy and Gendreau (2005b) for an extensive review of route construction, local search (LS), and metaheuristic approaches proposed for solving the hierarchical objective problem.

The edge exchange procedures mainly developed for TSP were implemented for VRPTW first (Savelsbergh, 1992). Rousseau et al. (2002) proposed a variable neighbourhood search (VNS) approach while Tan et al. (2001) compared the performances of four different approaches, namely  $\lambda$ -interchange local search, simulated annealing (SA), tabu search (TS), and genetic algorithm (GA). Utilizing the insertion heuristic 1 of Solomon (1987) and applying a local-post optimization, Jung and Moon (2002) developed a hybrid GA and reported very good results with considerably little computational effort. Pisinger and Ropke (2007) proposed an adaptive LNS framework an extension of the large neighbourhood search (LNS) of Shaw (1998). Ombuki et al. (2006) addressed VRPTW as a dual-objective problem and



proposed a GA approach using Pareto ranking technique. Alvarenga et al. (2007) proposed a two-phase column generation heuristic (CGH) that comprised an efficient GA and a set partitioning formulation. In Brandão de Oliveira and Vasconcelos (2010) a hybrid solution approach consisting of SA, hill climbing (HC), and random restart procedures was developed.

Recently, Muter et al. (2010) presented an algorithmic framework (MetaOpt) that combines metaheuristics with exact algorithms. In this framework, the metaheuristic searches for new solutions and the newly found columns are introduced into the exact algorithm. The information extracted from the exact algorithm is fed back to the metaheuristic as a guiding mechanism to better search the solution space. More recently, Garcia-Najera and Bullinaria (2011) has proposed an evolutionary algorithm (EA) to optimize the total distance and total number of vehicles simultaneously. They designed a bi-objective EA to obtain pareto optimal solutions by the help of similarity measures. The distance (time) minimization objective has also been considered in various VRPTW variants, within the context of soft time windows (e.g. (Koskosidis et al., 1992; Badeau et al., 1997)) and time-dependent travel times (e.g. (Fleischmann et al., 2004; Eglese et al., 2006)) as such.

The first ACO approach developed for VRPTW is the multiple ACS (MACS-VRPTW) of Gambardella et al. (1999). MACS-VRPTW is designed for solving VRPTW with hierarchical objective by using the coordination of two ant colonies simultaneously. The first colony, ACS-VEI, reduces the number of vehicles while the second, ACS-TIME, optimizes the travel times of the feasible solutions found by ACS-VEI. However, the two colonies utilize independent pheromone trails. Ellabib et al. (2002) proposed another ACO approach for the same problem where the basic idea is to let the ACS perform its search in the space of local minima rather than in the search space of all feasible tours. The VRPTW is transformed to TSP as in MACS-VRPTW.

### **3. Time-based ant system (TbAS)**

Ants in nature deposit an aromatic chemical called as pheromone on the path they walk. Pheromone is a communication mechanism among the ants and there is a positive correlation between the probability of a path being selected and the amount of pheromone on the path. In other words, the more pheromone on the path, the more ants follow that path, which further increases the pheromone level. The amount of the pheromone deposited depends on the quality of the food source as well. In time, all ants

are expected to follow the shortest path between the food source and their nest (Dorigo, 2010).

AS uses artificial ants to solve combinatorial optimization problems by simulating the above mentioned behaviour. The solution quality of the problem is analogous to the quality of the food source. Artificial ants incrementally build solutions utilizing a common memory. Imitating the trails in the real model, this memory can be read and modified by any of these ants (Pardalos and Resende, 2002). AS has two main components: pheromone trail intensity ( $\tau_{ij}$ ) and the visibility ( $\eta_{ij}$ ) between  $i$  and  $j$ , where  $i$  and  $j$  denote the customers in VRP (cities in TSP).  $\tau_{ij}$  is the common memory of artificial ants whereas  $\eta_{ij}$  is a heuristic information representing the desirability of visiting customer  $j$  after customer  $i$  and is set to the inverse of the distance between  $i$  and  $j$ .

In the route construction phase,  $K$  ants, which move in parallel, are initially placed at the  $K$  nearest customers to the depot. Each ant at a specific customer selects the next customer using the following random proportional rule:

$$p_{ij}^k = \frac{[\tau_{ij}]^\alpha [\eta_{ij}]^\beta}{\sum_{l \in N_i^k} [\tau_{il}]^\alpha [\eta_{il}]^\beta}, \quad \text{if } j \in N_i^k \quad (1)$$

where  $N_i^k$  denotes the set of not yet visited cities and the probability of selecting a city outside  $N_i^k$  is 0.  $\alpha$  and  $\beta$  are non-negative parameters to control the relative weight of pheromone information  $\tau_{ij}$  and heuristic information  $\eta_{ij}$ .

Usually a local search procedure is applied to further improve the routes obtained after each ant has constructed its tour. In the final phase, the pheromone trails are updated through the pheromone evaporation and pheromone reinforcement processes. The pheromone evaporation refers to uniformly decreasing the pheromone values on all arcs. The aim is to prevent the rapid convergence of the algorithm to a local optimal solution by reducing the probability of repeatedly selecting certain cities. The pheromone reinforcement process, on the other hand, allows each ant to deposit a certain amount of pheromone on the arcs belonging to its tour. The aim is to increase the probability of selecting the arcs frequently used by the ants that construct short tours. The pheromone update procedure is performed as follows:

$$\tau_{ij} \leftarrow (1 - \rho)\tau_{ij} + \sum_{k=1}^K \Delta \tau_{ij}^k, \quad \forall(i, j) \quad (2)$$

where  $\rho$  ( $0 < \rho \leq 1$ ) is the pheromone evaporation parameter, and  $(i, j)$  refers to the arc between cities  $i$  and  $j$ .  $\Delta \tau_{ij}^k$  is the amount of pheromone deposited on arc  $(i, j)$  by ant  $k$  and is computed as:

$$\Delta \tau_{ij}^k = \begin{cases} \frac{1}{L^k}, & \text{if ant } k \text{ uses arc } (i, j) \text{ on its tour} \\ 0, & \text{otherwise} \end{cases} \quad (3)$$

where  $L^k$  is the length of tour constructed by ant  $k$ .

This selection mechanism in equation (1) does not involve any information associated with the time of the visit. It may not be time-wise desirable to visit customer  $j$  after customer  $i$  after at a certain point in the scheduling horizon. However equation (1) assigns the same probability for visiting  $j$  after  $i$  at any time, which may lead to unnecessary waiting times. Using this motivation we introduce a multi-layer pheromone network structure which takes the time window characteristics of the problem into consideration. First, the scheduling horizon is divided into  $Z$  time intervals  $T_1, T_2, \dots, T_Z$  and the pheromone network is constructed accordingly. In other words, a third dimension is added to the pheromone information. Now, the implicit heuristic information that depends on the service time windows of the customers is included in the pheromone trail between customer  $i$  and  $j$  in the “time interval  $z$ ”.

An illustrative example consisting of 25 customers is given in Figure 1. The horizon is discretized into three time intervals  $(T_z)_{1 \leq z \leq 3}$ , with a different pheromone network (layer) associated with each interval (Figure 1.a). In this example, the length of each time interval is found by dividing the time window length of the depot by the number of intervals; however, the lengths of the time intervals are not necessarily equal in general and can be determined according to the specific environment. When an ant leaves customer  $i$  at time  $t_i \in T_z$  the pheromone information on the corresponding layer  $z$  is utilized to select the next customer it will visit. It can be observed that on the first layer at the bottom the arcs directed from the depot to the customers have higher pheromone levels whereas on the third layer at the top the pheromone levels on the arcs directed from the customers to the depot are higher. In the

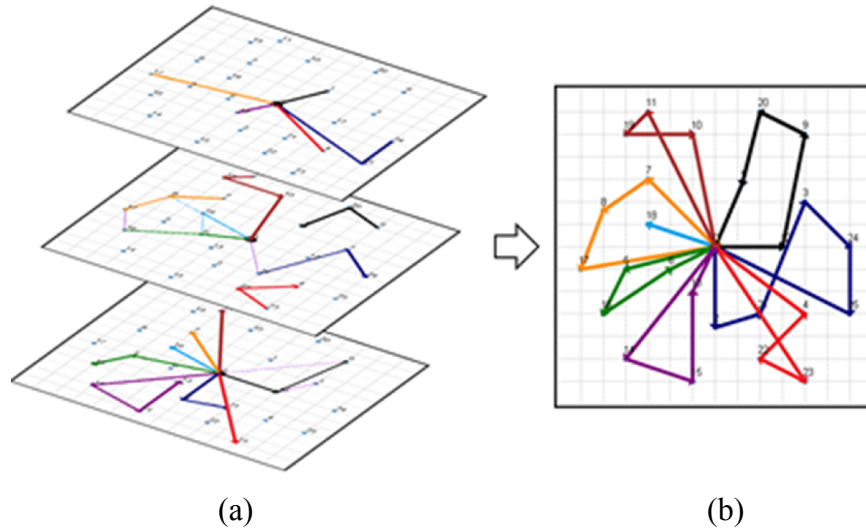


Figure 1. A three-layer pheromone network and the final route assignments

second network in the middle, pheromone is accumulated only on the arcs which are traversed in the second time interval. In Figure 1.b we see that the final routes strictly overlap with the pheromone trails.

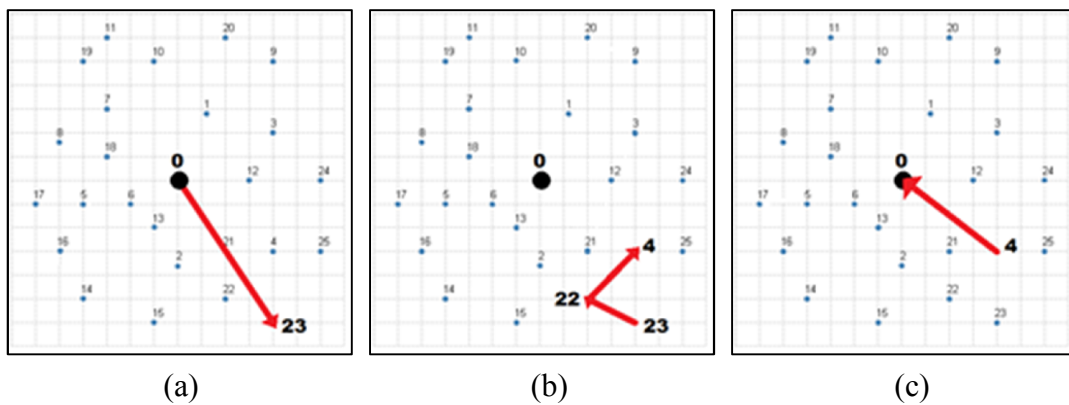


Figure 2. Illustration of the multi-layer pheromone network (a) Layer 1, (b) Layer 2, and (c) Layer 3

The mechanism of the proposed multi-layer pheromone network approach is depicted in Figure 2. Consider the vehicle route 0-23-22-4-0 in Figure 1. 0 indicates the depot and 23, 22, and 4 are the customers that the vehicle services in the given order where their time-windows are [10:00, 16:00], [12:00, 18:00], and [13:00, 16:00], respectively. Let the three time intervals be  $T_1 = [9:00, 12:00)$ ,  $T_2 = [12:00, 15:00)$ , and  $T_3 = [15:00, 18:00)$  and assume a 30-minute service time for all customers. To select the

first customer to visit, the ant uses the pheromone information on layer 1 since it starts its tour from the depot at 9:00. Suppose customer 23 is selected and arc (0, 23) is traversed in 2 hours 40 minutes. Then, the arrival time to customer 23 is 11:40 and the departure time is 12:10. Next, suppose the ant selects customer 22 followed by customer 4 using the pheromone trails on layer 2 and traverses arc (23, 22) during 12:10-13:20 and arc (22, 4) during 13:50-14:50. Finally, after servicing customer 4 the ant returns to the depot by traversing arc (4, 0) from 15:20 to 17:30 and the route terminates.

The above described multi-layer network structure makes use of the implicit information on the most suitable time interval for the travel between each customer pair. To better analyze the gain obtained by this new structure, we do not utilize any explicit heuristic information. Thus, the visibility parameter ( $\beta$ ) is set to 0 and without loss of generality the pheromone parameter is set to 1. The random selection rule (1) is then applied as follows:

$$p_{ijz}^k = \frac{\tau_{ijz}}{\sum_{l \in N_i^k} \tau_{ilz}}, \quad \text{if } j \in N_i^k \quad (4)$$

where  $\tau_{ijz}$  denotes the pheromone trail on layer (time interval)  $z$ . When  $Z = 1$  the algorithm reduces to AS.

Similar to most AS approaches TbAS has three main phases: route construction, local search, and pheromone update. The route construction is performed using equation (4). To put a limit on the exploration and to speed up the algorithm, we use a candidate list which consists of the nearest  $CL$  (candidate list size) feasible neighbours of a customer. Neighbouring customers that satisfy the following conditions are included in the candidate list: (i) the vehicle departing from customer  $i$  arrives at the candidate customer before its latest possible arrival time (also referred to as due date); (ii) the remaining capacity of the vehicle can accommodate the demand of the candidate customer; and (iii) after visiting the candidate customer the vehicle can return to the depot before the depot's due date. If the candidate list is empty, the vehicle completes its tour and returns to the depot. A new vehicle starts its tour at time 0 from the feasible customer that has the largest pheromone value from the depot.

After a solution is obtained we utilize "1-exchange" and "1-move" local search procedures to further reduce the total distance travelled. 1-exchange procedure exchanges two customers in a single route (intra-route) or between routes (inter-route)

whereas the 1-*move* procedure attempts to improve the solution by removing a customer and inserting it between two other customers, intra-route or inter-route.

The pheromone update is performed in a similar way as in Bullnheimer et al. (1999b); however, its implementation is slightly different due to the multi-layer nature of the pheromone network: first we evaporate the pheromone trails on all arcs on all networks; then we allow the ant to deposit pheromone between a customer pair only in the time interval when it has passed through. The procedure is as follows:

$$\tau_{ijz} \leftarrow (1 - \rho)\tau_{ijz} + \sum_{k=1}^{w-1} (w - k)\Delta\tau_{ijz}^k + w\Delta\tau_{ijz}^{bs}, \quad \forall(i, j) \quad (5)$$

where  $\Delta\tau_{ijz}^k$  is the amount of pheromone deposited on arc  $(i, j)$  on layer  $z$  by ant  $k$ ,  $w$  is the number of elite (best-performing) ants that are allowed to deposit pheromone, and superscript *bs* represents the best-so-far ant. The amount of the pheromone deposited is computed in the same way as in (3):

$$\Delta\tau_{ijz}^k = \begin{cases} \frac{1}{L^k}, & \text{if ant } k \text{ uses arc } (i, j) \text{ in time interval } z \text{ on its tour} \\ 0, & \text{otherwise} \end{cases} \quad (6)$$

where  $L^k$  ( $L^{bs}$ ) is the length of tour constructed by ant  $k$  (best-so-far ant *bs*). Note that the best-so-far ant is allowed to deposit pheromone after  $P$  iterations (referred to as preliminary iterations) to avoid a quick stagnation.

#### 4. Computational study

The computational study aims at investigating the role of the multi-pheromone network structure and the effect of the parameters on the solution quality as well as on the computational effort. As the main contribution of TbAS comes from the multi-layer pheromone network, the main parameter to be determined is the number of layers. The number of ants whose solutions are further improved through LS (referred to as *#LS\_ants*) is also a key parameter. So, we focus on the sensitivity of the solution quality to the value of these two parameters.

We use the well-known Solomon instances (Solomon, 1987) using real numbers. These instances comprise three main problem sets where the customers are clustered (C), randomly distributed (R), and both clustered and randomly distributed (RC) over a 100x100 grid. Each set has also two subsets, type 1 and type 2, which differ not only by the length of the time windows but also by the vehicle capacity. There are a total of 56

instances grouped in 6 problem classes (namely C1, C2, R1, R2, RC1, and RC2), which all involve  $n = 100$  customers.

Table 1. Values used for parameter tuning

Parameter	Value
Evaporation ( $\rho$ )	0.05, 0.10, <b>0.15</b>
Number of elite ants ( $w$ )	6, <b>12</b> , 18
Candidate list size ( $CL$ )	<b>25</b> , 50, 100
Number of iterations	<b>100</b> , 200
Preliminary iterations ( $P$ )	<b>25</b> , 50, 75

For parameter tuning, we performed a series of preliminary tests using the values given in Table 1 and determined the values shown in bold as the best-fitting values. The lengths of the time intervals are equal and are determined as the ratio of the length of the depot’s time-window to the number of layers. The algorithm is coded in C# programming language and executed on a Pentium 2.33 GHz processor. To observe the robustness of the algorithm, we performed 30 runs for each instance.

#### 4.1. Analysis of the number of layers

In this experiment we investigate the sensitivity of the solution quality and the computational effort to changing number of layers. To make a better analysis, we consider both small and large  $\#LS\_ants$  values in the LS phase, i.e.  $n/20$  (5) and  $n$  (100) ants, respectively. Table 2 reports the average of the best distances for each class implementing 1 to 5 layers. The layer 1 case corresponds to the standard single-layer pheromone network. “TD” is the total distance, “NV” is the number of vehicles, “CT” is the computation time, and “Avg” refers to the average. The values in bold show the best layer option for each problem class and  $\#LS\_ants$  configuration.

We observe that on the average multi-layer approach provides better distances than the traditional single-layer approach, for both 5 and 100  $\#LS\_ants$  cases. Among the multiple layers, the three-layer pheromone network outperforms the others. If we examine the individual classes, we see that the three-layer network is the best performer except RC classes, for which it lags by a very small margin. Overall, we observe that the multi-layer approach is capable of finding better routes compared to the single-layer approach.

To better analyze each class, we take the average of the distances for 5 and 100  $\#LS\_ants$  cases and report the normalized values in Figure 3. The normalization is

Table 2. Average of the best total distance values, number of vehicles, and the average computation times (in minutes)

		Number of layers									
		#LS ants = 5					#LS ants = 100				
		1	2	3	4	5	1	2	3	4	5
C1	TD	<b>828.38</b>	<b>828.38</b>	<b>828.38</b>	<b>828.38</b>	<b>828.38</b>	<b>828.38</b>	<b>828.38</b>	<b>828.38</b>	<b>828.38</b>	<b>828.38</b>
	NV	10.00	10.00	10.00	10.00	10.00	10.00	10.00	10.00	10.00	
	CT	0.33	0.32	0.34	0.34	0.33	5.15	5.23	5.52	5.48	5.19
C2	TD	589.93	589.93	590.27	<b>589.86</b>	590.27	<b>589.86</b>	<b>589.86</b>	<b>589.86</b>	<b>589.86</b>	<b>589.86</b>
	NV	3.00	3.00	3.00	3.00	3.00	3.00	3.00	3.00	3.00	
	CT	0.60	0.59	0.61	0.62	0.60	12.54	13.08	13.80	13.71	13.03
R1	TD	1191.93	1189.58	<b>1189.29</b>	1190.04	1190.43	1183.55	1182.94	<b>1181.88</b>	1184.42	1183.33
	NV	13.58	13.50	13.50	13.50	13.42	13.42	13.33	13.33	13.33	13.33
	CT	0.37	0.36	0.37	0.38	0.37	5.18	5.12	5.18	5.17	4.90
R2	TD	911.56	907.47	899.65	<b>899.20</b>	902.15	903.40	898.66	<b>892.37</b>	893.78	894.98
	NV	5.73	5.36	5.36	5.64	5.36	5.82	5.73	5.36	5.45	5.45
	CT	1.62	1.62	1.61	1.63	1.61	30.68	31.06	31.53	31.24	30.24
RC1	TD	1369.34	1368.10	<b>1367.06</b>	1367.73	1371.06	1349.04	<b>1347.85</b>	1349.61	1351.47	1349.22
	NV	13.38	13.38	13.25	13.25	13.38	13.00	13.00	13.13	13.25	13.13
	CT	0.42	0.40	0.42	0.43	0.41	4.79	4.78	4.83	4.86	4.71
RC2	TD	1044.62	1033.08	<b>1032.66</b>	1036.16	1035.01	1025.25	<b>1017.19</b>	1018.15	1017.82	1021.62
	NV	6.25	6.38	6.00	6.13	5.88	6.63	6.13	5.88	6.25	6.25
	CT	1.20	1.20	1.21	1.21	1.19	22.15	22.69	22.68	22.72	21.75
Avg	TD	996.73	993.60	<b>991.84</b>	992.45	993.48	987.65	985.27	<b>984.20</b>	985.24	985.46
	NV	8.88	8.80	8.73	8.80	8.71	8.86	8.75	8.66	8.75	8.73
	CT	0.77	0.76	0.77	0.78	0.76	13.60	13.83	14.09	14.02	13.47

performed by dividing each class average with respect to the number of networks by the minimum distance. These results show that the multi-layer approach outperforms the single-layer approach in all the classes but C1 where all the settings perform the same. In addition, we see that when the scheduling horizon is longer and the time-windows are larger (i.e. type 2 problems), our multi-layer approach performs better as a result of the usage of the implicit time information, particularly in R2 and RC2 classes.

