On the Online Dial-Ride-Problem

By

Athanasios Lois

University of Thessaly
Department of Mechanical Engineer
In partial fulfillment of the requirements for the degree of
Doctor of Philosophy
2010

Title of dissertation: **On the Online Dial-A-Ride-Problem**

Athanasios Lois, Phd Canditate 2010

Directed by: **Professor Athanasios Ziliaskopoulos**

Department of Mechanical Engineering

In recent years, there has been considerable progress in the area of logistics. The main forces behind this evolution are computer based technologies such as geographic information systems (GIS), global positioning systems (GPS) and intelligent highway information systems. All these technologies facilitate better decisions on routing, assignment, distribution as well as many other logistics-related problems. Most of the research in the last twenty years was focused on the optimization of networks either static or deterministic. A network can be defined as static when all its parameters are known *a priori*, it does not change over time, and is deterministic when all parameters are deterministically known. Models that anticipate real-time information and can account for parameter uncertainty are called either dynamic or stochastic. These types of models are receiving increasingly more interest from many researches in the area. This is the case because these models allow for more accurate representation of reality, by continually updating their data with new input from the real world. These models are certainly more difficult to solve. However, given the evolution of information technology, we could safely claim that they are the best choice for logistics problem solving. Researchers tackle these problems, by constructing algorithms, which are typically problem specific and extensions of the corresponding static algorithms. The traditional design and analysis of static algorithms assumes that an algorithm has complete knowledge of

the entire input. However, this assumption is often unrealistic in practical applications. Many of the practical algorithmic applications, however, are online by nature. For these applications the input is only partially available because some data becomes available in the course of operation. Often, this information is unavailable at the time routing decision must be made. An online algorithm must generate output without knowledge of the entire input.

The problem of concern in this dissertation is the well known “Dial-a-Ride” or “Pickup and Delivery” Problem and its variation called “Online Dial-a-Ride”. This problem can be considered as a special case of the general “Vehicle Routing Problem”. The “Dial-a-Ride Problem” is an important and difficult problem encountered in several contexts. It is becoming increasingly important as the pressure for optimal utilization of the available capacity of an existing fleet of vehicles intensifies, due to energy and environmental impact concerns. An extension to the traditional “Dial-a-Ride problem” (DARP) is the “Online Dial-a-Ride problem” which deals with the continuous flow of trip demands during course of operation. What is required is to construct online algorithms, capable of adapting to the continuous stream of incoming information. We can imagine the “Online Dial-a-Ride problem” as a “Dial-a-Ride problem” that has to be solved by an online algorithm each time a new trip demand occurs dynamically. We will examine this problem in more detail later in this document. For this dissertation, six different dial-a-ride algorithms have been implemented. The first four are static dial-a-ride algorithms while the remaining two can be considered as pure online algorithms. Finally, a methodology concerning the profitability of a proposed Demand Responsive Transportation System has been developed, as well as the underlying algorithm.

On the Online Dial-Ride-Problem

By

Athanasios Lois

University of Thessaly
Department of Mechanical Engineer
In partial fulfillment of the requirements for the degree of
Doctor of Philosophy
2010

Advisory Committee:
Professor Athanasios Ziliaskopoulos
Professor George Lymberopoulos
Professor Leandros Tasioulas
Associate Professor Spyros Karamanos
Associate Professor Serafim Kapros
Assistant Professor Dimitris Pandelis
Lecturer George Kozanidis

DEDICATION

to my children

ACKNOWLEDGEMENTS

Ἀνατίθημι τοῦτο το σύγγραμμα τοῖς μεν ἐμοῖς καθηγηταῖς οἳ παρεῖχον μοι το καιρόν ταύτη

τῆς μελέτης περί ἑνός διαφερόντως μεγίστου μαθήματος τοις δε οἰκείοις ἕνεκα του ὀάκνω

συμπαριστασθαί μοι ώστε σπουδάζειν και περαίνειν αὐτό. Το μεν πειρᾶσθαι μανθάνειν καί

γνῶναι, δῆλον ὅτι συνεβάλετο μέγα μοι οὐ μόνον ἐις το στοχάζεσθαι ὰλλά και ἐις τ

ἐπιτηδεύειν την ἐμή ὀρετήν και σύνεσιν.

I would like to express my profound appreciation to my advisor, Professor Athanasios Ziliaskopoulos for initiating this research, providing me freedom to follow my own interests, and giving me guidance and encouragement throughout the entire research. I am very grateful for his availability whenever I needed advice. The discussions were always helpful and inspiring. It has been a real pleasure and privilege working with him. I would like to thank my advisory committee members as well for their valuable comments and suggestions on this research.

Table of Contents
List of Tables
List of Figures
List of Abbreviations

List of Tables

List of Figures

List of Abbreviations

General Terms

VRP=Vehicle Routing Problem

PDPTW=Pickup and Delivery problem with Time Windows

DARP = Dial-a-Ride problem

TSP=Travelling Salesman Problem

DRT = Demand Responsive Transportation System

New terms specific to this disseration

InsertionH = Static insertion algorithm for the Dial-a-Ride problem

DP Exact = Static exact algorithm for the Dial-a-Ride problem

RegretH = Static Regret algorithm for the Dial-a-Ride problem

VLSN = Static Very large Scale Neighborhood algorithm for the Dial-a-Ride problem

OLDARP = General Online Dial-a-Ride problem

OR-DARP = Online Regret algorithm for the Dial-a-Ride problem

OP-DARP = Online Regret with Probabilities algorithm for the Dial-a-Ride problem

Chapter 1 : Introduction

In recent years, there has been considerable progress in the area of logistics. The main forces behind this evolution are computer based technologies such as Geographic Information Systems (GIS), Global Positioning Systems (GPS) and Intelligent Highway Information Systems. All these technologies facilitate better decisions on routing, assignment, distribution and many other logistics-related problems. Most of the research in the last twenty years has been focused on optimizing networks that where static and deterministic. A network can be defined as static when all information concerning its state is known *a priori*, it does not change over time, and is deterministic when all parameters are deterministically known. Models that anticipate real-time information and can account for uncertainty are called dynamic or stochastic networks. These types of models are attracting increasingly more interest from many researches in the area. Because the advantage of these models is that they allow for a more accurate representation of reality, by continually updating their data with new input from the real world. These models are more difficult to solve. However, given the evolution of Information Technology, we can safely claim that they are the best choice for a new era in logistics science. Attempts to deal with these problems typically focus on constructing problem specific algorithms. The traditional design and analysis of algorithms assumes that an algorithm, that generates a certain output, has complete knowledge of the entire input in advance. However, this assumption is often unrealistic in practical applications. Many of the algorithmic problems that arise in practice are online. For these problems the input is only partially available because some relevant input data becomes available in the course of operation. Thus it is not available at the time decisions need to be made. An online algorithm must generate output without knowledge of the entire input.

1.1 Motivation

Algorithmic development for online problems is an active field of research for important practical applications in logistics and transportation. Advances in Information and Communication Technologies have substantially increased the role of online information, which can be valuable if processed by specially designed algorithms. For all classic problems in logistics science there is a demand for extending static approaches to online ones. The majority of these problems belong to what is known as NP-Hard problems, which are usually intractable. Till now there is little progress in the field of mathematical formulation and algorithm construction for online problems. For some of these problems the state of the art is limited to heuristics. Given, however, the nature of heuristic algorithms, it is difficult to define precisely how close their solutions might be to the optimal solution. For the DARP, which is a special case of the general "Vehicle Routing Problem", there is no exact algorithm – at least not one running in acceptable CPU time – for reasonably sized problems. Instead, there is a lot of work on heuristic approaches that produce solutions close to the ones considered optimal. The whole research area, though, is fairly open to new approaches.

The "Online Dial-a-Ride" problem (OLDARP) is more interesting both theoretically and practically, because the state of the system changes on a real-time basis, due to cancellations and new requests. This situation is common in real world Transportation Systems.

1.2 Dissertation Objectives

This research is focused on Static and Online Algorithms for the DARP and OLDARP problems. Emphasis is given on applications for DRT systems.

The specific objectives are:

1. To study the "Online Dial-a-Ride" problem (ODARP), though an extensive literature study. New trends in research are considered, in order to get deep understanding of

the practical application needs that guide the research on this field. Issues like the closeness of the proposed solutions to the optimal solutions are considered.

2. To identify the problem properties that can be useful in the overall understanding of the online problem. This is a challenging objective as we aim to identify specific problem properties concerning the problem solution space; this in turn will lead to understanding what the difficulties concerning the solution are. It will also indicate possible solution paths.
3. To construct the appropriate set of static algorithms. These can be used as sub-modules to the online algorithms.
4. To construct the appropriate set of online algorithms. With special emphasis on adapting optimization procedures that are continuously improved as the time progresses, taking advantage of the time that the system remains idle.
5. To evaluate these algorithms in terms of solution optimality, speed of execution and correctness of the results.
6. To build a working framework, that utilizes online algorithms within a practically implementable solution for actual applications.

1.3 Dial-a-Ride Services as part of the Demand Responsive Transportation (DRT) Systems

A DRT system is a public transit service that can provide shared-ride door-to-door service with flexible routes and schedules. DRT was initially designed for service to the general public. It provides a higher quality of service (e.g. negligible access time, wait at home and no transfers). However it increases operating cost due to lower vehicle productivity (e.g., passenger trips per vehicle hour) than conventional bus services. Due to the required subsidy, DRT service to the general public is usually limited to suburban areas or time periods with low demand densities. It is also used as a feeder to line-haul systems, in those situations where they operate more cost-effectively. In some cities, DRT is limited to use by a special group of persons with mobility difficulties (handicapped or elderly persons), who are not able to access other public transportation services or airport transport services where there is one major origin or destination. A DRT system is made up of a control center and a fleet of small vehicles (usually < 20 seats) compared to conventional buses where higher capacity vehicles are used on a fixed route and schedule. The vehicles, operating with flexible routes and schedules, respond to requests for transportation as they are received by the control center. Each customer will provide information about the locations of his/her origin and destination, the desired time of pickup or delivery, and the number of riders. The dispatcher in the control center will combine the customer information with information regarding vehicle positions and their tentative routes, to plan the new routes for vehicle. This may be done using manual or automated dispatching techniques. The passengers are provided the expected pickup time. Unlike taxi service, DRT services allow ridesharing and thus reduce the cost per passenger. Early demand-responsive systems used manual dispatching techniques. With technological advances in computer hardware and software, automatic computer dispatching algorithms are available to many current services. Furthermore, Advanced Vehicle Location (AVL), Global Position Systems (GPS), Geographical Information Systems (GIS) and other similar systems are making the real-time dispatching more feasible. In the existing systems or algorithms, two types of service requests are considered: advance requests and immediate requests.

Advance requests usually refer to those received at least one day before the service is provided, so that routes and schedules can be planned at the start of the day of service. Service provided to handicapped persons often requires advance requests. If all the requests are provided in advance (and assuming all other factors, such as traffic conditions, are predictable), then the determination of the routes and schedules is a based on a static Dial-A-Ride Problem (DARP).

Immediate requests are those asking for service as soon as possible with a designated desired pickup or delivery time. In practice, a reasonable online service provided to the general public would allow the service requests, probably with a minimum time in advance (e.g. 10 minutes) for the efficient route and schedule planning. This kind of service is considered in this study. Except when serving only previous-day advance requests, the routing and scheduling of a DRT service is a dynamic problem, in which decisions for requests coming throughout the operating period are made in real time.

DRT services may be classified as many-to-many, many-to-few and many-to-one, depending on the demand patterns and the targeted service quality. Many-to-many means passengers can differ in their origins and destinations. If all the passengers are picked up or delivered at the same location (e.g. a shopping center or an airport), the service is many to-one. One example of many-to-one service is the feeder service in a local area, in which all passengers are collected to feed a metro station. Many-to-few lies between those two extreme conditions, where there are a few common origins and/or destinations.

In the USA and Canada DRT services are used extensively as part of the American with Disabilities Act (ADA). In many cities in the USA, a DRT taxi-based system exists, mainly for airport service and other intercity trips offering a cheap alternative to a regular taxi. While it is not exactly a door-to-door DRT application, a popular transit option in Turkey, called “dolmus”, is a good application example of a new public transportation mode that is more flexible than the public bus and still more affordable price compared to taxi/car option.

Many EU countries such as Italy, Finland, Sweden, Netherlands and Belgium have employed DRT systems, mainly funded by the European Commission. Experience from Europe shows that strategically it is more straightforward to implement DRT systems in regulated environments, as there is less conflict with other public transportation modes. In 2000, the UK Government pledged in its Ten Year Plan for transport to remove or (at least) relax constraints on the development of flexible bus services and to promote a greater role for community-based services (DETR, 2000a). In addition, research commissioned by the UK Department of the Environment Transport and the Regions (DETR) argues that flexible public transport services - provided by local authorities and bus operators in partnerships with employers, stores and leisure centers - would help to break down social exclusion (DETR, 2000b). Similar initiatives have been reported in Ireland (ADM, 1999). More recently, in 2001, the UK Rural White Paper proposals for the extension of Bus Service Operators Grant (BSOG) –formerly Fuel Duty Rebate (FDR) – to community transport were adopted. Finally, the recent successes of local authorities in winning substantial funding under the Rural and Urban Bus Challenge programs for the implementation of Demand Responsive Transport Services confirms this new interest in flexible forms of transport.

1.4 Contributions till now

The contribution of the dissertation to the static and online Dial-a-Ride problem may be defined as follows:

- Construction of a set of algorithms concerning the static Dial-a-Ride problem.
    - The InsertionH algorithm and its variations
    - The DP Exact algorithm
    - The Very Large Search Neighborhood VLSN hybrid algorithm
    - The RegretH heuristic algorithm
- Construction of a set of algorithms concerning the online Dial-a-Ride problem.
    - The online regret OR-DARP algorithm.

    - The online regret with probabilities OP-DARP algorithm.

- The construction of the "Convergence Process" in order to understand the profitability of a DRT system
- The comparison of results (online solutions vs a posteriori solutions)
- Testing of those online algorithms via simulation programs.

1.5 Organization of the Dissertation

Chapter 2 focuses on the review of the literature regarding the existing static and online DARP problems with emphasis on the online approach.

Chapter 3 focuses on the mathematical formulation of the static DARP. Some results concerning the MILP formulation are presented to indicate the NP nature of the problem.

Chapter 4 focuses on static algorithms for the DARP. In the framework of this dissertation we have developed four static algorithms: The first one is a simple Insertion heuristic; the second one is DP implementation of an exact approach; the third is a hybrid algorithm called VLSN (Very Large Scale Neighborhood) search algorithm; the last one is a heuristic that uses the regret technique as optimization procedure. This procedure helps us to find out which of these algorithms can be utilized for the online implementations.

Chapter 5 focuses on the online algorithms. Based on the findings in chapter 4 we developed two online algorithms. The first one is the online version of the static regret algorithm, while the second one is an online probabilistic regret algorithm.

Chapter 6 focuses on an actual application and on the Convergence Methodology regarding the profitability of a DRT system under consideration.

Chapter 7 presents the conclusions of this dissertation and discusses further research directions.

Chapter 2 : Critical Evaluation of the Literature

In this section we present a review of the existing literature on online problems. First we should define the term online. The terms online and dynamic are not sufficiently clear due to different usage by various authors.

Some of the them, use the term dynamic to refer to problems where the number of demands, network features such as travel times – even the fleet size – are not known a priori but varies with time. Others use the term to refer to problems where only network features vary. The term online is used invariably to refer only to problems where demand varies through the time. In the next sections when we refer to studies for dynamic DARP problems we mean problems where the number of trip demands is not known a priori while network and fleet features remain unchanged.

Initially, dynamic problems for the general Dynamic Vehicle Routing problem will be discussed as well as the Dynamic Vehicle Allocation problem. Then we will focus on the general DARP problem. Finally we will discuss the DARP problem in much greater detail, with emphasis on variations such as single vehicle, multi vehicle, as well as online operation.

2.1 The Dynamic Vehicle Routing Problem

This category of problems includes many vehicle routing problems and various approaches have been presented mostly for specific problem applications. Bagchi and Nag [112] proposed an expert system approach to dynamic vehicle routing. The Gavish system [45] makes use of a combination of optimization-based heuristics. The Un-capacitated Dynamic Travelling Repairman problem is a classic dynamic assignment problem. A Special version of this problem is the capacitated version. In this problem customers must be clustered into tours which satisfy vehicle capacity constraints. Bertsimas and van Ryzin [14] examine the case where a specific region is served by a homogeneous fleet of *m* vehicles, operating out of *m* depots whose locations need not be distinct, where each vehicle is restricted to visiting at most *q* customers before returning to the depot. Fleischmann et al. [110] proposed a framework that adapts to online traffic information for a specific application of the dynamic routing problem, consisting of three strategies (planning by assignment rules, planning by assignment algorithm, planning by insertion algorithm) that can cover a number of applications.

2.2 The Dynamic Vehicle Allocation problem

The vehicle allocation problem arises when a common carrier must manage a fleet of vehicles to maximize profits over a planning horizon. Demand materializes when shippers call the carrier requesting a vehicle to be available in a specific location, on specific day and time, to carry something to a given destination. This request is usually referred to as load. Each load must be served by a single vehicle and the vehicle is dedicated to that load. No vehicle can be shared by more than one loads. Time is split into intervals, and on each day, the operator must either assign each vehicle to a requested demand, or move it empty to another region to pickup a requested or expected demand, or to keep it inactive until the next day.

Vehicle allocation problems have many applications in everyday life. Most transport and logistics applications are based on vehicle allocation approaches. That may partly explain the popularity of this subject in the general operations research area.

In the case of rail transportation we can refer to Feeney [37], Leddon and Wrathall [72], Gorenstein et al. [51] and Herren [52],[53] where we can see early examples of efforts to optimize fleets of rail cars. Misra [79] formulates the problem as a linear program, while

White [107] presents a dynamic transshipment network over a finite planning horizon. White and Bomberault [108] use the dynamic structure of the network to develop a specialized algorithm. Mendiratta [75] and Mendiratta and Turnquist [76] present inventory models for managing empty cars, taking into account the decentralized nature of the decision-making process. Jordan and Turnquist [63] present the first stochastic model of the empty car management problem. Ratcliffe et al. [95] use a simulation model of empty freight cars. Glickman and Sherali [49] address the problem of pooled fleets of empty cars, recognizing that railroads share fleets of cars. Shan [97] uses a dynamic, multi-commodity network flow model to handle multiple car types, using resource directive decomposition to solve the resulting network. Chih [21] extends this model to handle multiple railroad vehicles. An array of myopic models can be referred to also, such as the transportation formulations given in Turnquist [104] and Turnquist and Markowicz [105]. Other researchers are going beyond the problem of managing empty cars. Haghani [54] presents a combined model for train makeup and empty car repositioning, representing one of the earliest efforts to address the flows of both loaded and empty cars. Chih et al. [22] consider the problem of managing locomotives. Smith and Sheffi [99] present a locomotive distribution model that accounts for uncertainty in the need for locomotives, using a simple recourse strategy to handle the effects of uncertainty. Kraay et al. [70] address the problem of dynamically managing the movement of trains over a rail line, which requires optimizing the use of sidings to allow for train passing. For trucking truckload applications, the primary problem addressed in the literature is the dynamic vehicle allocation problem for managing large fleets of trucks. Powell et al. [91] present a nonlinear dynamic network model for accounting for uncertainties in forecasted demands. Powell [85] refines this model to allow for stochastic vehicle inventories. This model extends [91] to allow for the possibility that trucks that are not needed will remain in inventory, and extends the model in [63] by tracking both loaded and empty movements. Powell [86] further extends this model by providing the most realistic model of future vehicle trajectories. Whereas prior research makes very restrictive assumptions about how a truck may be used under uncertainty, Powell [86] provides a very general model of truck dispatching under uncertainty. A more formal mathematical model is provided in Powell [87], where several different models are formulated for the same problem. Frantzeskakis and Powell [46] introduce a new heuristic for solving multistage dynamic networks with random arc capacities, motivated by truckload motor carriers. These results have been further extended in Powell and Cheung [88]. A real application of these results is given in Powell et al. [90]. For sea transportation, applications of dynamic models come into effect when planning the movement of ocean vessels and the optimization of fleets of containers over a global logistics network. Dantzig and Fulkerson [29] provide one of the earliest applications of optimization over a dynamic network to minimize the number of tankers required to meet a given schedule. Other efforts to optimize the movement of vessels include Brown et al. [18], Psaraftis et al. [93], Fisher and Rosenwein [38], and Perakis and Papadakis [83]. Ermoliev et al. [35] and Florez [39] consider the optimization of containers. Crainic et al. [26] considers the dynamic management of containers over land within a region near a port. Finally, in air transportations the problem is realized as the assignment of aircraft to routes, crew scheduling, pricing and booking problems, and the dynamic management of aircraft between airports (known commonly as air traffic control problems). Dantzig and Ferguson [28] use the fleet assignment problem as an early example of linear programming under uncertainty. Magnanti and Simpson [216] describe a series of dynamic network models with side constraints to handle fleet assignment. The common approach used in this area is based on set partitioning problems to choose from among the best set of possible crew schedules. A different approach is suggested by Ball and Roberts [4] which uses a matching algorithm to sequentially generate possible crew schedules. Another interesting problem is the flow management problem in air traffic control. The main concept for these problems is the dynamic management of aircraft moving between airports. An important relevant reference is Andreatta and Romanin-Jacur [1]. Odoni [237] and Mulvey and Zenios [80] give a nonlinear, dynamic network model for routing aircraft. Bielli et al. [15] also formulate the flow control problem as a dynamic network.

2.3 The Static Dial and Ride Problem

The Dial-a-Ride problem class has received considerable attention in the literature. Several versions of the DARP have been studied over the past 30 years. While none is identical to another, it helps comparatively evaluating them to assess their suitability for online applications. Since the definition of the DARP varies from one author to the other, we only consider these cases where time window constraints are imposed. In the absence of time windows, the problem only addresses precedence relationships on pick-up and drop-off, which do not fully capture the true nature of the DARP.

The first publications in this area date back to the late 1960s and early 1970s (Wilson and Weissberg [126]; Wilson and Colvin [127]). Surveys on solution methods can be found in Cordeau and Laporte (2003b, 2007) [122], [128]

Next, some of the major developments in the Dial-a-Ride Problem are discussed:

(I) Exact algorithms

The single-vehicle problem has been first studied by Psaraftis [129], who developed an exact dynamic programming algorithm for the case where time windows are imposed on each pick-up and drop-off. User dissatisfaction is measured and controlled through a Maximum Position Shift (MPS) constraint, limiting the difference between the position of a user in the list of requests and its actual position in the vehicle route. The algorithm defines the MPS constant to control the maximum ride time for each user. Only very small instances can be handled through this algorithm. It uses the methodology of states to represent each execution step as the current state and makes use of three arrays called

FEASBL, V, NEXT - for the full description of feasible states and for the next step - each one consisting of $(2N+1)3^N$ storage locations, where N is the number of requests. The algorithm runs recursively and the time is bounded to $(2N+1)^2 3^N$ processing steps. A dynamic version of the problem has also been defined by the same author. In the dynamic version when a new request comes up, the algorithm resets the origin time and re-runs the static version of the problem. Because of that limit $(2N+1)3^N$, only small problem instances can be solved even with current computing power. But at least this algorithm provides an exact solution. Waiting times are not considered in this study.

Desrosiers [124] reformulated the single vehicle DARP as an integer dynamic program. The formulation includes time windows as well as vehicle capacity and precedence constrains. It is solved exactly by dynamic programming. According to the authors, the effectiveness of the proposed algorithm is largely due to the use of efficient elimination criteria for states which are infeasible, because of the additional constrains on the route. The solution method can be described as follows: Problem is solved using forward dynamic programming. The vehicle is initially located at the departure node 0. At the first iteration, problem states created of routes visiting a single node chosen from the origins. At each subsequent iteration *k* (2<= k <=2n), the states are constructed form the states of the previous iteration and are made up of routes visiting one additional node chosen from the origins and destinations. To implement states elimination procedure, the authors use a set of 9 criteria concerning the feasibility of each state. Criteria can be categorized as follows:

1. Criteria concerning the visiting sequence
2. Criteria concerning vehicle capacity
3. Criteria concerning time constraints (like earliest and latest visiting times)
4. Criteria concerning the case where several clients are at the same location.

A double labeling system (cost and time) is also used to mark each feasible route. Only a subset of all possible labels (or routes) is stored. According to the authors memory usage for storing feasible states is not an issue, because of their small number. Optimal solutions were obtained for 40 demands (n = 40). Dynamic features were not included at all in this study. Authors recommended the use of the proposed algorithm as sub-algorithm in the multi vehicle DARP, However, this algorithm approach was not used extensively by subsequent studies.

Dumas et al [119] presented an exact algorithm which solves the pickup and delivery problem when transporting goods. The algorithm uses a column generation specific scheme with a constrained shortest path as a sub-problem. The presented algorithm solves only some instances of the (Pickup and Deliver problem with Time Windows) "PDPTW" and is not designed to handle large-scale problems. The first step of the proposed methodology is to produce admissible paths, by using shortest path with constrains. After that, the master problem is solved by a column generation algorithm and a branch and bound exploration tree. Given a set o columns or admissible paths, the restricted master problem is solved using the simplex algorithm. The authors use a new branching strategy which is applied directly to the requests, more precisely to the pickup sequence. This branching strategy has the advantage of eliminating the possible number of branches by half. Also in order to reduce execution time several strategies have been used. One is to judiciously impose a cut on the number of vehicles far before the optimal linear solution of the master problem is reached. Another is to reduce the shortest path problems to a small part of the network comprising 30% of the best networks arcs. The problem size ranged between 19 to 55 requests (20 to 112 nodes, including depot nodes). The proposed solution gives good results only for small instances. There is also too much customization in order to speed up the execution of the algorithm. The dynamic nature is not considered in this methodology and maybe hard to implement an addition to this methodology.

Kozanidis - Ziliaskopoulos [125] presented a DP exact algorithm for the 1-vehicle and 2-vehicle DARP problem. The algorithm is based on a data implementation which is an extension of the one introduced by Psaraftis [114]. Numerical results depict clearly the exponential behavior of that algorithmic implementation. The largest problem size that the single vehicle algorithm can solve is limited to nine (N=9) demands. The largest problem size that the 2-vehicle algorithm can solve is limited to seven (N=7) demands. Of course the limited number of requests that can be solved restrict the use of this algorithmic implementation to cases where the problem size is very small. What is important in this study is that there is an approach for the optimal solution for more than one vehicle.

Kikuchi [151] develops a balanced LP transportation problem for the multi vehicle case, minimizing empty vehicle travel as well as idle times, and thus fleet size. In a preprocessing step the service area is divided into zones, the time horizon into time periods. Every request is classified according to an origin and a destination zone as well as a departure and an arrival time period. An example with four zones is presented.

Cordeau [130] proposes a branch and cut algorithm for the static DARP. The algorithm is based on a 3-index mixed-integer problem formulation. New valid inequalities as well as previously developed ones for the PDP and the VRP are employed. The largest instance solved to optimality comprises 36 requests.

Ropke-Cordeau-Laporte [131] introduced two new formulations for the PDPTW and the closely related dial-a-ride problem (DARP) in which a limit is imposed on the elapsed time between the pickup and the delivery of a request. Several families of valid inequalities are introduced to strengthen these two formulations. These inequalities are used within branch-and-cut algorithms. They have been tested on several instance sets for both the PDPTW and the DARP. Instances with up to eight vehicles and 96 requests (194 nodes) have been solved to optimality.