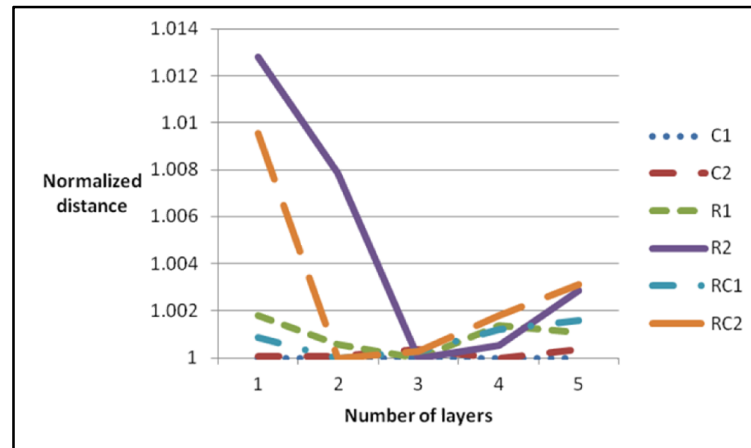


Figure 3. Normalized total distances

The only difference between the time complexity of the traditional AS approach and the proposed approach lies in  $Z$ . As  $Z$  is not expected to be large the additional



computation time that the pheromone update procedure requires is marginal. The results in Table 2 show that the multi-layer approach does not require any additional computational effort as the average run times are similar for all problem classes. On the other hand, we notice that the increase in  $\#LS\_ants$  has a significant effect on the computation time regardless of the number of layers: The average computational time using 100 ants is nearly 20 times the time needed in 5 ants. However, the average improvement in the total distance using 100 vs. 5  $\#LS\_ants$  is only 1.53%. We will further investigate the impact of  $\#LS\_ants$  on both the solution quality and the computational effort in the next section.

#### 4.2. Analysis of the number of ants used in the local search

$\#LS\_ants$  can be static or dynamic throughout the solution process. Furthermore, the ants can be selected randomly or based on their performance. In this experimental study we investigate the performance of TbAS for  $\#LS\_ants$  values of  $n/20$ ,  $n/10$ ,  $n/5$ ,  $n/2$ , and  $n$  (i.e. 5, 10, 20, 50, and 100 ants, respectively). The number of layers is fixed to three in accordance with the previous experiment. The results are reported in Table 3.

Table 3. Comparison of the number of the ants by distance

		$\#LS\_ants$				
		5	10	20	50	100
C1	TD	830.40	828.88	828.57	828.41	828.38
	NV	10.00	10.00	10.00	10.00	10.00
	CT	0.36	0.56	0.99	2.39	5.11
C2	TD	593.21	590.99	590.50	590.34	590.24
	NV	3.03	3.01	3.00	3.00	3.00
	CT	0.62	1.17	2.30	5.99	13.06
R1	TD	1205.30	1198.49	1194.00	1190.88	1189.47
	NV	13.80	13.80	13.79	13.78	13.74
	CT	0.34	0.54	0.95	2.29	4.89
R2	TD	928.81	922.10	916.76	913.52	910.89
	NV	5.72	5.78	5.78	5.84	5.84
	CT	1.46	2.73	5.38	14.02	30.56
RC1	TD	1394.41	1387.02	1379.56	1372.49	1368.45
	NV	13.68	13.65	13.61	13.55	13.54
	CT	0.33	0.51	0.91	2.19	4.68
RC2	TD	1067.02	1057.42	1048.97	1043.36	1039.72
	NV	6.33	6.52	6.50	6.55	6.56
	CT	1.05	1.96	3.86	10.06	21.93
Average	TD	1010.56	1004.79	1000.39	997.22	995.29
	NV	8.98	9.01	9.00	9.01	9.00
	CT	0.70	1.26	2.43	6.23	13.54

We observe that using more ants in the LS phase results in better exploration of the solution space and yields better solutions. The average distance in each class tends to decrease with the increasing value of  $\#LS\_ants$ . On the other hand, the improvement on the overall average distance is only 0.19% for 50 ants compared to 100 ants. While

increasing  $\#LS\_ants$  enhances the solution quality, the trade-off is the significant increase in the computational effort. Using the best 5 ants in LS as compared to all 100 ants deteriorates the average solution quality by 1.53% whereas the computational time reduces by 95%. Decreasing  $\#LS\_ants$  from 100 to even 50 leads to a 54% reduction in the average computational time at the expense of only 0.19% decline in the average total distance.

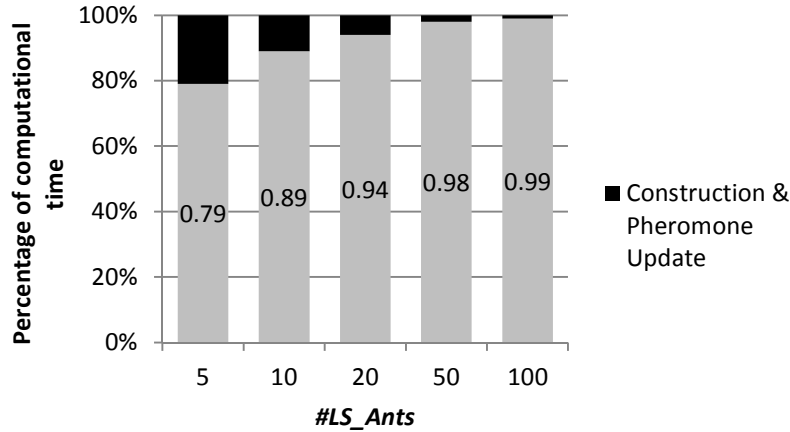


Figure 4. Percentages of computation times spent for different phases of TbAS

Although different numbers of pheromone networks do not affect the computation times, the above observation reveals that increasing the number of ants utilized in LS leads to substantially higher computational effort. So, in Figure 4 we compare the time spent to the main AS mechanisms, namely route construction and pheromone update phases, to that of the LS phase. We observe that when 5 ants are utilized LS takes 79% of the total computation time whereas it takes 99% of the time when 100 ants are utilized. We note that in the former case, the route construction and pheromone update procedures consume 20.6% and 0.4% of the total time, respectively. While LS consumes a significant amount of time, nevertheless, ACO is more efficient equipped with a LS mechanism, in parallel with the observation in the literature. From a practical point of view,  $\#LS\_ants$  can be rationally determined to obtain fairly good solutions in reasonable time.

#### 4.3. Analysis of the best and first improvement approaches in local search

State-of-the-art ACO algorithms are combined with LS for an enhanced performance (Floudas and Pardalos, 2009). The probability that a solution constructed by an ant will

be improved by an adequate LS is quite high as the neighbourhood structures of ACO and LS are different. Besides, the probabilistic, adaptive solution generation process of ACO provides proper initial solutions for LS which alone suffers from finding these solutions (Dorigo and Stützle, 2004). The proposed TbAS is not exceptional and similar to most best performing ACO algorithms it benefits from the LS to be able to achieve high quality solutions.

In the experiments so far, we implemented the best improvement LS approach where all possible neighbours are investigated and the one that leads to the largest improvement is performed. In this experiment, we also test the performance of TbAS using the first improvement (greedy) approach, i.e. as soon as an improving neighbour is found it is realized as the next solution. We noted earlier that the three-layer network approach performed best with respect to the average of best solutions. However, we observed that the average results obtained by the four-layer network approach are also competitive. Hence, in this experiment we compare both three- and four-layer settings implementing best and first improvement local search approaches. The average results are reported in Table 4. In this table, the rows “Best” (“Avg”) refers to the set average of the best distances (of the average distances) and “CV” denotes the coefficient of variation (the ratio of the standard deviation to the mean). “TbAS(3)/(4)” and “Best/First” columns indicate the configuration as described above.

Although these results do not show any superiority of one number of layers to the other, we observe that four-layer network structure tends to provide better solutions in type 2 problems whereas three-layer network structure is slightly better in type 1 problems, particularly when best neighbour approach is used in the LS procedure. This may be due to the fact that more number of layers provides better time-wise discriminated pheromone trails for type 2 problems with longer scheduling horizon and larger time-windows. In terms of the LS technique, the greedy approach performs slightly better than the best improvement approach. Although this may seem non-intuitive, note that LS is performed after the routes are constructed with the ant system and the post-optimization with the greedy approach may better explore the solution space. Besides, the difference is marginal. Finally, the small CV values show the robustness of the algorithm.

Table 4. Average distances, number of vehicles and the coefficient of variation for best and first improvement LS approaches

Problem Set	TbAS(3)					TbAS(4)					
	Best		First			Best		First			
	TD	NV	TD	NV	CV	TD	NV	TD	NV	CV	
C1	Best	828.38	10.00	828.38	10.00	828.38	10.00	828.38	10.00	828.38	10.00
	Avg	828.38	10.00	828.41	10.00	828.39	10.00	828.45	10.00	828.45	10.00
	CV	0.0000	0.0000	0.0001	0.0000	0.0001	0.0000	0.0002	0.0000	0.0002	0.0000
C2	Best	589.86	3.00	589.86	3.00	589.86	3.00	589.86	3.00	589.86	3.00
	Avg	590.24	3.00	590.16	3.00	590.61	3.00	590.28	3.00	590.28	3.00
	CV	0.0009	0.0000	0.0004	0.0000	0.0023	0.0075	0.0021	0.0075	0.0021	0.0075
R1	Best	1181.88	13.33	1180.91	13.25	1184.42	13.33	1183.26	13.25	1183.26	13.25
	Avg	1189.47	13.74	1189.59	13.65	1190.05	13.77	1189.46	13.63	1189.46	13.63
	CV	0.0028	0.0160	0.0033	0.0205	0.0023	0.0153	0.0031	0.0212	0.0031	0.0212
R2	Best	892.37	5.36	893.32	5.18	893.78	5.45	890.45	5.36	890.45	5.36
	Avg	910.89	5.84	906.80	5.58	911.21	5.82	905.24	5.55	905.24	5.55
	CV	0.0102	0.0949	0.0076	0.0909	0.0103	0.1012	0.0079	0.0958	0.0079	0.0958
RC1	Best	1349.61	13.13	1344.65	13.00	1351.47	13.25	1343.96	13.00	1343.96	13.00
	Avg	1368.45	13.54	1362.63	13.35	1366.08	13.48	1360.99	13.33	1360.99	13.33
	CV	0.0074	0.0249	0.0069	0.0256	0.0070	0.0233	0.0068	0.0181	0.0068	0.0181
RC2	Best	1018.15	5.88	1018.81	6.13	1017.82	6.25	1022.73	6.38	1022.73	6.38
	Avg	1039.72	6.56	1036.37	6.29	1038.01	6.50	1036.44	6.28	1036.44	6.28
	CV	0.0106	0.0911	0.0081	0.0689	0.0110	0.0821	0.0074	0.0776	0.0074	0.0776
Overall	Best	984.20	8.66	983.56	8.63	985.24	8.75	983.96	8.70	983.96	8.70
	Avg	995.29	9.00	993.20	9.49	994.95	8.98	992.66	8.85	992.66	8.85
	CV	0.0053	0.0378	0.0044	0.0343	0.0055	0.0382	0.0046	0.0367	0.0046	0.0367

Table 5. Comparison with the best-known distances from the literature for type 1 problems

Inst	Best-known		TbAS(3)					TbAS(4)				
	TD	NV	Best		First			Best		First		
	TD	NV	TD	NV	TD	NV	CV	TD	NV	TD	NV	CV
C101	828.94 [RT]	10	<b>828.94</b>	10	<b>828.94</b>	10		<b>828.94</b>	10	<b>828.94</b>	10	
C102	828.94 [RT]	10	<b>828.94</b>	10	<b>828.94</b>	10		<b>828.94</b>	10	<b>828.94</b>	10	
C103	828.06 [RT]	10	<b>828.06</b>	10	<b>828.06</b>	10		<b>828.06</b>	10	<b>828.06</b>	10	
C104	824.78 [RT]	10	<b>824.78</b>	10	<b>824.78</b>	10		<b>824.78</b>	10	<b>824.78</b>	10	
C105	828.94 [RT]	10	<b>828.94</b>	10	<b>828.94</b>	10		<b>828.94</b>	10	<b>828.94</b>	10	
C106	828.94 [RT]	10	<b>828.94</b>	10	<b>828.94</b>	10		<b>828.94</b>	10	<b>828.94</b>	10	
C107	828.94 [RT]	10	<b>828.94</b>	10	<b>828.94</b>	10		<b>828.94</b>	10	<b>828.94</b>	10	
C108	828.94 [RT]	10	<b>828.94</b>	10	<b>828.94</b>	10		<b>828.94</b>	10	<b>828.94</b>	10	
C109	828.94 [RT]	10	<b>828.94</b>	10	<b>828.94</b>	10		<b>828.94</b>	10	<b>828.94</b>	10	
R101	1642.87 [AMT]	20	1642.88	20	1642.88	20		1642.88	20	1642.88	20	
R102	1472.62 [AMT]	18	1472.81	18	1472.81	18		1472.82	18	1472.81	18	
R103	1213.62 [JM]	14	<b>1213.62</b>	14	<b>1213.62</b>	14		1218.61	14	<b>1213.62</b>	14	
R104	976.61 [JM]	11	977.55	11	984.34	11		995.17	11	991.09	11	
R105	1360.78 [JM]	15	1365.85	15	<b>1360.78</b>	15		1368.91	16	1363.74	15	
R106	1240.41 [GB]	13	1240.55	13	1241.35	13		1241.55	13	<b>1240.26</b>	13	
R107	1073.34 [JM]	11	1085.56	12	<b>1073.01</b>	11		1080.43	11	1074.32	11	
R108	947.55 [JM]	10	<b>946.42</b>	10	<b>944.44</b>	10		949.38	10	<b>946.42</b>	10	
R109	1151.84 [JM]	13	<b>1151.84</b>	13	<b>1151.84</b>	13		<b>1151.84</b>	13	<b>1151.84</b>	13	
R110	1072.41 [JM]	12	1072.42	12	<b>1072.41</b>	12		1072.42	12	1082.22	12	
R111	1053.50 [JM]	12	<b>1053.50</b>	12	1053.80	12		<b>1053.50</b>	12	<b>1053.50</b>	12	
R112	953.63 [RT]	10	959.58	10	959.58	10		965.49	10	966.39	10	
RC101	1623.58 [RT]	15	1657.91	17	1642.48	16		1644.78	17	1638.00	16	
RC102	1461.23 [JM]	14	1477.87	14	1464.35	14		1477.20	14	1461.44	14	
RC103	1261.67 [S]	11	1276.05	12	1277.08	12		1262.68	11	1277.08	12	
RC104	1135.48 [C]	10	1148.83	10	1141.66	10		1147.34	10	1143.56	10	
RC105	1518.58 [JM]	16	<b>1518.58</b>	16	<b>1518.58</b>	16		<b>1518.58</b>	16	<b>1518.58</b>	16	
RC106	1377.35 [AMT]	13	<b>1377.35</b>	13	<b>1376.99</b>	13		1393.96	14	1381.58	13	
RC107	1212.83 [JM]	12	1212.95	12	<b>1212.83</b>	12		1238.69	13	1213.89	12	
RC108	1117.53 [JM]	11	1127.38	11	1123.26	11		1128.57	11	<b>1117.53</b>	11	

AMT: (Alvarenga et al., 2007), C: (Cordeau et al., 2001), GB: (Garcia-Najera and Bullinaria, 2011), JM: (Jung and Moon, 2002), RT: (Rochat and Taillard, 1995), S: (Shaw, 1998),

#### 4.4. Comparison with the best-known distances

Table 5 and Table 6 compare the best solutions found by TbAS to the best distances published in the literature for type 1 and type 2 problems, respectively. We use the same configurations as in Section 5.3. For all clustered problems (C1 and C2 problem sets) TbAS finds the best-known distances using any of the layer/LS configurations. For the other problems, we observe that TbAS is able to find relatively good solutions. The average gap between the best-known distances and our best distances is only 0.06% for type 1 problems and 0.75% for type 2 problems. In 4 instances the best-known distances are improved: R106, R107, R108, and RC106. The solutions of these problems are provided in the Appendix. The numbers in bold show the distances that are better than or same as the best-known distances. The new best distances are italicized. The convergence graphs of these instances are depicted in Figure 5.

Table 6. Comparison with the best-known distances from the literature for type 2 problems

Inst	Best-known		TbAS(3)				TbAS(4)			
	TD	NV	Best TD	NV	First TD	NV	Best TD	NV	First TD	NV
C201	591.56 [RT]	3	<b>591.56</b>	3	<b>591.56</b>	3	<b>591.56</b>	3	<b>591.56</b>	3
C202	591.56 [RT]	3	<b>591.56</b>	3	<b>591.56</b>	3	<b>591.56</b>	3	<b>591.56</b>	3
C203	591.17 [RT]	3	<b>591.17</b>	3	<b>591.17</b>	3	<b>591.17</b>	3	<b>591.17</b>	3
C204	590.60 [RT]	3	<b>590.60</b>	3	<b>590.60</b>	3	<b>590.60</b>	3	<b>590.60</b>	3
C205	588.88 [RT]	3	<b>588.88</b>	3	<b>588.88</b>	3	<b>588.88</b>	3	<b>588.88</b>	3
C206	588.49 [RT]	3	<b>588.49</b>	3	<b>588.49</b>	3	<b>588.49</b>	3	<b>588.49</b>	3
C207	588.29 [RT]	3	<b>588.29</b>	3	<b>588.29</b>	3	<b>588.29</b>	3	<b>588.29</b>	3
C208	588.32 [RT]	3	<b>588.32</b>	3	<b>588.32</b>	3	<b>588.32</b>	3	<b>588.32</b>	3
R201	1147.80 [BV]	8	1162.59	8	1157.65	9	1157.86	9	1155.80	8
R202	1034.35 [JM]	8	1036.60	7	1037.08	7	1042.05	7	1038.41	8
R203	874.87 [JM]	6	880.61	6	877.48	6	882.15	6	875.62	6
R204	735.80 [BV]	3	748.52	4	746.98	5	749.05	4	750.50	5
R205	954.16 [ORH]	5	972.87	5	972.55	5	964.64	5	973.81	5
R206	879.89 [JM]	5	899.76	6	900.76	4	902.82	6	892.95	5
R207	797.99 [BV]	4	823.50	5	820.49	4	829.55	4	805.70	4
R208	705.45 [JM]	4	717.78	4	719.42	3	722.22	3	711.37	3
R209	859.39 [JM]	5	876.33	5	892.31	5	879.14	6	888.29	5
R210	910.70 [JM]	5	923.50	5	915.49	5	928.59	6	918.79	6
R211	755.82 [BV]	4	774.06	4	786.31	4	773.51	4	783.71	4
RC201	1265.56 [JM]	9	1275.66	9	1278.14	8	1272.63	9	1278.09	8
RC202	1095.64 [JM]	8	1104.92	7	1104.92	7	1112.85	8	1111.16	8
RC203	926.89 [BV]	5	944.54	5	938.19	6	942.09	5	938.19	6
RC204	786.38 [JM]	4	806.85	4	802.10	4	800.48	4	802.95	4
RC205	1157.55 [JM]	7	<b>1157.55</b>	7	1159.06	7	<b>1157.55</b>	7	<b>1157.55</b>	7
RC206	1054.61 [JM]	7	1079.12	6	1072.78	6	1072.08	6	1084.71	7
RC207	966.08 [JM]	6	972.74	5	980.30	6	992.21	6	990.89	6
RC208	779.31 [JM]	4	803.83	4	815.00	5	792.65	5	818.28	5

BV: (Brandão de Oliveira and Vasconcelos, 2010), JM: (Jung and Moon, 2002), ORH: (Ombuki et al., 2006), RT: (Rochat and Taillard, 1995)

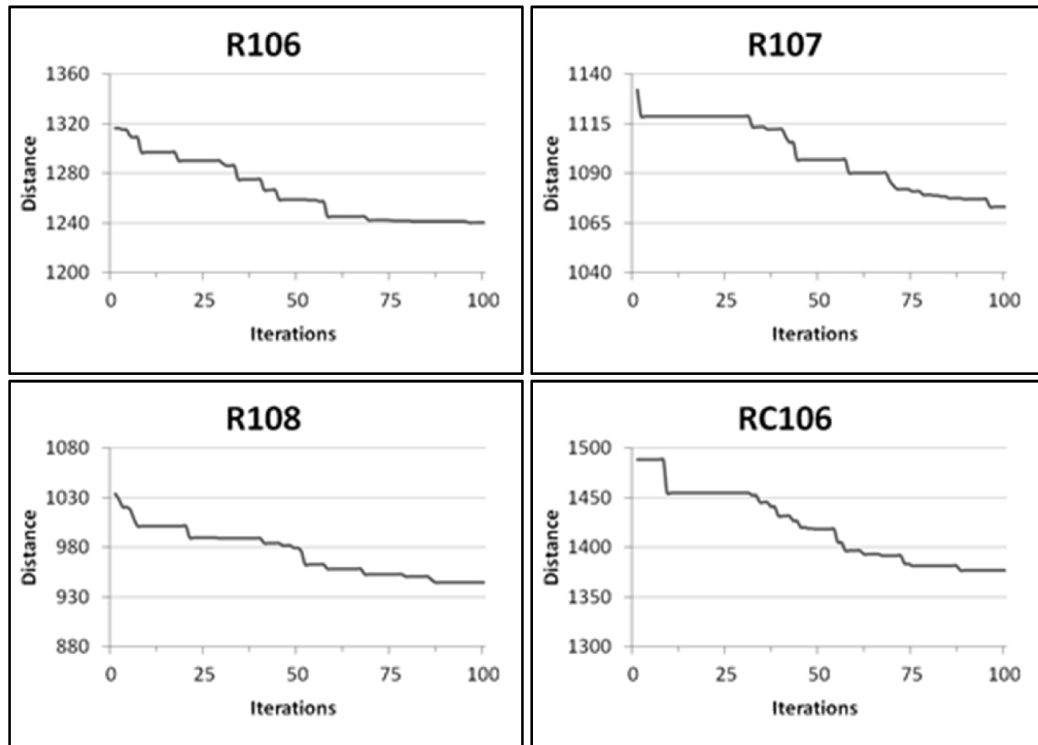


Figure 5. Convergence graphs for the instances with an improved best-known distance

## 5. Conclusion

In this study, we presented TbAS, a new AS algorithm that utilizes a multi-layer pheromone network approach for solving VRPs involving time-windows and conducted an extensive computational study to test its performance. Our tests on VRPTW instances showed that the multi-layer pheromone network approach outperformed the classical single-layer counterpart. We observed that equipped with LS, TbAS is capable of obtaining good solutions by implicitly using the service time information, especially in problems with longer scheduling horizon and wider time windows. We also compared the distances obtained by TbAS to those published in the literature. The results reveal that TbAS is effective in finding short distances.