(ii) Heuristic algorithms

Fisher – Jaikumar [117] presented a variant of the classic vehicle routing problem, in which a fleet of vehicles delivers products stored at a central depot to satisfy customer orders. The proposed algorithm is a generalized assignment problem with constraints and an objective function, that approximates the cost of the traveling salesman problem tours required for each vehicle to serve its assigned customers. Once this assignment has been made, a complete solution is obtained by applying any traveling salesman problem heuristic or optimizing algorithm, to obtain the delivery sequence for the customers assigned to each vehicle. Firstly, because the problem feasibility constraints are included in the generalized assignment problem, the heuristic will always find a feasible solution if one exists. Second, the generalized assignment problem is solved while considering the impact of a customer assignment to a vehicle on every other possible assignment considering vehicle capacity constraints. This avoids a problem faced by sequential assignment or limited adjustment heuristics, that can "paint themselves into a corner" by unknowingly making initial assignments, which lead to very expensive subsequent assignments in order to maintain feasibility. The authors’ opinion is that this method can easily be adapted to accommodate a number of important problem complexities, including multiple depots, multiple time periods, the option of not delivering to a customer at a penalty, constraints on the time duration of a vehicle route, and multiple capacity constraints (e.g., weight and volume).

The assignment algorithm is based on a special method that makes clusters of customers based on seeds – produced by a special process – and then a TSP algorithm finds the best optimized route. This algorithm has been tested with a maximum of 10 vehicles and 100 requests. Time windows and service times are considered in this algorithm, while the clustering method is based on a heuristic.

Sexton and Bodin [117] proposed a heuristic algorithm for the “Multi Vehicle Static Dial-and-Ride Problem”. Proposed algorithm is as follows: First, partition the set of customers into M vehicle clusters. Second, solve the single vehicle for each vehicle cluster. Third: move customers from one vehicle to another while attempting to reduce total customer inconvenience. Fourth: for each of the resultant vehicle clusters solve the resultant Single Vehicle Route Schedule. The algorithm handles two types of inconvenience: “Excess Ride” – defined as the actual ride time minus the direct ride time and “Time and Delivery” – defined as the desired delivery time minus the actual delivery time - Time Deviation. To solve the “Single Vehicle Static Ride and Dial problem”, the author uses an approach based on two different modules. The first module is responsible to identify the optimal scheduling sequence for a specific route. The second module uses a heuristic approach to identify a sequence of possible routes. For the scheduling module the authors use Benders decomposition and they show that the scheduling problem is the dual of network maximum flow problem, the structure of which allows exact solutions to be found quickly using a one pass algorithm. For the routing model the authors present a heuristic algorithm for finding an initial solution and after that they present a second heuristic algorithm for the improvement of the initial route. The algorithm finds the initial route based on a set from the current last task on a route to all other tasks feasible as immediate successors. The second heuristic algorithm makes the route improvement motivated by a langrangean relaxation and searches for feasible positions inside the same route. For every feasible route the best scheduling is computed by the scheduling module. Results are reported on several data sets from Baltimore and Gaithersburgh involving between 7 and 20 users.

Their proposed solution methodology is a combination of a heuristic part (route identification) and an exact part (scheduling optimization), that can be used by a superior heuristic algorithm for multi vehicle dial and ride problem. There is no guarantee the optimum solution. Also the online nature of the problem isn't mentioned at all.

Jaw et al. [132] have analyzed a version of the problem where windows are imposed on the pick-up time of pickup requests and on the drop-off time of delivery requests. The algorithm emphasizes flexibility and user's convenience. The basic concept of the proposed algorithm is simple. The algorithm tests the feasible assignment for the current request (pickup or delivery) for each available vehicle. For each assignment a COST function is calculated. From all available feasible insertions the one with the smaller COST is selected. After that, there is another procedure where the best position inside the scheduled route should be found because Active Pickup and Active Deliver Time should be calculated. Clearly this is the most difficult part of the optimization procedure. However, the same author has shown that the problem to find the "optimal" insertion times for Actual Pickup and Actual Delivery time is equivalent to minimizing a single – variable convex function, this variable being the amount by which the current time schedule of the schedule-block in question should be shifted. Furthermore, the disutility is formulated by two factors. DU_{di}, which is disutility due to deviation from the most desired time, DU_{ri} which is the disutility due to excess ride time. Those factors also include quadratic terms in order to model situations where the disutility is non-linear. The total cost is the result of a cost factor which includes the additional active vehicle time, plus the vehicle slack time, plus a factor that describes the utilization of the vehicles. This algorithm makes use of 8 constants and tries through that to find a balance. The alteration of those constants is manual and based mainly on experience. The quality of the solution cannot be estimated precisely by an automated procedure. Besides that, in the COST formulation (parts concerning cost) the author is not so sure whether his approach describes the best solution (see quadratic factors). Computational results are provided on artificial instances involving 250 users and on a real-life dataset with 2617 users and 28 vehicles.

Ioachim [133] presented an optimization based mini-clustering algorithm. It uses column generation to obtain mini-clusters and an enhanced initialization procedure to decrease processing times. The operation model presented by this approach can be described by two phases as follows:

Phase1: Starts with the transportation requests as input, create lists of neighboring requests using proximity features, generate a network and finally solve the restricted multi vehicle PDPTW by using column generation.

Phase 2: Using vehicle itineraries and the generated mini-clusters generate a network and solve a multivehicle TSP with time windows problem by column generation.

They also presented some comparison results between the proposed approach and a parallel insertion mini clustering approach. Results show that their approach outperformed the parallel insertion mini clustering approach in terms of the internal travelling time by an average 9.7% for problems up to 250 trip demands. For large scale problems consisting of more than 2500 requests, the proposed method obtained a significant 5.9% improvement in terms of the total travelling time.

Toth and Vigo [120] presented a heuristics for a real life transportation problem in Bologna, Italy. Users specify requests with a time window on their origin and destination. A limit proportional to the direct distance is imposed on the time spent by a user in the vehicle. Transportation is supplied by a fleet of capacitated minibuses and by the occasional use of taxis. The objective is to minimize the total cost of service. The problem is solved by a heuristic consisting of assigning requests to routes by means of a parallel insertion procedure. This procedure consists of an initialization procedure and enroute trip insertion procedure. The initialization procedure makes the choice of the initial route by using only a fraction of requests and some kind of score which is produced by various user defined factors. After that, an insertion procedure follows, using a special cost matrix created for that purpose. Optionally a tabu search based heuristic - by using intraroute and interroute exchanges - is executed to further improve the total solution. Computational results presented by the authors demonstrated that, the final result is considerably improved compared with the insertion algorithm. CPU time is however considerably greater (27secs compared to 3706 seconds). Results were reported on instances involving 300 trips. Also there is no reference for the integration of any dynamic feature concerning the demands such as real time demands.

Cordeau [134] tried to give a solution to that problem, by introducing a tabu search heuristic. In this algorithm users specify transportation requests between origins and destinations. They may provide a time window on their desired departure or arrival time. Transportation is supplied by a fleet of vehicles based at a common depot. The aim is to design a set of least cost vehicle routes capable of accommodating all requests. Side constraints related to vehicle capacity, route duration and the maximum ride time of any user. Solution methodology is based mainly on tabu search technique. According to that technique the search for feasible solutions can be done through infeasible solutions and a continuous diversification mechanism is put in place in order to reduce the likelihood of being trapped in a local optimum. To avoid cycling, solutions possessing some attributes of recently visited solutions are declared forbidden, or tabu, for a number of iterations, unless they produce a new incumbent.

Solutions are evaluated by using a cost function defined as $f(s) = c(s) + \alpha q(s) + \beta d(s) + \gamma w(s) + \tau t(s)$ where: $c(s)$ defined as the cost function produced by the arc values cost, $q(s)$ defined as the total violation of load, $d(s)$ defined as the total violation of duration, $w(s)$ defined as the violation of total time window, $t(s)$ defined as the violation of ride time constrains. α,β,γ,τ are self-adjusting positive parameters. By dynamically adjusting the values of the four parameters during the search, this relaxation mechanism facilitates the exploration of the solution space and is particularly useful for tightly constrained instances. A procedure for neighborhood evaluation is also proposed. This adjusts the visit times of the vertices on the routes so as to minimize route duration and ride times. The tabu search algorithm is considered as one of the most optimized algorithms in the DARP problems area.

Diana-Dessouky [135] presented a parallel regret insertion heuristic to solve a dial-a-ride problem with time windows. A new route initialization procedure is implemented, that keeps into account both the spatial and the temporal aspects of the problem. A regret insertion is then performed to serve the remaining requests. The operating scenario is representative of a large-scale dial-a-ride program in Los Angeles County. The proposed algorithm was tested on data sets of 500 and 1000 requests built from data of paratransit service in this area. The proposed parallel regret insertion can be described as follows: The basic idea is to find for each un routed request its best insertion (i.e. the one that minimizes the related cost, defined as an increment of the value of the objective function) in each itinerary. In this manner, they build an incremental cost matrix in which the rows represent the requests and the columns the routes. If a request has no feasible insertion in a route, the corresponding incremental cost is set to an arbitrarily large value. After that, we compute for each request its regret, given by the sum of the differences between all the elements of the corresponding row and the minimum one. The request with the largest regret will be inserted in the previously computed position. These steps are iterated until all the requests are inserted or until all the regret costs are zero. In the latter case, the corresponding requests cannot be inserted in any of the existing routes.

2.4 The Online Dial and Ride Problem

The traditional design and analysis of algorithms assumes that an algorithm, which generates output, has complete knowledge of the entire input. However, this assumption is often unrealistic in practical applications. Many of the algorithmic problems that arise in practice are online. In these problems the input is only partially available because some relevant input becomes available during the course of operation. An online algorithm must generate output without knowledge of the entire input.

In general the online version of the problem works as follows. A static solution is constructed on a basis of the requests known at the start of the planning horizon. When a new request arrives, the algorithm performs a feasibility check, and then it searches for a feasible solution to include the new service request. If the new request is accepted, the algorithm performs post-optimization. Feasibility checks are executed just to initially assess the possibility to accept the new request. Post-Optimization executed later gives the best position and time limits for the new request. Parallel computing techniques are used to speed up the process.

The different solution techniques developed are described in the following paragraphs. Predominantly heuristic methods have been used to solve online versions of the DARP. Psaraftis [129] presented an exact method for the dynamic DARP as an adaptation of the static version for the same problem. Although Psaraftis' version has been limited to very small problem instances due to the combinatorial nature of the problem, it remains one of the few exact methods that help gain insight to the problem.

Madsen et al. [163] have presented an algorithm for a real-life multi-vehicle dynamic DARP consisting of up to 300 daily requests for the transportation of elderly and handicapped people in Copenhagen. The problem had many constraints such as time windows,multi-dimensional capacity restrictions, customer priorities and a heterogeneous vehicle fleet. Many objectives taking into account user satisfaction and service costs were considered through the use of weight parameters. When a new request arrives, it is inserted in a current route using an eficient insertion algorithm based on that of Jaw et al[132].Computational results on real-life instances with up to 300 requests and 24 vehicles haveshown that the algorithm was fast enough to be used in a dynamic context and that it is capable of producing good quality solutions.

Teodorovic and Radivojevic [136] developed a fuzzy logic model for the online Dial-a-Ride problem. They combine fuzzy logic reasoning in the insertion procedure to make the decision about which vehicle will accept the new request and to design the new route and schedule for the vehicle chosen to serve the new request. The model is based in the use of two algorithms. The first algorithm addresses the approximate reasoning concerning the selection of the appropriate vehicle which will serve the trip request. The reasoning process needs the subjective perception of the dispatchers. It is based on a set of nine (9) rules. The second algorithm proposes an approximate reasoning for inserting the destination of the new request. The model was tested for a set of 900 trip demands and for a fleet of 30 vehicles with seemingly reasonable results.

Colorni and Righini [137] proposed an online approach for the DARP problem. Their system assumes that a negotiation with users takes place in order to discourage them from imposing unduly tight time windows. Because the main purpose of the proposed model is the approximation and not the optimization, the use of an iterative algorithm based on local search rather the constructive one is proposed. The approximation algorithm is interfaced with a simulator, to tune all necessary parameters off line. They broke down the dial-a-ride problem into two sub problems: that of clustering customers in a number of subsets equal to the number of vehicles, and that of routing each vehicle through pickup and delivery points of the customers in its subset. The two problems are solved alternatively as in many well-known two phase algorithms.

They have tested three different objectives, namely the maximization of serviced requests, the maximization of the perceived level of service by users, and the minimization of traveled distance. The insertion mechanism alternates between a clustering phase and a routing phase. The routing algorithm applies branch-and-bound to a set of requests whose time windows are not too distant in the future. The clustering phase works with exchanges. The authors report that they have performed experiments with their system (called DARIA) in Crema and Verbania, located in northern Italy, but they do not report any computational results.

Diana et al [159] proposed a probabilistic model that requires only the knowledge of the demand distribution over the service area, and the quality of the service. The quality of service is defined in terms of time windows associated with pickup and delivery nodes. Given a number of *n* trip requests in a service area, the objective is to estimate the number of vehicles needed to serve these requests. Authors propose the exponential distribution as the probability density function of the time intervals between two successive pickup times. The probability density function of the distance between two successive points in a route -served by one vehicle- is defined by using an approximation close to lognormal distribution. In order to benchmark the model, they compare it to a simulation approach that requires knowledge of the complete daily schedule. The requests were scheduled using a parallel regret insertion algorithm (Diana [135]). Computational results proved that the probabilistic model produced better results concerning the minimum number of vehicles required to service all trip requests. However for the largest problem instance, the model gave worse results.

Fu [122] presented an approach on the Dial-a-Ride scheduling problems arising in paratransit service systems that are subject to tight service constraints and time varying stochastic traffic congestion. Fu does not present a new algorithm, but rather extents conventional heuristics to address the new constraints with only marginal increase of computational complexity. All the time windows are assumed to be probabilistic while for the actual time of departure or arrival the probability is set almost in all experiments to 0.9. The algorithmic structure comprises 4 functions called iteratively: Function 1 is initialization, 2 find the vehicle to insert, 3 find the best positions to insert, and 4 perform schedule optimization. The author's opinion is that the probabilistic nature of travel times is important, because in urban environments, travel times are inherently varying and stochastic due to temporal and spatial variation of traffic congestion. The dial-a-ride problem where the real-time information includes vehicle status or traffic congestion conditions or number of requests has not been considered in this study.

Horn [157] provides a software environment for fleet scheduling and dispatching of demand responsive services. The system can handle advance as well as immediate requests. New incoming requests are inserted into existing routes according to least cost insertion. A steepest descent improvement phase is run periodically. Also automated vehicle dispatching procedures, to achieve a good combination of efficient vehicle deployment and customer service, are included. The system was tested in the modeling framework LITRES-2 (Horn [158]), using a 24-hours real life data set of taxi operations with 4282 customer requests.

Attanasio [123] presented a modified version of the static tabu search algorithm presented by Cordeau [134] in order to handle the online nature of the DARP. The online algorithm can be described as follows: A static solution is constructed on the basis of the requests known at the start of the planning horizon. When a new request arrives, the algorithm performs a feasibility check, i.e., it searches for a feasible solution to include the new service request. If the new request is accepted, the algorithm performs post-optimization, i.e., it tries to improve the current solution. Different combinations concerning the number of processors and the types of communication (concerning the knowledge information between the processors) has been presented. One practical problem with this approach is the difficulty of solving the problem in a shorter time interval than the updating interval.

Mitrovic-Minic et. al [138] presented a double horizon heuristic algorithm for the online Dial-a-Ride problem, which solves the problem with a short-term and a long-term goal. It differs from the rolling horizon heuristic introduced by Psaraftis [92] which operates with a dynamically redefined short-term horizon. This algorithm has three other algorithms as sub modules. The first and the second address the routing problem. The first is a constructive heuristic consisting of a cheap insertion procedure as well as a cheap reinsertion procedure. The second is a similarly heuristic, but contains an improvement procedure. The cheapest insertion procedure is applied to new requests accumulated over a certain time period of length *d*. It is then applied to the reinsertion of all scheduled requests whose pickup location has not yet been served. Before insertion these requests are sorted in increasing order of slack time. The improvement procedure is based on tabu search. It is applied after the reinsertion procedure and it runs while new requests are being accumulated. The tabu search procedure is a simplified version of the method introduced by Gendreau et al [150] with neighborhoods defined by means of an ejection chains fast insertion algorithm, the second one is the same as the first one with some marginal parameter improvements. Once a partial vehicle route is defined, the scheduling sub problem consists of determining the departure time of each vehicle from each yet unvisited location of its planned route. The Drive-First (DF) strategy is used for the scheduling algorithm.

Coslovich et al. [139] presented an algorithm which follows a two-phase strategy for the insertion of a new request into an existing route. An off-line phase is first used to create a feasible neighborhood of the current route through a 2-opt solution improvement mechanism. An on-line phase is then used to insert the new request with the objective of minimizing user dissatisfaction. Each time an unexpected customer is accepted, the current route of a vehicle is updated and, accordingly, a new neighborhood must be computed. The main steps of the proposed algorithm can be described as follows:

- Initialization Phase (Off line). In this phase algorithm generates the current route on the basis of requests of the customers registered in advance.
- First Phase UPDATE NEIGHBORHOOD (off-line). In this phase route neighborhoods are determined and within these neighborhoods the best route is selected.

- Second Phase: INSERT UNXEPECTECD CUSTOMERS (on-line). In this the best insertion of the delivery point is determined, taking into consideration information about neighborhoods

Xiang [140] presented a flexible scheduling scheme to dynamically cope with different stochastic events, such as the travelling time fluctuation, new requests, customer absenteeism, vehicle breakdowns, cancellations of requests, traffic jams and so on. A fast heuristic is proposed to re-optimize the schedule when a new event occurs. This heuristic consists of a properly organized local search strategy and uses a secondary objective function to drive the search out of local optima. Four Rolling Time Horizons of 1, 2, 3 hours are used, as well as an infinitely long time horizon.

Beaudry et all[154] presented a study on patients' transportation problem arising in large hospitals. Transportation requests arrive in a pure online way and the methodology for solving the problem should therefore aim at the immediate insertion of new requests into the current vehicle routes. Contrary to standard dial-a-ride problems, the certain problem includes several complicating constraints specific to a hospital context. This study provides a detailed description of the problem and proposes a two-phase heuristic procedure that deals with many options of the problem. A simple insertion scheme used to generate a feasible solution in the first phase. In the second phase this is improved with a tabu search algorithm. In the first phase the insertion algorithm utilizes the concept of spatial and temporal proximity between the new request and all other requests already scheduled but not yet serviced. A list of tentative neighbors is created and used as reference positions for inserting the new request. The second phase is a tabu search algorithm that takes into account intermediate infeasible solutions during the search process in an attempt to find a better solution. The authors provided computational results for three different approaches called P1, P2, and P3 accordingly. For the P1 approach only the insertion phase is executed. For the P2 approach the insertion phase and the tabu search are executed with a variant of the inter-route step. For the P3 approach the insertion phase and the tabu search are executed with inter-route and intra routes steps. As expected, the P3 approach consumed more time than the P1, P2 approaches. In fact, the P3 approach was 16 times more time consuming on average than the p1 approach making it ineffective in rush hours.

Marco Diana[160] presented a study about the importance of information flow temporal attributes for a dynamic DRT system. Authors focused on three characteristics of the information flow. These characteristics were: Percentage of real time requests, interval between call-in and requested pickup time, and the length of computational time. On the contrary they considered the duration of the customer's phone call to the call center and the time needed to transmit the schedule to the drivers. They finally assumed the time needed for the customer to be ready to be picked up to be negligible. They handled demands in batch mode. Demands were grouped and processed by algorithms in time slots of 5 minutes, 1 minute, 10 seconds. Because of this grouping and batch processing, customers have to be called back to be notified of the acceptance or rejection of their request. The authors tested two different algorithms to draw conclusions about the influence on the solution quality. The first algorithm was the dynamic version of the algorithm proposed by Jaw[132] and it is based on the best insertion method. The second algorithm was the dynamic version of an older algorithm proposed by the authors, based on the regret method. In their formulation the quality of the schedule is given by the number of rejected requests and by the value of the objective function z. The formulation of z is expressed by three numbers each one representing a specific quantity. First number is 0.45 and represents total distance, second number is .50 representing the excess ride time and the third number is 0.05 representing the idle time. Their experiments were based on trip requests gathered by the transportation service for elderly and disabled people in Los Angeles County. In order to run specific scenarios authors made some assumptions concerning: the number of requests known in advance, the expected value of time intervals between call-in and requested pickup time, the number of vehicles and the cycle time. By the term cycle time authors define a specific time period where the underlying algorithm processes trips requests without interruption. Results showed that:

1. When the percentage of requests known in advance is below 40% there is significant improvement to the solution
2. The expected value of time intervals between call-in and requested pickup time was a very significant factor
3. The algorithm that was used was a significant factor
4. The number of vehicles wasn't a significant factor. In experiments where the fleet size was 30% more it wasn't sufficient to satisfy all demands.

The proposed methodology is not effectively applicable to real situations, because they handle trip requests in specific cycle times where nothing can interrupt the processing. In real applications where trip requests come continuously and sometimes in burst mode the need for fast response is critical. The "productivity" of the underlying algorithm - in terms of real profit- in comparison to rejected trips is not considered, although it is a critical factor for its effectiveness especially in very large scale problems.

2.5 Competitive Ratios for the Online Dial-A-Ride Problem

The definition of the concept of competitive ratio gives a better understanding of the online dial-a-ride problem. The competitive ratio measures the impact of the lack of information on the performance of online algorithms - with unrestricted computational power available - and the closeness of the generated solution to the optimum offline one. So far competitive ratios for the online dial-a-ride problem deal with the minimization of the makespan i.e. the time when the last server has completed its tour or the minimization of the sum of completion times.

Hiiler[156] proved that “No deterministic algorithm for Online Darp can achieve a competitive ratio smaller than $\frac{5}{3}$ with response to completion time”.

Feuerstein [141] defined competitive ratios for the online single server Dial-a-Ride problem. More specifically, he defined competitive ratios for the Dial-a-Ride problem where the server has infinite capacity and the objective is the minimization of the time required to serve the last destination; he designed an algorithm that has best competitive ratio of 2 independely of the server capacity. He also proposed a simple a 2.5-competitive algorithm called DLT (for Don't Listen while Traveling), for the case of finite capacity. When the objective is the minimization of the sum of completion times of the rides he proved that the lower bound on the competitive ratio of any algorithm to be 1 + √2 for a server with any capacity, and 3 for a server with capacity 1.

Fink, Krumke,Westphal (2009) [142] have presented the latest work on competitive ratios .They defined that competitive ratio for k-servers as 3 for the lower bound and $\frac{2+\sqrt{2}}{\ln(1+\sqrt{2})}$ = 3.8738 for upper bounds. Those lower bounds were calculated for specific problem conditions:

1. Metric space
2. Server maximum capacity is one

No other restrictions very usual to dial ride problems like time windows or maximum vehicle ride times, are considered.

Ascheuer[161] proposed three online algorithms (REPLAN, IGNORE and SMARTSTART) for the online DARP in which vehicle capacity is 1 and the objective is to minimize the makespan. In the algorithm REPLAN, when a new request is received, the server completes the carried request (if there is one), and then replans an optimal tour considering all the yet unserved requests. The authors have proved that REPLAN is 2.5-competitive. The algorithm IGNORE is the same as the Don’t Listen while Traveling presented in the previous paragraph and was independently proved to be 2.5-competitive. Finally, the authors have shown that the more sophisticated algorithm SMARTSTART is 2-competitive. The algorithm SMARTSTART, which was extended to handle a server with any capacity, is thus optimal since it closes the gap with respect to the lower bound.

Hauptmeier et al[162] have studied the online algorithms REPLAN and IGNORE in a continuously operating environment, i.e., an environment in which the time horizon is not bounded and where requests arrive indefinitely. Clearly, the objective of minimizing the makespan or the latency are inadequate since both will always be infinite. Thus, they have considered the objectives of maximal and average flow time. The flow time of a request under a solution *s* is the diference between its completion time using solution *s* and its release time. The authors have then shown that, under a clearly defined reasonable load restriction, there is a bound for the maximal and average flow time in the algorithm IGNORE, but no bound exists in the case of the algorithm REPLAN. This result is in contrast with the property proved by Ascheuer et al[161] that both algorithms have the same competitive ratio for the online DARP.

Xiang [140] defined a competitive ratio as $\frac{Online\_\cos t - Static\_\cos t}{Static - \cos t}100\%$ dealing with the total travel distance. The simulations used were based on data according to the activity-scanning world view (Banks et al., 2001).

Although for large scale problems it is not possible to find the offline exact solution due to the combinatorial nature of the problem, we can define some competitive ratios for the same heuristic or hybrid algorithms for online and static version accordingly.

2.6 Conclusions

Based on the presented literature we can come to the following conclusion:

1. The single vehicle DARP problem is a hard combinatorial problem and only relatively small instances can be solved to optimality. This is due the fact that sharp lower

bounds on the objective value are hard to derive. Since exact approaches are in general inadequate, heuristics are commonly used in practice.

2. For the multi vehicle Dial and Ride problem none of the proposed solutions solved the problem optimally for large scale problem instances.
3. Most methodologies are based on combinations of heuristic methods. The online nature of Dial and Ride problem is a difficult task to be integrated and it is not implemented sufficiently.
4. The most preferred heuristics are combinations of inter-route, intra-route exchanges, and clustering methods,because they provide solutions in reasonable execution time and can be used as working tools in real life situations. On the other hand, there is no estimation at all, concerning the closeness to the optimal solutions.
5. There is no clear view on how to integrate online features into the solution or even what the most significant dynamic parameters are, that affect the solution.
6. The optimization procedures for online problems are based on time horizons. No one has shown so far how we can use the optimization procedure continuously as the algorithm runs.
7. The response time – a critical feature – for the online DARP problems has not been studied sufficiently.

Chapter 3 : Problem Formulation

The typical DARP problem is defined as follows: Given a set of vehicles and a set of trip demands, compute the optimum (for passengers and/or the operator) itinerary, for every vehicle, without violating certain constraints. For every vehicle a depot location is typically defined ("start" and "end" that may not necessarily be the same). In addition a capacity and a maximum vehicle drive time are also defined. The trip demands are characterized by the pickup and delivery locations, time windows for pickup and delivery, and maximum trip drive time. Those features are primary features for a trip request. More features can be taken into consideration, such as "customer specific constraints", i.e. for persons with disabilities. In the current study we stick with the primary features.

The basic notation concerning Vehicles and Trip demands are presented in Table 3-1:

Table 3-1.Basic Notation for the static Dial-a-Ride problem

V = {1,2,3, ,|V|} = *Set of Vehicles*
$Q_v = \{ Q_1 .. Q_{\|v\|} \}$ *Vehicles capacity for all v in V*
$MVRT_V = \{ MVRT_1 .. MVRT_{\|V\|} \}$ = *is the maximum vehicle ride time for every* $v \in V$
N *= is the number of trip demands. Each trip request consists of one pickup and one deliver node.*
VSD *= {**0**} = Set of "start" depots for all vehicles.*
VED *= {* ***2N +1****} = Set of "end" depots for all vehicles.*
VSED = { ***0, 2N +1****}= Set of "start" & "end" depots for all vehicles.*
$P^+ = \{1, 2, \dots N\}$ = *Set of pickup nodes in the network.*
$P^- = \{N+1, N+2, \dots 2N\}$ = *Set of deliver nodes in the network.*
$P = P^+ \cup P^-$ = *Set of pickup and deliver nodes in the network.*
$P^{all} = P \cup VED \cup VSD$ = *Set of total nodes (pickup, deliver, "start" depot, "end" depot) in the network.*
$P^{ved} = P \cup VED$ = *Set of pickup, deliver, "end" depot nodes in the network.*
$P^{vsd} = P \cup VSD$ = *Set of pickup, deliver, "start" depot nodes in the network.*
$[e_i\ l_i]$ = *time windows [earlier, late] for each node (pickup, deliver, "start", "end" depot nodes)* $i \in P^{all}$.