Further research on this topic may focus on developing an efficient visibility function and/or investigating other neighbourhood structures in the local search phase to further enhance the solution quality. In addition, TbAS may be easily adapted to other VRPTW variants. Summarizing, TbAS may be promising approach for solving hard combinatorial optimization problems which involve time information.

## A. Appendix

We report here the new best solutions we obtained for problems R106, R107, R108, and RC106. The values in parentheses show the total distance achieved.

Route	Distance
R106 (1240.257)	
1	0-92-37-14-44-38-86-43-100-98-93-0
2	0-50-33-65-71-66-20-32-70-1-0
3	0-48-47-36-19-49-46-82-7-52-0
4	0-94-59-42-15-57-87-97-95-13-0
5	0-69-30-51-81-9-35-34-3-77-0
6	0-83-45-8-84-17-5-60-0
7	0-27-62-88-18-89-0
8	0-73-41-22-75-56-74-2-58-0
9	0-21-72-39-23-67-55-4-25-26-0
10	0-12-29-78-79-68-54-24-80-0
11	0-96-85-91-16-61-99-6-0
12	0-28-76-40-53-0
13	0-63-64-11-90-10-31-0
R107 (1073.009)	
1	0-28-76-79-78-29-24-68-80-12-0
2	0-2-57-15-41-22-75-56-74-4-21-58-0
3	0-33-81-65-71-9-35-34-3-77-0
4	0-42-43-14-44-38-86-16-91-100-37-98-0
5	0-60-83-45-46-8-84-5-17-61-85-93-0
6	0-48-47-36-64-49-19-82-18-89-0
7	0-53-40-0
8	0-52-7-62-11-63-90-32-66-20-51-50-0
9	0-27-69-30-88-31-10-70-1-0
10	0-73-72-39-23-67-55-25-54-26-0
11	0-94-96-92-59-99-6-87-97-95-13-0
R108 (944.441)	
1	0-53-0
2	0-6-96-99-5-84-17-61-85-93-59-94-0
3	0-2-57-15-43-42-87-41-22-74-73-21-40-0
4	0-31-88-62-11-64-49-36-47-19-7-52-0
5	0-27-69-1-51-9-35-34-78-81-33-50-0
6	0-95-92-98-44-14-38-86-16-91-100-37-97-0
7	0-89-18-82-48-46-8-45-83-60-13-58-0
8	0-10-63-90-32-66-65-71-20-30-70-0
9	0-72-75-56-23-67-39-55-4-25-54-0
10	0-26-12-76-3-79-29-24-80-68-77-28-0

Route	Distance
RC106 (1376.993)	
1 0-83-64-19-23-21-18-48-25-77-0	131.07
2 0-65-52-87-59-75-97-58-74-0	133.8
3 0-92-67-31-29-30-32-89-0	143.69
4 0-72-38-39-40-36-35-37-54-0	102.292
5 0-61-42-44-43-41-70-68-0	93.345
6 0-82-99-86-57-22-49-20-24-0	103.77
7 0-2-45-5-8-7-6-46-4-3-1-100-0	109.751
8 0-62-33-28-26-27-34-50-91-80-0	128.16
9 0-95-63-85-76-51-84-56-66-0	104.989
10 0-81-71-94-93-96-0	66.456
11 0-11-12-14-47-15-16-9-10-13-17-0	126.269
12 0-88-78-73-79-60-55-0	91.352
13 0-69-98-53-90-0	42.049

## References

- Alvarenga, G. B., G. R. Mateus and G. de Tomi (2007). "A genetic and set partitioning two-phase approach for the vehicle routing problem with time windows." Computers & Operations Research **34**(6): 1561-1584.
- Badeau, P., F. Guertin, M. Gendreau, J.-Y. Potvin and E. Taillard (1997). "A parallel tabu search heuristic for the vehicle routing problem with time windows." Transportation Research Part C: Emerging Technologies **5**(2): 109-122.
- Brandão de Oliveira, H. and G. Vasconcelos (2010). "A hybrid search method for the vehicle routing problem with time windows." Annals of Operations Research **180**(1): 125-144.
- Bräysy, O. and M. Gendreau (2005a). "Vehicle Routing Problem with Time Windows, Part I: Route Construction and Local Search Algorithms." Transportation Science **39**(1): 104-118.
- Bräysy, O. and M. Gendreau (2005b). "Vehicle Routing Problem with Time Windows, Part II: Metaheuristics." Transportation Science **39**(1): 119-139.
- Bullnheimer, B., R. F. Hartl and C. Strauss (1999a). "An improved Ant System algorithm for the Vehicle Routing Problem." Annals of Operations Research **89**: 319-328.
- Bullnheimer, B., R. F. Hartl and C. Strauss (1999b). "A new rank-based version of the ant system: A computational study." Central European Journal of Operations Research **7**(1): 25-38.
- Cordeau, J. F., G. Laporte and A. Mercier (2001). "A unified tabu search heuristic for vehicle routing problems with time windows." Journal of the Operational Research Society **52**(8): 928-936.
- Dorigo, M. (2010). "Ant colony optimization." Scholarpedia **2**(3): 1461-1461.
- Dorigo, M. and L. M. Gambardella (1997). "Ant colony system: a cooperative learning approach to the traveling salesman problem." IEEE Transactions on Evolutionary Computation **1**(1): 53-66.
- Dorigo, M., V. Maniezzo and A. Colomi (1996). "Ant system: optimization by a colony of cooperating agents." IEEE transactions on systems, man, and cybernetics. Part B, Cybernetics : a publication of the IEEE Systems, Man, and Cybernetics Society **26**(1): 29-41.
- Dorigo, M. and T. Stützle (2004). Ant Colony Optimization. London, The MIT Press.



- Eglese, R., W. Maden and A. Slater (2006). "A Road Timetable to aid vehicle routing and scheduling." Computers & Operations Research **33**(12): 3508-3519.
- Ellabib, I., O. A. Basir and P. Calamai (2002). "An Experimental Study of a Simple Ant Colony System for the Vehicle Routing Problem with Time Windows." Ant Algorithms Lecture Notes in Computer Science **2463**: 53-64.
- Fleischmann, B., M. Gietz and S. Gnutzmann (2004). "Time-Varying Travel Times in Vehicle Routing." Transportation Science **38**(2): 160-173.
- Floudas, C. A. and P. M. Pardalos (2009). Encyclopedia of Optimization. New York, Springer.
- Gambardella, L. M., É. Taillard and G. Agazzi (1999). MACS-VRPTW: a multiple ant colony system for vehicle routing problems with time windows. D. Corne, M. Dorigo and F. Glover. London, McGraw-Hill: 63-76.
- Garcia-Najera, A. and J. A. Bullinaria (2011). "An improved multi-objective evolutionary algorithm for the vehicle routing problem with time windows." Computers & Operations Research **38**(1): 287-300.
- Jung, S. and B. R. Moon (2002). "A Hybrid Genetic Algorithm For The Vehicle Routing Problem With Time Windows." 1309-1316.
- Koskosidis, Y. A., W. B. Powell and M. M. Solomon (1992). "An Optimization-Based Heuristic for Vehicle Routing and Scheduling with Soft Time Window Constraints." Muter, İ., Ş. İ. Birbil and G. Şahin (2010). "Combination of Metaheuristic and Exact Algorithms for Solving Set Covering-Type Optimization Problems." INFORMS Journal on Computing **22**(4): 603-619.
- Ombuki, B., B. J. Ross and F. Hanshar (2006). "Multi-Objective Genetic Algorithms for Vehicle Routing Problem with Time Windows." Applied Intelligence **24**(1): 17-30.
- Pardalos, P. M. and M. G. C. Resende (2002). Handbook of Applied Optimization. New York, New York, USA, Oxford University Press.
- Pisinger, D. and S. Ropke (2007). "A general heuristic for vehicle routing problems." Computers & Operations Research **34**(8): 2403-2435.
- Rochat, Y. and É. D. Taillard (1995). "Probabilistic diversification and intensification in local search for vehicle routing." Journal of Heuristics **1**(1): 147-167.
- Rousseau, L.-M., M. Gendreau and G. Pesant (2002). "Using Constraint-Based Operators to Solve the Vehicle Routing Problem with Time Windows." Journal of Heuristics **8**(1): 43-58.
- Savelsbergh, M. W. P. (1992). "The Vehicle Routing Problem with Time Windows: Minimizing Route Duration." ORSA Journal on Computing **4**(2): 146-154.
- Shaw, P. (1998). Using constraint programming and local search methods to solve vehicle routing problems. M. Maher and J. F. Puget. Berlin, Springer: 417-431.
- Solomon, M. M. (1987). "Algorithms for the Vehicle Routing and Scheduling Problems with Time Window Constraints." Operations Research **35**(2): 254-265.
- Stützle, T. and H. Hoos (1997). MAX-MIN Ant System and local search for the traveling salesman problem, IEEE.
- Tan, K. C., L. H. Lee, Q. L. Zhu and K. Ou (2001). "Heuristic methods for vehicle routing problem with time windows." Artificial Intelligence in Engineering **15**(3): 281-295.
- Toth, P. and D. Vigo (2002). The vehicle routing problem. Philadelphia, PA, SIAM Monographs on Discrete Mathematics and Applications.
- Zufferey, N. (2011). "Optimization by ant algorithms: possible roles for an individual ant." Optimization Letters **6**(5): 963-973.

## **6 GREEN VEHICLE ROUTING**

## 6.1 Introduction

Green route planning on a time-dependent network where the travel speeds are subject to fluctuations throughout the day requires the greenest path between every pair of nodes to be determined first. Given these data, a fuel consumption/GHG matrix between the customers and between the depot(s) and the customers can be established for any instance.

The aforementioned facts in Section 2.1 show that bringing up the environmental costs and other issues that are related to the sustainable logistics requires new ways of doing business and planning approaches. Thus, ensuring an effective and efficient road transport will also play a major role in reducing the negative environmental effects of the logistics. For example, if all the vehicles collecting and distributing goods serve with full capacity and their routes are determined by considering the GHG factor, the threat on the environment by these activities will be greatly reduced. Also, better loading of the vehicles, preventing the empty vehicle circulation and reducing the distance traveled, in other words an effective route and schedule planning will constitute a relief on the traffic density and congestion yielding a decrease on the GHG indirectly.

Route planning forms the basis of the distribution and collecting activities for the road transport. The classical vehicle routing problem (VRP) aims to find routes that start and end at a central depot and serve certain pickup and delivery points. The demand at each point is known and each point is visited only once. All the vehicles are identical with a certain capacity and each vehicle serves only on a single route. The main objective is to minimize the total distance traveled (travel time). The VRP and its extensions with different objectives and constraints have been extensively studied in the literature for almost 50 years, especially after 90's. However, the environmental factors are often disregarded.

A detailed network structure is depicted in Figure 6.1. In this example, 0 denotes the depot while the customers are denoted by nodes 3, 5, 12, 16, 20 and 27. The speeds on each arc may differ from each other. So, the greenest path between node pairs will differ by considering different departure times. In Figure 6.1(a), the vehicle that departs at  $t=0$  and heads east (right) will follow the route  $0 \rightarrow 3 \rightarrow 5 \rightarrow 12 \rightarrow 0$  whereas the vehicle that heads west (left) will follow the route  $0 \rightarrow 16 \rightarrow 27 \rightarrow 20 \rightarrow 0$ . In Figure 6.1 (b), the orders of the customer visits are the same with the previous case for both of the vehicles which depart at  $t=1$ . However, the vehicle on the east route heading for the

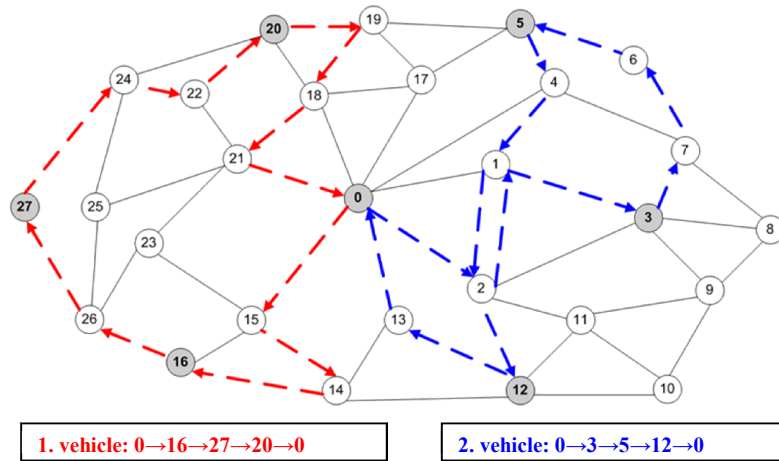
first customer follows the arcs (0,2)-(2,3) instead of the arcs (0,2)-(2,1)-(1,3) and the vehicle on the west route heading for the depot from the last customer follows the arcs (20,19)-(19,17)-(17,0) instead of the arcs (20,19)-(19,18)-(18,21)-(21,0). In Figure 6.1 (c) where the vehicles depart at  $t=2$ , the route of the vehicle on east route is changed to  $0 \rightarrow 5 \rightarrow 3 \rightarrow 12 \rightarrow 0$  which also changes the arcs followed.

## 6.2 Literature

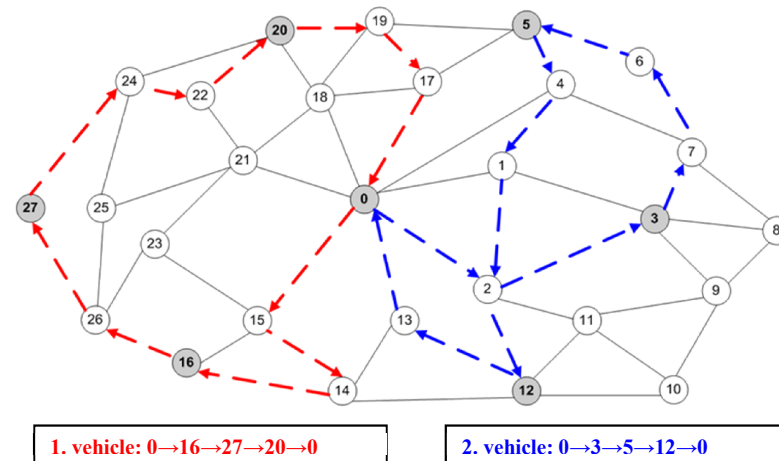
The time-dependent (TDVRP) is an extension of the VRP where the travel times between the customers are not only a function of the distance but also subject to variation due to weather and traffic conditions, accidents and similar random events and the characteristic features of the road such as the structure and the number of the lanes. In addition, the hourly, daily, weekly and even seasonal cycles in the vehicle volume on the road also cause temporary fluctuations in the travel time (Malandraki and Daskin, 1992). Despite the relatively scarce number of studies in the literature compared to the other types of VRP, research on the TDVRP has recently gained momentum with the increasing focus on green logistics and green supply chain concepts. These studies aim to obtain a least cost distribution plan by taking advantage of time-dependent travel times. The necessity and importance of examining the VRP models taking into consideration the environmental and social impact of transportation as well as the economic costs are emphasized by McKinnon (2007) and Sbihi and Eglese (2010); the studies that considers the fuel consumption and carbon emissions within the concept of green framework have recently begun to appear in the literature.

As the above example indicates, the greenest path between each node pair on the network and for each possible departure time should be found. For the solution of the VRP problems in this chapter, we will use the paths and GHG/fuel consumption amount data found via the GPA proposed in Chapter 2.

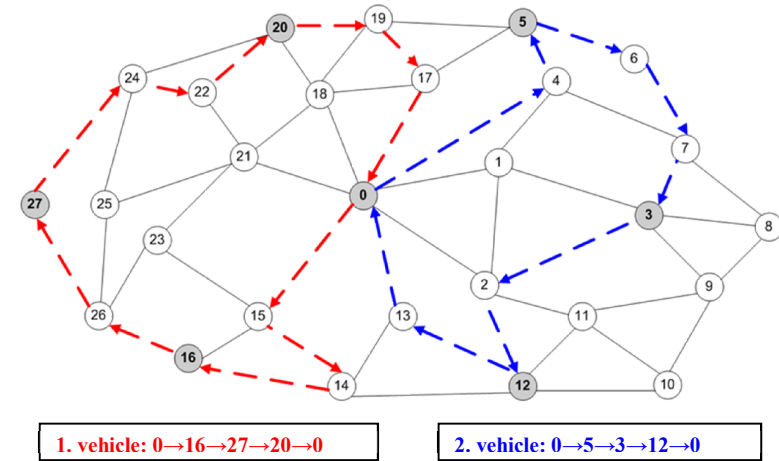
In the TDVRP literature time-dependency is taken into consideration in two ways: stochastic travel times and deterministic travel times. The first study where the time-dependency is built upon deterministic setting belongs to Ahn and Shin (1991). In this study, the important non-passing or first-in-first-out (FIFO) property was introduced and good results were obtained using the basic routing heuristics in the literature. Malandraki and Daskin (1992) examined mixed integer linear programming formulations for the VRP as well as for the TSP and presented several nearest neighbor



(a) Routes followed when the departure time is 0



(b) Routes followed when the departure time is 1



(c) Routes followed when the departure time is 2

Figure 6.1. A network structure example where the routes followed change with the changing times of the departure (departure times: (a)  $t=0$ , (a)  $t=1$ , (a)  $t=2$ )

heuristic based algorithms. Hill and Benton (1992) proposed a time-dependent travel speed based model for the VRP. However the FIFO property is disregarded in both Malandraki and Daskin (1992) and Hill and Benton (1992). In Park and Song (1997), the model of Hill and Benton (1992) was modified and savings, proximity priority searching and insertion techniques were utilized. In that study the travel times are defined as a function of different passing areas and discrete time intervals which were also introduced in this study. In a similar study (Park, 2000), an algorithm called BC-savings was introduced to minimize two conflicting objectives, which are the operation time and the total weighted tardiness. Taking the rush hours into account, Ichoua et al. (2003) divided the scheduling horizon into three time intervals and considered three types of roads which also affect the travel time. They implemented a parallel tabu search approach and tested its performance both in dynamic and static environments. Furthermore, Zheng and Liu (2006) employed a hybrid intelligent algorithm to minimize the total distance traveled by regarding the travel time as a fuzzy variable. Donati et al. (2008) used ant colony optimization in a multi-colony setting where the first colony was utilized to minimize the total number of vehicles whereas the second colony was utilized to minimize the total distance traveled. A speed distribution related with the road length accounted for the time dependency.

Stochastic travel times in VRP were first introduced in Laporte et al. (1992). They presented three mathematical programming models and used a branch-and-cut approach. Kenyon and Morton (2003) examined the same problem by developing two models. The first model aimed at minimizing the expected completion time whereas the objective of the second model was maximizing the probability that the operation is completed without exceeding a preset target time. The actual travel times of the routes regarding the random travel times were computed after the route construction phase. The stochastic nature of the travel times in Potvin et al. (2006) arises from the short term bias factor that depends on a random variable distributed uniformly. Woensel et al. (2007) incorporated the traffic congestion into their model through a queuing approach by modeling the behavior of the traffic flows. They used the mean of the speed distributions as the expected total travel time. Solving the best neighbor choice sub-problem using dynamic programming, Hashimoto et al. (2008) handled the travel times and costs as a function of time and used iterated local search to obtain the overall solution. For the time-dependent travel times, the three different time zones and three different road types approach was adopted in Ichoua et al. (2003). Kuo et al. (2009)

improved the initial solutions that are obtained by the nearest neighbor and sweep algorithms using the tabu search algorithm. In addition, the proposed method was also tested on a real life problem. Different from the previously mentioned studies, Soler et al. (2009) proposed an approach to obtain the optimal solution for the TDVRP by converting it to the asymmetric capacitated VRP.

Another study that belongs to the second group and also aims to minimize the fuel consumption is Kuo (2010). This study focused on the load of the vehicles and used simulated annealing to obtain a least fuel consuming plan. Kara et al. (2007) addressed energy minimizing VRP by using a weighted load function (load x distance) instead of only the total distance. A weight-based objective function that is based on the basic rules of physics was introduced. However, other factors such as speed, vehicle weight, friction and air resistance were not considered. In a similar study, Hsu et al. (2007) addressed the distribution of perishable food where the commodity is subject to quality changes due to the time-varying temperatures and time-dependent travel times. Besides the transportation costs, they tried to minimize the inventory, energy and penalty costs related to late deliveries. Analyzing a distribution problem in England as a case study Maden et al. (2010) observed that CO<sub>2</sub> emission could be decreased by 7% by using a heuristic approach. In a similar problem, Jabali et al. (2012) defined the emission amount as a nonlinear function of travel speed and aimed to find the optimal speeds using iterative tabu search methods.

Although these studies use time-dependent routing approach, the environmental effects and carbon emission are treated indirectly or the fuel consumption is considered based on the load carried. Instead of directly obtaining the route that yields minimum fuel consumption/GHG, it is assumed that these amounts are related to the length of the road, the load, and speed. One of the most recent studies in this subject belongs to Bektaş and Laporte (2011). This study analyzes how the solutions differ under four different objective functions that are (i) distance traveled, (ii) the load of the vehicle under constant speed, (iii) energy consumption under variable speed, and (iv) carbon emission, driver cost and fuel consumption. The optimal solutions are sought using small instances. In addition, Demir et al. (2012) tried to solve larger instances using adaptive large neighborhood search. Nevertheless, neither of these studies which emphasized the need for sustainable routing approaches used a time-dependent transportation network but assumed that the path between any customer pair is constant.