DRT_i = { DRT_1, .. DRT_N}** Set of requests maximum ride time for every trip request, $i \in$ **P^+

***τ_{ij} =** travel time from node i to j where i,j $\in$ **P^{all}** , .*

***c_{ij} =** travel cost from node i to j where i,j $\in$ **P^{all}** , .*

***s_i =** service time at node i $\in$ **P^{all}**.*

***d_i =** load at node (pickup or deliver) i $\in$ P^{all}. (+1 for pickups, -1 for delivers, 0 for "start", "end" depots)*

$x_{ij}^v, i, j \in P^{all}, i \neq j$ *= 1, if i^th request is served by vehicle v. else 0.*

$T_i^v, i \in P^{all}$ *Is the time variable which describes the time that the service begins at node i.*

$L_i^v, i \in P^{all}$ *Is the load variable which describes the vehicle load on every node.*

A commonly used objective function minimizes total travel cost as follows:

$$\sum_{v \in V} \sum_{i \in P^{all}} \sum_{j \in P^{all}} x_{ij}^v c_{ij} .$$

With the following set of constraints:

1. Trip time window constraints (**[e_i l_t]**)
2. Maximum vehicle ride time constraints (***$MVRT_V$***)
3. Maximum trip request ride time constraints (***DRT_i***)
4. Maximum Vehicle load restriction constraints (***Q_v***)

The Mixed Integer Linear Program that can solve the above defined problem is as follows:

$$\min \sum_{v \in V} \sum_{i \in P^{all}} \sum_{j \in P^{all}} x_{ij}^v c_{ij} \tag{1}$$

ST:

$$\sum_{v \in V} \sum_{j \in P^{all}} x_{ij}^v = 1, i \in P^+ \tag{2}$$

$$\sum_{j \in P^{all}} x_{ij}^v - \sum_{j \in P^{all}} x_{ji}^v = 0, i \in P, v \in V \tag{3}$$

$$\sum_{j \in P^+} x_{ij}^v = 1, i \in VSD, v \in V \tag{4}$$

$$\sum_{i \in P^-} x_{ij}^v = 1, j \in VED, v \in V \tag{5}$$

$$\sum_{j\in P^{all}} x_{ij}^{v} - \sum_{j\in P^{all}} x_{j(N+i)}^{v} = 0, i \in P^{+}, v \in V \qquad (6)$$

$$T_i^v + s_i + \tau_{i(i+N)} \le T_{(i+N)}^v, i \in P^+, v \in V \qquad (7)$$

$$x_{ij}^v = 1 \Rightarrow T_i^v + s_i + \tau_{ij} <= T_j^v, i, j \in P \ \ , v \in V \qquad (8)$$

$$x_{ij}^v = 1 \Rightarrow T_i^v + s_i + \tau_{ij} <= T_j^v, i \in VSD, j \in P^+ \qquad (9)$$

$$x_{ij}^v = 1 \Rightarrow T_i^v + s_i + \tau_{ij} <= T_j^v, j \in VED, i \in P^- \qquad (10)$$

$$x_{ij}^v = 1 \Rightarrow L_i^v + d_j = L_j^v, i \in P \ \ , j \in P^+, v \in V \qquad (11)$$

$$x_{ij}^v = 1 \Rightarrow L_i^v - d_{j-N} = L_j^v, i \in P, j \in P^-, v \in V \qquad (12)$$

$$x_{ij}^v = 1 \Rightarrow L_i^v + d_j = L_j^v, i \in VSD, j \in P^+, v \in V \qquad (13)$$

$$L_i^v = 0, i \in VSD \qquad (14)$$

$$e_i \le T_i^v \le l_i, i \in P, v \in V \qquad (15)$$

$$e_i \le T_i^v \le l_i, i \in VSD \ \ , v \in V \qquad (16)$$

$$e_i \le T_i^v \le l_i, i \in VED \ \ , v \in V \qquad (17)$$

$$d_i \le L_i^v \le Q^v, i \in P^+, v \in V \qquad (18)$$

$$T_{(i+N)}^v - T_i^v \le DRT_i, i \in P^+, v \in V \qquad (19)$$

$$T_i^v \le MVRT_v, i \in P^{all}, v \in V \qquad (20)$$

$$x_{ij}^v, binary$$

The Objective function (1) minimizes the total cost produced by all vehicles.

Constraint (2) guarantees that one vehicle is leaving one pickup location and heading to a unique deliver location.

Constraint (3) guarantees that, the number of vehicles entering stop location i will leave location i.

Constraint (4) guarantees that, there is only one unique pickup location after the “start” depot location for each vehicle.

Constraint (5) guarantees that, there is one unique deliver location before “end” depot location for each vehicle.

Constraint (6) is the pairing constraint, meaning that the same vehicle visits node *i, i+n*.

Constraints (7-10) are precedence constraints ensuring that node *i* is visited before node i+n and the compatibility between routes and schedule.

Constraints (11-14) ensure the consistence of load variables for every trip request.

Constraint (15) ensures that each request time windows is not violated.

Constraints (16-17) ensure that each vehicle time windows constraint is not violated.

Constraint (18) makes sure that each vehicle’s maximum load capacity is not violated.

Constraint (19) maintains each trip maximum ride time.

Constraint (20) maintains each vehicle maximum drive time.

This mathematical program has been implemented in AMPL (one vehicle)and solved with CPLEX solver (See Appendix A). Basic goals of this MILP implementation were:

1. Confirmation of the adequacy of the exact DP algorithm, proposed in the next chapter
2. Exploration of solving times by MILP formulation and therefore the utilization of this formulation - or other exact algorithms - as sub-algorithms by heuristic algorithms

The specific experiment characteristics were:

Vehicles: 1

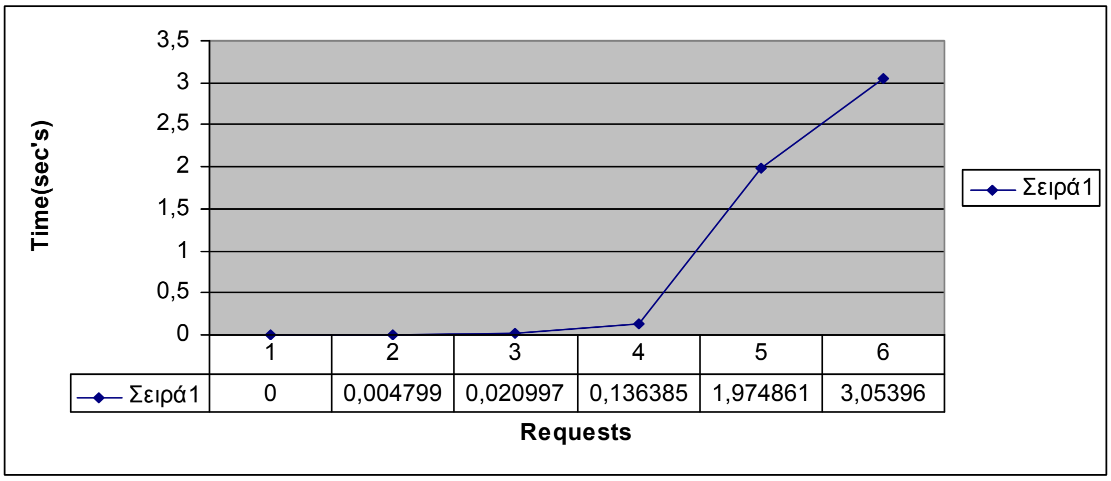
Demand Requests: 1 to 10. (Each request represented by one pickup and one deliver point)

Network: A two dimensional network (representing a square area with edge length 100 km's) was created by using random coordinations for each pickup and delivery point. (Minimum value was 0.00, maximum value was 100.00)

Distances between various points (pickup and deliver) are based on the triangular property of Euclidean networks. The following table shows the average CPU time. Variability is high and depends mainly of the locations of pickup, delivery, start depot locations, and end depot locations. Figure 3.1 represents the system behavior (in terms of execution time)

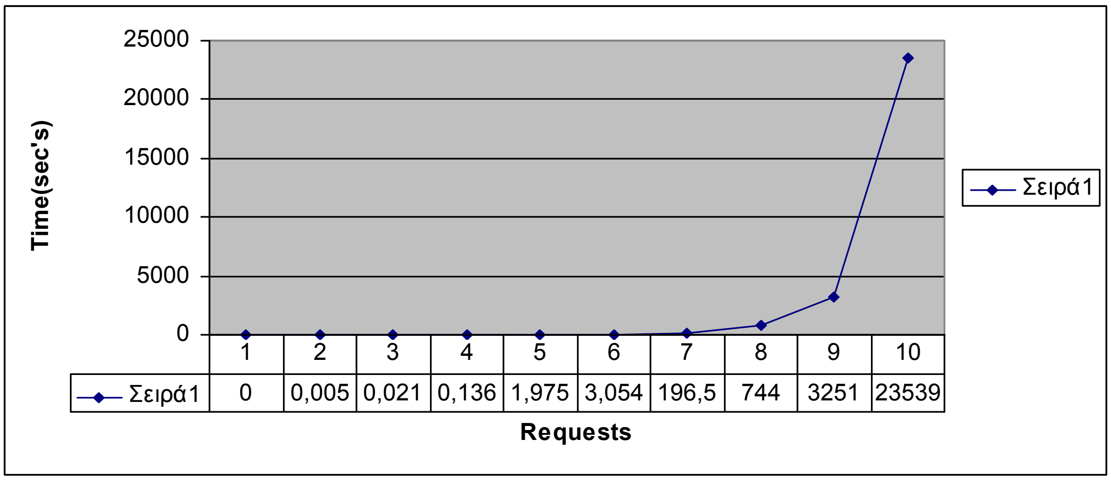
Table 3-2. Experiments Table for MILP implementation of the static Dial-a-Ride exact solution

No of Requests	*Time (sec's) On Average*	*Time (sec's) Max*	Time (sec's) Min
1	0	0	0
2	0,004799	0,008998	0,001
3	0,020997	0,036993	0,002
4	0,136385	0,6589	0,016
5	1,974861	10,2044	0,05
6	3,05396	9,23	0,576
7	196,5368	1096,5	7,26
8	743,976	3173	69,3
9	3250,994	6447	0,95
10	23539	30727	16351



	1	2	3	4	5	6
Σειρά1	0	0,004799	0,020997	0,136385	1,974861	3,05396

Figure 3-1.Experiments execution time (1 through 6 requests)



	1	2	3	4	5	6	7	8	9	10
Σειρά1	0	0,005	0,021	0,136	1,975	3,054	196,5	744	3251	23539

Figure 3-2.Experiments execution time (1 through 10 requests)

<u>Some Conclusions:</u>

1. The time needed to solve more than 8 requests (8 pickups, 8 delivers) increases "explosively", making this implementation inappropriate for real applications, where more than 10 requests must be handled.
2. Such solution times denote that this implementation cannot be used as sub-solver to any Online DARP algorithm, which is designed to be used in real life.
3. It is necessary to reduce the CPU times and to increase the problem size. This can be achieved if we use some kind of pre-processing that can eliminate as many as possible of the variables x_{vij} representing infeasible problem states. For example: Trips

such as $(n_t + i, i), i \in P^+ or (j, i), j \in VSD, i \in P^-$, or requests time windows **[e*i* *l*i]** that overlap with each other.

4. If we achieve a significantly smaller execution time, then this formulation can be used extensively as an exact sub-solver to the online algorithm, proposed later for the online DARP problem.

Chapter 4 : Static Dial-a-Ride Algorithms

4.1 The Insertion Algorithm InsertionH

4.1.1 Basic concept

The insertion algorithm we developed, is a variation of the one presented by Jaw [132]. The basic function of this insertion algorithm is to create specific trip to vehicle assignments, as an initial solution which can be refined later by other optimization heuristics.

4.1.2 Nomeclature

V = {1,2,3,…..,|V|} = *Set of Vehicles*

R $_{V\{v=1,2,,,,|V|\}}$ *= is the route schedule for every* $v \in V$

RC $_{V\{v=1,2,,,,|V|\}}$ *= is the route cost for the for every* $v \in V$

N *= is the number of trip requests. Each trip request is consisted by one pickup and one deliver node.*

TR $_{i\{i=1,2,,,N\}}$ *= is the sequence of trip demands.*

EPTi $_{\{i=1,2,,,N\}}$ *= Earlier pickup time of trip request i*

BIP$_{iv}$ *= the best insertion position of i-th trip request* TR_i *to Route schedule* R_v

MinCostBIP$_{iv}$ *= is the cost of the best insertion position of the trip request* TR_i *to Route schedule* R_v

4.1.3 Algorithm Description

In a preprocessing step the algorithm creates an empty route for every vehicle. Then for every trip request the algorithm finds the best position -in terms of cost- to insert the trip by searching all vehicle routes. If such a position is found, then the trip request is inserted; otherwise, if it is not possible to find at least one position where the request could be inserted, the trip request is rejected. At the end of the execution, specific trips have been assigned to specific vehicles. The main drawback of this algorithm is that it always tends to load the first vehicle with more demands. That is why we implemented a variation where the search for each demand starts with a different vehicle each time. A pseudo code description of InsertionH algorithm follows:

```
Step0: for every $v$ in $V$ build an empty $R_v$
Step1: Sort $R_v$ in descending order according to $RC_v$
Step2: Sort $TR_i$ in ascending order according to demands $EPT_i$
Step3: for every $TR_{i\ \{i=1,2,,,N\}}$
    Step3.1: Sort $R_v$ in ascending order according to $RC_v$
    Step3.2:for every $R_{v\ \{v=1,2,,,,|V|\}}$ do
        find the $BIP_{iv}$
        Calculate $MinCostBIP_{iv}$
    Step3.3: If no $BIP_{iv\ \{v=1,2,,,,|V|\}}$ found goto step 3.6
    Step3.4: From all $MinCostBIP_{iv}$ select the minimum one
    Step3.5: Assign that $TR_i$ to the appropriate $R_v$ and goto step3
    Step3.6: Reject that trip request and repeat step3
end for
```

4.1.4 Algorithm computational effort

Computational effort of the InsertionH algorithm can be simply calculated by the number of possible insertions that the algorithm should check in order to find the best position in terms of solution cost (given that this is the objective). The worst scenario is the case that only one vehicle is available. The best scenario is the case that the number of vehicles is equal to the number of trip requests and each vehicle has one trip request assigned.

Given a number of n demands – not including the first demand - and one available vehicle, the total number of possible searches for the insertion algorithm can be calculated as the following sum:

$1+\sum_{i=1}^{n}\sum_{k=1}^{2*i+1}k$ (4-1).The closed form of this expression is $1+\frac{1}{6}n(17+15n+4n^2)$ (4-2).

The following example explains clearly the above mathematical formula.

Example : One Vehicle, Three trip demands (where P1, D1,P2,D2,P3,D3 represent the pickup, delivery of the respective trip demands. For the first trip there is only one possible combination defined as:

P1D1

Total Number of combinations is: 1

After the first trip allocation, the number of possible combinations for the assignment of the second trip is 6:

P2 D2 P1 D1
P2 P1 D2 D1
P2 P1 D1 D2

P1 P2 D2 D1
P1 P2 D1 D2

P1 D1 P2 D2
Total Number of combinations for the assignment for trips one and two is 7 (6+1)

After the second trip allocation, the number of possible combinations for the assignment of the third trip is 15:

P3 D3 P2 D2 P1 D1
P3 P2 D3 D2 P1 D1
P3 P2 D2 D3 P1 D1
P3 P2 D2 P1 D3 D1
P3 P2 D2 P1 D1 D3

P2 P3 D3 D2 P1 D1
P2 P3 D2 D3 P1 D1
P2 P3 D2 P1 D3 D1
P2 P3 D2 P1 D1 D3

P2 D2 P3 D3 P1 D1
P2 D2 P3 P1 D3 D1
P2 D2 P3 P1 D1 D3

P2 D2 P1 P3 D3 D1
P2 D2 P1 P3 D1 D3

P2 D2 P1 D1 P3 D3

Total Number of combinations for the assignment for trips one, two and three is 22 (15+6+1)

The algorithm complexity is $O(n^3)$. For a total number of 1000 demands the worst scenario (one vehicle) gives a total number of 669,169,501 possible trip combinations.

Figure 4-1 presents graphically the InsertionH algorithm possible trip combinations. The x–axis represents the number of trip demands, and the y-axis represents the number of possible trip combinations during the insertion procedure.

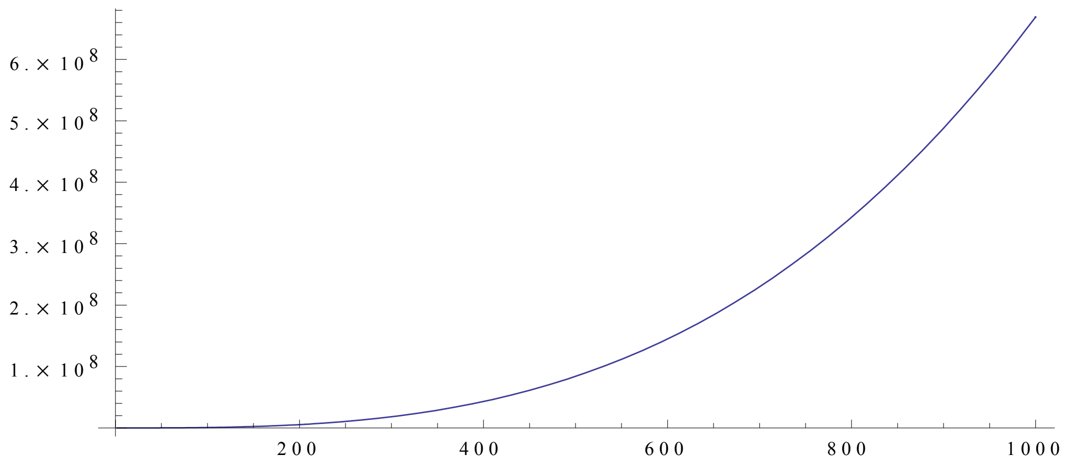


Figure 4-1.InsertionH algorithm's search computational performance

4.2 The DP Exact algorithm

4.2.1 Basic Concept

The Exact algorithm is a DP implementation. The algorithm examines all possible valid combinations of vehicles and trip assignments. During the execution the algorithm improves the solution continuously i.e., when a combination of possible demand assignments has higher cost than a previous solution, then further searching based on this combination is discontinued

4.2.2 Algorithm Description

The proposed DP algorithm can be described as a greedy algorithm. The algorithm searches for the exact solution in all feasible solution combinations. This kind of search requires long computational time in order to find the exact solution. Although the introduction of some restrictions can reduce significantly the solution space that algorithm searches for the optimal solution. The certification of the correctness of the proposed DP algorithm can be authenticated by comparing the results of this algorithm with the results to the well known MILP mathematical formulation. In this chapter we present two implementations. The DP implementation and the MILP implementation with AMPL and CPLEX. Computational results for both implementations are listed in order to draw relevant conclusions from the comparison of these two outputs.

As it is depicted in the computational results later, our implementation of the DP seems to outperform the MILP approach in terms of both computational time and memory requirements. Due to the nature of the problem, if we want to get the exact solution for the DARP problem it is necessary to evaluate every feasible combination of trip demands and vehicles that we cannot rule out based on some technique. Given the problem's combinatorial nature it is obvious that the number of combinations explodes, making any exact approach meaningless for realistically sized problems. If we reduce somehow that number of combinations, then the solution space is restricted making the effort more promising. Memory requirements are another issue. We need to reduce memory requirements in order to execute the algorithm in a realistic memory space. Initial attempts for the exact solution for one vehicle were introduced by Psaraftis[114] . He calculated that the required memory locations in order to keep track of all states are of the order $O(3^N)$.

Keeping the above observations in mind, we tried to enrich our exact algorithm with features that offer: (a) restriction of the solution space as much as possible, and (b) the independence of memory resources. For these reasons a DP implementation was developed by using recursive techniques. Recursive techniques offer tracking of the last execution point in the algorithm. Every time we have a recursive return we know exactly what the next point to move to is.   It is similar to the Depth First Algorithm, which is well known in graph theoretical approaches. The algorithm starts with an initial combination. Based on that combination, it finds the next one, which contains the parent - previously produced - combination. The process continues that way, until finding the “final” or “bottom” combination.

The notation ViPjViDj….ViDj (i=1,2,…|v|, j=1,2,….N) is used in order to describe the route sequence. For example, given two vehicles (denoted as V1,V2) and two trip demands (denoted as P1,D1 for picking and delivering trip request 1, P2,D2 for picking and delivering trip request 2), the algorithm starts by using first vehicle V1. It can select the first trip request (only pickup is possible) “V1P1” or select the second trip request (only pickup is possible) “V1P2 to be assigned to vehicle V1. Let’s suppose that the first trip request is selected “V1P1”. At that point the algorithm puts a “backtracking” (+1) point.

For the “V1P1” combination, the combinations that could follow are: To deliver what was picked-up form first trip request “V1P1V1D1”, or to pick up the second trip request “V1P1V1P2” or to let the second vehicle pickup the second trip request “V1P1V2P2”. Another backtracking point (+2) should be placed here.

For the “V1P1V1D1” combination, the combinations that could follow are: The first vehicle picks up the second trip request “V1P1V1D1V1P2” or the second vehicle picks up the second trip request “V1P1V1D1V2P2”. Another backtracking point (+3) should be placed here.

For the “V1P1V1D1V1P2” combination one more combination has been left in order to construct a full trip. That combination is “V1P1V1D1V1P2V1D2”. After the evaluation of that full trip, the algorithm backtracks to the previous backtracking point in order to search the next combination that can follow: combination “V1P1V1D1V2P2”. The following figure depicts this search procedure. Red dashed arrows show the search sequence.

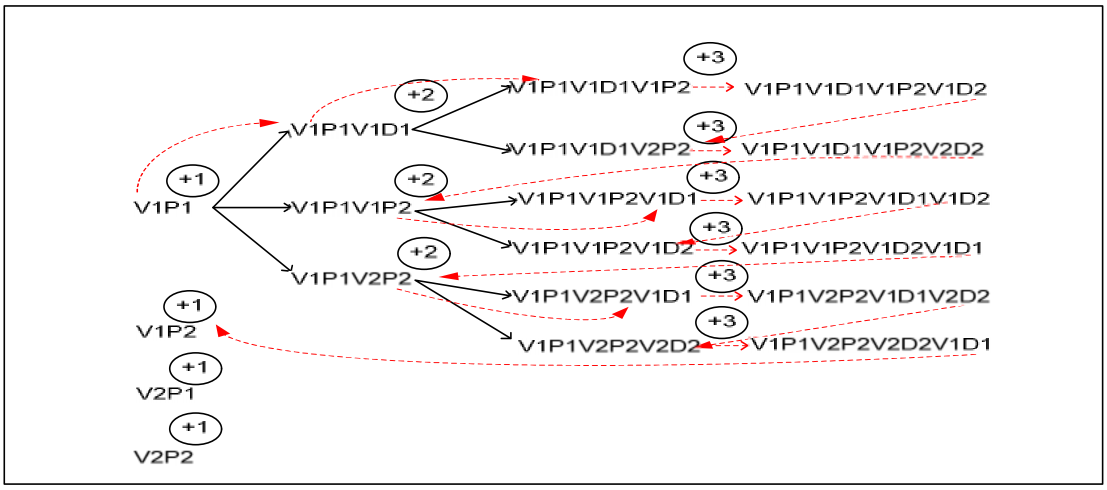


Figure 4-2.DP Exact Algorithm’s recursive Graph

The notation (+1), (+2), (+3) signifies track points that the algorithm puts in order to remember where to backtrack, after the recursive return.

Trip combination V1P1V1D1V1P2V1D2 is a full trip, satisfying all trip demands. Every combination is checked for feasibility and if feasible, is evaluated by calculating travel cost. The example above shows how algorithms produce possible trip combinations. By using track points there is no need to store all combinations because every time we know exactly where to go. The cost produced from the first full trip solution is used as a reference point in subsequent searches. If a trip combination is not feasible or the calculated cost until that time is larger than the previously calculated one, the search using that combination is discontinued. The term “feasible” means that there is no violation of:

- Time windows concerning trip demands
- Time violation concerning maximum trip request ride time
- Time violation concerning maximum vehicle ride time
- Violation concerning maximum vehicle capacity

As a result, the search is limited in a restricted solution space. Of course the number of possible combinations is combinatorial, increasing as a function of the trip demands and vehicles numbers. That technique obviously cannot be used in order to solve actual large problems. However, knowing an initial feasible solution we could substantially improve the search process. To achieve an initial feasible solution we run the InsertionH algorithm just to produce an initial solution that can be used as a starting point of our “DP Exact” algorithm.

In order to describe the recursive nature of the proposed exact algorithm we provide the necessary information concerning notation and the recursive relationship

Let

$T = \{t_1, t_2, ..., t_n\}$ = the set of unassigned trips

$T_k = \{t_1, t_2, ..., t_k\}, k \le n$ = the set of assigned demands in a feasible solution. Starting value of T_k is { }.

S_{T_k} = the set of feasible solutions resulting from all possible trip – belongs to set T_k -combinations concerning routing and scheduling.

$MinCost$ =represents the lower cost limit that allows the exact algorithm to reject all feasible solutions with larger costs.

$MinCost$ Could be:

Initial Arbitrary Large Value
Or
The cost of a feasible solution produced by an InsertionH algorithm in a preprocessing phase
Or
The cost of a feasible solution - containing all demands of set T - produced by the DP Exact algorithm.

$PS(S_{T_k}, t_i, MinCost)$= returns all feasible solutions by combining each solution contained in S_{T_k} ,with the new trip t_i ,given the MinCost.

The recursive relationship and the $S_{T_{k+1}} = PS(S_{T_k}, t_{k+1}, MinCost)$ problem reduce to $S_{T_{k=n}}$. In this case set $S_{T_{k=n}}$ contains all demands included in T . After the evaluation of every feasible solution included in $S_{T_{k=n}}$ the optimal solution is finally found.

A short pseudo-code description of the exact algorithm follows:

```
Step 0: $T_k = \{ \}, k = 0$
 $S_{T_k} = \{\}$
 $MinCost = Min$ (arbitrary large number, Cost of the best feasible solution produced by any InsertionH algorithm)

Step 1: For every demand $t_i$ in set T

Step2: $S_{T_{k+1}} = PS(S_{T_k}, t_i, MinCost)$
```

```
Step3 Reject from $S_{T_{k+1}}$ all feasible solutions with cost more than MinCost
Step4: $T_k = T_k \cup t_i, T = T \cap t_i$
Step 5: If $T \neq \{\}$ recursive call of Step1
else
    MinCost = Min($S_{T_{k+1}}$)
    Mark solution with cost equal to MinCost
    Recursive return to the previous state
End if
```

A specific example has been constructed in order to evaluate the algorithm's computational performance and the results are presented in detail in Appendix B. The example uses 2 vehicles and 2 trip demands. Each vehicle is denoted as Vi (i=1,2) and each trip request is denoted as Pj (j=1,2) for pickups and Dj (j=1,2) for deliveries accordingly.

Before executing the exact algorithm, we use the InsertionH algorithm to produce some initial feasible solution. The heuristic finds a feasible solution with cost of 193.861 distance units. This cost is used as an upper bound for the DP Exact algorithm. The exact algorithm produces output – see appendix B - which clarifies how it works. Not all possible combinations are presented here, because for some cases it is useless to search more. E.g. consider a combination V1P1V1D1V1P2 where there is no reason to search for further combinations because already this combination is not feasible. As the algorithm progresses it finally finds the best combination which is V1P2V1D2V1P1V1D1 with cost 191.941 distance units.

This exact approach – limited only by execution time- can solve any DARP problem. It may run for unreasonably long CPU times, so clearly it is practically limited to fairly small and probably not very useful (application-wise) problems; typically for 1 vehicle problem, it can solve up to 10 trip demands, for 2 vehicles up to 8 trip demands, for 3 vehicles up to 7 trip demands and for 4 vehicles up to 6 trip demands, which is of no practical value. Where the value of this approach lies, as discussed in detail later, is using the 2-vehicle approach to explore a fairly large neighborhood solution as part of a VLSN algorithm search approach.

In order to test the exact algorithm, problem instances were randomly produced. The test field was a square area of 100X100 Kilometers. Pickup and delivery positions of every trip request were selected randomly over that area. Vehicle “Start”, “End” depots were selected randomly over that area. Time windows for pickup were randomly selected between 0–1440, time windows for delivery were produced from pickup time windows plus the 1.5*Cartesian distance between those points. Maximum vehicle ride time was defined as 1440. Vehicle speed was assumed to be 1 Km per Minute and vehicle capacity was limited to 3 seats. The graphic representation of the solution space may give us some ideas on how we can use specific features of the problem in order to find the optimal solution. Or may highlight some patterns which exist within the solution space. For these reasons, we present two graph representations of the solution space and solution search process using as input the DP Exact algorithm search output. A typical DARP problem with 3 vehicles and 5 trip demands was prepared and solved for this purpose (network characteristics and trip features remain the same as those described in the preceding paragraph). In figure 4-3 x-axis represents the solution cost while y-axis represents the solution ordering number   i.e. 1,2,3…5000

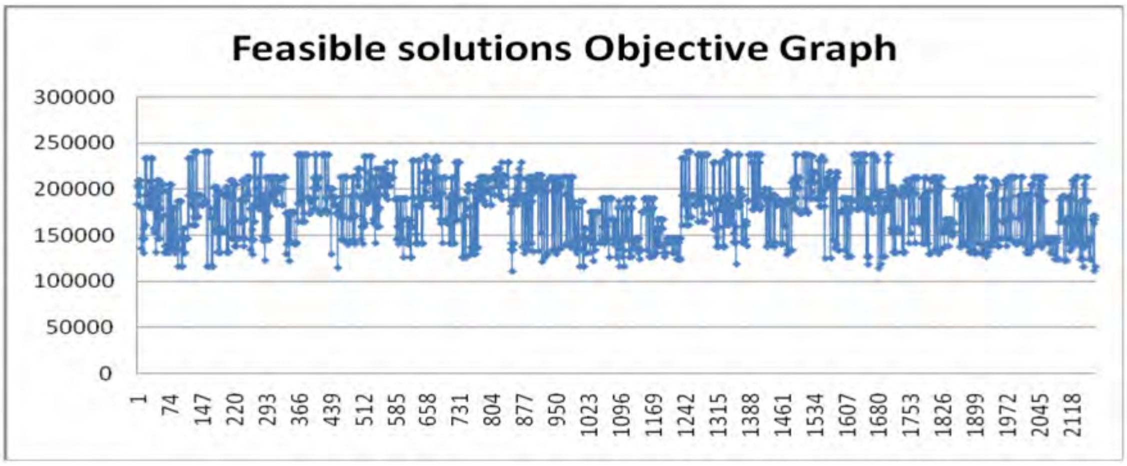


Figure 4-3.Solution space graph for the static DARP problem (3 Vehicles,5 Trips)

Figure 4-3 presents all feasible solutions produced after the execution of the DP Exact algorithm. What is interesting in this figure is that we cannot find a pattern that can be used as guidance to search for the optimal solution. This may be the reason, heuristics are so easily trapped to local optimums. The following figure 4-4 shows the way the search process improves the solution cost every time a feasible solution is found with lower cost than the previous one.

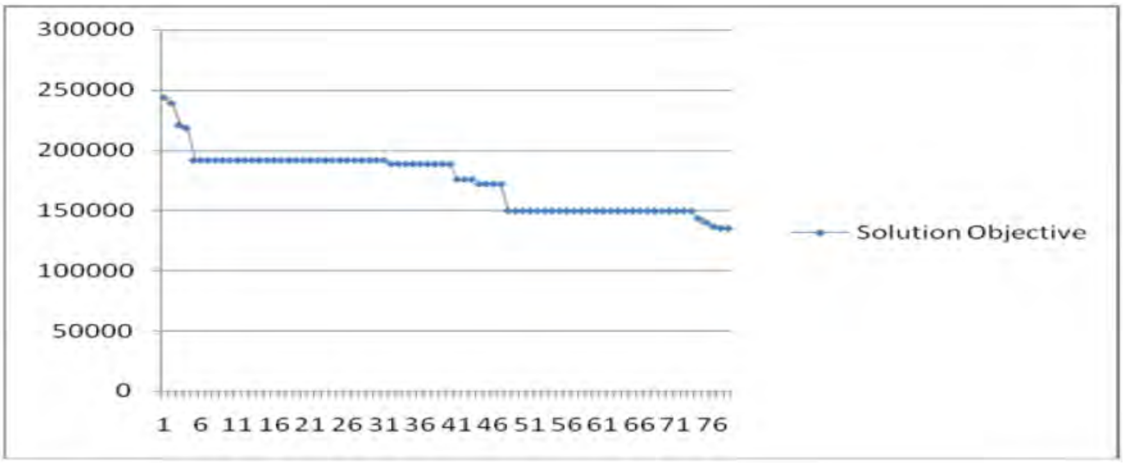


Figure 4-4.DP Exact searching process graph

Every time the exact algorithm finds a feasible solution that has a lower objective than the previous one, it uses this objective as a reference for the subsequent searches. By using this technique we substantially limit the number of possible trip schedules. However, because of the NP-Hard [155] nature of the Dial-a-Ride problem, it is obvious that the exact algorithm cannot solve large scale problem instances within reasonable time.

4.2.3 Algorithm computational effort for the DP Exact algorithm.

It has been proven [155] that DARP problem is NP hard. Solution space explodes when the number of demands increases. For the 1-vehicle problem the number of possible solutions is given by the following formula: $d!(d!+\sum_{x=1}^{d-1}\sum_{y=1}^{d-x}\sum_{z=1}^{d-1}(d-z)!)$ (4-3) where d is the number of trip demands.

The following examples explain clearly the above mathematical formula.

Example 1: One Vehicle, Two trip demands (where P1, D1 represents the pickup, delivery of the first trip demand, P2, D2 represents the pickup, delivery of the second trip)

Number of valid Pickup, Deliver combinations is 6:

P1P2D1D2
P1P2D2D1
P2P1D1D2
P2P1D2D1
4 combinations
P1D1P2D2
P2D2P1D1
2 combinations
Total: 6 combinations
Example 2: One Vehicle, Three trip demands (where P1, D1 represents the pickup, delivery of the first trip demand, P2, D2 represents the pickup, delivery of the second trip, P3, D3 represents the pickup, delivery of the third trip)

Number of valid Pickup, Deliver combinations is 90:

P1P2P3D1D2D3
    ……….D1D3D2
    ……….D2D1D3

……….D2D3D1

……….D3D1D2

……….D3D2D1

P1P3P2 . .

P2P1P3 . .

P2P3P1 . .

P3P1P2 . .

P3P2P1D1D2D3

….D1D3D2

….D2D1D3

….D2D3D1

….D3D1D2

….D3D2D1

36 (6x6) Combinations

P1P2D1P3D2D3

P1P2D1P3D3D2

P1P2D1D2P3D3

P1P2D2D1P3D3

P1D1P2P3D2D3

P1D1P2P3D3D2

P1P2D2P3D1D3

P1P2D2P3D3D1

P1D1P2D2P3D3

P3P2D3P1D1D2

P3P2D3P1D2D1

P3P2D3D2P1D1

P3P2D2D3P1D1

P3D3P2P1D1D2

P3D3P2P1D2D1

P3P2D2P1D1D3

P3P2D2P1D3D1

P3D3P2D2P1D1

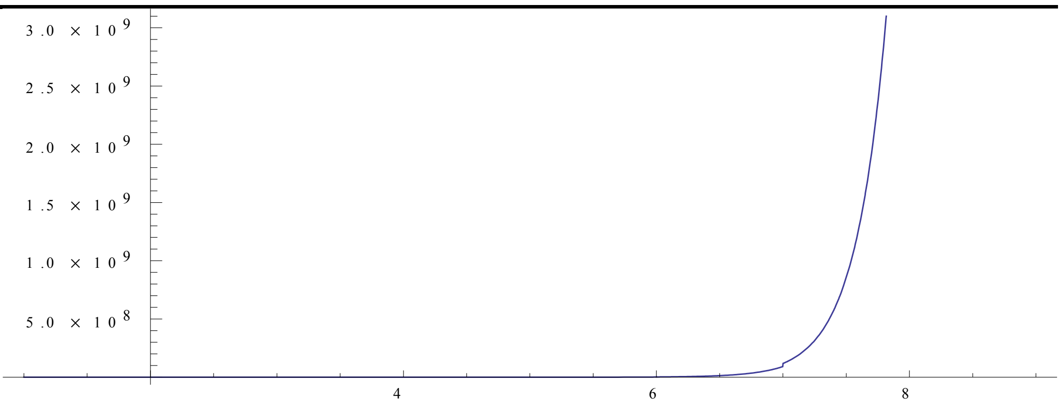
54(6x9) combinations

Total: 90 combinations

Table 4.1 and Figure 4-5 shows clearly the nature of the solution space as a function of the number of demands.

Table 4-1: Solution Space size for one vehicle

Demands Number	*Combinations*
1	1
2	6
3	90
4	1872
5	54000
6	2170800
7	117799920
8	8301242880
9	735655011840
10	79974705888000
15	1460690908759555304832000
20	6528387048161452132873560504238080000
50	24060288862563820952726974199966940990389145053095343821684131102673683060167111988404460384061720374345228032252313600000000000000



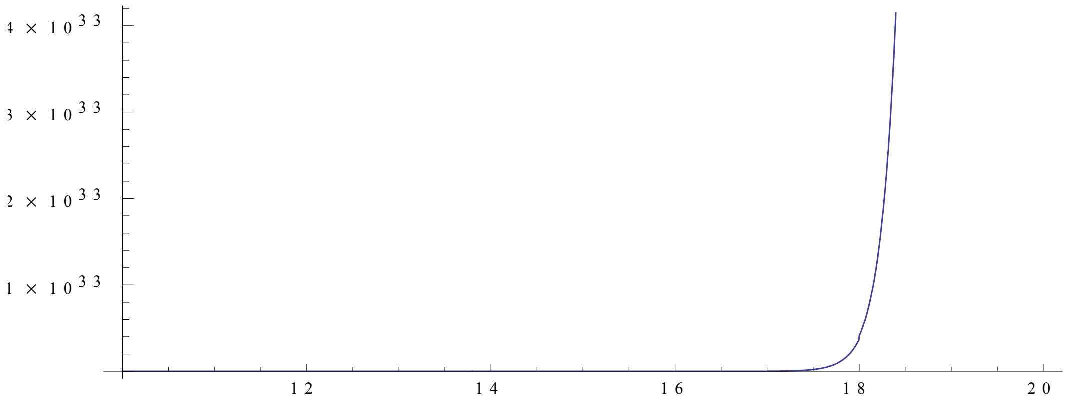


Figure 4-5.DP Exact algorithm searching graph (x-axis = number of trip demands, y-axis = number of pickup and deliver combinations for all trip demands)

It is obvious that any exact algorithm that searches in a brutal way for the optimal solution is not able to provide the optimal solution for large scale problems in affordable execution time.

For the m-vehicle problem the number of possible combinations is given by the following formula: $\sum_{L=1}^{v}\sum_{d=1}^{vd}\frac{vd!}{(vd-d)!d!}(d!(d!+\sum_{x=1}^{d-1}\sum_{y=1}^{d-x}\sum_{z=1}^{d-1}(d-z)!))$ (4-4) where $\boldsymbol{v}$ is the number of vehicles and $\boldsymbol{vd}$ is the number of trip demands.

The above relationship is produced as follows:

Given a number of v vehicles and the number of vd demands, the number of all possible permutations of the trip demands assignments to a specific vehicle is produced by the formula $\sum_{d=1}^{vd}\frac{vd!}{(vd-d)!d!}$ (4-5). For all vehicles this number is produced by the formula $\sum_{L=1}^{v}\sum_{d=1}^{vd}\frac{vd!}{(vd-d)!d!}$ (4-6). The following example explains clearly the above mathematical formula.

Given a number of 2 vehicles V1,V2 and a number of 3 trip demands TD1,TD2,TD3 the number of possible permutations is $\sum_{L=1}^{2}\sum_{d=1}^{3}\frac{vd!}{(vd-d)!d!}=14$. These permutations can be categorized as follows :

All trip demands assigned to one vehicle. (V1 or v2)
V1TD1TD2TD3
V2TD1TD2TD3
Two demands assigned to one vehicle (V1 or v2)
V1TD1TD2
V1TD1TD3
V1TD2TD3
V2TD1TD2
V2TD1TD3
V2TD2TD3
One demand assigned to one vehicle (V1 or V2)
V1TD1
V1TD2
V1TD3

V2TD1
V2TD2
V2TD3.

On the other hand the number of assignments, for all vehicles, is produced by the formula v^{vd} (4-7). The following example – given the number of two vehicles (V1,V2) and three (3) trip demands (TD1,TD2,TD3)– explains 4-7.

V1TD1TD2TD3
V2TD1TD2TD3
V1TD1TD2-V2TD3
V1TD1TD3-V2TD2
V1TD2TD3-V2TD1
V2TD1TD2-V1TD3
V2TD1TD3-V1TD2
V2TD2TD3-V1TD1
Total number of combinations equal to eight ($8=2^3$)

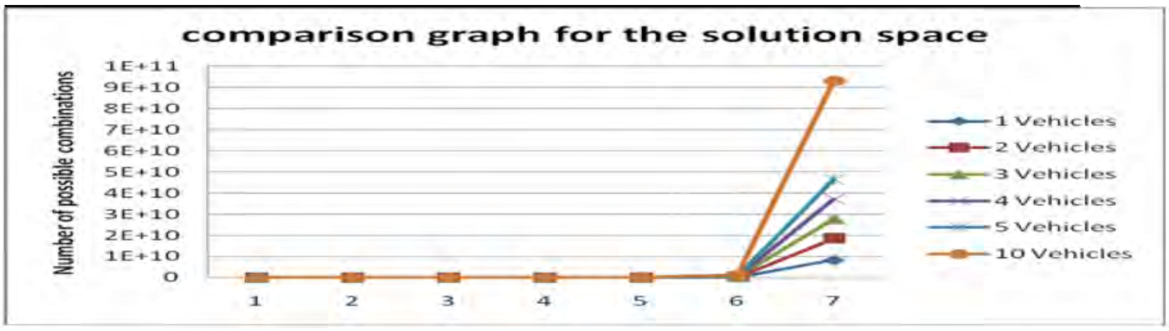
The total computational effort is given by combining formula (4-3) and formula (4-6) which results in the formula

$$\sum_{L=1}^{v}\sum_{d=1}^{vd}\frac{vd!}{(vd-d)!d!}(d!(d!+\sum_{x=1}^{d-1}\sum_{y=1}^{d-x}\sum_{z=1}^{d-1}(d-z)!))\,.$$

The following table gives some comparison results concerning the computational effort expressed as the number of all possible trip combinations produced by formula 4-4.

Table 4-2: comparison for computational effort for 1,2,3,4,5,10 vehicles

Dems	*1 Vehicles*	*2 Vehicles*	*3 Vehicles*	*4 Vehicles*	*5 Vehicles*	*10 Vehicles*
2	6	16	24	32	40	80
3	90	222	333	444	555	1110
4	1872	4544	6816	9088	11360	22720
5	54000	128650	192975	257300	321625	643250
6	2170800	5049552	7574328	10099104	12623880	25247760
7	117799920	268396646	402594969	536793292	670991615	1341983230
8	8301242880	18615169792	27922754688	37230339584	46537924480	93075848960



4.2.4 Algorithm computational results and correctness of the solution

The Proposed algorithm was implemented in C++ and tested on a Linux machine with a dual core processor. The exact algorithm has been tested extensively in order to prove its analytical convergence to the optimum solution. We chose to compare that algorithm with the Mixed Integer Linear Program (MILP) implementation of the DARP problem. The mathematical formulation presented in Section 2 has been used to implement the MILP in AMPL. Both implementations DP and MILP gave always the same results concerning objective function and trips pickup and deliver sequence. The following table presents the results of the comparison between two implementations.

Table 4-3: Comparison Results between DP and MILP implementation

Vehicle Number	*Trip demands*	*CPU Time for DP (Sec's)*	*CPU Time for MILP*
1	4	0.120	0.295
1	5	1.030	0.400
1	6	22.480	9.124
1	7	263.000	267.973
1	8	578.964	3527.330
2	4	1.150	-
2	5	15.450	-
2	6	493	-
2	7	1559	-
3	4	1.260	-
3	5	155.910	-
3	6	604.653	-
3	7	8149	-
4	4	2.450	-
4	5	17.620	-
4	6	3583	-
5	4	2.040	-
5	5	149.420	-
5	6	459.850	-
5	7	19592	-

Execution times show clearly that the DP implementation is faster than MILP one. For that reason the DP implementation has been chosen to be used as sub-module in the heuristic – exact algorithm implementation.

4.3 *The Very Large Scale Neighborhood hybrid algorithm*

4.3.1 Basic Concept

The results of the above DP Exact algorithm show that the use of this algorithm is limited due to its long execution times. The same happens in the case of MILP implementation. A different approach is to use algorithms that will use the exact algorithm as a sub-algorithm. The algorithm presented in this chapter, called Very Large Scale Neighborhood (VLSN) is based on the iterative usage of the exact DP algorithm for every pair of vehicles. The VLSN algorithm can be used in order to solve DARP problems by repetitively applying the DP Exact algorithm for every m vehicles from a total number of v vehicles, where 2 <= m <= v-1. This technique achieves feasible solutions in a broader neighborhood solution space. The number of vehicles used each time by the exact algorithm affects the neighborhood size. The following figure 4-6 gives an artistic representation of VLSN algorithm solution search. The outer ellipse represents the whole range of possible solutions. Each inner ellipse represents the solution neighborhood explored by m-vehicles DP Exact algorithm. Larger number of vehicles used by DP exact algorithm results in larger neighborhood search.

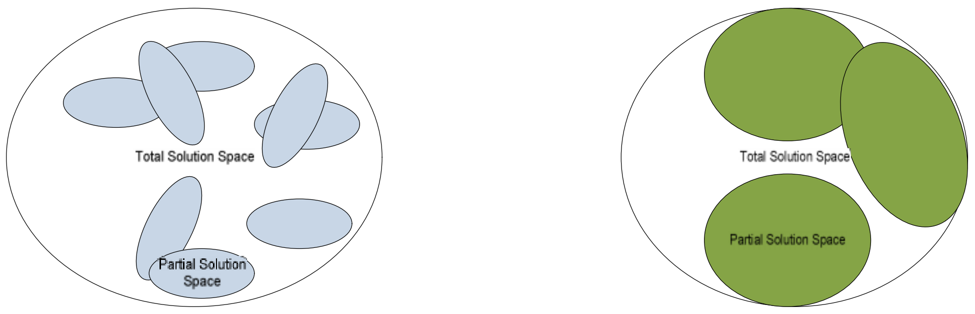


Figure 4-6.VLSN Algorithm artistric graph representation for search process

4.3.2 Nomenclature

V = {1,2,3,…..,|V|} = *Set of Vehicles*

ExactDP2 = *is the Exact DP algorithm for 2 Vehicles*

VP $_{v1\{v1=1,2,\dots|V|\},\ v2\{v2=1,2,\dots|V|\}}$ *= is the pair of vehicles that will be used by ExactDP2 algorithm*

RT *=* $\{VP_{1,2}, VP_{1,3} .. VP_{1,|v|} ,\dots, VP_{|v-1|,|v|}\}$ *is the set for every pair of vehicles permutations*

$\boldsymbol{RC}_{v\{v=1,2,\dots|V|\}}$ *= is the route cost for vehicle v*

TRC = $\sum_{1}^{|V|} RC_v$ *is the total cost for all routes*

4.3.3 Algorithm Description

The main idea is to search in a greater solution space neighborhood by using the analytical 2-vehicle approach which classifies it as Very Large Scale Neighborhood (VLSN) heuristic. (Ahuja[149]). The approach is continuously applying the above presented DP Exact algorithm for 2 vehicles for every possible combination of vehicle routes. For example, if we use 4 vehicles (Vi, i=1,2,3,4), all possible 2-vehicle combinations are: V1-V2, V1-V3, V1-V4, V2-V3, V2-V4, V3-V4. For every combination we process, the “ExactDP2” algorithm is guaranteed to provide the best solution (in terms of cost if that is what the objective function optimizes), because of the exhaustive search. If the solution provided by this run is different than the one previously calculated, it is recorded as the new best solution. After all the pairs have been processed the procedure starts over. At some point the algorithm will go through a complete processing cycle of all pairs, without finding a better solution for any pair. This indicates that no further improvement is possible, and the algorithm terminates. For this algorithm to work, an initial solution is required. Therefore, the InsertionH algorithm is executed first to provide this solution.

A short verbal description of the heuristic-exact algorithm follows:

First, run the InsertionH algorithm. This produces a feasible solution and the appropriate vehicle routes $(rt_i, i = 1,2,3....|V|)$, where rt_i is the route for vehicle i and |V| is the total number of vehicles. The next step forms all possible pairs between vehicle routes. For example, if our system has 4 vehicles, all possible vehicle route pairs are $rt_1 – rt_2$, $rt_1 – rt_3$, $rt_1 – rt_4$, $rt_2 – rt_3$, $rt_2 – rt_4$, $rt_3 – rt_4$. The DP Exact2 algorithm executed for every pair of vehicle routes. After the execution, pair $rt_1 – rt_2$ is changed. The new routes $rt'_1 – rt'_2$ are different from the original because of inter-route and intra-route interchanges of trips. They are also guaranteed to constitute a better solution, as mentioned before. The next pair is between V1 & V3 vehicles. Now the DP Exact2 algorithm processes the pair $rt'_1 – rt_3$. The process continues until all possible combinations of vehicle routes have been examined. When this first cycle ends, a new cycle of the same vehicle combinations starts. The whole process continues until one cycle of all route combinations has been examined and no optimization has been detected.

A description of the VLSN algorithm in pseudo code is as follows:

```
Step0:  Run InsertionH algorithm for the initial solution
        Calculate TRC
        OldTRC=TRC
Step1:  for every v in V
         RT= RT $\cup$ $VP_{v,v+1}$
Step2:  While Profit > 0
           For every $VP_{v1,v1}$ in RT
                  Run ExactDP2
        Calculate TRC
        Profit = OldRC - TRC
        OldTRC=TRC
```

4.3.4 Neighborhood Definition and Computational Effort

Neighborhood definition for the proposed algorithm is critical as it affects execution time and the solution space under investigation by VLSN algorithm. The following mathematical formulation could be useful for the understanding of the neighborhood size calculation. Given vd trip demands, v vehicles, and m-vehicles partition, the number of neighborhoods for m-vehicles is $\frac{v!}{(v-m)!m!}$ (4-7), while the number of trip exchanges between any m-vehicles partition could be at most n (equal the number of total trip demands). The Total Computational effort for achieving the best solution for the m-vehicles partition can be produced by combining (4-4), (4-7) resulting to $\frac{v!}{(v-m)!m!}(\sum_{L=1}^{m}\sum_{d=1}^{vd}\frac{vd!}{(vd-d)!d!}(d!(d!+\sum_{x=1}^{d-1}\sum_{y=1}^{d-x}\sum_{z=1}^{d-1}(d-z)!)))$ (4-8). In our case for 2 – vehicles partition the size of possible trip assignments for every 2-pairs combination, given a number of vd trip demands is $\frac{v!}{(v-2)!2!}\sum_{L=1}^{2}\sum_{d=1}^{vd}\frac{vd!}{(vd-d)!d!}(d!(d!+\sum_{x=1}^{d-1}\sum_{y=1}^{d-x}\sum_{z=1}^{d-1}(d-z)!))$ (4-9). Since the VLSN algorithm has a specific stopping criterion described in section 4.3.2 as "one cycle of all 2-vehicles permutations without change" we cannot calculate the total computational effort in advance. Practically computational effort is the computational effort described in (4-9) multiplied by the number of cycles for which the solution continues to improve.

A major issue concerning the 2-vehicles VLSN algorithm is the study of the size of the neighborhood area processed by the algorithm in comparison to the size of the entire solution space.

Given a number of v vehicles and vd trip demands the total number of all possible tip demands assignments to all vehicles is given by formula v^{vd}. The number of all possible pair combinations for all vehicles is given by the formula (4-7) where m = 2 resulting formula $\frac{v!}{(v-2)!2!}$ (4-10). For 2 vehicles the number of possible trip demands assignments to be 2^{vd}. By combining (4-10), 2^{vd} the size of neighborhood area is defined as $\frac{v!}{(v-2)!2!}\,2^{vd}$ (4-11). The term 4-11 should be corrected in order not to take into account those trip assignments - where all trips assigned to one vehicle – more than once. After this correction, the final formula that gives the size of neighborhood area is given by the formula $\frac{v!}{(v-2)!2!}2^{vd}-2\frac{v!}{(v-2)!2!}+v$ (4-12). The percentage of neighborhood area size in comparison to total solution area can be calculated by combining (4-12) and (4-7) resulting $\frac{\frac{v!}{(v-2)!2!}2^{vd}-2\frac{v!}{(v-2)!2!}+v}{v^{vd}}$ (4-13). I.e. for v=4, vd=3 the total solution space is 4^3 =64 trip combinations, while the solution space investigated by VLSN algorithm is $\frac{\frac{4!}{(4-2)!2!}2^3-2\frac{4!}{(4-2)!2!}+4=40}{4^3=64}$ which means that the 2-vehicle VLSN algorithm looks at maximum at 62.5% of the total solution space. Following example explains clearly the above mathematical formula.

Given a number of 4 vehicles called A,B,C,D and a number of 3 trip demands named as 1,2,3 the total number of trip assignment combinations for all vehicles, is produced by formula $4^3=64$ (see section 4.2.3). On the other hand the possible trip assignments for each pair of vehicles are:

PAIR A,B	PAIR A,C	PAIR A,D	PAIR B,C	PAIR B,D	PAIR C,D
A123	**A123**	**A123**	**B123**	**B123**	C123
A12-B3	A12-C3	A12-D3	B12-C3	B12-D3	C12-D3
A13-B2	A13-C2	A13-D2	B13-C2	B13-D2	C13-D2
A23-B1	A23-C1	A23-D1	B23-C1	B23-D1	C23-D1
A1-B23	A1-C23	A1-D23	B1-C23	B1-D23	C1-D23
A2-B13	A2-C13	A2-D13	B2-C13	B2-D13	C2-D13
A3-B23	A3-C23	A3-D23	B3-C23	B3-D23	C3-D23
B123	**C123**	**D123**	**C123**	**D123**	D123

The total number of assignment trip combinations is $\frac{4!}{(4-2)!2!}2^3=48$. Trip combinations A123, B123, C123, D123 appeared 3 times each one. In order to guarantee that these trip combinations appear once we need to introduce the term $-2\frac{v!}{(v-2)!2!}+v=8$ where v=4. Finally the total number of unique combinations is 40 and the space investigated by VLSN algorithm is $\frac{40}{64} \approx 0.625$. The remaining (unchecked) combinations are the following 24:

A1B2C3	A2B1C3	A3B1C2	C1D2A3	C2D1A3	C3D1A2
A1B2D3	A2B1D3	A3B1D2	C1D2B3	C2D1B3	C3D1B2
A1B3C2	A2B3C1	A3B2C1	C1D3A2	C2D3A1	C3D2A1
A1B3D2	A2B3D1	A3B2D1	C1D3B2	C2D3B1	C3D2B1

Figure 4-7 indicates clearly the (VLSN Neighborhood area/ Total Solution Space) ratio as function of demands and vehicles number.