### 6.3 Computational Study

In this chapter, we use TbAS, proposed in Chapter 5, with slight modifications in order to handle the time-dependent characteristics of the problem. We conduct tests on real data set of Washington D.C. We create an instance with a single depot and 25 customers.

Table 6.1. Customer details of the instance

Node ID	Demand	Node ID	Demand	Node ID	Demand
9328	-	5562	27	9441	5
9524	13	6184	13	9028	17
378	29	3270	16	8539	14
9454	17	3661	15	9488	25
179	30	3728	10	9358	17
1577	11	4166	17	8251	12
2742	10	4803	22	9420	24
2172	12	6476	14	5092	21
5479	17	6250	21		

The details of the customers are given in Table 6.1. Note that, the Node ID column refers to the original ID values of the nodes in the real data. We use node ID and customer ID interchangeably. The first node with ID 9328 refers to the depot. The demand values are distributed between 5 and 30 units. The capacity of a single vehicle is 80 units. The distribution of the customers is shown in Figure 6.2. The depot is shown with a yellow square and the customers are shown with blue circles.

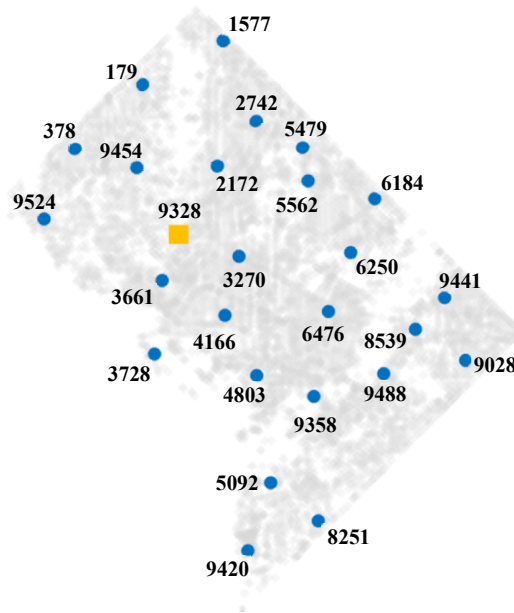


Figure 6.2. VRP instance based on Washington DC data



We have two different solutions where the time-dependent, GHG minimizing Green VRP is compared with the time-independent version. The results are summarized in Table 6.2. The second and the third columns give the distance values of each solution in detail. The fourth and the fifth columns give the corresponding GHG emission values in grams. The total distance of the Green VRP is 2.78% higher whereas it generates 5.31% less GHG emission.

Table 6.2. Comparison of time independent and time-dependent VRP solutions

	Distance (km)		GHG emission (g)	
	Time-Independent VRP	Green VRP	Time-Independent VRP	Green VRP
Route 1	17.25	17.59	3,591.76	3,213.33
Route 2	20.31	22.56	4,284.10	4,157.88
Route 3	21.00	22.09	4,242.79	4,094.42
Route 4	20.34	20.34	3,717.50	3,717.50
Route 5	29.99	30.20	6,524.00	5,686.15
Route 6	31.06	31.06	5,708.90	5,708.90
TOTAL	139.95	143.85	28,069.04	26,578.17

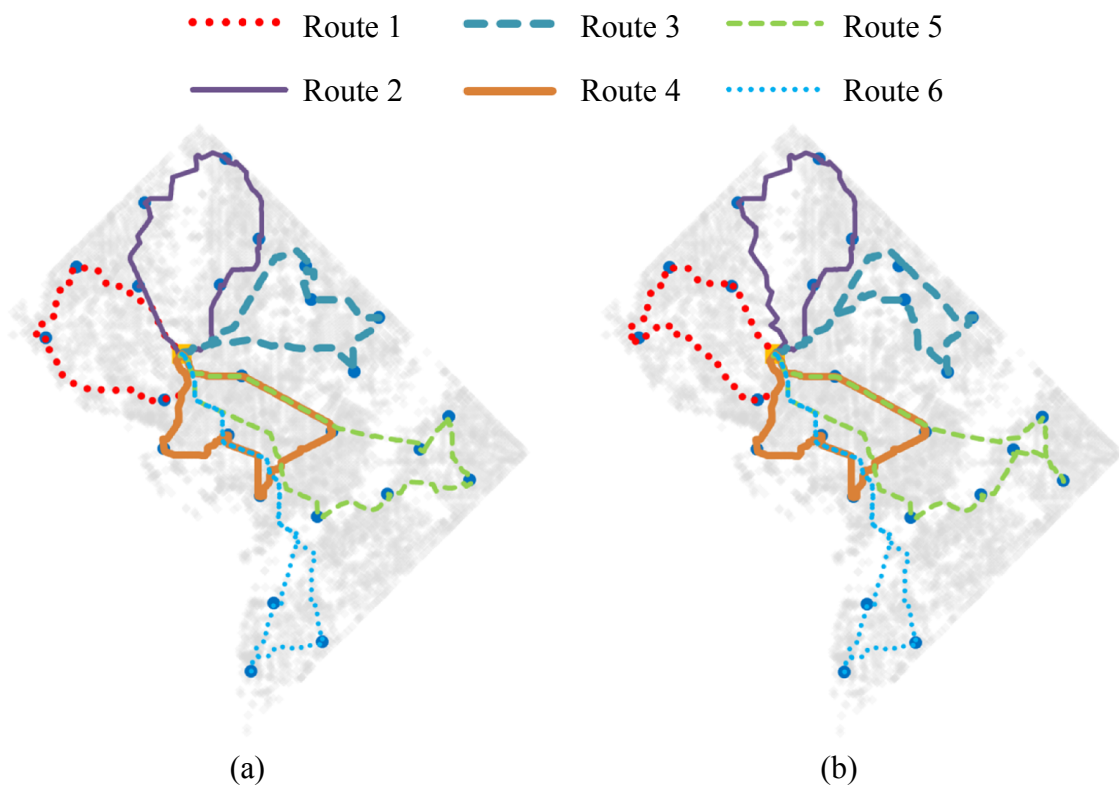


Figure 6.3. Solving VRP using (a) time-independent and (b) time-dependent information.

The routes generated are shown in Figure 6.3. Both methods yield six routes, two of them, namely route 4 and 6, being exactly the same. In the Green VRP case, customer 9454 is served by the first route instead of route 2, decreasing the total GHG emission by 504.65 grams. Although the third route serves the same set of customers in both of the solutions, the order of the customers are different. The congestion between the customers 5479 and 5562 causes the customer 5562 to be visited last in the route in Green VRP. Similarly, the highly congested link between the customers 9028 and 9488 is not preferred in the Green VRP which helps to decrease the GHG emission of route 5 by 837.85 grams.

#### **6.4 Conclusion and Future Research**

In this section, we compare the effects of the time-independent and time-dependent green objective on the GHG emissions. We observe that, building the routes by taking the congestion and the time-dependent travel times into account yields a decrease in the emission values as expected. This gain not only comes from using different links and routes but also from visiting the same customers in a different order.

For the future research, we will solve the Green VRP with time-windows for which we anticipate the gain to be less compared to the case where there are no time-windows. We expect the limitation on the visit times to force the vehicle to travel in the congestion even if just occasionally.

#### **6.5 References**

- Ahn, B.-H. and J.-Y. Shin (1991). "Vehicle-routing with Time Windows and Time-varying Congestion." J Oper Res Soc **42**(5): 393-400.
- Bektaş, T. and G. Laporte (2011). "The Pollution-Routing Problem." Transportation Research Part B: Methodological **45**(8): 1232-1250.
- Demir, E., T. Bektaş and G. Laporte (2012). "An adaptive large neighborhood search heuristic for the Pollution-Routing Problem." European Journal of Operational Research **223**(2): 346-359.
- Donati, A. V., R. Montemanni, N. Casagrande, A. E. Rizzoli and L. M. Gambardella (2008). "Time dependent vehicle routing problem with a multi ant colony system." European Journal of Operational Research **185**(3): 1174-1191.
- Hashimoto, H., M. Yagiura and T. Ibaraki (2008). "An iterated local search algorithm for the time-dependent vehicle routing problem with time windows." Discrete Optimization **5**(2): 434-456.
- Hill, A. V. and W. C. Benton (1992). "Modelling Intra-City Time-Dependent Travel Speeds for Vehicle Scheduling Problems." The Journal of the Operational Research Society **43**(4): 343-351.

- Hsu, C.-I., S.-F. Hung and H.-C. Li (2007). "Vehicle routing problem with time-windows for perishable food delivery." Journal of Food Engineering **80**(2): 465-475.
- Ichoua, S., M. Gendreau and J.-Y. Potvin (2003). "Vehicle dispatching with time-dependent travel times." European Journal of Operational Research **144**(2): 379-396.
- Jabali, O., T. Van Woensel and A. G. de Kok (2012). "Analysis of Travel Times and CO2 Emissions in Time-Dependent Vehicle Routing." Production and Operations Management **21**(6): 1060-1074.
- Kara, İ., B. Kara and M. K. Yetis (2007). Energy Minimizing Vehicle Routing Problem. Combinatorial Optimization and Applications. A. Dress, Y. Xu and B. Zhu, Springer Berlin Heidelberg. **4616**: 62-71.
- Kenyon, A. S. and D. P. Morton (2003). "Stochastic vehicle routing with random travel times." Transportation Science **37**(1): 69-82.
- Kuo, Y. (2010). "Using simulated annealing to minimize fuel consumption for the time-dependent vehicle routing problem." Computers & Industrial Engineering **59**(1): 157-165.
- Kuo, Y., C.-C. Wang and P.-Y. Chuang (2009). "Optimizing goods assignment and the vehicle routing problem with time-dependent travel speeds." Computers & Industrial Engineering **57**(4): 1385-1392.
- Laporte, G., F. Louveaux and H. Mercure (1992). "The vehicle routing problem with stochastic travel times." Transportation Science **26**(3): 161-170.
- Maden, W., R. Eglese and D. Black (2010). "Vehicle Routing and Scheduling with Time Varying Data: A Case Study." Journal of the Operational Research Society **61**(3): 515-522.
- Malandraki, C. and M. S. Daskin (1992). "Time Dependent Vehicle Routing Problems: Formulations, Properties and Heuristic Algorithms." Transportation Science **26**(3): 185-200.
- McKinnon, A. (2007). CO2 Emissions from Freight Transport in the UK. Technical Report, London, UK, Commission for Integrated Transport.
- Park, Y.-B. (2000). "A solution of the bicriteria vehicle scheduling problems with time and area-dependent travel speeds." Computers and Industrial Engineering **38**(1): 173-187.
- Park, Y.-B. and S.-H. Song (1997). "Vehicle scheduling problems with time-varying speed." Computers & Industrial Engineering **33**(3-4): 853-856.
- Potvin, J., Y. Xu and I. Benyahia (2006). "Vehicle routing and scheduling with dynamic travel times." Computers & Operations Research **33**: 1129-1137.
- Sbihi, A. and R. Eglese (2010). "Combinatorial optimization and Green Logistics." Annals of Operations Research **175**(1): 159-175.
- Soler, D., J. Albiach and E. Martínez (2009). "A way to optimally solve a time-dependent Vehicle Routing Problem with Time Windows." Operations Research Letters **37**(1): 37-42.
- Woensel, T., L. Kerbache, H. Peremans and N. Vandaele (2007). "A Queueing Framework for Routing Problems with Time-dependent Travel Times." Journal of Mathematical Modelling and Algorithms **6**(1): 151-173.
- Zheng, Y. and B. Liu (2006). "Fuzzy vehicle routing model with credibility measure and its hybrid intelligent algorithm." Applied Mathematics and Computation **176**(2): 673-683.

**7 A PARALLEL MATHEURISTIC FOR SOLVING THE VEHICLE  
ROUTING PROBLEMS**

# A parallel matheuristic for solving the vehicle routing problems

Umman Mahir Yıldırım, Bülent Çatay

## Abstract

In this chapter, we present a matheuristic approach for solving the Vehicle Routing Problems (VRP). Our approach couples the Ant Colony Optimization (ACO) algorithm with solving the Set Partitioning (SP) formulation of the VRP. As the ACO algorithm, we use a rank-based ant system approach where an agent level-based parallelization is implemented. The interim solutions which correspond to single vehicle routes are collected in a solution pool. To prevent duplicate routes, we present an elimination rule based on an identification key that is used to differentiate the routes. After a pre-determined number of iterations, the routes accumulated in the solution pool are used to solve the SP formulation of the problem to find a complete optimal solution. Once the optimal solution is obtained it is fed back to ACO as an elite solution that can be used in the pheromone reinforcement procedure. Our experimental study using the well-known VRP with Time-Windows benchmark instances of Solomon shows that the proposed methodology provides promising results.

**Keywords** Vehicle routing problem, matheuristic, ant colony optimization.

## 1. Introduction

This chapter deals with one of the most widely known combinatorial optimization problems, namely the Vehicle Routing Problem (VRP). The basic VRP aims to serve a set of geographically dispersed customers with known demands, using a homogeneous fleet of capacitated vehicles located at a central depot. The objective is to determine the best set of routes that minimizes either the total distance travelled or the number of routes while complying with the following constraints: (i) every route starts and ends at the central depot, (ii) each customer is assigned to a single route, and (iii) the vehicle capacity is not exceeded. In the vast literature on the VRP and its variants, exact methods, heuristics and metaheuristics are widely used. In addition, hybridization of these heuristics/metaheuristics as well as the exact methods has received notable attention. Yet, articles presenting matheuristic approaches for solving the VRPs are recently gaining momentum.

Matheuristics may be considered as a special case of hybrid heuristics. Boschetti et al. (2009) claim that the interoperation of metaheuristics and mathematical programming techniques yields the matheuristics and the features derived by the mathematical model of the problem are further exploited by the metaheuristic. On the

other hand, Bertazzi and Speranza (2012) define a matheuristic as any heuristic that utilizes mathematical programming in one of its solution steps. In this notion, the mathematical model can be embedded in the solution procedure in several ways such as solving sub-problems, solving parts of an instance, restricting the search space and exploring neighborhoods. Some recent matheuristic approaches and applications can be found in Maniezzo et al. (2010).

Doerner and Schmid (2010) classify the matheuristics for the VRP under three categories based on local branching, decomposition and set-partitioning/set-covering formulations. Our approach falls within the last category. In this category, first a heuristic/metaheuristic method generates preferably high quality solutions. Also, giving more importance to the solution diversification could be preferred as it may help to escape local optima and also generate a high quality solution. Then, these solutions are fed as columns for the set-partitioning/set-covering formulation of the problem. This approach has been adopted for solving different VRPs such as the capacitated VRP (Kelly and Xu, 1999; Groër et al., 2010), the VRP with time-windows (VRPTW) (Alvarenga et al., 2007), the periodic VRPTW (Pirkwieser and Raidl, 2009) and the stochastic VRP (Mendoza and Villegas, 2013).

For the split delivery VRP Archetti et al. (2008) implemented a Tabu Search (TS) approach. They identified the promising parts of the solution space with the TS and further explored them using the integer programming (IP). Gulczynski et al. (2011) developed an IP-based heuristic for the periodic VRP. In their parallel algorithm, Groër et al. (2011) combined a local search heuristic with IP for solving the VRP. Recently, Subramanian et al. (2012) and Subramanian et al. (2013) have coupled iterated local search (ILS) with mixed IP (MIP) in a matheuristic environment.

Matheuristic approaches have been implemented for many other routing problem variants such as the truck and trailer routing problem (Villegas et al., 2013), the dial-a-ride problem (Wolfler Calvo and Touati-Moungla, 2011), the traveling salesman problem (Rodríguez-Martín and Salazar-González, 2011) and the technician routing and scheduling problem (Pillac et al., 2013).

In this study, we present a parallel matheuristic approach, namely MathAnt, for solving the VRP. Our approach couples the ACO approach with solving the SP formulation of the VRP. To the best of our knowledge, this is the first attempt to integrate these two methods to solve a combinatorial optimization problem.

The remainder of this chapter is structured as follows. The next section contains a general description of our algorithmic approach. Section 3 proposes an elimination method to handle duplicate routes. The computational results are presented in Section 4. In Section 5, we give the concluding remarks and the future research directions.

## **2. A parallel matheuristic method: MathAnt**

The proposed method is based on the idea that the solutions generated by an algorithm may contain a subset of partial solutions which, when combined, can yield a better solution. Nevertheless, to generate such a promising subset, the solution method itself should be able to produce both distinct and good partial solutions. One such method is the ACO which is a constructive algorithm that builds diverse solutions at each iteration by using the foraging behaviour of ants. For the VRP, the algorithm has the potential to find high quality partial solutions, i.e. vehicle routes that can be combined to obtain improved complete solutions. Building upon this potential, MathAnt integrates the ACO with IP in an attempt to efficiently solve the VRPs. It basically solves the SP formulation of the VRP at certain iterations of the ACO using the routes constructed by ant colonies. Gendreau and Potvin (2005) claim that running several threads concurrently in a parallel exploration context seem to be very promising compared with the various implementations reported in the literature. In addition, the foraging behaviour of the ants in the ACO is suitable for parallelization. So, to further enhance the performance of the algorithm, even if not in terms of the solution quality, we implemented an agent level-based parallelization.

The general scheme of the algorithm is given in Figure 1. In our implementation, we use IBM ILOG CPLEX for solving the SP formulation. In the ACO phase, we use the Time-based Ant System (TbAS) presented in Yildirim and Çatay (2012). TbAS may only be applied to the VRP with time-windows since it uses the time-window nature of the problem in the visibility mechanism. It has a multi-layer pheromone network structure to distinguish the pheromone levels belonging to different time intervals and utilizes the timing of the visit as implicit heuristic information in the route construction phase. Basically, it takes into account time-wise desirability to travel from one customer to the next with-in the random selection rule. In TbAS, each foraging individual ant of the colony moves independently until reaching the food source, which stands for a complete solution. In the pheromone update procedure only the best-so-far ant and the elite ants are allowed to deposit pheromone. The amount of the pheromone deposited is

inversely proportional to the rank of the ant in the colony in terms of solution quality. In other words, for the  $w-1$  elite ants and the best-so-far ant, the pheromone amounts of the  $k^{\text{th}}$  elite ant and the best-so-far ant ( $bs$ ) are  $(w-k)/L_k$  and  $w/L_{bs}$  respectively. Here,  $L_k$  and  $L_{bs}$  denote the total length of the complete solution of the  $k^{\text{th}}$  elite ant and  $bs$ . We refer the interested reader to (Yildirim and Çatay, 2012) for the details of the TbAS approach.

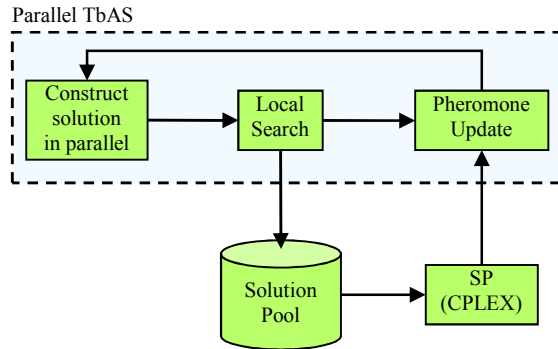


Figure 1. Flow chart of MathAnt

In the MathAnt implementation, in addition to the best-so-far ant and the elite ants (referred to as ACO-ants) the optimal solution obtained by solving the SP formulation is also used to further enhance the pheromone trails. The optimal solution is achieved by the so-called CPLEX-ant. When the CPLEX-ant is used to update the pheromone network, it is given a weight proportional to the weight of  $bs$ . Hence, to intensify the search near the CPLEX-ant, one can amplify its relative weight with respect to that of  $bs$  since it corresponds to the highest quality solution.

### 3. Handling duplicate routes

In the ACO, the ants in the colony do not necessarily construct distinct routes within the same iteration or in different iterations. The same route can be found multiple times by different ants, particularly when the algorithm is converging or stagnating. These duplicate routes can be handled in two ways. One alternative is to eliminate them while the pool of routes is being updated at the end of each iteration by comparing a newly constructed route against the existing ones in the pool. In this case, the whole route information should be recorded to make a full comparison. This will obviously be very time consuming; however, the SP problem will be solved using only unique routes, which may reduce the presolve processing time of CPLEX significantly. The other



alternative is to add the constructed routes to the pool without any comparison and let CPLEX perform the elimination using its presolve process. In this case, the CPLEX presolve process may be more efficient in CPU time, nevertheless, keeping the duplicate routes will increase the size of the pool, which may require significant additional memory.

In the former case, each route should be assigned a unique value (referred to as identification key), if possible. The goal is to minimize the number of false eliminations, i.e. counting two different routes as the same. Matching each customer number with a unique prime number and multiplying the corresponding prime numbers of the customers in the route will perfectly and uniquely represent any route and will prevent false eliminations. However, as the number of customers in the route increases, this multiplication becomes intractable. The multiplication of the first 20 prime numbers only yields a value of 5.58E+26, which is far larger than the maximum value that can be stored in any programming language. Using any single characteristic of the route such as the total distance, total time, the number of customers, etc. as the discrimination criterion may yield many false eliminations. So, we considered and analyzed four different criteria as summarized in Table 1.