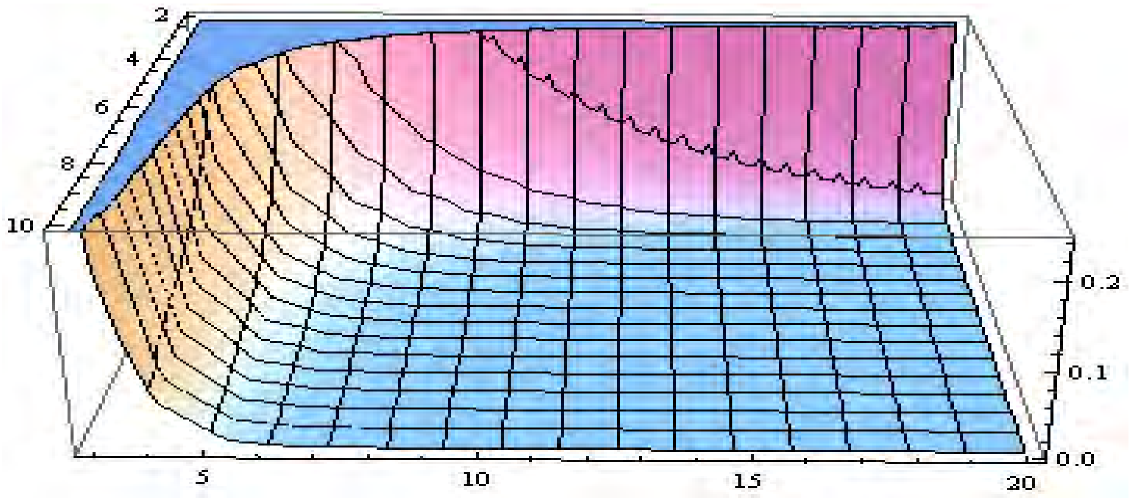


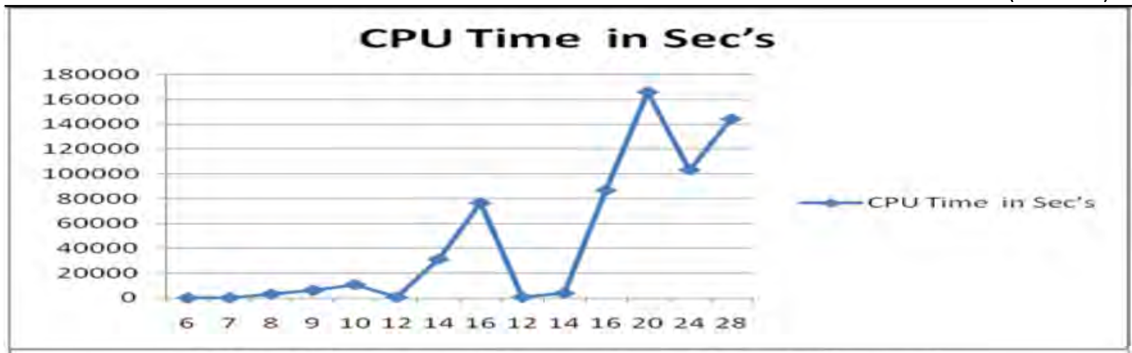
Figure 4-7.VLSN Algorithm (Neighborhood area / Total Solution Space) ratio. x –axis represents the number of trip demands (3 to 20), y-axis represents the number of vehicles (2 to 10), z – axis represents the ratio value

4.3.5 Algorithm computational results

The Proposed algorithm was implemented in C++ and tested on a Linux machine with a dual core processor. A variety of problem configurations where solved in order to test the performance of the algorithm. As mentioned above, the main algorithm stopping criterion is "one combination cycle without improvement". But the "ExactDP2" algorithm which is used as a sub problem suffers in terms of execution time when trip demands are more than 8. That is why we added an additional stopping criterion: the execution time for the exact algorithm can be no more than 7200 seconds. Problem instances were produced randomly. The test field was a square area 100X100 Kilometers. Pickup and deliver positions of every trip request were selected randomly over that area. Vehicle “Start”, “End” depots were selected randomly over that area. Time windows for pickup were randomly selected between 0–1440, time windows for deliveries were produced from corresponding pickups time windows plus 1.5*(Cartesian distance) between the two points. Maximum vehicle ride time was 480. Vehicle speed was 1Km per Minute. Vehicle capacity was 3 seats.

Table 4-4: Computational Results for Small Instances of VLSN implementation

Problem	*Vehicle Number*	*Vehicle Capacity*	*Trip demands*	*CPU Time in Sec’s*	*Cost(Initial Heuristic)*
1	3	4	6	18.28	688.057 (708.166)
2	3	4	7	110.5	539.661 (566.256)
3	3	4	8	3002.22	827.165 (845.014)
4	3	4	9	6108.24	566.066 (578.211)
5	3	4	10	10710.6	843.97 (959.223)
6	5	6	12	567.82	943.145 (1043.34)
7	5	6	14	31139.6	1385.29 (1500.38)
8	5	6	16	76412.8	1588.61 (1774.05)
9	5	8	12	567.82	943.145 (1043.34)
10	5	8	14	3830.84	1321.7 (1425.91)
11	5	8	16	86619.8	1483.95 (1756.4)
12	5	8	20	165650	1692.49 (1988.71)
13	5	8	24	102995	1991.97 (2041.72)
14	5	12	28	144001	2191.74 (2394.34)



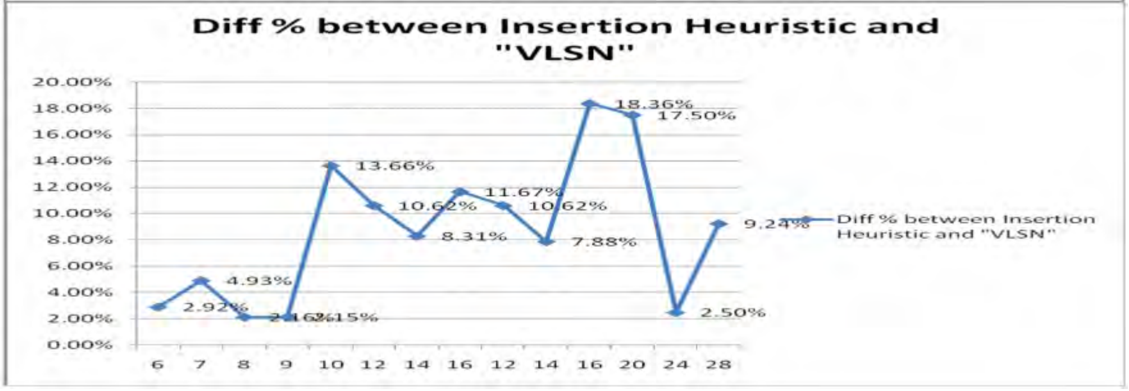


Figure 4-8.VLSN Execution times and the cost differences between the Insertion heuristic and “VLSN” hybrid algorithm

Travel cost is the objective function optimized by the VLSN algorithm. The cost in parentheses is the objective function produced by the InsertionH algorithm. The new heuristic-exact algorithm improves the objective function. For those problem instances that the execution time is longer than 7200sec, the relevant stopping criterion for the DP Exact2 algorithm was activated.

Closeness to the Optimal Solution

Although, there is no methodology that estimates the closeness of any solution produced by a heuristic to the optimal solution, we can provide some computational results that indicate that the solution produced by that algorithm and the solution produced by the exact algorithm are very similar in terms of cost. The problem size is of course limited to small instances because of the limited ability of the exact algorithm to solve large problem sizes. However it is worthwhile to present those results.

Table 4-5: Comparative evaluation of the VLSN with an exact approach

Problem	*Vehicle Number*	*Vehicle Capacity*	*Trip demands*	*Exact Execution Time*	*VLSN Execution Time*	*Objective function Difference*
1	3	3	**5**	1587.03	20.55	0
2	3	3	**5**	1819.85	1.12	0
3	3	3	**5**	1635.13	1.74	0
4	3	3	**5**	1779.86	14.59	6.742
5	3	3	**5**	1183.11	13.29	0
6	3	3	**6**	2038.55	80.6	0
7	3	3	**6**	2064.41	4.42	0
8	3	3	**6**	1336.86	242.57	0
9	3	3	**6**	3133.78	18.58	0
10	3	3	**6**	2049.84	245.66	0

4.4 The Static Regret based heuristic algorithm RegretH

4.4.1 Basic Concept

The Regret based Heuristic algorithm aims to reduce the chances to get trapped in a severely suboptimal solution. It does this by considering the opportunity which is defined as the difference between the cost actually calculated by the algorithm (total distance in our case) and the cost calculated for a better position that could be obtained if a “different” course of action had been chosen. In our case the term “different” course can be defined as the calculation of the best position to which an already assigned demand can be reassigned to produce a better solution in terms of cost. This reassignment could be intra-route or inter-route. The basic idea is to use a fast algorithm in order to quickly get a feasible suboptimal solution. Usually this is an insertion algorithm. The insertion algorithm gives us a myopic sub-optimal solution based on the positions of the previously assigned demands. Then, based on the existing solution, we try to find a better solution by reassigning the most expensive demands. The reallocation which provides the largest gain in the system is chosen. Unlike the tabu-search algorithm, regret algorithm never explores non-feasible solutions. The advantage of the regret algorithm is that it has much less computational burden. The disadvantage is that it can be easily trapped in local minima.

4.4.2 Nomenclature

V = {1,2,3,…..,|V|} = *Set of Vehicles*

R $_{V\{v=1,2,,,,|V|\}}$ = *is the route schedule for every v* $\in$ *V*

| R $_V$***|*** *= size of route schedule defined as the number of demands included in this route*

RC $_{v\{v=1,2,\dots|V|\}}$ = *is the route cost*

TRC = $\sum_{1}^{|V|} RC_v$ *is the total cost for all routes*

DR $_{kv\ \{k=1,2,\dots|Rv|\},\ \{v=1,2,\dots|V|\}}$ = *the assigned **k** demand belongs to route **v***

DMaxR $_{v\ \{v=1,2,\dots|V|\}}$ = *the maximum demand cost concerning route **v***

RCostDMaxR $_{vm\ \{v=1,2,,,,|V|\},\{l=1,2,\dots|V|\},\ v<>l}$ = *is the extra cost incurred in route **m**, if we move the demand (with the maximum cost) that belongs to route **v** to the route **m***

CostMatrix[v,m] $_{\{m=1,2,\dots|V|\},\ \{v=1,2,\dots|V|\}}$ = *is 2-d matrix cost for the Regret Algorithm. Diagonal positions **CostMatrix[v,v]** contains **DMaxR*** $_v$ *while other positions **Cost-Matrix[v,m]*** $_{\{m<>v\}\ \{m=1,2,\dots|V|\},\ \{v=1,2,\dots|V|\}}$ *contains **RCostDMaxR*** $_{vm}$

RegretCostMatrix[v] $_{\{v=1,2,\dots|V|\}}$ ***=*** *is the one dimensional matrix containing the maxi-*

mum value of the difference between the element with the least $\boldsymbol{RCostDMaxR}_{vm\{m=1,2,\ldots|V|\}}$ and the corresponding $\boldsymbol{DMaxR}_{v}$.

4.4.3 Algorithm Description

The main objective is the minimization of the total cost for all vehicles. Our regret algorithm uses the original regret concept where the absolute difference between the "best" lower cost and the second "best" lower cost alternative is used as a metric for guiding problem solving.

Our regret algorithm uses a fast heuristic in order to produce an initial solution. The fast heuristic we use is the InsertionH heuristic mentioned in section 4.1. Then we start the regret process. We build the regret matrix.

Each row represents the route produced by the initial heuristic.

Each diagonal item in the 2-d matrix represents the cost of the most expensive demand of the route described by the corresponding row.

Each element of a column (except the one on the diagonal) represents the cost that will be produced if we insert the aforementioned most expensive demand to the route represented by the corresponding row.

If the insertion is not feasible, then we set the insertion cost to an arbitrarily large number.

By using this matrix we define the "profit" we gain if we move demands from one route to another. This can be done by using the following rule: for every column calculate the differences of diagonal element minus every other element. If at least one difference is positive select the greatest one. Then move the corresponding demand to the appropriate row (route in our case)

The Algorithm pseudo code follows

```
Step0:Run the InsertionH Algorithm
    Calculate TRC
    NewTRC= TRC
Step 1:While (TRC- NewTRC>0)
    TRC= NewTRC
    Step 1.1: for every v in V
              for every $DR_{kv\ \{k=1,2,\dots|RV|\}}$
                        find $DMaxR_v$
                        $CostMatrix[v,v] = DMaxR_v$
    Step 1.2: for every $DMaxR_{v\ \{v=1,2,\dots|V|\}}$
              for every m in V
                        Find $RCostDMaxR_{v,m\ \{v=1,2,,,,|V|\},\{m=1,2,\dots|V|\},\ v<>m}$
                        $CostMatrix[v,m]= RCostDMaxR_{vm}$
    Stap 1.3: for every v in V
                        $RegretCostMatrix[v]= positive\ Max(DMaxR_v - CostMatrix[v,m])$
              $_{\{m=1,2,\dots|V|\}}$ the greatest  positive difference.
                        Move demand $DMaxR_v$ from route v to route m
    Step 1.4: Calcuate new TRC
```

Computational effort of the RegretH, given n demands, is the sum of the computational effort of:

1. InsertionH algorithm used in the initial phase. This computational effort has already been calculated as $O(n^3)$.
2. The procedure that gives the most expensive demand of any route. This computational effort is obviously $O(n)$. This procedure removes temporarily one by one every demand and then calculates the solution cost produced in the absence of this demand. The difference between the previous solution cost and current solution cost is the cost of every demand in the solution.
3. The procedure that gives the additional cost that will be produced if we insert the most expensive demand of a route to another route. Consider a feasible solution produced by the InsertionH algorithm for *n* trip demands and *m* vehicle routes. The computational effort for this solution is the sum of all possible searches for the best point to insert the most expensive trip demand of every vehicle route to another vehicle routes. The number of possible searches is given by formula

$\sum_{k=1}^{2(n-1)+1} k = (2n^2 - n)$ (4-14). For all m vehicle routes the number of possible searches – the worst case- is $m(2n^2 - n)$ (4-15). Consequently the computational effort is O(mn^2). Figure 4-8 clearly presents the computational effort as a function of number of demands.

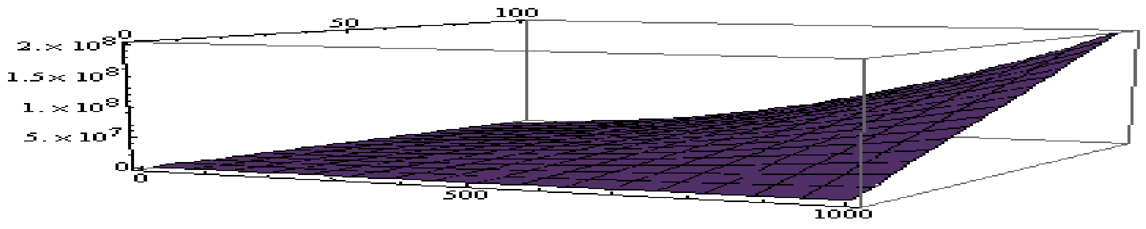


Figure 4-9.Computational effort of the Regret Procedures. (x-axis describes the number of trip demands(1-1000), y-axis (1-100) describes the number of vehicle routes, z-axis is the comutational effort(0- 2E-8)

The two aforementioned procedures are executed repetitively while the solution is optimized. The number of repetitions is unknown but larger than one. Taking *r*, the number of repetitions, the final computational load is O(n^3)+rO(mn^2) (4-16). The order of *r* as a function of *m,n* wasn't calculated and is therefore considered unknown. Figure 4-9 presents the total computational effort as a function of number of repetitions and number of demands. We see that the main factor affecting the computational effort is the number of demands.

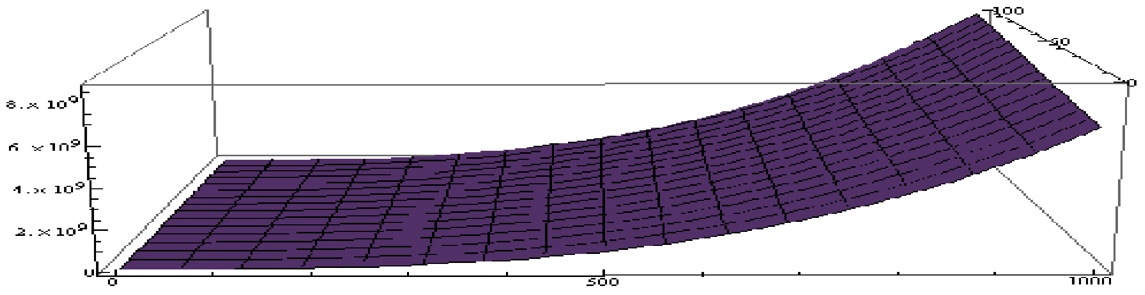


Figure 4-10.Computational effort of the RegretH algorithm. (x-axis describes the number of trip demands(1-1000), y-axis (1-100) describes the number of regret repetitions , z-axis is the comutational effort(0- 8E-9)

4.4.4 Algoritm computational results

The Proposed algorithm was implemented in C++ and tested on a Linux machine with a dual core processor. We tested our regret algorithm by using two different sets of data. The first set was produced artificially while the second was compiled from data collected from a real life application in the municipality of Philippi in northern Greece, where a DRT system has been established and tested for 45 days.

Artificial Data

Problem instances were produced randomly. The test field was a square area 100X100 Kilometers. Vehicle capacity was 8,12,and 16. The number of passengers for each trip demand was selected randomly between 4,6 or 8, depending on the maximum vehicle capacity. Pickup and deliver positions of every trip request were selected randomly over that area. Vehicle “Start”, “End” depots were selected randomly over that area. Time windows for pickup were 15 minutes and the earlier pickup time was randomly selected between 0 and 1440. Time windows for delivers were produced from pickup time windows plus 1.5*(Cartesian distance) between pickup and deliver. Maximum vehicle ride time was 1440. Vehicle speed was 1Km/minute

Real data

Problem instances were selected from the data collected during the pilot application in the municipality of Philippi. Road network consists of 33 nodes. The maximum number of passengers was 4 persons and the time window 15 minutes. The maximum ride time was 1.5 times the absolute shortest path time for that specific distance.

The following tables and graphs present the results of the algorithm's application on real and artificial data. In addition there is a comparison of the algorithms performance when applied to artificial data versus its performance when applied to real data. Tables 4-6 to 4-11 have the following structure:

The 1st column represents the number of the experiment

The 2nd column represents the number of requested demands

The 3rd column represents the number of vehicles used to solve the problem.

The 4th column represents the satisfied demands among the requested demands.

The 5th column represents the average number of customers per KM.

The 6th column represents the percentage of distance deviation.

The 7th column represents the percentage of time deviation.

The 8th column represents the solution cost. (In our case, "cost" was the total distance covered by all vehicles).

The 9th column represents the required time in order to get the solution

The 10th represents the total improvement over the initial solution after the application of the regret algorithm.

The 11th column represents the required time in order to get the improved solution by using regret algorithm.

The rest of the tables' and graphs' structures are obvious.

Table 4-6: Static Regret algorithm results for artificial data - 8 Seat Veh.

# of Exper	*Dem Reques*	*Vehicles*	*Demands*	*Avg Pass per KM*	*Avg Distance Deviation %*	*Avg Time Deviation %*	*Cost(in KM)*	*Time(Sec's)*	*Impv Perecentage %*	*Regret Time*
1	100	40	88	0.891418	1	23	7078.08	1.01	1.27166	6.36
2	250	48	224	0.916597	4	26	16974.9	8.77	1.63156	92.44
3	500	50	427	0.970287	4	16	23631.6	43.6	0.367968	98.59
4	750	50	628	0.994035	5	32	38891.6	124.53	0.366966	158.21
5	1000	50	700	1.10418	7	33	42371	213.33	0.39065	181.65
6	1250	50	780	1.17288	6	33	43413.8	346.66	0	45.93
7	1500	50	811	1.20192	7	35	44224.4	455.74	0	51.63
8	1750	50	836	1.25347	7	35	44072.6	593.76	0	59.44
9	2000	50	851	1.36185	8	36	43867	706.9	0	61.65

Table 4-7: Static Regret algorithm results for artificial data - 12 Seat Veh.

# of Exper	*Dem Reques*	*Vehicles*	*Demands*	*Avg Pass per KM*	*Avg Distance Deviation %*	*Avg Time Deviation %*	*Cost(In KM)*	*Time(Sec's)*	*Impv Perecentage %*	*Regret Time*
1	100	33	82	0.891335	2	24	6167.62	0.94	0.161993	2
2	250	49	225	0.917287	4	27	17171.9	6.92	3.44666	8
3	500	50	430	0.945412	4	29	30428.8	41.56	1.47551	185.9
4	750	50	623	0.949743	4	31	39408.1	120.23	0.469661	17
5	1000	50	727	1.06374	5	33	41560.1	231.86	0	41.2
6	1250	50	743	1.16695	7	34	43190.2	310.26	0	43.1
7	1500	50	803	1.22041	8	35	44236.9	456.5	0	50.8
8	1750	50	832	1.3179	8	36	44248.9	607.56	0.110481	117.6
9	2000	50	858	1.3725	8	36	44216.7	921.14	0	58.3

Table 4-8: Static Regret algorithm results for artificial data - 16 Seat Veh.

# of Exper	*Dem Reques*	*Vehicles*	*Demands*	*Avg Pass per KM*	*Avg Distance Deviation %*	*Avg Time Deviation %*	*Cost(in KM)*	*Time(Sec's)*	*Impv Perecentage %*	*Regret Time*
1	100	33	89	0.898842	3	23	7233.99	1.12	1.80876	7.3
2	250	44	227	0.902318	3	26	16585	8.64	2.32	57
3	500	50	435	0.984404	5	28	31148.1	42.17	1.27272	234
4	750	50	626	1.01759	5	31	39143.4	124.32	0.221305	83.8
5	1000	50	713	1.15315	7	34	42432.5	232.86	0.28181	135.9
6	1250	50	769	1.22487	8	35	44222.7	329.54	0.0931177	119.5
7	1500	50	816	1.21443	7	34	43935.3	475	0	50.7
8	1750	50	848	1.30354	8	35	44310.5	604.08	0	61.7
9	2000	50	866	1.35842	9	36	44100.4	761.93	0	61.6

Table 4-9: Static Regret algorithm results for real data - 8 Seat Veh.

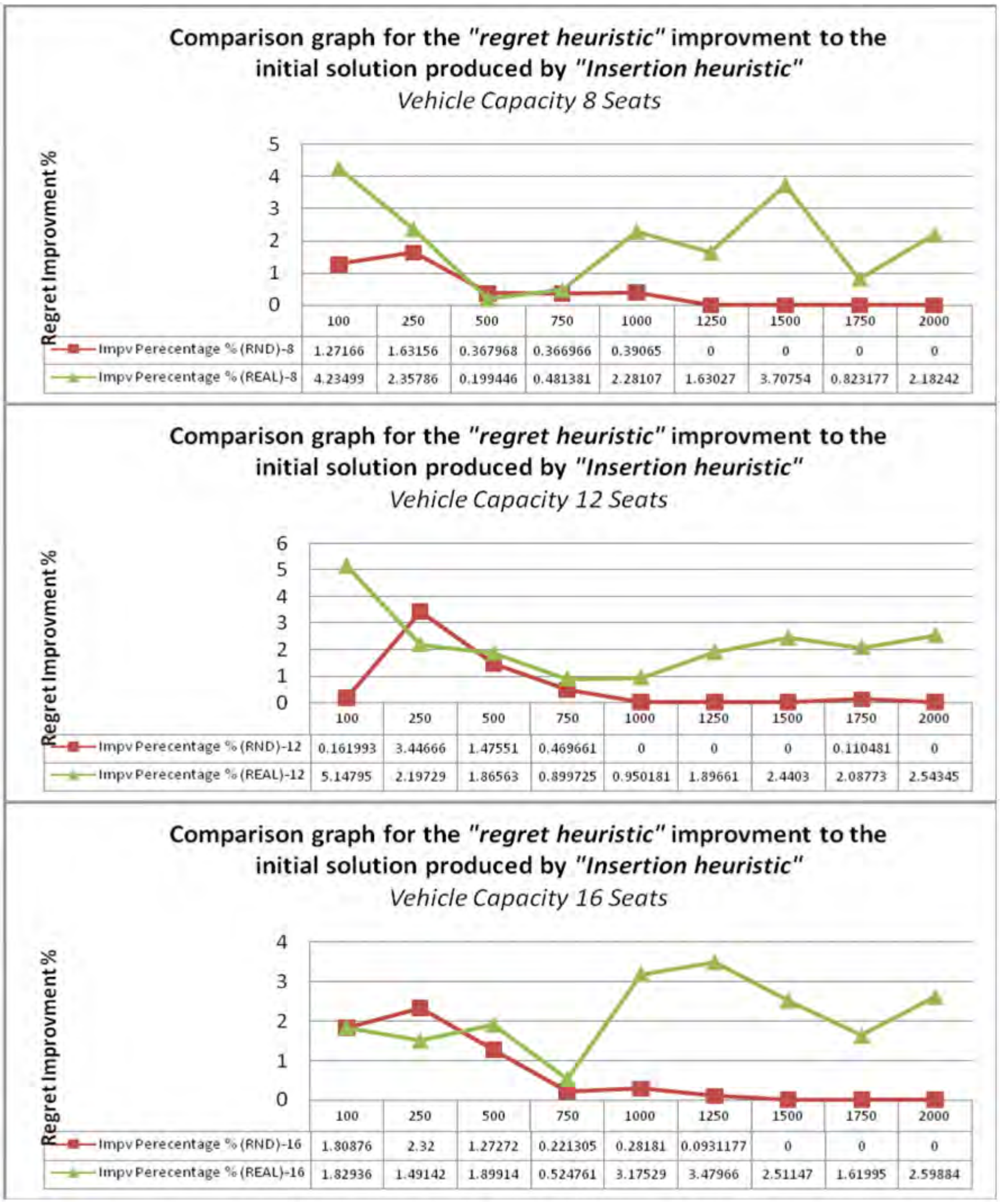
# of Exper	*Dem Reques*	*Vehicles*	*Demands*	*Avg Pass per KM*	*Avg Distance Deviation %*	*Avg Time Deviation %*	*Cost(In meters)*	*Time(Sec's)*	*Impv Perecentage %*	*Regret Time*
1	100	7	94	1.4106	1	1	701300	3.57	4.23499	5.1
2	250	10	226	1.696	0	1	1.20E+06	40.91	2.35786	50.0
3	500	22	415	2.2156	1	2	1.81E+06	146.9	0.19945	15
4	750	27	624	2.4134	1	2	2.51E+06	345.23	0.48138	294.8
5	1000	30	803	2.4475	1	2	3.02E+06	633.17	2.28107	794.5
6	1250	35	949	2.4774	1	1	3.58E+06	945.11	1.63027	1047.6
7	1500	39	1151	2.4338	1	2	4.34E+06	1653.97	3.70754	2722.2
8	1750	44	1370	2.365	2	3	5.28E+06	2170.99	0.82318	2686
9	2000	47	1619	2.4839	2	3	5.93E+06	3137.05	2.18242	3265.3

Table 4-10: Static Regret algorithm results for real data - 12 Seat Veh.