Table 1. Criteria used in eliminating duplicate routes

Criteria	Description
I1	$31^4D + 31^3FC + 31^2LC + 31NC$
I2	$31^4D + 31^3FC + 31^2LC + 31NC + TT$
S1	$D_S + '-' + FC_S + '-' + LC_S + '-' + NC_S$
S2	$D_S + '-' + FC_S + '-' + LC_S + '-' + NC_S + '-' + TT$

As illustrated in Table 1, we utilized five integer representative discriminative characteristics, namely the total distance (D), first customer ID number (FC), last customer ID number (LC), total number of customers (NC), and total tour time (TT). Any characteristic with a subscript S denotes its string counterpart. Using these characteristics in the given order, we have mainly 2 criteria groups based on integer (I) and string (S) values. In the integer subgroup, the criterion values are multiplied with a certain power of a prime number. Using a prime number in hashing is traditional. Here, we used 31 as the prime number. In addition to being an odd prime, 31 has a nice property that the multiplication can be replaced by a shift and subtraction for better performance (Blosch, 2008). So, the coefficients of  $D$ ,  $FC$ ,  $LC$ ,  $NC$  and  $TT$  are set to  $31^4$ ,  $31^3$ ,  $31^2$ , 31 and 1 respectively. In the string subgroup, the criterion values are

concatenated with a hyphen. The hyphen is used as a separator to differentiate routes such as “0-1-23-0” and “0-12-3-0”, where 0 denotes the depot and the remaining numbers are the customer ID numbers which show the sequence of their visits. The performances of these four criteria are tested in Section 4.1.

Table 2. Elimination criteria: false elimination

D	FC	LC	NC	TT	Identification Key I1	Identification Key I2
183	5	9	3	362	169162040	169162402
182	35	40	5	300	169162102	169162402

It is noteworthy to remark that the inclusion of each additional criterion most often decreases false eliminations, if not always. In other words, including an additional information does not always help to differentiate two routes. The example given in Table 2 shows how the inclusion of the tour time as additional information prevents distinguishing two different routes. The second and the third rows in this table provide information related with two distinct vehicle routes: 0-5-12-9-0 and 0-35-42-45-33-40-0. The identification keys for these two routes according to I1 yields different values as shown in the sixth column. On the other hand, the identification keys according to I2 returns the same value for both routes, which will result in a false elimination.

#### 4. Computational Analysis

We have tested the performance of the proposed approach on the well-known VRPTW instances of Solomon (1987). These instances have three main sets which differ by the distribution of the customers over a 100x100 grid. The customers are clustered (C), randomly distributed (R) or both clustered and randomly distributed (RC). Each set is also divided into two subsets as type 1 and type 2 which have different time window lengths and vehicle capacities. The parameters of TbAS have been set as described in Yildirim and Çatay (2012). We have used an Intel Core2 Quad 2.33 GHz computer with 8.0 GB RAM and 64-bit operating system. The IP-solver is IBM ILOG CPLEX version 12.2.

##### 4.1. Comparing the elimination methods for duplicate routes

To evaluate the four criteria described in Section 3 we performed a single run using the first and the last instances of each subset of Solomon data C1, C2, R1, R2, RC1 and

RC2. To better observe how the elimination methods reduce the pool size, we set the iteration limit to 300 to accumulate a large number of routes. The results are summarized in Table 3. The first column shows the instance. The second and third columns report the total number of routes in the pool and the total number of unique routes, respectively. The last four columns correspond to the number of routes in the reduced pool after applying the four elimination criteria.

Table 3. Comparing elimination criteria

Instance	Number of routes	Number of unique routes	Number of routes using I1	Number of routes using I2	Number of routes using S1	Number of routes using S2
C101	1,458,513	154,676	133,329	148,605	135,726	148,876
C109	972,487	492,785	427,913	464,680	436,073	465,016
R101	2,075,507	50,698	45,860	48,329	46,594	48,556
R112	1,005,643	644,038	480,661	563,178	516,548	546,686
RC101	1,735,879	91,973	79,206	89,300	81,502	89,459
RC108	1,136,336	537,137	405,375	473,012	434,536	474,441
Type 1 Average	1,397,394	328,551	262,057	297,851	275,163	295,506
C201	936,673	393,816	346,336	381,549	354,753	382,534
C208	602,936	402,030	383,520	400,883	386,667	400,972
R201	961,235	492,038	440,795	486,130	448,121	486,361
R211	428,969	392,145	376,448	390,833	379,762	390,913
RC201	1,002,893	460,026	409,595	452,793	420,498	453,093
RC208	481,032	405,270	386,936	400,252	389,806	400,323
Type 2 Average	735,623	424,221	390,605	418,740	396,601	419,033
Total Average	1,091,961	372,706	321,387	353,646	331,211	352,518

We first analyze the average number of routes by taking all 12 instances into consideration. We observe that on the average 1.091 million routes are obtained, out of which 372 thousand (35%) are unique. The number of all routes found in type 1 instances (C1, R1 and RC1) is nearly twice the number of routes found in type 2 instances (C2, R2 and RC2): 1.397 million routes compared to 735 thousand routes, respectively. This is basically the result of longer routes involving more customers typically obtained in type 2 problems. On the other hand, the total number of unique routes found in type 1 and type 2 instances are 329 thousand and 424 thousand, respectively. So, we see that wider time windows in type 2 instances extend the size of the solution space, as expected.

Applying any of the elimination criteria decreases the size of the pool but at the expense of eliminating unique routes as well. Integer criteria I1 and I2 eliminate 12.2% and 4.1% of the unique routes, respectively. On the other hand, the false eliminations by

using string criteria S1 and S2 are 10.0% and 4.2%, respectively. So, we observe that the number of different discrimination criteria plays a more important role compared to the main group of the criteria (integer or string). Both I2 and S2 involving 5 different characteristics are able to keep more than 95% of the unique routes. Nevertheless, as the number of nodes increases, a single string representation of a node uses more memory compared to that of integer (4 bytes). Thus, taking the memory usage into account we decided to implement I2 criterion.

#### **4.2. Elimination of routes: Elimination method vs CPLEX presolve process**

In this section, we analyze how to eliminate the duplicate routes, either via the proposed elimination method or CPLEX presolve. All the tests in this section are conducted using the 39 instances in R1, R2, RC1 and RC2 sets of Solomon. Since the optimal solutions can be easily obtained for the clustered instances of C1 and C2 sets, they are omitted as their sensitivity to parametric changes cannot be evaluated.

The detailed computational time analysis is given in Table 4. All time units are in seconds. The number of CPLEX calls directly affects the solution quality (analyzed in detail in Section 4.3) and the computational time. Thus, we tested 3 different CPLEX call frequency settings in a run with 100 iterations: 1, 2, and 5. Note that CPLEX is run only once at the end of the ACO procedure when CPLEX Call Frequency=1 whereas CPLEX Call Frequency=5 represents that optimization using CPLEX is performed 5 times, after every 20 iterations.

We observe that the computational time of the algorithm using duplicate route elimination is 2.96% longer compared to the elimination through CPLEX presolve. Taking into consideration this small margin one can question the benefit of implementing the duplicate route elimination method. However, when the size of the solution pool increases, the memory requirement and the time spent by CPLEX also increase and leaving the route elimination procedure to CPLEX may not be favorable. On a sample run with instance R101, when the number of the routes in the solution pool reached up to 8.5 million, CPLEX failed to solve the SP problem because of excessive memory requirements. Nonetheless, applying the elimination criteria beforehand kept the size of the solution pool at most 24,049 routes, which in turn allowed finding the optimal solution in seconds.

Table 4. Elimination of duplicate routes

CPLEX Call Frequency	Set	Duplicate Route Elimination		CPLEX Presolve	
		Number of Routes	Average CPU Time (sec)	Number of Routes	Average CPU Time (sec)
5	R1	37,991.60	165.45	147,806.18	148.25
	R2	59,170.24	545.20	72,515.71	525.72
	RC1	36,755.78	118.40	146,083.93	122.40
	RC2	55,443.15	361.48	79,038.03	355.21
	Average	47,291.36	303.12	112,110.84	291.87
2	R1	39,730.20	122.92	148,734.05	117.82
	R2	57,823.65	466.97	73,009.76	463.39
	RC1	37,311.23	100.65	146,683.90	94.38
	RC2	55,988.98	336.80	78,833.80	321.22
	Average	47,672.42	259.26	112,616.86	252.2
1	R1	37,266.07	110.56	149,681.03	105.75
	R2	57,569.75	445.76	73,347.18	438.98
	RC1	35,964.50	92.71	146,624.80	84.34
	RC2	54,822.25	321.16	78,938.78	314.19
	Average	46,327.03	244.64	113,012.82	238.10

### 4.3. Effect of parameters on the solution quality

The frequency of the CPLEX calls and the pheromone reinforcement weight of the SP-ant affect the solution quality. Table 5 reports the average solution quality of 5 runs for different parameter combinations. The average solution quality does not show a significant difference across different parameter settings. Nevertheless, the best solutions are obtained when  $\delta=5$ . Intensifying the search near the CPLEX solution in the solution space generates better solutions compared to equally exploring the solution space near the ACO and CPLEX solutions. Among different CPLEX frequency call values 5 yields the best results. The increasing frequency of the CPLEX calls helps better improve the solution, as expected. This comes at the expense of an increase in the computational effort. Calling CPLEX every 20 iterations increases the computational time by 23% compared to a single call at the end of the algorithm. In light of these results, we set  $\delta=5$  and CPLEX call frequency=5 in the following experiments.

Table 5. Solution quality for different parameter combinations

$\delta$	CPLEX Call Frequency		
	5	2	1
1	1097.91	1098.51	1099.47
2	1098.80	1099.01	1099.72
5	1097.06	1098.28	1099.04

#### 4.4. Performance against the best heuristic solutions

In this section, we compare the performance of MathAnt against the best performing heuristics and metaheuristics in the literature as well as TbAS of Yildirim and Çatay (2012) to investigate the benefit of hybridizing TbAS with IP. Table 6 and Table 7 summarize the results for type 1 and type 2 problems, respectively. Note that Yildirim and Çatay (2012) reported the results of 4 different implementations of their algorithm. In these tables, we consider the best results achieved.

Table 6. Comparison of results for type 1 problems

Instance	BKS	Ref*	TbAS	MathAnt	Gap (%) (BKS)	Gap (%) (TbAS)
R101	1642.87	[AMT]	1642.88	1642.88	0.00	0.00
R102	1472.62	[AMT]	1472.81	1472.82	0.01	0.00
R103	1213.62	[JM]	1213.62	<b>1213.62</b>	0.00	0.00
R104	976.61	[JM]	977.55	<b>976.61</b>	0.00	-0.10
R105	1360.78	[JM]	1360.78	<b>1360.78</b>	0.00	0.00
R106	1240.26	[YÇ]	1240.26	<b>1239.37</b>	-0.07	-0.07
R107	1073.01	[YÇ]	1073.01	1075.14	0.20	0.20
R108	944.44	[YÇ]	944.44	<b>938.20</b>	-0.66	-0.66
R109	1151.84	[JM]	1151.84	<b>1151.84</b>	0.00	0.00
R110	1072.41	[JM]	1072.41	1072.42	0.00	0.00
R111	1053.50	[JM]	1053.50	<b>1053.50</b>	0.00	0.00
R112	953.63	[RT]	959.58	955.68	0.21	-0.41
R1 Average	1179.63		1180.22	1179.40	-0.03	-0.09
RC101	1623.58	[RT]	1638.00	1623.59	0.00	-0.88
RC102	1461.23	[JM]	1461.44	<b>1461.23</b>	0.00	-0.01
RC103	1261.67	[S]	1262.68	<b>1261.67</b>	0.00	-0.08
RC104	1135.48	[C]	1141.66	1135.83	0.03	-0.51
RC105	1518.58	[JM]	1518.58	<b>1518.58</b>	0.00	0.00
RC106	1376.99	[YÇ]	1376.99	<b>1376.99</b>	0.00	0.00
RC107	1212.83	[JM]	1212.83	<b>1211.11</b>	-0.14	-0.14
RC108	1117.53	[JM]	1117.53	<b>1117.53</b>	0.00	0.00
RC1 Average	1338.49		1341.21	1338.32	-0.01	-0.20
Total Average	1243.17		1244.62	1242.97	-0.02	-0.13

\* AMT: (Alvarenga et al., 2007), JM: (Jung and Moon, 2002), YÇ: (Yildirim and Çatay, 2012), RT: (Rochat and Taillard, 1995), S: (Shaw, 1998), C: (Cordeau et al., 2001)

In both tables, the first column identifies the problem and the fifth column shows the results achieved by MathAnt. The second and the third columns give the best-known solutions (BKS) from the literature and the corresponding articles, respectively. The best results found by TbAS are given in the fourth column. Column six and seven report the percentage gaps between MathAnt and best-known solutions and TbAS results, respectively. A negative number shows an improvement. In general, we observe that the

Table 7. Comparison of results for type 2 problems

Instance	BKS	Ref*	TbAS	MathAnt	Gap (%) (BKS)	Gap (%) (TbAS)
R201	1147.8	[BV]	1155.8	1149.39	0.14	-0.55
R202	1034.35	[JM]	1036.6	1034.58	0.02	-0.20
R203	874.87	[JM]	875.62	877.23	0.27	0.18
R204	735.8	[BV]	746.98	740.98	0.70	-0.80
R205	954.16	[ORH]	964.64	957.33	0.33	-0.76
R206	879.89	[JM]	892.95	883.92	0.46	-1.01
R207	797.99	[BV]	805.7	810.91	1.62	0.65
R208	705.45	[JM]	711.37	712.93	1.06	0.22
R209	859.39	[JM]	876.33	<b>859.39</b>	0.00	-1.93
R210	910.7	[JM]	915.49	915.48	0.52	0,00
R211	755.82	[BV]	773.51	765.04	1.22	-1.09
R2 Average	877.84		886.82	882.47	0.58	-0.48
RC201	1265.56	[JM]	1272.63	1267.16	0.13	-0.43
RC202	1095.64	[JM]	1104.92	1096.75	0.10	-0.74
RC203	926.89	[BV]	938.19	937.76	1.17	-0.05
RC204	786.38	[JM]	800.48	789.26	0.37	-1.40
RC205	1157.55	[JM]	1157.55	<b>1157.55</b>	0.00	0.00
RC206	1054.61	[JM]	1072.08	1055.77	0.11	-1.52
RC207	966.08	[JM]	972.74	967.07	0.10	-0.58
RC208	779.31	[JM]	792.65	783.93	0.59	-1.10
RC2 Average	1004		1013.91	1006.91	0.32	-0.73
Total Average	930.96		940.33	934.86	0.47	-0.59

\* BV: (Brandão de Oliveira and Vasconcelos, 2010), JM: (Jung and Moon, 2002), ORH: (Ombuki et al., 2006), RT: (Rochat and Taillard, 1995)

proposed MathAnt method is able to generate good solutions. Combining TbAS with IP improves the solutions of type 1 and type 2 problems by 0.13% and 0.59%, respectively, compared to using TbAS alone. The average improvement of the MathAnt matheuristic over TbAS is 0.35%. When we compare MathAnt results against the best-known results from the literature, we see that the performance of MathAnt is better in type 1 problems as this was the case for TbAS as well. The average gap for type 2 problems is 0.47% whereas it is -0.02% for type 1 problems. The average results on type 1 problems reveal that MathAnt is the best performing method in the literature. Note that MathAnt improved the best-known solutions of R106, R108 and RC107 instances (as shown in bold-italic in Table 6) and matched the best-known results in 10 instances (as shown in bold in Table 6). In type 2 problems, MathAnt could match the best-known result in only 2 instances (as shown in bold in Table 7).

## 5. Conclusions

In this chapter we presented MathAnt, a parallel matheuristic that combines the ACO and the IP, for solving the VRP and its variants. We used TbAS (Yildirim and Çatay,

2012) as the ACO approach and CPLEX as the IP solver. An agent level parallelization was implemented and the pheromone reinforcement procedure was adapted so as to incorporate the CPLEX solution in TbAS. After determining the frequency of CPLEX calls we conducted experiments on the VRPTW instances to test the performance of MathAnt. The comparison results against the published best solutions in the literature show that MathAnt is capable of generating good solutions. MathAnt had superior performance on type 1 instances in particular where the time-window lengths are narrow and it improved three best-known results in the literature. Furthermore, an elimination method was investigated to cope with the duplicate routes generated by TbAS. We observed that our elimination method effectively decreases false eliminations and keeps the size of the pool of routes minimum. Although the computational time of MathAnt equipped with our elimination method is longer compared to leaving the elimination to CPLEX, the elimination method becomes advantageous especially when the solution pool size increases and CPLEX fails to generate a solution.

MathAnt can be easily implemented to solve any VRP variant using any ACO approach. In this study, we only considered the VRPTW. Further research will address the other VRP variants. Russell and Chiang (2006) state that using the set covering formulation instead of the set partitioning model in a VRP context may lead to an improved solution. We will investigate the impact of this relaxation on the performance of our algorithm as well. Moreover, we utilized parallelism for only reducing the computational time. However, a parallel implementation by devising multiple ant colonies evolving on different processors may lead to improved performance with respect to the solution quality as well as processor load balance.

## References

- Alvarenga, G. B., G. R. Mateus and G. de Tomi (2007). "A genetic and set partitioning two-phase approach for the vehicle routing problem with time windows." Computers & Operations Research **34**(6): 1561-1584.
- Archetti, C., M. G. Speranza and M. W. P. Savelsbergh (2008). "An Optimization-Based Heuristic for the Split Delivery Vehicle Routing Problem." Transportation Science **42**(1): 22-31.
- Bertazzi, L. and M. G. Speranza (2012). Matheuristics for Inventory Routing Problems. Hybrid Algorithms for Service, Computing and Manufacturing Systems: Routing and Scheduling Solutions. J. R. Montoya-Torres, A. A. Juan, L. H. Huatuco, J. Faulin and G. L. Rodriguez-Verjan. Hershey, PA, IGI Global: 1-14.
- Blosch, J. (2008). Effective Java. Boston, Addison-Wesley.



- Boschetti, M., V. Maniezzo, M. Roffilli and A. Bolufé Röbler (2009). Matheuristics: Optimization, Simulation and Control. Hybrid Metaheuristics. M. Blesa, C. Blum, L. Gaspero et al., Springer Berlin Heidelberg. **5818**: 171-177.
- Brandão de Oliveira, H. and G. Vasconcelos (2010). "A hybrid search method for the vehicle routing problem with time windows." Annals of Operations Research **180**(1): 125-144.
- Cordeau, J. F., G. Laporte and A. Mercier (2001). "A unified tabu search heuristic for vehicle routing problems with time windows." Journal of the Operational Research Society **52**(8): 928-936.
- Doerner, K. and V. Schmid (2010). Survey: Matheuristics for Rich Vehicle Routing Problems. Hybrid Metaheuristics. M. Blesa, C. Blum, G. Raidl, A. Roli and M. Sampels, Springer Berlin Heidelberg. **6373**: 206-221.
- Gendreau, M. and J.-Y. Potvin (2005). "Metaheuristics in Combinatorial Optimization." Annals of Operations Research **140**(1): 189-213.
- Groër, C., B. Golden and E. Wasil (2010). "A library of local search heuristics for the vehicle routing problem." Mathematical Programming Computation **2**(2): 79-101.
- Groër, C., B. Golden and E. Wasil (2011). "A Parallel Algorithm for the Vehicle Routing Problem." INFORMS J. on Computing **23**(2): 315-330.
- Gulczynski, D., B. Golden and E. Wasil (2011). "The period vehicle routing problem: New heuristics and real-world variants." Transportation Research Part E: Logistics and Transportation Review **47**(5): 648-668.
- Jung, S. and B. R. Moon (2002). "A Hybrid Genetic Algorithm For The Vehicle Routing Problem With Time Windows." 1309-1316.
- Kelly, J. P. and J. Xu (1999). "A Set-Partitioning-Based Heuristic for the Vehicle Routing Problem." INFORMS J. on Computing **11**(2): 161-172.
- Maniezzo, V., T. Stützele and S. Voß (2010). Matheuristics: hybridizing metaheuristics and mathematical programming. New York, Springer.
- Mendoza, J. and J. Villegas (2013). "A multi-space sampling heuristic for the vehicle routing problem with stochastic demands." Optimization Letters **7**(7): 1503-1516.
- Ombuki, B., B. J. Ross and F. Hanshar (2006). "Multi-Objective Genetic Algorithms for Vehicle Routing Problem with Time Windows." Applied Intelligence **24**(1): 17-30.
- Pillac, V., C. Guéret and A. L. Medaglia (2013). "A parallel matheuristic for the technician routing and scheduling problem." Optimization Letters **7**(7): 1525-1535.
- Pirkwieser, S. and G. Raidl (2009). Multiple Variable Neighborhood Search Enriched with ILP Techniques for the Periodic Vehicle Routing Problem with Time Windows. Hybrid Metaheuristics. M. Blesa, C. Blum, L. Gaspero et al., Springer Berlin Heidelberg. **5818**: 45-59.
- Rochat, Y. and É. D. Taillard (1995). "Probabilistic diversification and intensification in local search for vehicle routing." Journal of Heuristics **1**(1): 147-167.
- Rodríguez-Martín, I. and J. Salazar-González (2011). The Multi-Commodity One-to-One Pickup-and-Delivery Traveling Salesman Problem: A Matheuristic. Network Optimization. J. Pahl, T. Reiners and S. Voß, Springer Berlin Heidelberg. **6701**: 401-405.
- Russell, R. A. and W.-C. Chiang (2006). "Scatter search for the vehicle routing problem with time windows." European Journal of Operational Research **169**(2): 606-622.
- Shaw, P. (1998). Using constraint programming and local search methods to solve vehicle routing problems. M. Maher and J. F. Puget. Berlin, Springer: 417-431.
- Solomon, M. M. (1987). "Algorithms for the Vehicle Routing and Scheduling Problems with Time Window Constraints." Operations Research **35**(2): 254-265.