# of Exper	*Dem Reques*	*Vehicles*	*Demands*	*Avg Pass per KM*	*Avg Distance Deviation %*	*Avg Time Deviation %*	*Cost(In meters)*	*Time(Sec's)*	*Impv Perecentage %*	*Regret Time*
1	100	7	94	1.5387	1	1	662400	5.43	5.14795	6.9
2	250	8	226	1.846	0	1	1.07E+06	66.64	2.19729	63.1
3	500	16	415	2.3952	1	2	1.64E+06	205.22	1.86563	187.3
4	750	20	624	2.5544	1	3	2.30E+06	752.25	0.89973	375.2
5	1000	22	803	2.7643	1	2	2.64E+06	1268.09	0.95018	847.7
6	1250	26	949	2.7768	1	2	3.15E+06	1980.49	1.89661	2043.2
7	1500	28	1151	2.8621	1	3	3.64E+06	2894.08	2.4403	3375.6
8	1750	31	1370	2.8553	2	3	4.27E+06	4228.65	2.08773	4202
9	2000	29	1619	2.9792	2	4	4.78E+06	6912.65	2.54345	4462.9

Table 4-11: Regret algorithm results for real data - 16 Seat Veh.

# of Exper	*Dem Reques*	*Vehicles*	*Demands*	*Avg Pass per KM*	*Avg Distance Deviation %*	*Avg Time Deviation %*	*Cost(In meters)*	*Time(Sec's)*	*Impv Perecentage %*	*Regret Time*
1	100	6	94	1.5708	1	1	634100	6.1	1.82936	4.0
2	250	8	226	1.8775	0	1	1.07E+06	87.34	1.49142	39.2
3	500	16	415	2.5276	2	3	1.54E+06	304.22	1.89914	408.0
4	750	17	624	2.6921	1	3	2.13E+06	904.19	0.52476	272.1
5	1000	19	803	2.8659	1	2	2.52E+06	1700.63	3.17529	1369.3
6	1250	23	949	3.0272	1	2	2.86E+06	2885.84	3.47966	3285.9
7	1500	23	1151	2.9933	1	2	3.40E+06	4939.14	2.51147	4130
8	1750	26	1370	3.0433	2	3	3.98E+06	6243.56	1.61995	5989.4
9	2000	27	1619	3.2228	2	4	4.42E+06	9817.08	2.59884	7437.0



	100	250	500	750	1000	1250	1500	1750	2000
Impv Perecentage % (RND)-8	1.27166	1.63156	0.367968	0.366966	0.39065	0	0	0	0
Impv Perecentage % (REAL)-8	4.23499	2.35786	0.199446	0.481381	2.28107	1.63027	3.70754	0.823177	2.18242

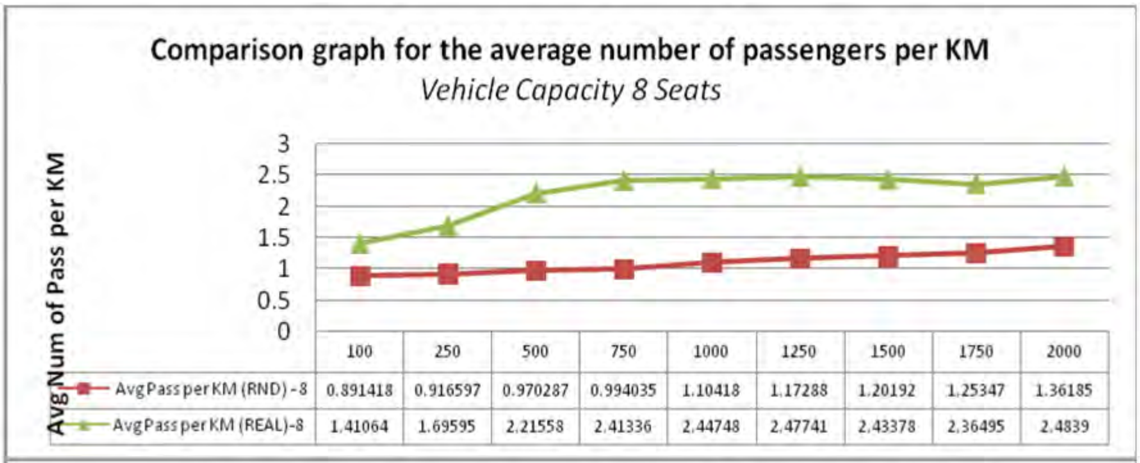
	100	250	500	750	1000	1250	1500	1750	2000
Impv Perecentage % (RND)-12	0.161993	3.44666	1.47551	0.469661	0	0	0	0.110481	0
Impv Perecentage % (REAL)-12	5.14795	2.19729	1.86563	0.899725	0.950181	1.89661	2.4403	2.08773	2.54345

	100	250	500	750	1000	1250	1500	1750	2000
Impv Perecentage % (RND)-16	1.80876	2.32	1.27272	0.221305	0.28181	0.0931177	0	0	0
Impv Percentage % (REAL)-16	1.82936	1.49142	1.89914	0.524761	3.17529	3.47966	2.51147	1.61995	2.59884

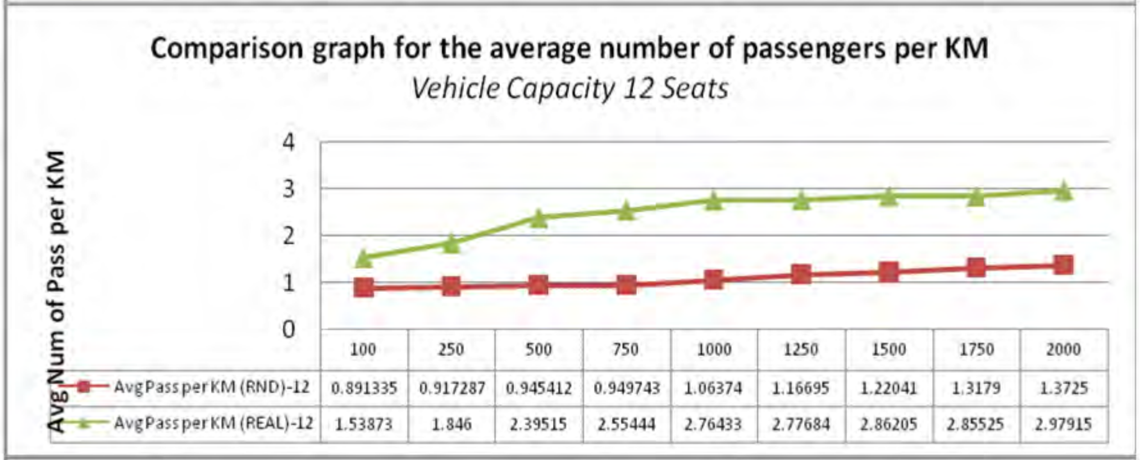
Figure 4-11.Comparison graphs concerning the improvement due to regret optimization application

Table 4-12: Comparison of the improvements after the application of the regret optimization

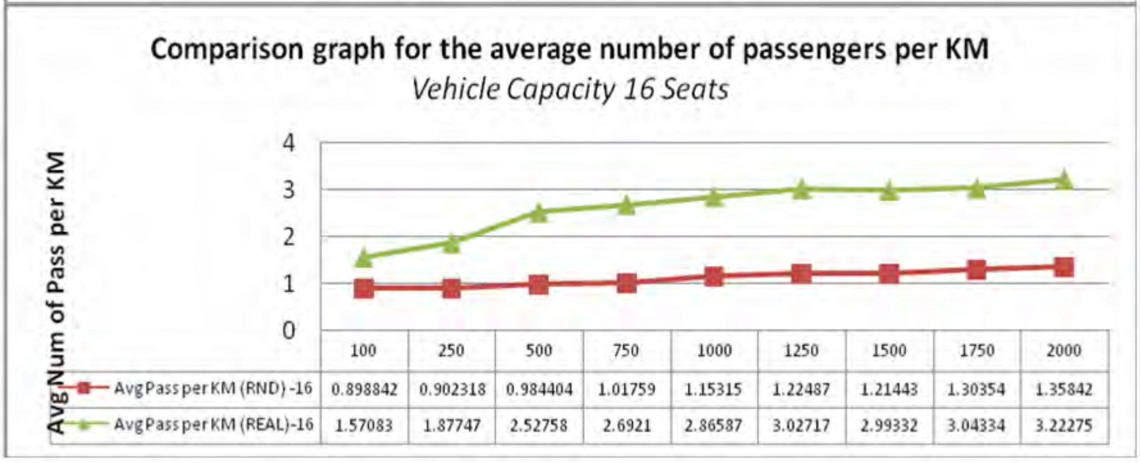
Capacity 8	*Dem Reques*		
# of Expert		**Improvement Percentage % (Artificial Data set)**	**Improvement Percentage % (Real Data set)**
1	100	1.27166	4.23499
2	250	1.63156	2.35786
3	500	0.367968	0.199446
4	750	0.366966	0.481381
5	1000	0.39065	2.28107
6	1250	0	1.63027
7	1500	0	3.70754
8	1750	0	0.823177
9	2000	0	2.18242
Capacity 12			
1	100	0.161993	5.14795
2	250	3.44666	2.19729
3	500	1.47551	1.86563
4	750	0.469661	0.899725
5	1000	0	0.950181
6	1250	0	1.89661
7	1500	0	2.4403
8	1750	0.110481	2.08773
9	2000	0	2.54345
Capacity 16			
1	100	1.80876	1.82936
2	250	2.32	1.49142
3	500	1.27272	1.89914
4	750	0.221305	0.524761
5	1000	0.28181	3.17529
6	1250	0.0931177	3.47966
7	1500	0	2.51147
8	1750	0	1.61995
9	2000	0	2.59884



	100	250	500	750	1000	1250	1500	1750	2000
🟥 AvgPass per KM (RND) -8	0.891418	0.916597	0.970287	0.994035	1.10418	1.17288	1.20192	1.25347	1.36185
▲ AvgPass per KM (REAL)-8	1.41064	1.69595	2.21558	2.41336	2.44748	2.47741	2.43378	2.36495	2.4839

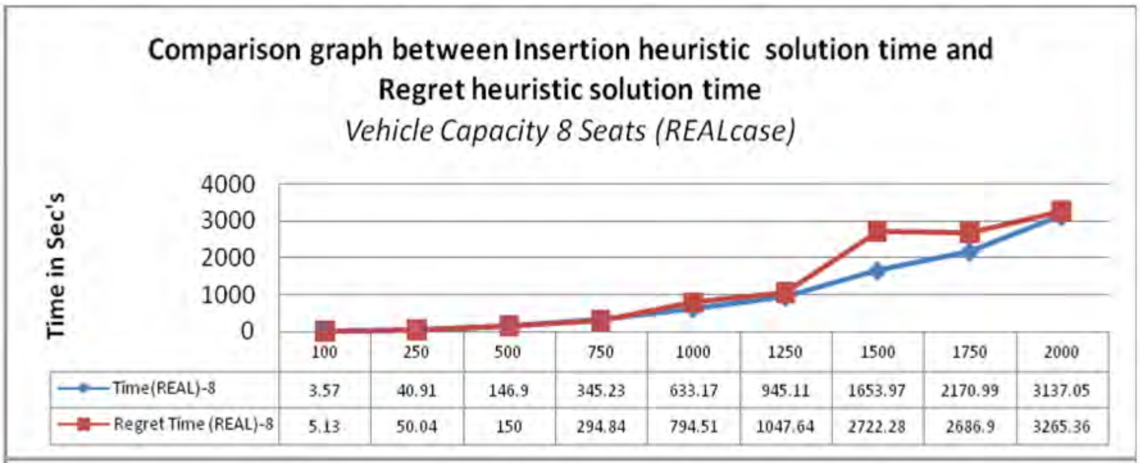


	100	250	500	750	1000	1250	1500	1750	2000
🟥 AvgPass per KM (RND)-12	0.891335	0.917287	0.945412	0.949743	1.06374	1.16695	1.22041	1.3179	1.3725
▲ AvgPass per KM (REAL)-12	1.53873	1.846	2.39515	2.55444	2.76433	2.77684	2.86205	2.85525	2.97915

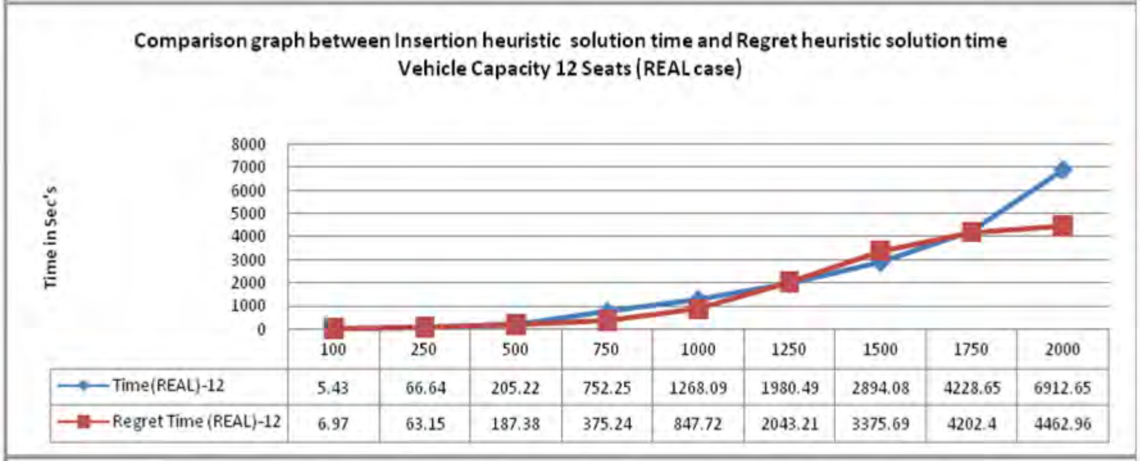


	100	250	500	750	1000	1250	1500	1750	2000
🟥 AvgPass per KM (RND) -16	0.898842	0.902318	0.984404	1.01759	1.15315	1.22487	1.21443	1.30354	1.35842
▲ AvgPass per KM (REAL)-16	1.57083	1.87747	2.52758	2.6921	2.86587	3.02717	2.99332	3.04334	3.22275

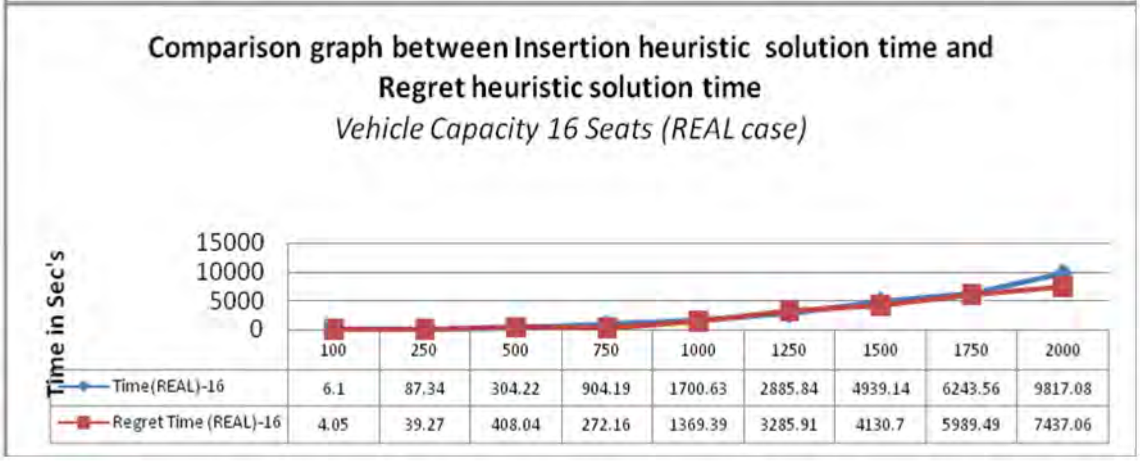
Figure 4-12. Comparison graphs concerning the average number of passenger per Km for artificial and real Data.



	100	250	500	750	1000	1250	1500	1750	2000
Time(REAL)-8	3.57	40.91	146.9	345.23	633.17	945.11	1653.97	2170.99	3137.05
Regret Time (REAL)-8	5.13	50.04	150	294.84	794.51	1047.64	2722.28	2686.9	3265.36

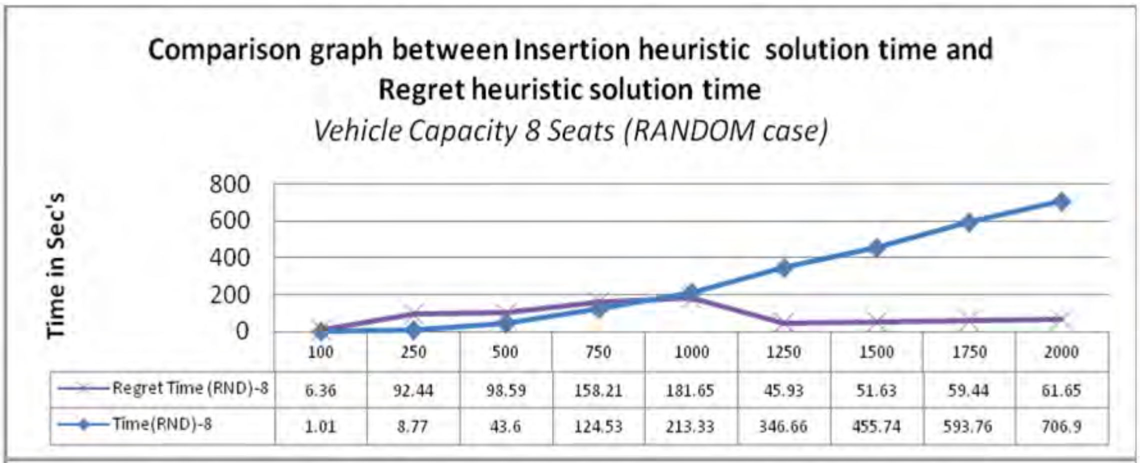


	100	250	500	750	1000	1250	1500	1750	2000
Time(REAL)-12	5.43	66.64	205.22	752.25	1268.09	1980.49	2894.08	4228.65	6912.65
Regret Time (REAL)-12	6.97	63.15	187.38	375.24	847.72	2043.21	3375.69	4202.4	4462.96

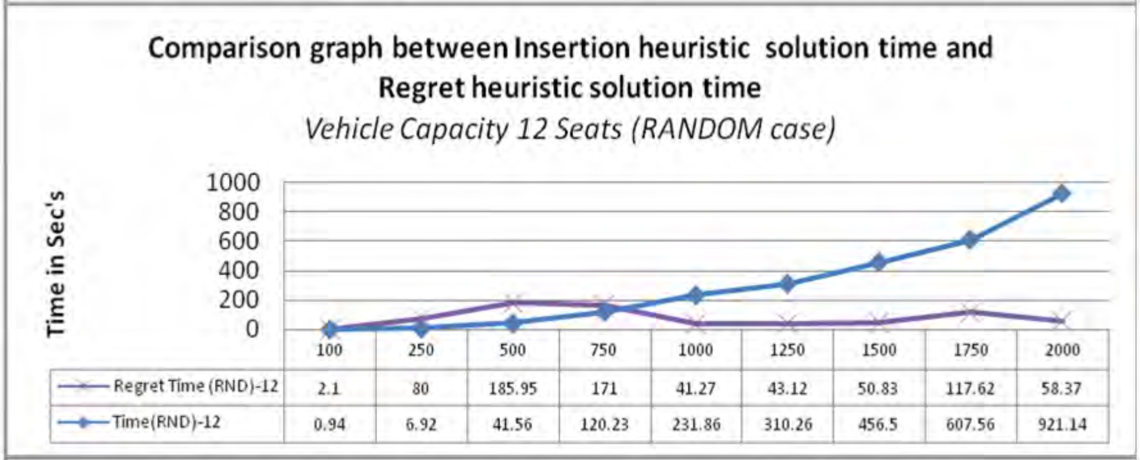


	100	250	500	750	1000	1250	1500	1750	2000
Time(REAL)-16	6.1	87.34	304.22	904.19	1700.63	2885.84	4939.14	6243.56	9817.08
Regret Time (REAL)-16	4.05	39.27	408.04	272.16	1369.39	3285.91	4130.7	5989.49	7437.06

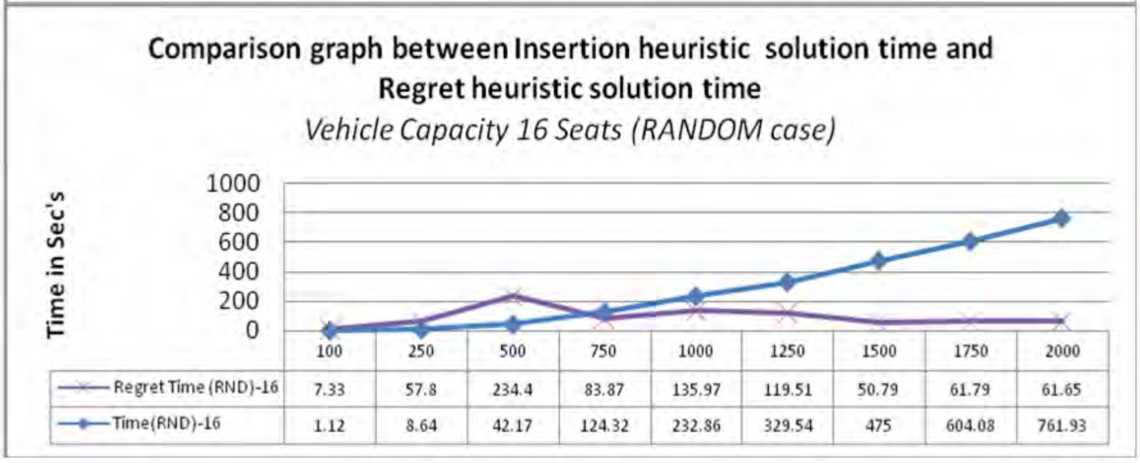
Figure 4-13.Comparison graphs concerning execution times for insertion and regret heuristics for real data



	100	250	500	750	1000	1250	1500	1750	2000
Regret Time (RND)-8	6.36	92.44	98.59	158.21	181.65	45.93	51.63	59.44	61.65
Time(RND)-8	1.01	8.77	43.6	124.53	213.33	346.66	455.74	593.76	706.9



	100	250	500	750	1000	1250	1500	1750	2000
Regret Time (RND)-12	2.1	80	185.95	171	41.27	43.12	50.83	117.62	58.37
Time(RND)-12	0.94	6.92	41.56	120.23	231.86	310.26	456.5	607.56	921.14



	100	250	500	750	1000	1250	1500	1750	2000
Regret Time (RND)-16	7.33	57.8	234.4	83.87	135.97	119.51	50.79	61.79	61.65
Time(RND)-16	1.12	8.64	42.17	124.32	232.86	329.54	475	604.08	761.93

Figure 4-14.Comparison graphs concerning execution times for insertion and regret heuristics for artificial data

4.4.5 RegretH Algorithm analysis and conclusions

The above figures show that the regret algorithm improves the solution more effectively when it is applied on real data than on random artificial data. The explanation could be because real data follows a specific pattern in terms of the type of demands (desired pickup and delivery points , desired travel time etc.) Given table 4-10 we found that :

- The minimum improvement percentage is 0 and it occurs only to those problem instances regarding artificial data.
- The maximum improvement percentage is **5.14795%** and it occurs for a real probem instance.
- All real problem instances, have an improvement ratio larger than 0.
- There is no recognizeable pattern in the way the regret algorithm improves the initial solution.
- Execution times (see figure 4-12,4-13) are comparable for the insertion algorithm and the regret algorithm. That means the regret algorithm is an acceptable option to improve our solution. The maximum solution time for the regret algorithm for the largest problem instance was almost 2 hours.

4.4.6 Algorithm applications

The heuristic regret algorithm is used as sub module for the online algorithms. Online algorithms need fast optimization modules to produce good solutions during idle times. The results in section 4.1.4 indicate that we can use this algorithm because of its short execution time.

Chapter 5 : Online Dial-a-Ride Algorithms

5.1 Online Regret Dial-a-Ride Algorithm(OR-DARP)

5.1.1 Basic Concept

Online algorithms by design are used to handle online demands. Online demand is the demand realized during the course of operation of a system. The proposed online algorithm has been developed to satisfy the following key features.

- **Fast response** concerning the acceptance or denial of the incoming demand. An online algorithm is used mainly for real applications. Prompt response is very critical to such applications. This is the main reason, that the main online algorithms give an answer as soon as possible. All trip demands are considered as urgent demands in order to be scheduled immediately
- **Continuous improvement** of the solution quality. It means that idle intervals at any time can be used by the optimization process.

5.1.2 Nomenclature

V = {1,2,3, ,|V|} = Set of Vehicles

EPT$_i$ = is the earliest pickup time of the *i-th* online demand

t$_c$ = is the current clock time as the online algorithm runs

InsertionH (t$_c$) = is the online regret sub-algorithm (fast insertion) without optimization. The *InsertionH (t$_c$)* algorithm searches only those positions that belong to assigned demands where $EPT_i > t_c$

RegretH (t$_c$) = is the one time step online regret sub-algorithm with optimization. The *RegretH (t$_c$)* algorithm searches those positions that belong to assigned demands where $EPT_i > t_c$

RS(t$_c$) = is the problem solution produced by the online regret algorithm with optimization at the time ***t$_c$***

TR(t$_c$) = is the new trip request that occurred at time t_c

rt = the real clock time

TDem = total number of online *trip demands*

$TdemStamp_d$ = is the timestamp of the *d-th* online trip demand where (*d=1,2,,TDem*)

DemQt* =** is the -online trip demands- queue at real clock time ***rt

***QueuingTime* =** is the queuing time that an online trip demand is allowed to wait for processing

5.1.3 Algorithm Description

The online regret algorithm is an online algorithm that handles online demands quickly, while optimizing continuously the produced solution. To achieve this goal, we developed two sub modules combined in the general regret online dial-a-ride algorithm. We describe the online regret sub-algorithm without optimization as an algorithm that uses the InsertionH algorithm to give an initial fast response for any incoming trip demand. The regret optimization works in the background to optimize solutions for all served demands.

A more descriptive section of the online regret algorithm follows:

The algorithm reads an initial input and uses the fast insertion heuristic to produce an initial solution. Dispatcher (described in section 5.1.4) is continuously monitoring the system and feeds the online algorithm with events. If there is no event, then the optimization sub-module performs a one-step optimization. If on completion there are still no new events, optimization continues for another step. If the optimization is completed (the regret algorithm produces no more optimization in a full cycle) the system goes to idle state while the only activity that continues is the dispatcher looking for new events.

In case of a new demand the fast insertion heuristic is called up in order to provide a solution. Fast insertion solution provides a solution within no more than a maximum threshold time (maxRT). The size of this predetermined threshold time is about the maximum acceptable system response time. The algorithm calls the regret module for one step optimization and the process continues till no further improvement can be achieved.

The most important feature of the online regret algorithm is that the algorithm has no need for time horizons. Instead it uses every single time unit i.e. one second in order to optimize the solution. The main features of the developed online regret algorithm are:

1. It handles every trip request in time no more than $maxRT$.
2. It runs the optimization continuously in single steps.

A description of the regret online algorithm in pseudo code is as follows:

```
Step0:  For each vehicle v (v=1,2,3…..|V|)
        Build an empty route
        Step 1: Define $maxRT$
        Step2: For each  time $t_c$ reading
            Step 2.1. Read Dispatcher input ($t_c$)
            Step 2.2 If there is a TR($t_c$)
                        Call InsertionH ($t_c$)
                     Else if RS($t_c$) can be optimized more
                        Call RegretH($t_c$)
```

5.1.4 Dispatcher Module for the OR-DARP Algorithm

In order to test online algorithms there is a need for a dispatcher simulator. The main task of the dispatcher simulator is to feed online algorithms with the appropriate demands. The dispatcher simulator software should imitate the basic features of a live operator. The operation of the developed software dispatcher can be described as follows:

At the first step the dispatcher loads into memory all online trips demands and sorts them in ascending order of the time of their appearance

The second step is to initialize the time counter and start feeding the online algorithm with the online demands when the appearance time of that demand is equal (or nearly equal) to the current timer.

The main feature of the software dispatcher is the queuing time. By the term queuing time we mean the maximum time that the online requested trip demand is allowed to wait before being processed or rejected. A very tight queuing time decreases the queue but rejects more trip demands. On the other hand a more relaxed queuing time could affect the users inconvenience due to long waiting times. The dispatcher algorithm works under three conditions concerning the response of the online algorithm

Condition 1: HARD_WORKING_STATE. When online algorithm signals dispatcher with this state, it means that the online algorithm executes an insertion of another trip request that came earlier. Or it executes a critical step of the optimization procedure, such as the exchange of demands among the routes. In both cases the running of the algorithm cannot be interrupted

Condition 2: SOFT_WORKING_STATE. When online algorithm signals dispatcher with this state. It means that the online algorithm executes a non critical step of the optimization procedure and it can be interrupted at the time a new online trip request shows up.

Condition 3: IDLE_STATE. Idle state means that the online algorithm executes neither insertion nor optimization and just waits for new trip demands.

Those conditions provide critical information to the dispatcher in relation to operational behavior

The software dispatcher pseudo code is as follows:

```
Sort all online demands in ascending order of their timestamp
Initialize rt
 Read current time rt
While rt < max (TdemStamp_d (d=1,2,,Tdem)) do
    Update DemQt
    If online algorithm signals IDLE_STATE or SOFT_WORKING_STATE
        If current demand in the DemQt satisfies rt - TdemStamp_d < QueuingTime and > 0
            a.  Send to the online algorithm the current demand of the DemQt
            b.  Remove current demand from the DemQt
        Else
          Do Nothing
        End if
    Else If online algorithm signals HARD_WORKING_STATE
     Do nothing
    End if
    Read rt
End while
```

5.1.5 Online Algorithm performance

For a specific online algorithm named OLALG - that serves all incoming demands- given:

- The current solution *S(n)* consists of *n* already served demands
- The operational time *T*
- The processing time for a new trip request named *PT(S(y+1))* ,where y represents the number of the already served demands

The maximum number *x* of demands that can be processed by OALG is given by solving the equation $T - \sum_{i=0}^{x-1} PT(S((n+i)+1)) \cong 0$ (4-17).

5.1.6 Algorithm computational study

The proposed algorithm and the software dispatcher were implemented in C++ and tested on a Linux machine with a dual core processor. In order to evaluate online regret algorithm performance we have designed a sequence of experiments based on real data gathered during a pilot testing in the municipality of Philippi. For every trip demand we have the basic features such as: “earlier and late pickup time, earlier and late pickup time, origin and destination, maximum travel time, quantity concerning the number of persons, special trip requirements” as well as the timestamp. Timestamp is the time that the trip request came to the operator for processing.

We made 9 different scenarios each of different size with regard to the number of requested trip demands (94,226,414,624,803,949,1151,1370,1619). Some details about the environment of the experiments are:

1. *Maximum fleet size 50 vehicles.*
2. *Vehicle Capacity 12 seats.*
3. *The Maximum time that an online trip request service is allowed to wait until processed or rejected, was 60 seconds. After that, the dispatcher proceeds to next demand.*
4. *Maximum pickup wait time 15 minutes*
5. *Maximum trip demand ride time 1.5 time the absolute shortest path time*
6. *The cost per distance units (Km in our case) was 0.32 currency units*
7. *The charging price per distance unit (Km in our case) was 0.32 currency units*
8. *Work period was one day (or 86400 seconds)*

Tables 5-1 to 5-7 present the basic behavior of the algorithm. A series of comparative results examine the performance of this online algorithm in comparison with its offline version.

Table 5.1 presents results concerning the number of served trip demands vs the number of requested trip demands.

The 2nd column represents the number of vehicles used in order to serve trip demands

The 3rd column represents the number of requested trip demands.

The 4th column represents the number of served trip demands

The 5th column represents the percentage difference between requested and served trip demands

The 6th column represents the average number of customers per Km.

The 7th column represents the average percentage of distance deviation

The 8th column represents the average percentage of time deviation

Table 5-1: Online Regret Algorithm test results for 1-day (86440 sec's) operation period

#of expert	*Vehicles (vehicle capacity 12)*	*Trip demands*	*Online trip demands Assigned*	*Dems Diff %*	*Avg Pass per KM*	*Avg Distance Deviation %*	*Avg Time Deviation %*
1	7	94	94	0.00%	1.40658	1%	1%
2	9	226	226	0.00%	1.74777	0%	1%
3	16	415	414	-0.24%	2.27061	1%	1%
4	22	624	622	-0.32%	2.43832	1%	2%
5	25	803	735	-8.47%	2.57096	1%	2%
6	24	949	727	-23.39%	2.51445	0%	1%
7	28	1151	873	-24.15%	2.46122	1%	2%
8	27	1370	890	-35.04%	2.40471	1%	3%
9	28	1619	1012	-37.49%	2.59163	2%	5%

Table 5-1 shows that the online regret algorithm reaches a critical limit after experiment #4 where the number of served demands starts to decrease monotonically in comparison with the requested trip demands. In the last experiment we found that the number of serviced demands is 37.5% less than the requested demands. This behavior is presented graphically in the figure 5-1.

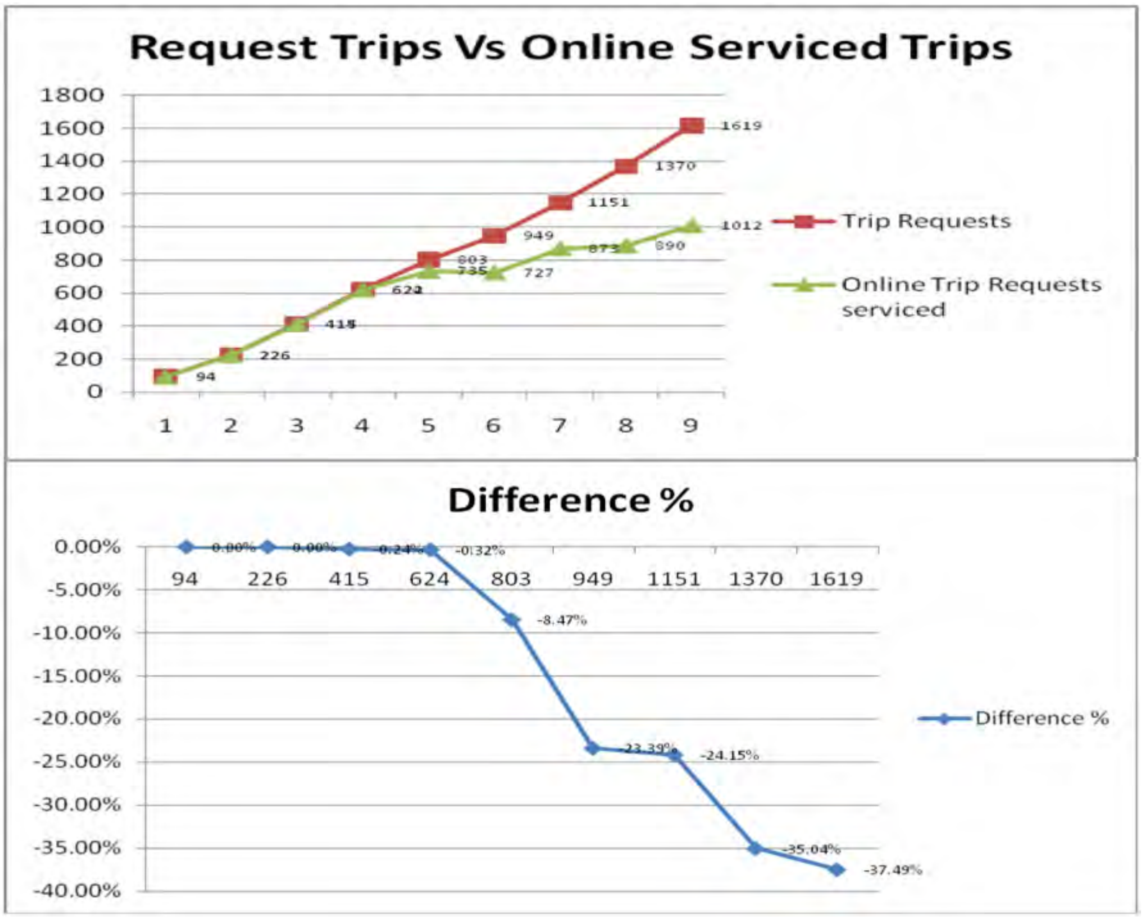


Figure 5-1.Comparison graph for Online serviced demands among requested demands

The main reason for this behavior is that during the optimization, some trip demands cannot get an answer within 60 seconds of the time they arrive at the dispatcher.

Table 5-2 presents results concerning the time required for the simple insertion of all demands into a solution and the time required for the regret optimization procedure.

The 5th column represents the total time needed by online regret sub-algorithm without optimization (only fast insertion) in order to satisfy all incoming requests represented by the 3rd column.

The 6th column represents the total time needed by the online regret sub-algorithm with optimization in order to optimize routes

The 8th column represents the maximum time required for the online regret sub-algorithm without optimization to successfully process a trip request

The 9th column represents the maximum time required for a full step of optimization by the online regret sub-algorithm with optimization

Table 5-2: Online regret Algorithm test results
1-day (86440 sec's) work period and for real data from the municipality of Philippi

#of exper	*Trip de-mands*	*Trip Requests Assigned*	*Cost*	*Ins Time(Sec's)*	*Regret Time (Sec's)*	*Profit (euros)*	*Max Ins Time*	*Max Reg Time*
1	94	94	705400	12.97	60.41	48.4799	0.410	3.330
2	226	226	1154500	182.69	1266.11	200.9	2.57	39.24
3	415	414	1683200	404.88	7330.59	554.2	2.82	196.83
4	624	622	2385600	730.92	15246.4	886.3	3.51	343.3
5	803	735	2630300	1862.51	29432.5	1096.4	7.160	790.9
6	949	727	2652700	2406.16	37519.2	1071.4	11.41	909.
7	1151	873	3172500	2456.50	41219.0	1263.6	9.2	768.0
8	1370	890	3361500	2857.13	44882.7	1225.1	12.37	709.7
9	1619	1012	3516600	3636.53	48463.1	1496.4	14.1	1207

We clearly see that the regret optimization procedure consumes the major part of the total execution time required by OR-DARP algorithm in order to provide a solution. In experiment #9 the execution time needed by the regret optimization procedure is almost 48463/(48463+3636) = 93% of the total execution time.

Table 5-3 presents some comparative results between the static regret (RegretH) algorithm solution cost and the online (OR-DARP) regret algorithm solution cost.

The 3rd column represents the solution cost produced by the static regret algorithm

The 4th column represents the solution cost produced by the online regret algorithm

The 5th column represents the cost difference percentage between static and online cost

Table 5-3: Comparison between the online regret algorithm cost and the static regret algorithm cost

#of exper	*Online Trip demands serviced*	*Static Cost*	*Online Cost*	*Cost Differ-ence %*	*Avg Pass per KM (Static)*	*Avg Pass per KM (online)*	*Avg Pass Difference %*
1	94	636900	705400	10.76%	1.556	1.40658	-9.58%
2	226	1088100	1154500	6.10%	1.858	1.74777	-5.91%
3	414	1659900	1683200	1.40%	2.330	2.27061	-2.56%
4	622	2248300	2385600	6.11%	2.595	2.43832	-6.05%
5	735	2562500	2630300	2.65%	2.589	2.57096	-0.68%
6	727	2522500	2652700	5.16%	2.606	2.51445	-3.50%
7	873	3015700	3172500	5.20%	2.577	2.46122	-4.48%
8	890	3162200	3361500	6.30%	2.512	2.40471	-4.26%
9	1012	3301000	3516600	6.53%	2.732	2.59163	-5.13%

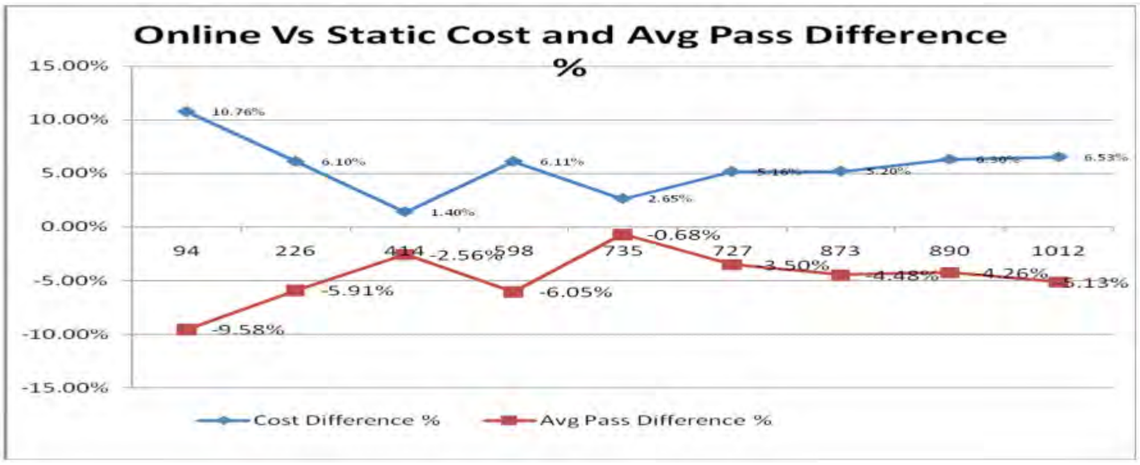


Figure 5-2.Comparison graph for the cost difference between Static regret and Online regret for the same data set

The cost difference between the static regret algorithm and the online regret algorithm varies from 10.76% to 1.4%. Figure 5-2 clearly presents that there is a mirror symmetry concerning the cost difference and the average number of passengers per Km. Cost difference and average passengers per KM difference, describe the same differences in nature between the online regret solution characteristics and the static regret solution.

Another interesting set of experiments is the comparison between the OR-DARP and the OR-DARP without regret optimization. The rationale of these experiments is to explore the cost and profits difference in order to determine the usability of the online regret algorithm with regard to the size of cost reduction (or profit increase). This set of experiments is presented in table 5-4, where

The 5th column represents the cost difference percentage

The 6th column represents the execution time of the OR-DARP algorithm without the regret optimization procedure

The 7th column represents the execution time of the OR-DARP algorithm.

The 8th column represents the solution profits of the OR-DARP algorithm without the regret optimization procedure

The 9th column represents the solution profits of the OR-DARP

The 10th column represents the profit difference percentage

Table 5-4: Comparison between online regret sub-algorithm with optimization and the online regret sub-algorithm without optimization for the same data set

#of expert	*Serviced Demands*	*Online Regret (no opt) Cost*	*Online Regret(opt) Cost*	*Diff Cost %*	*Online Regret (no opt) Time*	*Online Regret (opt) Time*	*Online Regret (no opt) profit*	*Online Regret (opt) Profit*	*Diff prof %*
1	94	713500	705400	-1.148%	10.11	73.38	46.975	48.4799	3.2%
2	226	1176500	1154500	-1.906%	147.5	1449	193.824	200.864	3.505%
3	414	1745900	1683200	-3.725%	477.8	7735	530.08	554.24	4.359%
4	622	2458900	2385600	-3.073%	1488	15977	858.719	886.271	3.109%
5	735	2716800	2630300	-3.289%	2092	31295	1068.7	1096.38	2.525%
6	727	2670700	2652700	-0.679%	1941	39925	1065.7	1071.36	0.528%
7	873	3185500	3172500	-0.410%	3808	43676	1255	1263.62	0.682%
8	890	3463900	3361500	-3.046%	4226	47740	1184.13	1225.09	3.343%
9	1012	3619200	3516600	-2.918%	6655	52100	1455.39	1496.41	2.741%

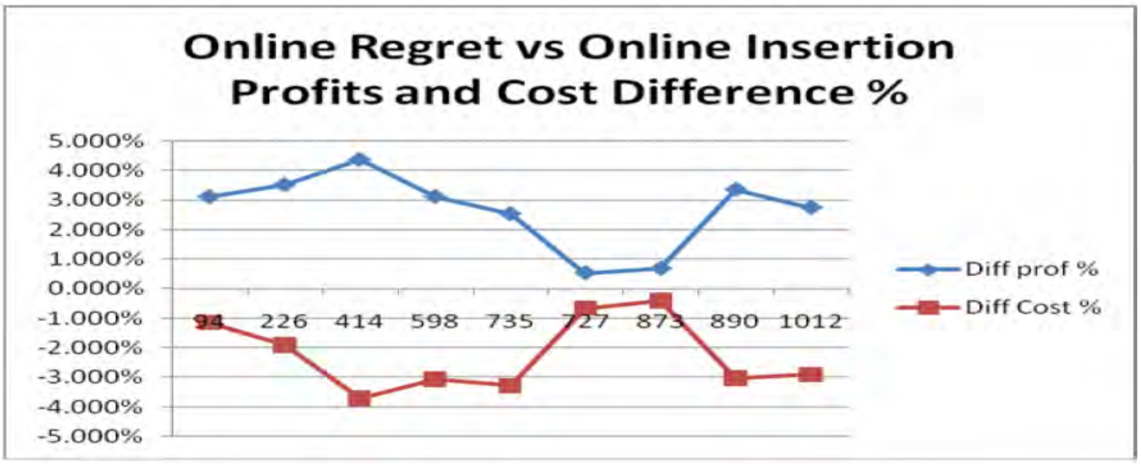


Figure 5-3.Comparative graph for the cost and profits difference between Online Insertion and Online regret for the same data set

The cost difference between the online regret sub-algorithm with optimization and online regret algorithm without optimization varies between -3.725% and -0.410%. Similarly the profit difference varies between 4.36% and 0.53%. For the same data set the online regret algorithm with optimization always outperforms the one without optimization. However, the execution time between these two different versions of the OR-DARP algorithm differs significantly.

Table 5-6 presents an interesting comparison between OR-DARP algorithm with regret optimization and OR-DARP algorithm without optimization. Both versions of the online algorithms have been tested on the initial set – the one presented in table 5-1 - of requested trip demands. The OR-DARP algorithm without regret optimization accepts all requested trip (see table 5-5) demands while OR-DARP algorithm with optimization accepts only a part of requested trip demands (see table 5-2).

Table 5-5: Online regret algorithm without optimization test results for 1-day (86440 sec's) work period

#of exper	*Vehicles*	*Trip demands*	*Trips Requests Assigned*	*Avg Pass per KM*	*Avg Distance Deviation %*	*Avg Time Deviation %*	*Cost*	*Time(Sec's)*	*profit*
1	7	94	94	1.3955	1%	1%	713500	10.1	46.9759
2	9	226	226	1.70881	0%	1%	1176500	148.4	193.824
3	16	415	415	2.21605	1%	3%	1745900	481.61	535.904
4	21	624	624	2.36126	1%	2%	2440300	1508.33	874.6
5	26	803	803	2.54391	1%	2%	2920000	2717.36	1180.61
6	28	949	949	2.75778	1%	2%	3158800	3881.55	1514.56
7	31	1151	1151	2.73839	1%	2%	3810300	7198.61	1794.9
8	33	1370	1370	2.69738	1%	3%	4549400	11249.2	2076.16
9	37	1619	1619	2.85987	2%	5%	5129000	17956.3	2521.28

By combining the 3rd(assigned trip demands), 4th (solution cost), 7th (profit) column in table 5-2 and the 4th (assigned trip demands),8th (solution cost), 10th (profit) column in table 5-5 we have table 5-6.

In table 5-6

The 2nd column represents the percentage difference between the 3rd column in table 5-2 and the 4th column in table 5-5.

The 3rd column represents the difference of solution costs (4th column in table 5-2), (8th column in table 5-5).

The 4th column represents the difference of solution profits (7th column in table 5-2), (10th column in table 5-5).

Table 5-6: Comparison between the online regret sub-algorithm without optimization and online regret sub-algorithm with optimization for different data sets

#of exper	*Trips Diff %*	*Cost Diff%*	*Profit Diff %*
1(94)	0.00%	-1.14%	3.20%
2(226)	0.00%	-1.87%	3.63%
3(415)	-0.24%	-3.59%	3.42%
4(624)	-0.32%	-2.24%	1.34%
5(803)	-8.47%	-9.92%	-7.13%
6(949)	-23.39%	-16.02%	-29.26%
7(1151)	-24.15%	-16.74%	-29.60%
8(1370)	-35.04%	-26.11%	-40.99%
9(1619)	-37.49%	-31.44%	-40.65%

The results of Table 5-6 are of special interest: In experiments 1 to 4, when using the online regret sub-algorithm with optimization instead of the online regret algorithm without optimization there is a very small (or zero) percentage of requested trips that cannot be served, while at the same time the profits are bigger (and the cost smaller).

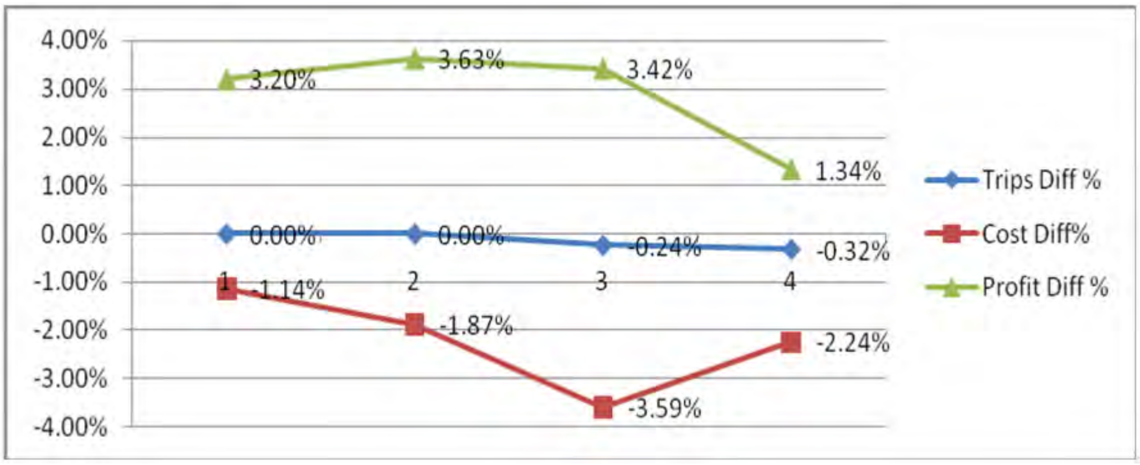


Figure 5-4.Comparison graph for the trips cost, and profits difference between online regret sub-algorithm without optimization and online regret sub-algorithm with optimization for experiments 1 to 4

In experiments 5 to 9, when using the online regret sub-algorithm with optimization instead of the online regret sub-algorithm without optimization there is a significant loss of trips demands (not served). It is interesting that the use of online regret algorithm with optimization has profit losses greater than cost reduction because of the smaller number of served demands. In experiment 6 we notice that cost reduction is 16.02% compared to a profit loss of 29.26%.

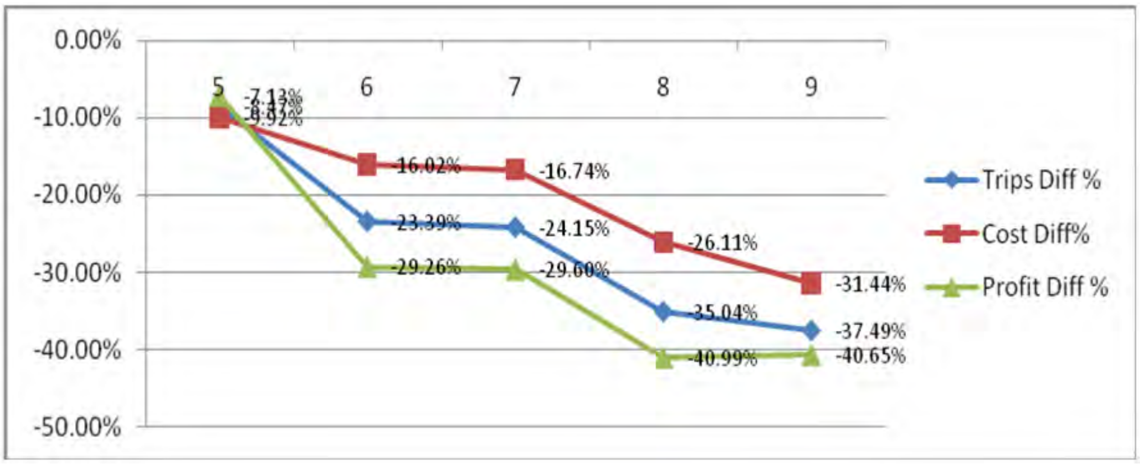


Figure 5-5.Comparison graph for the trips cost, and profits difference between online regret algorithm without optimization and online regret algorithm with optimization for different data set for experiments 5 to 9

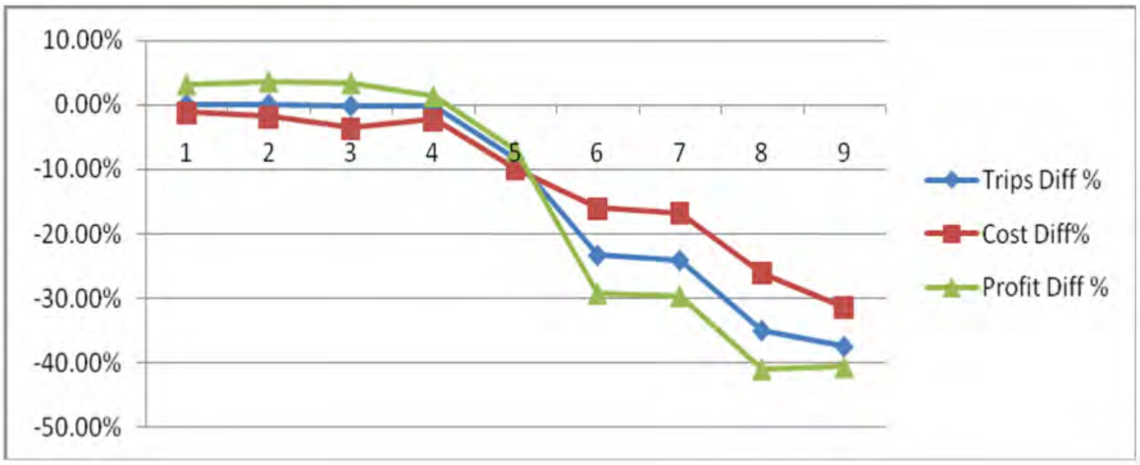


Figure 5-6.Comparison graph for the trips cost, and profits difference between online regret algorithm without optimization and online regret with optimization for different data set

Table 5-7 presents interesting results on queuing times and the served trip demands. By the term queuing time we mean the time a trip request is allowed to wait in a queue in order to get an answer. We used four different queuing times (the shortest time is 10 seconds while the longest is 120 seconds) covering the majority of possible acceptable response times in real life. The results, shows that short queuing times affect the number of served demands. Figure 5-9 indicate that the maximum difference between served demands with response time of 10 seconds and served demands with response time of 120 seconds has a the maximum value of 23.60% which is a significant difference.