- Subramanian, A., P. H. V. Penna, E. Uchoa and L. S. Ochi (2012). "A hybrid algorithm for the Heterogeneous Fleet Vehicle Routing Problem." European Journal of Operational Research **221**(2): 285-295.
- Subramanian, A., E. Uchoa and L. S. Ochi (2013). "A hybrid algorithm for a class of vehicle routing problems." Computers & Operations Research **40**(10): 2519-2531.
- Villegas, J. G., C. Prins, C. Prodhon, A. L. Medaglia and N. Velasco (2013). "A matheuristic for the truck and trailer routing problem." European Journal of Operational Research **230**(2): 231-244.
- Wolfler Calvo, R. and N. Touati-Moungla (2011). A Matheuristic for the Dial-a-Ride Problem. Network Optimization. J. Pahl, T. Reiners and S. Voß, Springer Berlin Heidelberg. **6701**: 450-463.
- Yildirim, U. M. and B. Çatay (2012). "A time-based pheromone approach for the ant system." Optimization Letters **6**(6): 1081-1099.

## **8 COMPUTATIONAL RESULTS ON A CLASS OF VRP USING THE PARALLEL MATHEURISTIC**

## **8.1 Introduction**

In this chapter, we extend our work in Yıldırım and Çatay (2014) by taking the addressed future research directions into account. We discuss the effect of incorporating set covering (SC) formulation instead of the set partitioning (SP) model. We test the proposed method on other VRP variants, namely the basic Capacitated VRP (CVRP), Open VRP (OVRP), Heterogeneous VRP (HVRP) and Multi-Depot VRP (MDVRP), as well as the VRPTW. The implementation details of the proposed method are given in Section 7.1.

## **8.2 A brief description of the VRP variants considered**

The classical version of the VRP is the CVRP which aims to serve a set of geographically dispersed customers with known demands, using a homogeneous fleet of capacitated vehicles located at a central depot. The objective is to determine the best set of routes that minimizes either the total distance travelled or the number of routes while complying with the following constraints: (i) every route starts and ends at the central depot, (ii) each customer is assigned to a single route, and (iii) the vehicle capacity is not exceeded.

The OVRP differs from CVRP as the vehicle does not have to return to the depot after the last visited customer. This problem can be observed in real life where a third party logistic company and a distance-based payment scheme are in use.

The VRPTW extends the CVRP by introducing the earliest and latest possible visiting times, namely time windows, for each customer. In the hard time windows case, a vehicle is not allowed to visit the customers out of these intervals whereas they can be visited early or late with a penalty if soft time windows are applied. In this study, we consider hard time windows for the VRPTW.

The HVRP extends the CVRP by introducing a fleet with a limited number of vehicles with different capacities. In addition, a fixed and a variable cost are also introduced for each vehicle. The problem is referred to as fleet size mix problem (FSM) when the fleet size is not limited.

When we have more than one depot available to serve the customers, then the problem is called the MDVRP. The number of customers that each depot can serve is not limited.

### 8.3 The proposed method

First, an initial solution for the ACO is obtained using the nearest neighbour heuristic. After initializing the global pheromone network, the members of the ant colony constructs different solutions in parallel. At the end of each iteration, the global pheromone network is updated. After updating the ant with the best solution found so far (the best-so-far ant), the routes are sent to the solution pool. This step includes duplicate check procedure. After predetermined number iterations, the SP model is solved and used to update the pheromone network with the optimal solution. We refer the reader to Yıldırım and Çatay (2014) for more details.

As for the proposed method in this study, both SP and SC models are solved at the end of the total number of iterations. Note that, the SC model may result in multiple visits for a customer. Thus, the algorithm includes a post-processing step to eliminate the revisited customers if any. In the post-processing phase, the visit with the highest cost is removed from the solution. The flow chart of the proposed method for a single run is given in Figure 8.1. For the sake of simplicity, some processes such as determining the Best-so-far Ant and pheromone update of the Best-so-far Ant are not shown in the figure.

Our matheuristic also differs from the one proposed in Yıldırım and Çatay (2014) for the HVRP and MDVRP problems. In the following, we give the details of the implementations for both HVRP and MDVRP problems.

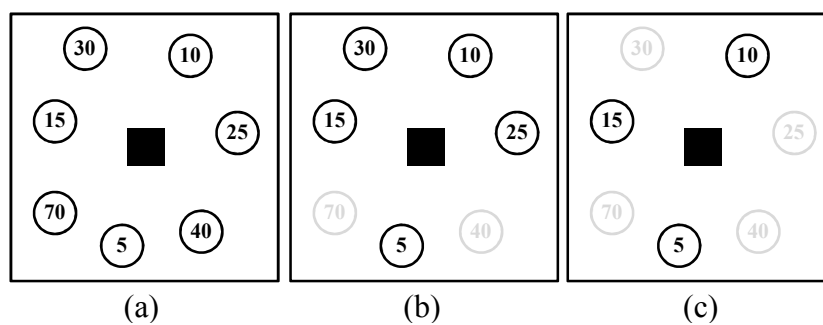


Figure 8.2. Solving the HVRP problem for each vehicle type separately

Figure 8.2 shows a sample network with a single depot and seven customers each which are shown with a square and circles respectively. The numbers in the circles represent the demands for the corresponding customer. Suppose that we are trying to solve a

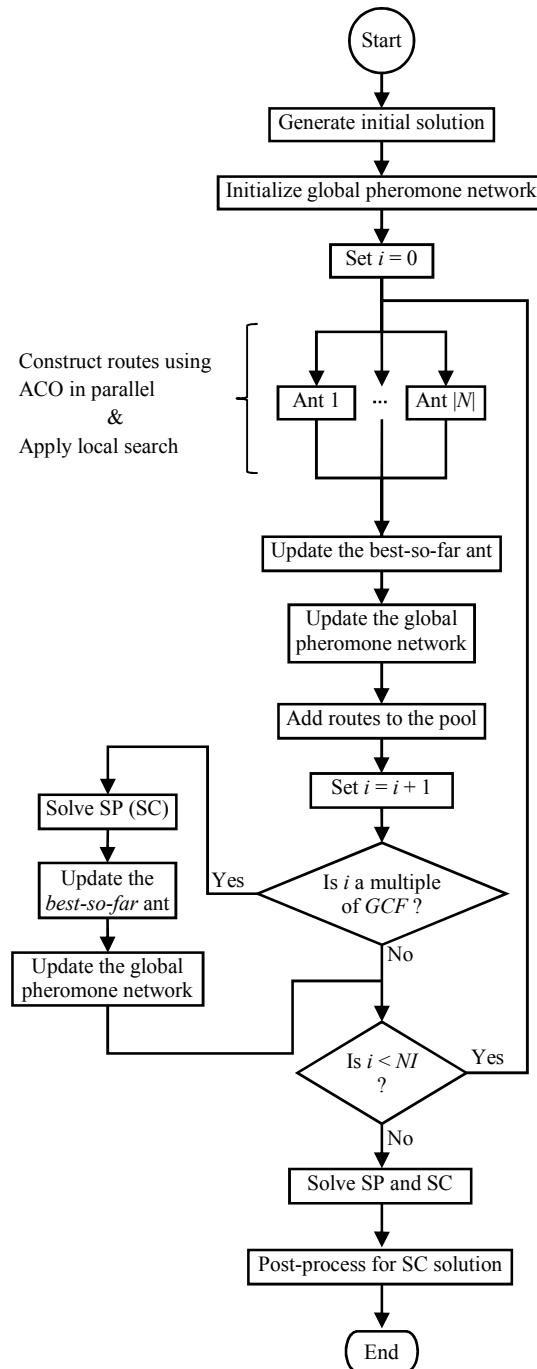


Figure 8.2. An agent based parallelization. *GCF*: global CPLEX frequency, *NI*: number of iterations.

HVRP problem with three different types of vehicles with capacities of 80, 30 and 20. The first type of vehicle is big enough to accommodate each customer separately. So, we can directly solve a homogeneous VRP on the network (Figure 8.2.a) with the first vehicle. However, the second type of vehicle cannot fit the demands of 70 and 40. So,

the network is reduced to Figure 8.2.b to obtain a feasible homogeneous VRP for the second vehicle. Furthermore, the customers with a demand value higher than the capacity of the third vehicle are eliminated to obtain the last network in Figure 8.2.c. Now, 3 different homogeneous VRP problems are solved disregarding the fleet size limits. After collecting all the solutions found for the different type of vehicles, the combined pool is fed into the SP and SC models with the original fleet size limits

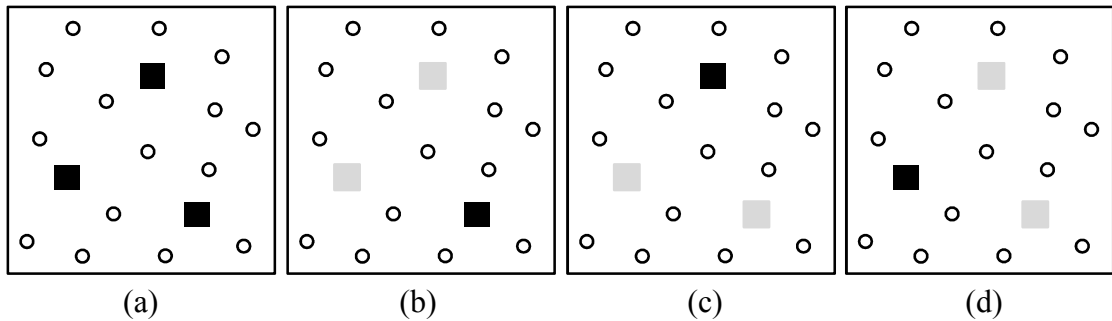


Figure 8.3. Solving the MDVRP problem for each depot type separately

A similar implementation is valid for the MDVRP problem where a sub problem is solved for each depot type. A sample network is given in Figure 8.3.a which three different depot locations. The network is modified by keeping only a single depot each time obtaining the networks in Figure 8.3.b, Figure 8.3.c and Figure 8.3.d. The idea behind is the same with the approach for solving the HVRP problem. After collecting all the solutions of the sub problems in a single pool, SP and SC models are solved to obtain a solution for the original problem.

#### 8.4 Computational Study

In this section we compare the performance of the proposed method on VRP variants in Section 8.1 with the best known/optimal solutions. All the problems are solved by using both SP and SC formulations. For comparing with the best known/optimal solutions, we use the best of the results of SC and SP formulations. A comparison of SC and SP formulations is presented separately.

We coded our algorithm in C#. We have used an Intel Core i7 4770 3.40 GHz computer with 16.0 GB RAM and 64-bit operating system. The IP-solver is IBM ILOG CPLEX version 12.6 where we limit the solving time to 300 seconds in each call. For

all the problems, we used the parameters in Yıldırım and Çatay (2014). The total number of runs is set to 30.

#### 8.4.1 Comparing with the best known/optimal solutions

##### 8.4.1.1 CVRP

The tests in this section are conducted on 14 instances of Christofides et al. (1979). Table 8.1 presents a comparison against the best heuristic solutions where BKS and CT refer to the best known solution value and the computational time in seconds respectively. The average gap is 0.25% where 9 best known solutions out of 14 problems are obtained. The gap for problems cmt05 and cmt10 with 199 customers are 2.21% and 0.64% respectively. For the remaining 12 problems with a maximum number of 150 customers, the average gap decreases to 0.06%.

Table 8.1. Comparison of results for CVRP

Instance	BKS	MathAnt	Gap (%)
cmt01	524.61	<b>524.61</b>	0.00
cmt02	835.26	<b>835.26</b>	0.00
cmt03	826.14	827.39	0.15
cmt04	1028.42	1031.82	0.33
cmt05	1291.29	1319.89	2.21
cmt06	555.43	<b>555.43</b>	0.00
cmt07	909.68	<b>909.68</b>	0.00
cmt08	865.94	<b>865.94</b>	0.00
cmt09	1162.55	<b>1162.55</b>	0.00
cmt10	1395.85	1404.72	0.64
cmt11	1042.11	1042.12	0.00
cmt12	819.56	<b>819.56</b>	0.00
cmt13	1541.14	1543.79	0.17
cmt14	866.37	<b>866.37</b>	0.00
Total Average	976.03	979.22	0.25

##### 8.4.1.2 OVRP

As the CVRP, OVRP problem is also tested on the 14 instances of Christofides et al. (1979). The average gap is 0.08% where the highest gap is 0.88% which is observed in cmt01, the smallest instance in size. 5 of the best-so-far solutions are obtained and 3 best known solutions (cmt08, cmt10 and cmt13) are improved by 0.12% on the average. The detailed routes for the new best results are given in Appendix.



Table 8.2. Comparison of results for OVRP

Instance	BKD	MathAnt	Gap (%)
cmt01	412.95	416.58	0.88
cmt02	564.06	<b>564.06</b>	0.00
cmt03	639.25	639.74	0.08
cmt04	733.13	<b>733.13</b>	0.00
cmt05	868.44	869.19	0.09
cmt06	412.95	412.96	0.00
cmt07	566.93	566.94	0.00
cmt08	642.11	<b>641.55</b>	-0.09
cmt09	741.44	<b>741.44</b>	0.00
cmt10	871.58	<b>869.80</b>	-0.20
cmt11	678.54	682.12	0.53
cmt12	534.24	<b>534.24</b>	0.00
cmt13	836.55	<b>835.90</b>	-0.08
cmt14	552.64	<b>552.64</b>	0.00
Total			
Average	646.77	647.20	0.08

### 8.4.1.3 MDVRP

We use test instances of (Cordeau et al., 1997) for testing the performance of the algorithm on the MDVRP. The average gap over 7 problems is 1.97%. The MDVRP implementation that solves a single depot VRP for each depot obtains solutions below 3.01% gap range, yet, is incapable of obtaining the best known solutions.

Table 8.12. Summary for MDVRP

Instance	BKS	MathAnt	Gap (%)
p01	576.87	584.99	1.41
p02	473.53	482.34	1.86
p03	641.19	657.61	2.56
p04	1001.04	1008.6	0.76
p05	750.03	772.63	3.01
p06	876.50	888.93	1.42
p07	881.97	906.44	2.77
Average	743.01	757.37	1.97

### 8.4.1.4 HVRP

In this section we give results for the HVRP. As in (Subramanian et al., 2012), we consider the cases where the fleet is limited (HVRP) as well as the cases where the fleet is unlimited (Fleet Size and Mix – FSM). More specifically, we tackle the following variants:

HVRPFV, limited fleet, with fixed and variable costs.

HVRPV, limited fleet, with variable costs but without fixed costs,

FSMFV, unlimited fleet, with fixed and variable costs.

FSMF, unlimited fleet, with fixed costs but without variable costs,

FSMV, unlimited fleet, with variable costs but without fixed costs”

Table 8.7. Summary for HVRPFV

Instance	BKS	MathAnt	Gap (%)
HVRPFV_c050_13mix	3185.09	<b>3185.09</b>	<b>0.00</b>
HVRPFV_c050_14mix	10107.53	10109.17	0.02
HVRPFV_c050_15mix	3065.29	3067.72	0.08
HVRPFV_c050_16mix	3265.41	3270.54	0.16
HVRPFV_c075_17mix	2076.96	2083.50	0.31
HVRPFV_c075_18mix	3743.58	3748.58	0.13
HVRPFV_c100_19mix	10420.34	10563.59	1.37
HVRPFV_c100_20mix	4788.49	4940.84	3.18
Average	5081.59	5120.42	0.66

The results for HVRPFV, HVRPV, FSMFV, FSMF and FSMV problems with solving only SP are given in Table 8.7-8.11 respectively. The average gap values are below 1.0% except HVRPV where the average gap is 1.34%. In fact, the method performs very bad on a particular instance, namely HVRPV\_c100\_19mix. Other than that, the results are satisfactory.

Table 8.8. Summary for HVRPV

Instance	BKS	MathAnt	Gap (%)
HVRPV_c050_13mix	1517.84	<b>1517.84</b>	<b>0.00</b>
HVRPV_c050_14mix	607.53	<b>607.53</b>	<b>0.00</b>
HVRPV_c050_15mix	1015.29	<b>1015.29</b>	<b>0.00</b>
HVRPV_c050_16mix	1144.94	<b>1144.94</b>	<b>0.00</b>
HVRPV_c075_17mix	1061.96	1064.07	0.20
HVRPV_c075_18mix	1823.58	1828.68	0.28
HVRPV_c100_19mix	1117.51	1205.94	7.91
HVRPV_c100_20mix	1534.17	1569.43	2.30
Average	1227.85	1248.10	1.34

Table 8.9. Summary for FSMFV

Instance	BKS	Best (SP)	Gap (%)
FSMFV_c020_03mix	1144.22	<b>1144.22</b>	<b>0.00</b>
FSMFV_c020_04mix	6437.33	<b>6437.33</b>	<b>0.00</b>
FSMFV_c050_13mix	2964.65	<b>2964.65</b>	<b>0.00</b>
FSMFV_c050_14mix	9126.9	9143.40	0.18
FSMFV_c050_15mix	2634.96	<b>2634.96</b>	<b>0.00</b>
FSMFV_c050_16mix	3168.92	<b>3168.92</b>	<b>0.00</b>
FSMFV_c075_17mix	2004.48	<b>2004.48</b>	<b>0.00</b>
FSMFV_c075_18mix	3147.99	<b>3147.99</b>	<b>0.00</b>
FSMFV_c100_19mix	8661.81	8756.58	1.09
FSMFV_c100_20mix	4153.02	4170.56	0.42
Average	4344.43	4357.31	0.17

Table 8.10. Summary for FSMF

Instance	BKS	MathAnt	Gap (%)
FSMF_c020_03mix	961.03	<b>961.03</b>	<b>0.00</b>
FSMF_c020_04mix	6437.33	<b>6437.33</b>	<b>0.00</b>
FSMF_c050_13mix	2406.36	<b>2406.36</b>	<b>0.00</b>
FSMF_c050_14mix	9119.03	9125.20	0.07
FSMF_c050_15mix	2586.37	<b>2586.37</b>	<b>0.00</b>
FSMF_c050_16mix	2720.43	<b>2720.43</b>	<b>0.00</b>
FSMF_c075_17mix	1734.53	1743.70	0.53
FSMF_c075_18mix	2369.65	2372.01	0.10
FSMF_c100_19mix	8661.81	8805.36	1.66
FSMF_c100_20mix	4037.9	4058.69	0.51
Average	4103.44	4121.65	0.29

Table 8.11. Summary for FSMV

Instance	BKS	MathAnt	Gap (%)
FSMV_c020_03mix	623.22	<b>623.22</b>	<b>0.00</b>
FSMV_c020_04mix	387.18	<b>387.18</b>	<b>0.00</b>
FSMV_c050_13mix	1491.86	<b>1491.86</b>	<b>0.00</b>
FSMV_c050_14mix	602.21	603.21	0.17
FSMV_c050_15mix	999.82	999.90	0.01
FSMV_c050_16mix	1131.00	<b>1131.00</b>	<b>0.00</b>
FSMV_c075_17mix	1038.60	1039.46	0.08
FSMV_c075_18mix	1800.80	1803.19	0.13
FSMV_c100_19mix	1105.44	1180.53	6.79
FSMV_c100_20mix	1530.43	1557.90	1.80
Average	1071.06	1081.75	0.90

### 8.4.1.5 VRPTW

All the tests in this section are conducted using the 39 instances in R1, R2, RC1 and RC2 sets of Solomon (1987). Since the optimal solutions can be easily obtained for the clustered instances of C1 and C2 sets, they are omitted. The algorithm is compared with the best known results for real valued distances and optimal results for truncated distance values.

Table 8.3. Comparison of results for type 1 problems for real valued distances

Instance	BKS	Ref*	MathAnt	Gap (%)
R101	1642.87	AMT	<b>1642.88</b>	0.00
R102	1472.62	AMT	1472.82	0.01
R103	1213.62	JM	<b>1213.62</b>	0.00
R104	976.61	JM	<b>976.61</b>	0.00
R105	1360.78	JM	<b>1360.78</b>	0.00
R106	1239.37	YÇ.b	<b>1239.37</b>	0.00
R107	1073.01	YÇ.a	<b>1072.12</b>	-0.08
R108	938.20	YÇ.b	<b>938.20</b>	0.00
R109	1151.84	JM	<b>1151.84</b>	0.00
R110	1072.41	JM	<b>1072.42</b>	0.00
R111	1053.50	JM	<b>1053.50</b>	0.00
R112	953.63	RT	<b>953.63</b>	0.00
R1 Average	1179.04		1178.98	-0.01
RC101	1623.58	RT	<b>1623.59</b>	0.00
RC102	1461.23	JM	<b>1461.23</b>	0.00
RC103	1261.67	S	<b>1261.67</b>	0.00
RC104	1135.48	CLM	<b>1135.52</b>	0.00
RC105	1518.58	JM	<b>1518.58</b>	0.00
RC106	1376.99	YÇ.a	<b>1376.99</b>	0.00
RC107	1211.11	YÇ.b	<b>1211.11</b>	0.00
RC108	1117.53	JM	<b>1117.53</b>	0.00
RC1 Average	1338.27		1338.28	0.00
Total Average	1242.73		1242.70	0.00

\* AMT: (Alvarenga et al., 2007), JM: (Jung and Moon, 2002), YÇ.a: (Yildirim and Çatay, 2012), YÇ.b: (Yıldırım and Çatay, 2014), RT: (Rochat and Taillard, 1995), S: (Shaw, 1998), CLM: (Cordeau et al., 2001)

The results using real distances for type 1 and type 2 problems are given in Table 8.3 and Table 8.4 respectively. Our method performs better on type 1 problems where the average gap values are 0.00% and 0.10% respectively. We reached all of the best-known values for type 1 problems with R102 being the only exception with a gap of only 0.01%. We also obtained a new best value for the instance R107. For type 2 problems the performance of the method is relatively worse. Nevertheless, we obtained new best results also for R210 and RC208 problems. The detailed routes for the new best results are given in Appendix.

Table 8.4. Comparison of results for type 2 problems for real valued distances

Instance	BKS	Ref*	MathAnt	Gap (%)
R201	1147.80	BV	<b>1147.80</b>	0.00
R202	1034.35	JM	<b>1034.35</b>	0.00
R203	874.87	JM	<b>874.87</b>	0.00
R204	735.80	BV	<b>735.80</b>	0.00
R205	954.16	ORH	955.82	0.17
R206	879.89	JM	880.12	0.03
R207	797.99	BV	802.24	0.53
R208	705.45	JM	707.99	0.36
R209	859.39	JM	<b>859.39</b>	0.00
R210	910.70	JM	<b>906.19</b>	-0.50
R211	755.82	BV	757.05	0.16
R2 Average	877.84		878.33	0.07
RC201	1265.56	JM	<b>1265.56</b>	0.00
RC202	1095.64	JM	<b>1095.64</b>	0.00
RC203	926.89	BV	937.45	1.14
RC204	786.38	JM	786.70	0.04
RC205	1157.55	JM	<b>1157.55</b>	0.00
RC206	1054.61	JM	<b>1054.61</b>	0.00
RC207	966.08	JM	<b>966.08</b>	0.00
RC208	779.31	JM	<b>778.93</b>	-0.05
RC1 Average	1004.00		1005.31	0.14
Total Average	930.96		931.80	0.10

\* BV: (Brandão de Oliveira and Vasconcelos, 2010), JM: (Jung and Moon, 2002), ORH: (Ombuki et al., 2006), RT: (Rochat and Taillard, 1995)

Table 8.5. Comparison of results for type 1 problems for truncated distances

Instance	Optimal	MathAnt	Gap (%)
R101	1637.7	<b>1637.7</b>	0.00
R102	1466.6	<b>1466.6</b>	0.00
R103	1208.7	<b>1208.7</b>	0.00
R104	971.5	<b>971.5</b>	0.00
R105	1355.3	<b>1355.3</b>	0.00
R106	1234.6	<b>1234.6</b>	0.00
R107	1064.6	<b>1064.6</b>	0.00
R108	932.1	<b>932.1</b>	0.00
R109	1146.9	<b>1146.9</b>	0.00
R110	1068.0	<b>1068.0</b>	0.00
R111	1048.7	<b>1048.7</b>	0.00
R112	948.6	<b>948.6</b>	0.00
R1 Average	1173.61	1173.61	0.00
RC101	1619.8	<b>1619.8</b>	0.00
RC102	1457.4	<b>1457.4</b>	0.00
RC103	1258.0	<b>1258.0</b>	0.00
RC104	1132.3	<b>1132.3</b>	0.00
RC105	1513.7	<b>1513.7</b>	0.00
RC106	1372.7	1373.5	0.06
RC107	1207.8	<b>1207.8</b>	0.00
RC108	1114.2	<b>1114.2</b>	0.00
RC1 Average	1334.49	1334.59	0.01
Total Average	1237.96	1238.00	0.00

Table 8.6. Comparison of results for type 2 problems for truncated distances

Instance	Optimal	MathAnt	Gap (%)
R201	1143.2	<b>1143.2</b>	0.00
R202	1029.6	<b>1029.6</b>	0.00
R203	870.8	<b>870.8</b>	0.00
R204	731.3	<b>731.3</b>	0.00
R205	949.8	951.3	0.16
R206	875.9	879.3	0.39
R207	794.0	803.4	1.18
R208	701.2	703.6	0.34
R209	854.8	<b>854.8</b>	0.00
R210	900.5	902.6	0.23
R211	746.7	752.3	0.75
R1 Average	872.53	874.75	0.28
RC201	1261.8	<b>1261.8</b>	0.00
RC202	1092.3	<b>1092.3</b>	0.00
RC203	923.7	929.5	0.63
RC204	783.5	787.4	0.50
RC205	1154.0	<b>1154.0</b>	0.00
RC206	1051.1	<b>1051.1</b>	0.00
RC207	962.9	963.3	0.04
RC208	776.1	<b>776.1</b>	0.00
RC1 Average	1000.68	1001.94	0.15
Total Average	926.48	928.30	0.22

The results using truncated distances for type 1 and type 2 problems are given in Table 8.5 and Table 8.6, respectively. The same pattern in instances with real valued distances is also observed for the instances with truncated distances. The average gaps for type 1 and type 2 problems are 0.00% and 0.22% respectively. For type 1 problems, we find the optimal solutions for all the problems, but RC106. For type 2 problems, we only succeed to find 9 of the optimal solutions.

#### 8.4.2 Comparison of SP and SC formulations

In Table 8.13, we report the average performance of SC and SP formulations on the VRP variants. We observe similar performance except OVRP where the average gap of SP and SC formulations are 0.83% and 0.41% respectively.

Table 8.13. The average results comparison of SP and SC formulations on VRP variants

VRP Type	SP	SC
	Average Gap (%)	Average Gap (%)
CVRP	0.69	0.66
OVRP	0.83	0.41
MDVRP	2.35	2.27
HVRPFV	0.82	0.80
HVRPV	2.41	2.34
FSMFV	0.25	0.25
FSMF	0.41	0.39
FSMV	1.35	1.31
VRPTW (real)	0.35	0.34
VRPTW (truncated)	0.12	0.11

## 8.5 Conclusion and Future Research

Extending Yıldırım and Çatay (2014), we tested the algorithm on different variants. We show that the proposed method performs well on a range of routing problems. The summary of the results is given in Table 8.14. Best known results are obtained in 59% of all instances and six best known results in the literature are improved. We observe relatively poor performance on HVRP and MDVRP variants in spite of the modifications tailored for the problem types.

Table 8.14. Summary of the performance of proposed method on different VRP variants

Problem type	Number of instances	Average gap with the best-known solutions (%)	Number of instances where the best known solution is obtained	Number of improved solutions
CVRP	14	0.25	8	-
OVRP	14	0.08	5	3
MDVRP	7	1.97	-	-
HVRPFV	8	0.63	-	-
HVRPV	8	1.72	3	-
FSMFV	10	0.17	7	-
FSMF	10	0.29	5	-
FSMV	10	0.90	4	-
VRPTW (real)	39	0.00	28	3
VRPTW (truncated)	39	0.11	29	-

We also tested the claim of Russell and Chiang (2006) and compared using set covering with using set partitioning, but we could not observe a significant gain using set covering rather than set partitioning. Moreover, we utilized parallelism for only reducing the computational time. However, a parallel implementation by devising multiple ant colonies evolving on different processors may lead to improved performance with respect to the solution quality as well as processor load balance.

## 8.6 Appendix. New best solutions

### *VRPTW*

Route	Distance
R107 (1072.118)	
1 0-72-0-28-76-79-78-29-24-68-80-12-0	82.100
2 0-53-0-26-39-23-67-55-4-25-54-0	127.042
3 0-62-0-52-7-62-11-63-90-32-66-20-51-50-0	114.944
4 0-71-0-2-57-43-15-41-22-75-56-74-72-73-21-0	103.079
5 0-159-0-53-40-58-0	24.359
6 0-33-0-48-47-36-64-49-19-82-18-89-0	126.000
7 0-49-0-60-83-45-46-8-84-5-17-61-85-93-0	113.470
8 0-71-0-94-96-92-59-99-6-87-13-0	62.747
9 0-19-0-95-97-42-14-44-38-86-16-91-100-37-98-0	105.742
10 0-90-0-27-69-30-88-31-10-70-1-0	86.165
11 0-63-0-33-81-65-71-9-35-34-3-77-0	126.471
R210 (906.187)	
1 0-95-92-42-15-23-67-39-75-72-73-21-40-53-0	125.795
2 0-6-94-96-99-59-87-97-13-58-0	57.818
3 0-18-83-45-61-16-86-44-38-14-43-57-2-41-22-74-56-4-55-25-54-26-0	198.200
4 0-28-12-76-3-79-29-78-81-9-20-66-32-90-63-10-70-31-0	150.767
5 0-52-7-82-48-47-36-19-88-62-11-64-49-46-8-84-17-85-98-37-100-91-93-5-60-89-0	219.950
6 0-27-69-1-30-51-33-71-65-35-34-24-80-68-77-50-0	153.657
RC208 (778.926)	
1 0-94-92-95-67-62-50-34-31-29-27-26-28-30-32-33-76-89-63-85-51-84-56-91-80-0	198.990
2 0-61-42-44-39-38-36-35-37-40-43-41-72-71-93-96-54-81-0	133.001
3 0-69-98-88-2-6-7-79-73-78-12-14-47-17-16-15-13-9-11-10-53-60-8-46-4-45-5-3-1-70-100-55-68-0	227.168
4 0-90-65-82-99-52-83-64-49-19-18-48-21-23-25-77-58-75-97-59-87-74-86-57-24-22-20-66-0	219.767



## OVRP

Route	Distance
CMT08 (641.553)	
1 0-13-94-95-97-87-2-57-15-43-42-14-44-38	88.268
2 0-53-58-40-21-73-72-74-22-41-75-56-23	57.123
3 0-89-18-83-60-5-99-59-92-98-37-100-91-16-86	65.099
4 0-6-96-93-85-61-84-17-45-8-46-36-49-64	98.449
5 0-28-76-77-3-79-78-34-35-71-66-65	76.914
6 0-27-69-1-50-33-81-9-51-20-30-70-10-62	80.560
7 0-26-12-80-68-29-24-54-4-55-25-39-67	92.894
8 0-52-31-88-7-82-48-47-19-11-63-90-32	82.246
CMT10 (869.800)	
1 0-152-58-137-2-115-178-144-57-15-43	46.222
2 0-180-198-110-155-4-139-187-39-56-186-23	50.974
3 0-111-50-102-157-33-185-78-34-164-135-35-136-65	62.935
4 0-166-83-199-114-8-174-46-124-168-47-36-143-49	68.318
5 0-105-26-149-195-179-54-130-165-55-25-170-67	56.202
6 0-53-40-21-73-171-74-72-197-75-133-22-41-145	50.509
7 0-156-147-60-118-5-84-173-113-17-45-125	51.228
8 0-27-176-1-122-51-81-120-9-103-161-71	57.898
9 0-112-183-6-96-104-99-59-93-85-61	31.261
10 0-94-95-97-87-172-42-142-14-192-119-44-140-38	59.145
11 0-154-138-12-109-177-80-150-68-134-163-24	39.199
12 0-146-52-153-106-194-7-182-88-148-62-159	39.253
13 0-167-127-190-31-10-189-108-90-126-63-181	46.939
14 0-132-69-162-101-70-30-32-131-160-128-20-188-66	60.569
15 0-28-184-76-196-116-77-3-158-79-129-169-121-29	43.132
16 0-89-18-82-48-123-19-107-175-11-64	60.559
17 0-13-117-92-151-37-98-100-193-91-191-141-16-86	45.456
CMT13 (835.898)	
1 0-120	7.071
2 0-107-67-69-70-71-74-72-75-78-77-76-73	76.693
3 0-2-1-3-4-5-6-7-9-10-11-15-14	71.582
4 0-109-21-20-23-26-28-31-34-36-35-32-29	106.217
5 0-104-103-68-79-80-56-58-55-53-52-54-57	111.563
6 0-40-43-45-59-65-61-62-64-60-63-66	135.274
7 0-88-82-111-86-87-92-89-91-90-114-18-118-108	30.117
8 0-119-81-112-85-84-117-113-83	30.704
9 0-105-106-102-101-99-100-116	20.850
10 0-95-96-93-94-97-115-110-98	30.261
11 0-13-8-12-17-16-19-25-22-24-27-30-33	105.085
12 0-37-38-39-42-41-44-46-49-47-48-50-51	110.481

## 8.7 References

- Alvarenga, G. B., G. R. Mateus and G. de Tomi (2007). "A genetic and set partitioning two-phase approach for the vehicle routing problem with time windows." *Computers & Operations Research* **34**(6): 1561-1584.
- Brandão de Oliveira, H. and G. Vasconcelos (2010). "A hybrid search method for the vehicle routing problem with time windows." *Annals of Operations Research* **180**(1): 125-144.

- Christofides, N., A. Mingozzi and P. Toth (1979). The vehicle routing problem. Combinatorial optimization. N. Chrisofides, A. Mingozzi, P. Toth and C. Sandi. Chichester, UK, Wiley: 315–338.
- Cordeau, J.-F., M. Gendreau and G. Laporte (1997). "A tabu search heuristic for periodic and multi-depot vehicle routing problems." Networks **30**(2): 105-119.
- Cordeau, J. F., G. Laporte and A. Mercier (2001). "A unified tabu search heuristic for vehicle routing problems with time windows." Journal of the Operational Research Society **52**(8): 928-936.
- Jung, S. and B. R. Moon (2002). "A Hybrid Genetic Algorithm For The Vehicle Routing Problem With Time Windows." 1309-1316.
- Ombuki, B., B. J. Ross and F. Hanshar (2006). "Multi-Objective Genetic Algorithms for Vehicle Routing Problem with Time Windows." Applied Intelligence **24**(1): 17-30.
- Rochat, Y. and É. D. Taillard (1995). "Probabilistic diversification and intensification in local search for vehicle routing." Journal of Heuristics **1**(1): 147-167.
- Shaw, P. (1998). Using constraint programming and local search methods to solve vehicle routing problems. M. Maher and J. F. Puget. Berlin, Springer: 417-431.
- Solomon, M. M. (1987). "Algorithms for the Vehicle Routing and Scheduling Problems with Time Window Constraints." Operations Research **35**(2): 254-265.
- Subramanian, A., P. H. V. Penna, E. Uchoa and L. S. Ochi (2012). "A hybrid algorithm for the Heterogeneous Fleet Vehicle Routing Problem." European Journal of Operational Research **221**(2): 285-295.
- Yildirim, U. M. and B. Çatay (2012). "A time-based pheromone approach for the ant system." Optimization Letters **6**(6): 1081-1099.
- Yıldırım, U. M. and B. Çatay (2014). A Parallel Matheuristic for Solving the Vehicle Routing Problems. Computer-based Modelling and Optimization in Transportation. J. F. Sousa and R. Rossi, Springer International Publishing. **262**: 477-489.

## 9 CONCLUSION

In this thesis we developed efficient methods for solving path finding and vehicle routing problems on road transportation, with a particular emphasis on environmental objectives. In this context, we first concentrated on the environmental aspect of the well-known shortest/fastest path problem and introduced the Greenest Path Problem (GPP) which aims at finding the least greenhouse gas (GHG) generating path on a time-dependent road network subject to varying traffic conditions. With the recent environmental concerns and new government regulations, this problem has the potential to arise in many applications. Then, we addressed Vehicle Routing Problems (VRPs) using metaheuristic and matheuristic approaches based on Ant Colony Optimization (ACO) and extended our work to the case with the environmental concerns. We referred to this problem as the Green VRP (GVRP) where the objective is to minimize the emissions rather than financial costs.

In the first part, we discussed the peculiar characteristics of the GPP that makes it a complicated optimization problem. We reviewed the current solution methods in the literature and observed why they would fail to solve the GPP on real road networks. Then, we proposed a new heuristic, namely the Greenest Path Algorithm (GPA), which was proven to be both fast and effective. Using upper bounds on the objective function, the method is able to obtain good quality solutions with little computational effort. Visualizing the solutions, we illustrated how the shortest and the greenest path differed under congestion. We also extended the GPP by modelling the case where the travel speed was considered as a decision variable.

In spite of the increasing interest in the time-dependent path finding and routing literature, the literature lacks a systematic methodology that mimics the characteristics of real road networks to generate realistic instances. So, we proposed a time-dependent and network-consistent speed generation scheme to fill this gap. The proposed method was successfully used to create synthetic data for our experimental studies.

In the second part, we first introduced a the Time-based Ant System (TbAS), a new ACO approach for solving the VRP with Time Windows and showed that it is capable of obtaining efficient routes when the distance is minimized. We also tested TbAS on the GVRP to show how taking the time dependency into account can significantly improve the solution quality. Note that, any route obtained by a time-independent method can cause significant burden in practice such as delayed deliveries

or overtime cost of the drivers. We further proposed a parallel matheuristic for solving the VRPs using an agent based parallelization. Testing it on VRPTW and extending to solve other VRP variants, we obtained promising results. In fact, for some problems types, the algorithm was shown to be the best performing algorithm in the literature.

We think that the proposed GPA can easily be applied to problems that have different objectives. One can find the most pedestrian friendly path by assigning weights to the structure of the pavement, the pollution on the path or the sight, and solve the problem with these weights as costs.

It would be interesting to test the proposed methods in practice. Keeping in mind that these algorithms are developed from a single user point of view, multiple users with the solutions of these algorithms on hand could create additional congestion on the roads. One possible, but utopic for sure, approach could be to guide the drivers to the paths with a centralized decision support/intelligent transport management system and making sure that the paths are strictly followed. Nevertheless, with the increasing number of self-driven cars, the approach may be tested on a small pilot area in the near future.

Other practical issues that have not been considered in this study are the road gradient, vehicle loads the, acceleration and deceleration of the vehicles, particularly during frequent starts and stops which significantly increase the GHG. We are planning to develop new methodologies that account for these cases as well as the case where the speed is a decision variable in our future studies.

## REFERENCES

- Ahn, B.-H. and J.-Y. Shin (1991). "Vehicle-routing with Time Windows and Time-varying Congestion." J Oper Res Soc **42**(5): 393-400.
- Ahuja, R. K., J. B. Orlin, S. Pallottino and M. G. Scutella (2002). "Minimum Time and Minimum Cost-Path Problems in Street Networks with Periodic Traffic Lights." Transportation Science **36**(3): 326-336.
- Ahuja, R. K., J. B. Orlin, S. Pallottino and M. G. Scutella (2003). "Dynamic Shortest Paths Minimizing Travel Times and Costs." Networks **41**(4): 197-205.
- Alvarenga, G. B., G. R. Mateus and G. de Tomi (2007). "A genetic and set partitioning two-phase approach for the vehicle routing problem with time windows." Computers & Operations Research **34**(6): 1561-1584.
- Archetti, C., M. G. Speranza and M. W. P. Savelsbergh (2008). "An Optimization-Based Heuristic for the Split Delivery Vehicle Routing Problem." Transportation Science **42**(1): 22-31.
- Badeau, P., F. Guertin, M. Gendreau, J.-Y. Potvin and E. Taillard (1997). "A parallel tabu search heuristic for the vehicle routing problem with time windows." Transportation Research Part C: Emerging Technologies **5**(2): 109-122.
- Bauer, R. and D. Delling (2008). SHARC: Fast and Robust Unidirectional Routing. 2008 Proceedings of the Tenth Workshop on Algorithm Engineering and Experiments (ALENEX): 13-26.
- Bektaş, T. and G. Laporte (2011). "The Pollution-Routing Problem." Transportation Research Part B: Methodological **45**(8): 1232-1250.
- Bellman, R. (1958). "On a Routing Problem." Quarterly of Applied Mathematics **16**: 87-90.
- Bertazzi, L. and M. G. Speranza (2012). Matheuristics for Inventory Routing Problems. Hybrid Algorithms for Service, Computing and Manufacturing Systems: Routing and Scheduling Solutions. J. R. Montoya-Torres, A. A. Juan, L. H. Huatuco, J. Faulin and G. L. Rodriguez-Verjan. Hershey, PA, IGI Global: 1-14.
- Blosch, J. (2008). Effective Java. Boston, Addison-Wesley.
- Boschetti, M., V. Maniezzo, M. Roffilli and A. Bolufé Röhrler (2009). Matheuristics: Optimization, Simulation and Control. Hybrid Metaheuristics. M. Blesa, C. Blum, L. Gaspero et al., Springer Berlin Heidelberg. **5818**: 171-177.
- Brandão de Oliveira, H. and G. Vasconcelos (2010). "A hybrid search method for the vehicle routing problem with time windows." Annals of Operations Research **180**(1): 125-144.
- Bräysy, O. and M. Gendreau (2005a). "Vehicle Routing Problem with Time Windows, Part I: Route Construction and Local Search Algorithms." Transportation Science **39**(1): 104-118.
- Bräysy, O. and M. Gendreau (2005b). "Vehicle Routing Problem with Time Windows, Part II: Metaheuristics." Transportation Science **39**(1): 119-139.
- Bullnheimer, B., R. F. Hartl and C. Strauss (1999a). "An improved Ant System algorithm for the Vehicle Routing Problem." Annals of Operations Research **89**: 319-328.
- Bullnheimer, B., R. F. Hartl and C. Strauss (1999b). "A new rank-based version of the ant system: A computational study." Central European Journal of Operations Research **7**(1): 25-38.
- Cai, X., T. Kloks and C. K. Wong (1997). "Time-varying shortest path problems with constraints." Networks **29**(3): 141-150.

- Chabini, I. (1998). "Discrete dynamic shortest path problems in transportation applications: Complexity and algorithms with optimal run time." Transportation Research Record **1645**: 170-175.
- Chen, M., R. A. Chowdhury, V. Ramachandran, D. L. Roche and L. Tong (2007). Priority Queues and Dijkstra's Algorithm. Technical Report, University of Texas, Austin.
- Christofides, N., A. Mingozzi and P. Toth (1979). The vehicle routing problem. Combinatorial optimization. N. Christofides, A. Mingozzi, P. Toth and C. Sandi. Chichester, UK, Wiley: 315-338.
- Cooke, K. L. and E. Halsey (1966). "The shortest route through a network with time-dependent internodal transit times." Journal of Mathematical Analysis and Applications **14**(3): 493-498.
- Cordeau, J.-F., M. Gendreau and G. Laporte (1997). "A tabu search heuristic for periodic and multi-depot vehicle routing problems." Networks **30**(2): 105-119.
- Cordeau, J. F., G. Laporte and A. Mercier (2001). "A unified tabu search heuristic for vehicle routing problems with time windows." Journal of the Operational Research Society **52**(8): 928-936.
- Daganzo, C. F. (2002). "Reversibility of the time-dependent shortest path problem." Transportation Research Part B: Methodological **36**(7): 665-668.
- Dean, B. C. (1999). Continuous-Time Dynamic Shortest Path Algorithms. Technical Report, Massachusetts Institute of Technology, Cambridge.
- Dean, B. C. (2004a). "Algorithms for minimum-cost paths in time-dependent networks with waiting policies." Networks **44**(1): 41-46.
- Dean, B. C. (2004b). Shortest Paths in FIFO Time-Dependent Networks: Theory and Algorithms. Technical Report, Massachusetts Institute of Technology, Cambridge.
- Dehne, F., M. Omran and J.-R. Sack (2012). "Shortest Paths in Time-Dependent FIFO Networks." Algorithmica **62**(1-2): 416-435.
- Dell'Amico, M., M. Iori and D. Pretolani (2008). "Shortest paths in piecewise continuous time-dependent networks." Operations Research Letters **36**(6): 688-691.
- Delling, D. (2008). Time-Dependent SHARC-Routing. Algorithms - ESA 2008. D. Halperin and K. Mehlhorn, Springer Berlin Heidelberg. **5193**: 332-343.
- Delling, D., P. Sanders, D. Schultes and D. Wagner (2009). Engineering Route Planning Algorithms. Algorithmics of Large and Complex Networks. J. Lerner, D. Wagner and K. Zweig, Springer Berlin Heidelberg. **5515**: 117-139.
- Delling, D. and D. Wagner (2007). Landmark-Based Routing in Dynamic Graphs. Experimental Algorithms. C. Demetrescu, Springer Berlin Heidelberg. **4525**: 52-65.
- Delling, D. and D. Wagner (2009). Time-Dependent Route Planning. Robust and Online Large-Scale Optimization. R. Ahuja, R. Möhring and C. Zaroliagis, Springer Berlin Heidelberg. **5868**: 207-230.
- Demir, E., T. Bektaş and G. Laporte (2012). "An adaptive large neighborhood search heuristic for the Pollution-Routing Problem." European Journal of Operational Research **223**(2): 346-359.
- Deo, N. and C.-Y. Pang (1984). "Shortest-path algorithms: Taxonomy and annotation." Networks **14**(2): 275-323.
- Dijkstra, E. W. (1959). "A note on two problems in connexion with graphs." Numerische Mathematik **1**(1): 269-271.
- Disser, Y., M. Müller-Hannemann and M. Schnee (2008). Multi-criteria Shortest Paths in Time-Dependent Train Networks. Experimental Algorithms. C. McGeoch, Springer Berlin Heidelberg. **5038**: 347-361.

- Doerner, K. and V. Schmid (2010). Survey: Matheuristics for Rich Vehicle Routing Problems. Hybrid Metaheuristics. M. Blesa, C. Blum, G. Raidl, A. Roli and M. Sampels, Springer Berlin Heidelberg. **6373**: 206-221.
- Donati, A. V., R. Montemanni, N. Casagrande, A. E. Rizzoli and L. M. Gambardella (2008). "Time dependent vehicle routing problem with a multi ant colony system." European Journal of Operational Research **185**(3): 1174-1191.
- Dorigo, M. (2010). "Ant colony optimization." Scholarpedia **2**(3): 1461-1461.
- Dorigo, M. and L. M. Gambardella (1997). "Ant colony system: a cooperative learning approach to the traveling salesman problem." IEEE Transactions on Evolutionary Computation **1**(1): 53-66.
- Dorigo, M., V. Maniezzo and A. Coloni (1996). "Ant system: optimization by a colony of cooperating agents." IEEE transactions on systems, man, and cybernetics. Part B, Cybernetics : a publication of the IEEE Systems, Man, and Cybernetics Society **26**(1): 29-41.
- Dorigo, M. and T. Stützle (2004). Ant Colony Optimization. London, The MIT Press.
- Dreyfus, S. E. (1969). "An Appraisal of Some Shortest-Path Algorithms." Operations Research **17**(3): 395-412.
- Eglese, R., W. Maden and A. Slater (2006). "A Road Timetable to aid vehicle routing and scheduling." Computers & Operations Research **33**(12): 3508-3519.
- Eglese, R. W. and D. Black (2010). Optimizing the Routing of Vehicles. Green Logistics: Improving the Environmental Sustainability of Logistics. A. McKinnon, S. Cullinane, M. Browne and S. Whiteing. London, Kogan Page: 215-228.
- Ellabib, I., O. A. Basir and P. Calamai (2002). "An Experimental Study of a Simple Ant Colony System for the Vehicle Routing Problem with Time Windows." Ant Algorithms Lecture Notes in Computer Science **2463**: 53-64.
- EMEP/CORINAIR (2007). EMEP/CORINAIR Emission Inventory Guidebook: Group 7 road transport.
- Figliozzi, M. A. (2010). "Vehicle Routing Problem for Emissions Minimization." Transportation Research Record: Journal of the Transportation Research Board **2197**: 1-7.
- Figliozzi, M. A. (2011). "The impacts of congestion on time-definitive urban freight distribution networks CO2 emission levels: Results from a case study in Portland, Oregon." Transportation Research Part C: Emerging Technologies **19**(5): 766-778.
- Fleischmann, B., M. Gietz and S. Gnutzmann (2004). "Time-Varying Travel Times in Vehicle Routing." Transportation Science **38**(2): 160-173.
- Floudas, C. A. and P. M. Pardalos (2009). Encyclopedia of Optimization. New York, Springer.
- Franceschetti, A., D. Honhon, T. Van Woensel, T. Bektaş and G. Laporte (2013). "The time-dependent pollution-routing problem." Transportation Research Part B: Methodological **56**(0): 265-293.
- Gambardella, L. M., É. Taillard and G. Agazzi (1999). MACS-VRPTW: a multiple ant colony system for vehicle routing problems with time windows. D. Corne, M. Dorigo and F. Glover. London, McGraw-Hill: 63-76.
- Garcia-Najera, A. and J. A. Bullinaria (2011). "An improved multi-objective evolutionary algorithm for the vehicle routing problem with time windows." Computers & Operations Research **38**(1): 287-300.
- Gendreau, M. and J.-Y. Potvin (2005). "Metaheuristics in Combinatorial Optimization." Annals of Operations Research **140**(1): 189-213.
- Groër, C., B. Golden and E. Wasil (2010). "A library of local search heuristics for the vehicle routing problem." Mathematical Programming Computation **2**(2): 79-101.

- Groër, C., B. Golden and E. Wasil (2011). "A Parallel Algorithm for the Vehicle Routing Problem." INFORMS J. on Computing **23**(2): 315-330.
- Gulczynski, D., B. Golden and E. Wasil (2011). "The period vehicle routing problem: New heuristics and real-world variants." Transportation Research Part E: Logistics and Transportation Review **47**(5): 648-668.
- Hall, R. (1986). "The fastest path through a network with random time-dependent travel times." Transportation Science **20**(3): 182-186.
- Halpern, J. (1977). "Shortest route with time dependent length of edges and limited delay possibilities in nodes." Zeitschrift für Operations Research **21**(3): 117-124.
- Hart, P. E., N. J. Nilsson and B. Raphael (1968). "A Formal Basis for the Heuristic Determination of Minimum Cost Paths." Systems Science and Cybernetics, IEEE Transactions on **4**(2): 100-107.
- Hashimoto, H., M. Yagiura and T. Ibaraki (2008). "An iterated local search algorithm for the time-dependent vehicle routing problem with time windows." Discrete Optimization **5**(2): 434-456.
- Hickman, J., C. Hassel, R. Joumard, Z. Samaras and S. Sorenson (1999). MEET Methodology for Calculating Transport Emissions and Energy Consumption. Technical Report.
- Hill, A. V. and W. C. Benton (1992). "Modelling Intra-City Time-Dependent Travel Speeds for Vehicle Scheduling Problems." The Journal of the Operational Research Society **43**(4): 343-351.
- Horn, M. E. T. (2000). "Efficient modeling of travel in networks with time-varying link speeds." Networks **36**(2): 80-90.
- Hsu, C.-I., S.-F. Hung and H.-C. Li (2007). "Vehicle routing problem with time-windows for perishable food delivery." Journal of Food Engineering **80**(2): 465-475.
- Hvattum, L. M., I. Norstad, K. Fagerholt and G. Laporte (2013). "Analysis of an exact algorithm for the vessel speed optimization problem." Networks **62**(2): 132-135.
- Ichoua, S., M. Gendreau and J.-Y. Potvin (2003). "Vehicle dispatching with time-dependent travel times." European Journal of Operational Research **144**(2): 379-396.
- Jabali, O., T. Van Woensel and A. G. de Kok (2012). "Analysis of Travel Times and CO2 Emissions in Time-Dependent Vehicle Routing." Production and Operations Management **21**(6): 1060-1074.
- Jung, S. and B. R. Moon (2002). "A Hybrid Genetic Algorithm For The Vehicle Routing Problem With Time Windows." 1309-1316.
- Kanoulas, E., D. Yang, X. Tian and Z. Donghui (2006). Finding Fastest Paths on A Road Network with Speed Patterns. Data Engineering, 2006. ICDE '06. Proceedings of the 22nd International Conference on.
- Kara, İ., B. Kara and M. K. Yetis (2007). Energy Minimizing Vehicle Routing Problem. Combinatorial Optimization and Applications. A. Dress, Y. Xu and B. Zhu, Springer Berlin Heidelberg. **4616**: 62-71.
- Kaufman, D. E. and R. Smith (1990). Minimum travel time paths in dynamic networks with application to intelligent vehicle-highway systems. Ann Arbor, Mich., University of Michigan, Transportation Research Institute.
- Kaufman, D. E. and R. Smith (1993). "Fastest Paths in Time-Dependent Networks for Intelligent Vehicle-Highway Systems Application." Journal of Intelligent Transportation Systems **1**(1): 1-11.
- Kelly, J. P. and J. Xu (1999). "A Set-Partitioning-Based Heuristic for the Vehicle Routing Problem." INFORMS J. on Computing **11**(2): 161-172.
- Kenyon, A. S. and D. P. Morton (2003). "Stochastic vehicle routing with random travel times." Transportation Science **37**(1): 69-82.



- Koskosidis, Y. A., W. B. Powell and M. M. Solomon (1992). "An Optimization-Based Heuristic for Vehicle Routing and Scheduling with Soft Time Window Constraints."
- Kuo, Y. (2010). "Using simulated annealing to minimize fuel consumption for the time-dependent vehicle routing problem." Computers & Industrial Engineering **59**(1): 157-165.
- Kuo, Y., C.-C. Wang and P.-Y. Chuang (2009). "Optimizing goods assignment and the vehicle routing problem with time-dependent travel speeds." Computers & Industrial Engineering **57**(4): 1385-1392.
- Laporte, G., F. Louveaux and H. Mercure (1992). "The vehicle routing problem with stochastic travel times." Transportation Science **26**(3): 161-170.
- Lecluyse, C., K. Sorensen and H. Peremans (2013). "A network-consistent time-dependent travel time layer for routing optimization problems." European Journal of Operational Research **226**(3): 395-413.
- Liberti, L., S. Cafieri and F. Tarissan (2009). Reformulations in Mathematical Programming: A Computational Approach. Foundations of Computational Intelligence Volume 3. A. Abraham, A.-E. Hassanien, P. Siarry and A. Engelbrecht, Springer Berlin Heidelberg. **203**: 153-234.
- Maden, W., R. Eglese and D. Black (2010). "Vehicle Routing and Scheduling with Time Varying Data: A Case Study." Journal of the Operational Research Society **61**(3): 515-522.
- Malandraki, C. and M. S. Daskin (1992). "Time Dependent Vehicle Routing Problems: Formulations, Properties and Heuristic Algorithms." Transportation Science **26**(3): 185-200.
- Maniezzo, V., T. Stützle and S. Voß (2010). Matheuristics: hybridizing metaheuristics and mathematical programming. New York, Springer.
- McKinnon, A. (2007). CO2 Emissions from Freight Transport in the UK. Technical Report, London, UK, Commission for Integrated Transport.
- Mendoza, J. and J. Villegas (2013). "A multi-space sampling heuristic for the vehicle routing problem with stochastic demands." Optimization Letters **7**(7): 1503-1516.
- Miller, E. D., H. S. Mahmassani and A. Ziliaskopoulos (1994). Path search techniques for transportation networks with time-dependent, stochastic arc costs IEEE International Conference on Systems, Man and Cybernetics.
- Muter, İ., Ş. İ. Birbil and G. Şahin (2010). "Combination of Metaheuristic and Exact Algorithms for Solving Set Covering-Type Optimization Problems." INFORMS Journal on Computing **22**(4): 603-619.
- Nannicini, G. (2009). Point-to-Point Shortest Paths on Dynamic Time-Dependent Road Networks. PhD, Ecole Polytechnique, Palaiseau.
- Nannicini, G., D. Dellinger, L. Liberti and D. Schultes (2008). Bidirectional A \* Search for Time-Dependent Fast Paths. Experimental Algorithms. C. McGeoch, Springer Berlin Heidelberg. **5038**: 334-346.
- Nielsen, L., D. Pretolani and K. Andersen (2009). Bicriterion Shortest Paths in Stochastic Time-Dependent Networks. Multiobjective Programming and Goal Programming. V. Barichard, M. Ehrgott, X. Gandibleux and V. T'Kindt, Springer Berlin Heidelberg. **618**: 57-67.
- Norstad, I., K. Fagerholt and G. Laporte (2011). "Tramp ship routing and scheduling with speed optimization." Transportation Research Part C: Emerging Technologies **19**(5): 853-865.
- Ombuki, B., B. J. Ross and F. Hanshar (2006). "Multi-Objective Genetic Algorithms for Vehicle Routing Problem with Time Windows." Applied Intelligence **24**(1): 17-30.

- Orda, A. and R. Rom (1990). "Shortest-path and minimum-delay algorithms in networks with time-dependent edge-length." J. ACM **37**(3): 607-625.
- Orda, A. and R. Rom (1991). "Minimum weight paths in time-dependent networks." Networks **21**(3): 295-319.
- Pallottino, S. and M. G. Scutellà (1998). Shortest Path Algorithms In Transportation Models: Classical and Innovative Aspects. Equilibrium and Advanced Transportation Modelling. P. Marcotte and S. Nguyen, Springer US: 245-281.
- Pallottino, S. and M. G. Scutellà (2003). "A new algorithm for reoptimizing shortest paths when the arc costs change." Operations Research Letters **31**(2): 149-160.
- Pardalos, P. M. and M. G. C. Resende (2002). Handbook of Applied Optimization. New York, New York, USA, Oxford University Press.
- Park, Y.-B. (2000). "A solution of the bicriteria vehicle scheduling problems with time and area-dependent travel speeds." Computers and Industrial Engineering **38**(1): 173-187.
- Park, Y.-B. and S.-H. Song (1997). "Vehicle scheduling problems with time-varying speed." Computers & Industrial Engineering **33**(3-4): 853-856.
- Pillac, V., C. Guéret and A. L. Medaglia (2013). "A parallel matheuristic for the technician routing and scheduling problem." Optimization Letters **7**(7): 1525-1535.
- Pirkwieser, S. and G. Raidl (2009). Multiple Variable Neighborhood Search Enriched with ILP Techniques for the Periodic Vehicle Routing Problem with Time Windows. Hybrid Metaheuristics. M. Blesa, C. Blum, L. Gaspero et al., Springer Berlin Heidelberg. **5818**: 45-59.
- Pisinger, D. and S. Ropke (2007). "A general heuristic for vehicle routing problems." Computers & Operations Research **34**(8): 2403-2435.
- Potvin, J., Y. Xu and I. Benyahia (2006). "Vehicle routing and scheduling with dynamic travel times." Computers & Operations Research **33**: 1129-1137.
- Qian, J. and R. Eglese (2014). "Finding least fuel emission paths in a network with time-varying speeds." Networks **63**(1): 96-106.
- Rochat, Y. and É. D. Taillard (1995). "Probabilistic diversification and intensification in local search for vehicle routing." Journal of Heuristics **1**(1): 147-167.
- Rodríguez-Martín, I. and J. Salazar-González (2011). The Multi-Commodity One-to-One Pickup-and-Delivery Traveling Salesman Problem: A Matheuristic. Network Optimization. J. Pahl, T. Reiners and S. Voß, Springer Berlin Heidelberg. **6701**: 401-405.
- Rousseau, L.-M., M. Gendreau and G. Pesant (2002). "Using Constraint-Based Operators to Solve the Vehicle Routing Problem with Time Windows." Journal of Heuristics **8**(1): 43-58.
- Russell, R. A. and W.-C. Chiang (2006). "Scatter search for the vehicle routing problem with time windows." European Journal of Operational Research **169**(2): 606-622.
- Sanders, P., C. Vetter, D. Delling and G. V. Batz (2009). Time-Dependent Contraction Hierarchies. 2009 Proceedings of the Eleventh Workshop on Algorithm Engineering and Experiments (ALENEX): 97-105.
- Savelsbergh, M. W. P. (1992). "The Vehicle Routing Problem with Time Windows: Minimizing Route Duration." ORSA Journal on Computing **4**(2): 146-154.
- Sbihi, A. and R. Eglese (2010). "Combinatorial optimization and Green Logistics." Annals of Operations Research **175**(1): 159-175.
- Shaw, P. (1998). Using constraint programming and local search methods to solve vehicle routing problems. M. Maher and J. F. Puget. Berlin, Springer: 417-431.

- Sherali, H. D., A. G. Hobeika and S. Kangwalklai (2003). "Time-Dependent, Label-Constrained Shortest Path Problems with Applications." Transportation Science **37**(3): 278-293.
- Soler, D., J. Albiach and E. Martínez (2009). "A way to optimally solve a time-dependent Vehicle Routing Problem with Time Windows." Operations Research Letters **37**(1): 37-42.
- Solomon, M. M. (1987). "Algorithms for the Vehicle Routing and Scheduling Problems with Time Window Constraints." Operations Research **35**(2): 254-265.
- Stützle, T. and H. Hoos (1997). MAX-MIN Ant System and local search for the traveling salesman problem, IEEE.
- Subramanian, A., P. H. V. Penna, E. Uchoa and L. S. Ochi (2012). "A hybrid algorithm for the Heterogeneous Fleet Vehicle Routing Problem." European Journal of Operational Research **221**(2): 285-295.
- Subramanian, A., E. Uchoa and L. S. Ochi (2013). "A hybrid algorithm for a class of vehicle routing problems." Computers & Operations Research **40**(10): 2519-2531.
- Sung, K., M. G. H. Bell, M. Seong and S. Park (2000). "Shortest paths in a network with time-dependent flow speeds." European Journal of Operational Research **121**(1): 32-39.
- Tan, K. C., L. H. Lee, Q. L. Zhu and K. Ou (2001). "Heuristic methods for vehicle routing problem with time windows." Artificial Intelligence in Engineering **15**(3): 281-295.
- Tian, Y., K. C. K. Lee and W.-c. Lee (2009). Monitoring Minimum Cost Paths on Road Networks With Time-Dependent Edge Availabilities. Proceedings of the 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems GIS '09.
- Toth, P. and D. Vigo (2002). The vehicle routing problem. Philadelphia, PA, SIAM Monographs on Discrete Mathematics and Applications.
- Villegas, J. G., C. Prins, C. Prodhon, A. L. Medaglia and N. Velasco (2013). "A matheuristic for the truck and trailer routing problem." European Journal of Operational Research **230**(2): 231-244.
- Wellman, M. P. (1990). "Fundamental concepts of qualitative probabilistic networks." Artificial Intelligence **44**(3): 257-303.
- Wellman, M. P., M. Ford and K. Larson (1995). Path planning under time-dependent uncertainty. Proceedings of the Eleventh conference on Uncertainty in artificial intelligence. Montreal, Quebec, Canada, Morgan Kaufmann Publishers Inc.: 532-539.
- Wen, L., B. Çatay and R. Eglese (2014). "Finding a minimum cost path between a pair of nodes in a time-varying road network with a congestion charge." European Journal of Operational Research **236**(3): 915-923.
- Wikipedia. (2014). "Google driverless car." Retrieved June 01, 2014, from [http://en.wikipedia.org/wiki/Google\\_driverless\\_car](http://en.wikipedia.org/wiki/Google_driverless_car).
- Woensel, T., L. Kerbache, H. Peremans and N. Vandaele (2007). "A Queueing Framework for Routing Problems with Time-dependent Travel Times." Journal of Mathematical Modelling and Algorithms **6**(1): 151-173.
- Wolfler Calvo, R. and N. Touati-Moungla (2011). A Matheuristic for the Dial-a-Ride Problem. Network Optimization. J. Pahl, T. Reiners and S. Voß, Springer Berlin Heidelberg. **6701**: 450-463.
- Yıldırım, U. M. (2014). "Speed Generator." from <http://myweb.sabanciuniv.edu/mahiryldrm/speedGenerator>.
- Yildirim, U. M. and B. Çatay (2012). "A time-based pheromone approach for the ant system." Optimization Letters **6**(6): 1081-1099.

- Yıldırım, U. M. and B. Çatay (2014). A Parallel Matheuristic for Solving the Vehicle Routing Problems. Computer-based Modelling and Optimization in Transportation. J. F. Sousa and R. Rossi, Springer International Publishing. **262**: 477-489.
- Zheng, Y. and B. Liu (2006). "Fuzzy vehicle routing model with credibility measure and its hybrid intelligent algorithm." Applied Mathematics and Computation **176**(2): 673-683.
- Ziliaskopoulos, A. (1994). Optimum path algorithms on multidimensional networks: Analysis, design, implementation and computational experience, Technical Report, University of Texas, Austin.
- Ziliaskopoulos, A. K. and H. S. Mahmassani (1993). "Time-Dependent , Shortest-Path Algorithm for Real-Time Intelligent Vehicle Uighway System Applications." Transportation Research Record **1408**: 94-100.
- Zufferey, N. (2011). "Optimization by ant algorithms: possible roles for an individual ant." Optimization Letters **6**(5): 963-973.