Table 5-7: Comparison table between different queuing times for OR-DARP algorithm

#of exper	*10 secs (Trip demands)*	*30 secs (Trip demands)*	*60 secs (Trip demands)*	*90 secs (Trip demands)*	*120 secs (Trip demands)*	*Diff Between 120 secs, 10 Secs queueing time*
1	94	94	94	94	94	0.00%
2	226	226	226	226	226	0.00%
3	408	413	414	415	415	1.72%
4	572	587	622	616	613	7.17%
5	637	678	735	757	759	19.15%
6	682	692	727	792	787	15.40%
7	799	806	873	906	921	15.27%
8	822	870	890	945	1016	23.60%
9	897	914	1012	1011	961	7.13%

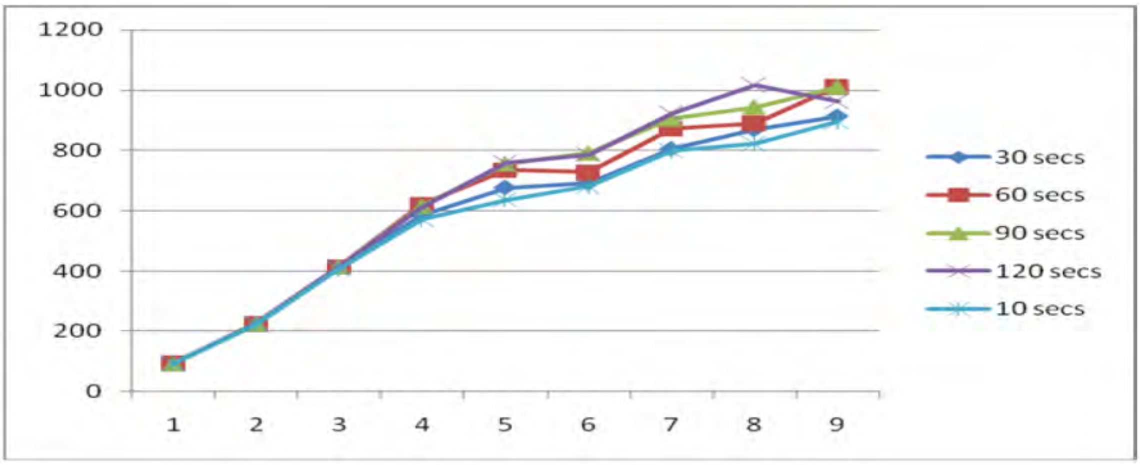


Figure 5-7. Comparison graph for the number of serviced trip demands for different response times

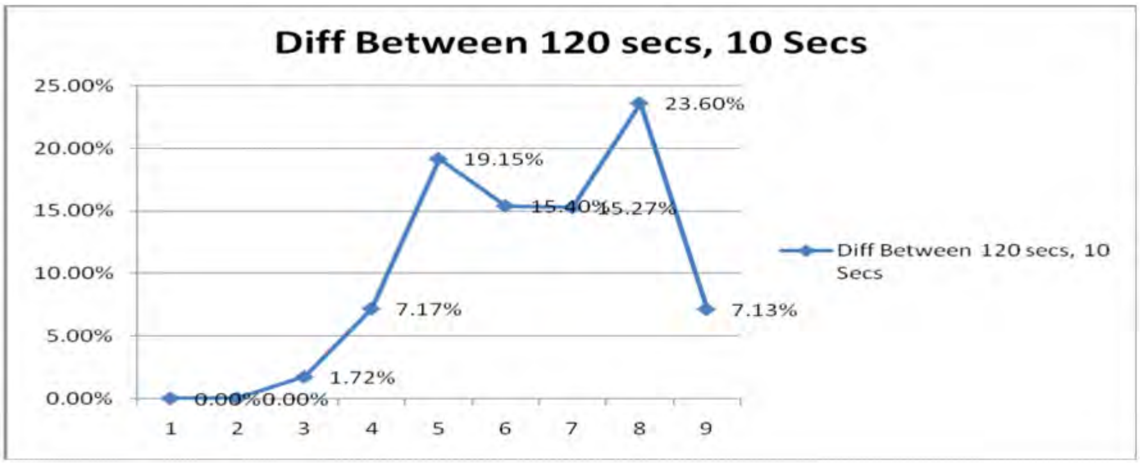


Figure 5-8. Percentage difference concerning serviced trip demands for response times 10secs and 120 secs.

5.1.7 Analysis of OR-DARP Algorithm

Computational results in Section 5.1.6 demonstrate that the OR-DARP algorithm always improves the solution produced by OR-DARP without regret optimization when applied. Although as expected the solution cost is more in comparison with solution cost provided by the static regret algorithm.

The benefit of the proposed online regret algorithm is the absence of time horizons which means that all demands are handled as emergency ones getting immediate response.

An important result is that, the side effect of the OR-DARP algorithm is the rejection of some requested trip demands. Table 5-1 clearly shows that, for large scale programs OR-DARP rejects some demands. The main reason for that behavior is the additional time required for the optimization procedure even for a single step of optimization.

Another important result is in table 5-6 and the relevant graphs figure 5-5, figure 5-6,figure 5-7. We made this comparison between different data sets in order to define a critical point regarding the costs and profits. There is a point that identifies the number of demands, where the use of OR-DARP algorithm instead of OR-DARP algorithm without optimization becomes unprofitable. The main reason for this, is that cost decrease is less in comparison with profit loss due to less served demands. This means that beyond a specific number of requested trip demands it is more profitable to use OR-DARP without optimization in order to serve more requests.

5.1.8 OR-DARP Algorithm Competitive Ratio

It is already mentioned in section 2.5 that the basic feature describing the closeness of an online algorithm with respect to the optimal algorithm is the competitive ratio.

Due to the presence of specific restrictions - specific pickup and delivery order and presence of time windows – it is hard to find a complete mathematical formula for the competitive ratio problem of the dial-a-ride Problem.

Feuerstein [141] defined competitive ratios for the online single server Dial-a-Ride problem. Fink, Krumke,Westphal (2009) [142] defined that competitive ratio for k-servers has 3 as the lower bound. These lower bounds were calculated under specific conditions such us the same start end depot location, the absence of time windows, the homogeneity of vehicles and they deal with the minimization of the makespan i.e. the time when the last server has completed its tour or the minimization of the sum of completion times.

Since it is impossible to find the optimal solution for large-scale problems with the existing algorithms, competitive ratio can be calculated in a different way. The basic idea is to use the same algorithm - not necessarily the algorithm that finds the optimal solution - in two different versions. The first version is the offline algorithm while the second version is the online version.

Xiang [140] defined competitive ratio as follows: $\frac{Online\_\cos t - Static\_Cost}{Stati\_Cost} 100\,\%$ (4-18).

Where online_cost represents the solution cost calculated by the online version of the algorithm while static_cost represents the solution cost calculated by the static version of the algorithm. The above formula is not an analytic mathematical formula, which means that we cannot use it in order to forecast the online cost given the static cost and vice versa.

Given a set of *n* online demands $OD_n = \{OD_{s_{1i}d_1}, .., OD_{s_nd_n}\}$ where $OD_{s_id_i}$ is the online demand with source s_i and destination d_i, the competitive ratio can be calculated as $\frac{OLALG(OD_n)}{OPTALG(OD_n)}$ (4-19) where OLALG(OD_n) is any online algorithm which provides a feasible solution while OPTALG(OD_n) is the optimum offline algorithm for the same set of demands if known before. Clearly, OPTALG(OD_n) will always be at least as good as any online algorithm, since any online solution (i. e., sequence of schedules) can be converted to an offline schedule. Suppose we have a set of static demands called as $SD_m = \{SD_{s_{1i}d_1}, .., SD_{s_m d_m}\}$. The OPTALG($SD_n$) algorithm provides the optimized solution for this set of static demands. When a new $od_{s_a d_a}$ online demand shows-up, the new solution can be no worse than OPTALG(SD_m) + D(o,s_a)+D(s_a,d_a)+D(d_a,o) (4-14) where o=is the origin of vehicles start, end depot and D(x,y) is the distance between points x,y located in our space.

According to the above relations the maximum overhead cost for every online demand is *D(o,s_i)+D(s_i,d_i)+D(d_i,o)* assuming that the vehicle that serves this demands operates as a taxi.

In case of $SD_{m=0} = \{\}$ and $OD_n = \{OD_{s_{1i}d_1}, .., OD_{s_n d_n}\}$ competitive ratio *CR* cannot be larger than $\frac{\sum_{i=1}^{n}(D(o,s_i)+D(s_i,d_i)+D(d_i,o))}{OPTALG(OD_n)}$ (4-20). The difficult part in this equation is the calculation of the solution cost produced by the optimal algorithm. We should give a special example for this. Our Hypothesis:

1. Fleet consists of one vehicle with infinite capacity.
2. For every online demand $od_{s_a d_a}$ the showing time is larger than, the time where the previous served demand is completed and the vehicle that serves this demand returns to its depot again.
3. The source and destination of all online demands is the same

Given these constrains the optimum solution for all demands produced by OPTALG is *D(o,s_i)+D(s_i,d_i)+D(d_i,o)* (*i*=1 or 2 or .. n). In this case the competitive ratio can be no larger than $\frac{\sum_{i=1}^{n}(D(o,s)+D(s,d)+D(d,o))}{D(o,s)+D(s,d)+D(d,o)} = n$ (4-21). The mathematical formula presented in 4-21 leads to the same results as those presented by the following theorem "There is no competitive deterministic online algorithm for the Online Darp with response to the total travel distance" [156].

The most basic measure in a routing context is the number of online requests relative to the total. We define this ratio as the degree of "onlinenism" of the system considered and denote it as *oratio* .

This degree defined as *oratio* = $\frac{Online\_equests}{Total\_\mathrm{Re}quests}$ (4-22).

To the best of our knowledge, no methodology has examined the competitive ratio as a function of the *oratio*. We constructed a series of experiments for the study of this behavior. The basic idea of these experiments relies on the fact that – given the total number of demands (offline and online) included in the solution – the number of online demands has been gradually increased over the number of offline demands. It is expected that the quality of the solution will deteriorate, as the proportion of online demands involved in the solution increases compared to the proportion of offline demands. The main aim is to find a mathematical approaching formula which allows us to predict with some accuracy the quality of the solution given a specific combination of offline and online demands.

It is already mentioned that the maximum overhead - called OC_a - due the presence of an online demand $od_{s_a d_a}$ is $D(o,s_a)+D(s_a,d_a)+D(d_a,o)$ given the *o* point as end, start depot. For a given set of *n* online demands such as $OD_n = \{OD_{s_{1i}d_1},..,OD_{s_n d_n}\}$ the expected overhead cost is given by formula $E\left[\sum_{i=1}^{n} OC_i\right]$(4-23). A linear approach between the cost produced by the OPTALG algorithm and the OLALG algorithm could be approached by formula *ax+b* (4-24) where *x* is the number of online demands. Factor *a* in (4-24) represents the extra incurred overhead cost introduced by the online algorithm every time a new online demand shows-up, while factor *b* represents the cost produced by OPTALG for the *SD*$_m$ set. The worst case of overhead is produced when every online demand is handled the in the taxi way. If this is the case, factor a is equal to $E\left[\sum_{i=1}^{n} OC_i\right]$. Although our experiments – presented later – showed that this number is significantly smaller.

Our experiments were based on artificial data in order to reduce the presence of hidden patterns concerning the trips nature.

The testing field was a square area 100X100 Kilometers. Vehicle capacity was 8, 12, 16 seats. The maximum number of passengers for each trip demand was selected randomly between 4, 6 or 8 in accordance with vehicle capacity. Pickup and delivery positions of every trip request were selected randomly over that area. Vehicle "Start", "End" depots were selected at the center of this area (50, 50). Time windows for pickup were 15 minutes and the earlier pickup time was randomly selected between 0–1440, time windows for deliveries were produced from pickups time windows plus the 1.5*(Cartesian distance) between pickup and delivery. Maximum vehicle ride time was 1440. Vehicle speed was 1Km/minute.

The number of experiments for each distinctive combination of demands number and vehicles capacity was set to 20 and the confidence level for this experiment was set to 95%.

The underlying algorithms were the RegretH algorithm for offline demands and the online OR-DARP algorithm for online demands.

In Table 5-8 we present the results concerning the comparison between the pure offline version and the pure online version for various instances of our. The columns of table 5-8 are self-explanatory.

Table 5-8: Comparison between the online regret sub-algorithm without

Experiment	*Requested Demands*(Avg serviced demands)	*Capacity*	*Full Online(Avg)*	*Full Static(Avg)*	*Percentage Difference*
1	500(425)	8	32290	28368.83	13.82%
2	500(426)	12	32764.67	28739	14.01%
3	500(427)	16	33611.17	29066.17	15.64%
4	750(506)	8	37604.33	33622.67	11.84%
5	750(526)	12	37601.5	33232.5	13.15%
6	750(522)	16	37868.5	33245.67	13.91%

The cost difference between the full online solution and the full static solution is always larger than 10%. Vehicles capacity affects cost difference as well. It is expected that larger vehicle capacity provides more opportunities to improve the solution due to the fact that this restriction is relaxed. Due to fact that the above table presents only the values of the full offline, the online solution doesn't provide us with information regarding the intermediate states.

We chose to present an example to show all intermediate states. From each experiment group we choose the most representative experiment which is closest to the averages concerning features like full online and static solution costs and the number of serviced demands. The following charts figure 5-9, 5-10 present all intermediate states.

The structure of the charts is the following.

1. The Y-axis represents the cost of the solution
2. The X- axis represents the number of offline demands
3. The blue points represent the cost of each solution.
4. The black line represents the linear approximation of the solutions set
5. The green line represents the 2nd degree polynomial approximation of the solutions set
6. The red line represents the 4th degree polynomial approximation of the solutions set

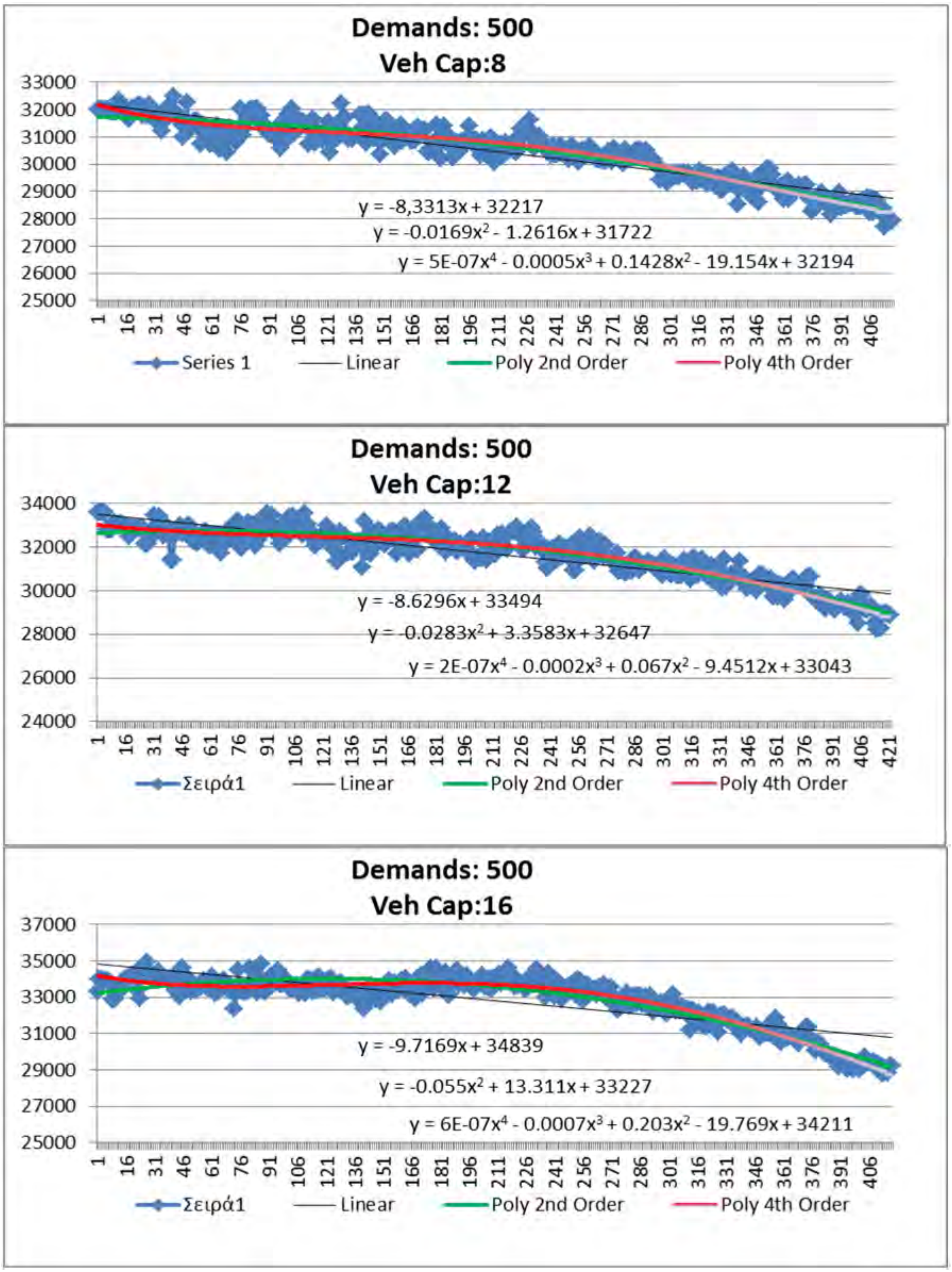
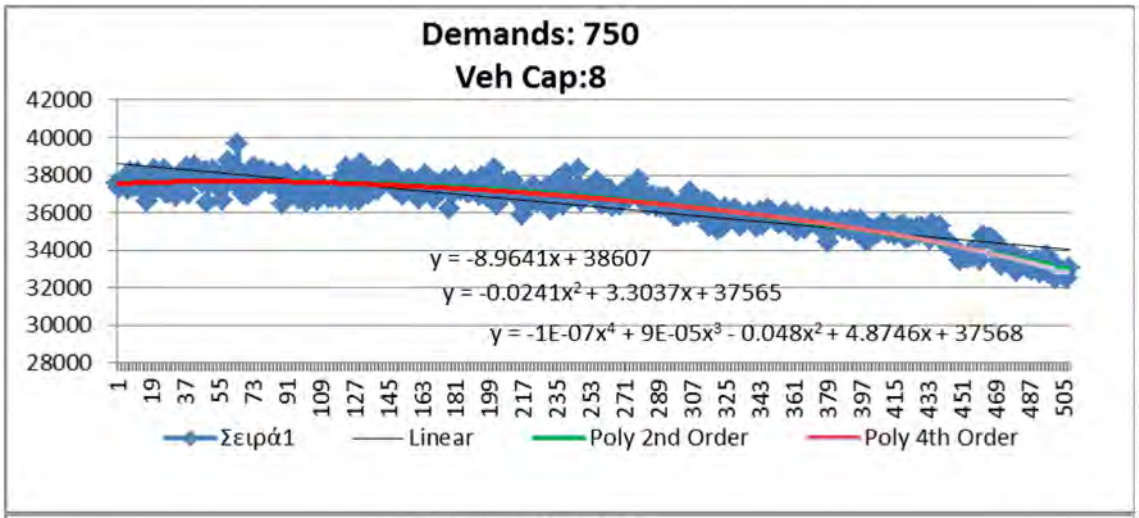
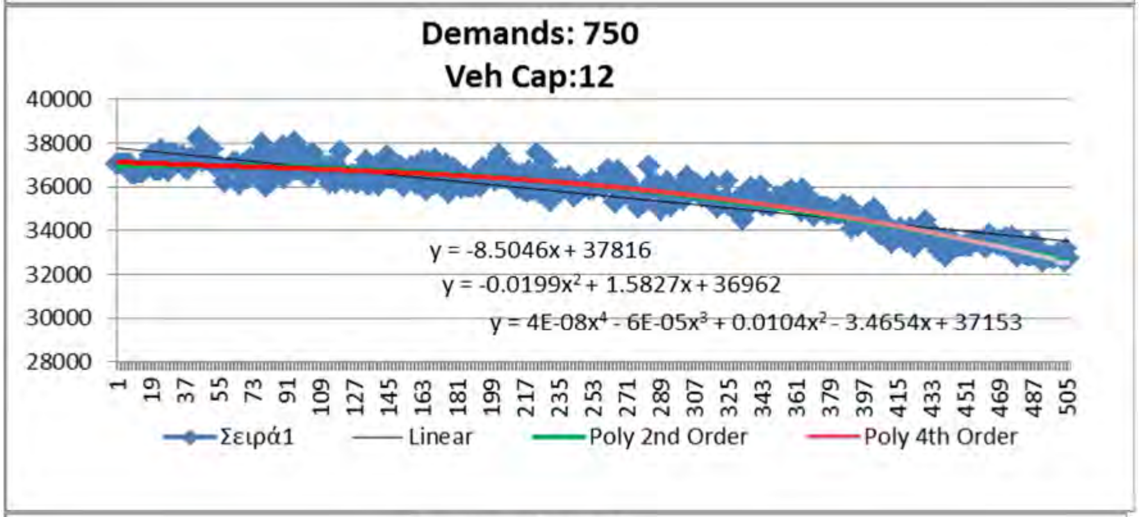


Figure 5-9. Solution cost in relation to number of online and offline demands. Starts with one static demand, 499 online demands and ends to one online demand, 499 static demands





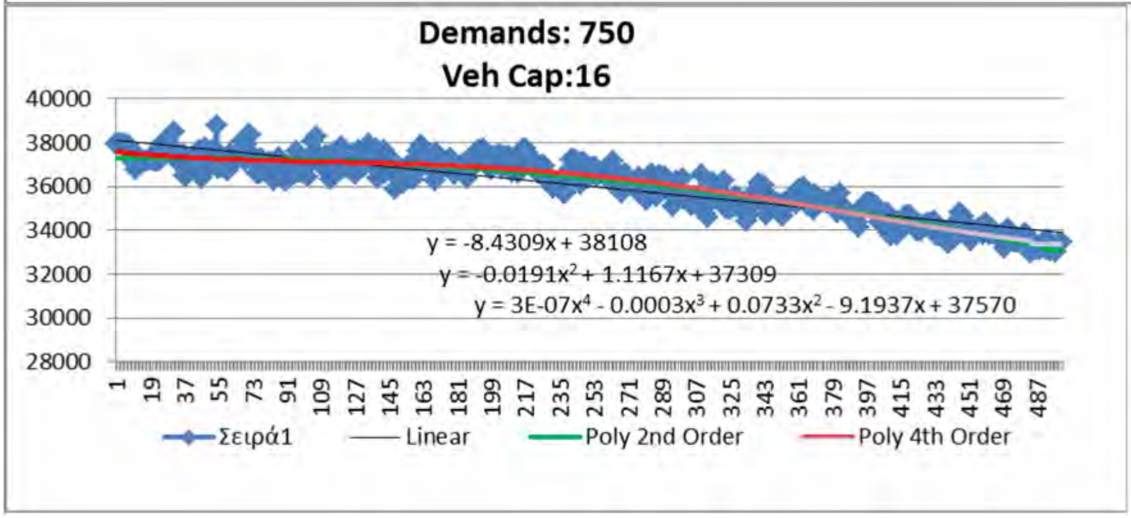


Figure 5-10. Solution Quality in relation to number of online and offline demands. Starts with one static demand, 749 online demands and ends to one online demand, 749 static demands

In all cases we observe that the polynomial approximation is much more accurate compared to the linear approach.

In 5 of the 6 experiments we observe that the 2nd degree polynomial approximation is almost identical to the 4th degree polynomial approximation.

The basic idea of these approaches is that we can use them in order to have a good approximation for the cost of a full online solution in comparison to the cost of the full offline one. This gives us the opportunity to estimate with reasonable accuracy the cost of a full online solution without knowing in advance the full number of online demands.

Table 5-9 presents the linear approach in order to describe the overhead cost incurred due the presence of online demands.

The 4th column describes the factor α of the linear approximation while the last column describes the confidence value for 95% confidence level. The 5th and 6th columns describe the minimum and the maximum value of the factor *a* according to the last column. Factor *a* describes the cost incurred by every online demand in the solution. I.e. in the first experiment the additional overhead cost is 8.313 meaning that each online demand introduces 8.313 cost units

Table 5-9: Approximation functions for oratio

Experiment	*Requested Demands(Avg serviced demands)*	*Capacity*	*a-factor(Avg)*	*a-factor(Min)*	*a-factor(Max)*	*Confidence (95% level)*
1	500(425)	8	9,2252013	7,870672	10,57973	1,35453
2	500(426)	12	9,4376278	8,200871	10,67438	1,236756
3	500(427)	16	10,652912	9,970224	11,3356	0,682689
4	750(506)	8	7,8788124	6,978787	8,778837	0,900025
5	750(526)	12	8,3031367	7,300445	9,305829	1,002692
6	750(522)	16	8,8669036	8,001019	9,732788	0,865885

We notice that the factor *a* increases in accordance with the vehicle capacity and the number of demands. This is quite normal since the offline algorithm takes advantage of these constraints in order to produce better solutions. By using table (4-23) in order to evaluate the parameter a of the linear approximation ax+b and calculate the worst case solution cost for each experiment we get the table 5-10. The 5th column of table 5-10 describes the value of factor a as the expected value $E\left[\sum_{i=1}^{n} OC_i\right]$.

Table 5-10: Comparison between proposed online algorithm and the worst case online algorithm

Experiment	*Demands*	*Actual Serviced Demands (Average)*	*Vehicles Capacity*	*Full Online (Average)*	$\lvert a\rvert = E\left[\sum_{i=1}^{n} OC_i\right]$	*Worst Case Online solution Cost (Average)*	*Percentage Difference (Average)*
1	500	425	8	32290	134.96	57360	77.6%
2	500	426	12	32764	133.47	56859	73.5%
3	500	427	16	33611	135.56	57886	72.2%
4	750	506	8	37604	125.39	63448	68.7%
5	750	526	12	37601	124.97	65737	74.8%
6	750	522	16	37686	128.79	67232	78.4%

The above table presents clearly the way that the proposed OR-DARP algorithm improves the solution in comparison to worst case solution concerning these online demands.

5.2 Online Regret Algorithm with probabilistic demands (OP-DARP)

5.2.1 Basic Concept

For the OP-DARP algorithm we follow a different concept than the one presented in section 5.1. The basic idea is to use probabilistic demands in order to utilize the knowledge regarding the incoming demands. This knowledge can be proved valuable to provide better solutions. The basic assumption is that we have substantial computational power so that the execution time is of no concern.

The influence of probabilistic demands to the solution quality is an important issue that deserves to be investigated. However an analytical mathematical model which can describe this influence is not easily attainable. A widely approved way to study such problems is the Monte Carlo method. Generally speaking the Monte Carlo method is a technique that involves the use of random numbers and probability to solve problems. The general concept of this method is as follows:

1. Define a domain of possible inputs.
2. Generate inputs randomly from the domain by using a specific probability distribution.
3. Execute a deterministic computation using the inputs.
4. Aggregate the results of the individual computations into the final result.

In our case the sampling has to do with the selection of probabilistic demands from a domain of possible inputs. Our domain of possible inputs came from real data collected during a pilot project in the municipality of Philippi. In section 5.2.4 we define in detail this data source. Each trip demand belonging to this domain of possible inputs is characterized by the probability of it occurring. Random inputs were produced via uniform random numbers concerning the showing up probability. These random inputs of probabilistic demands are used by the OP-DARP algorithm in order to provide a solution when combined with the real demands.

Given the fact of the existence of real trips the data domain input can be created by:

1. A probability distribution that closely fits the collected data for trip demand occurrence.
2. A simple empirical distribution, if none of the known probability distributions is an acceptable fit.

Every time the OP-DARP algorithm has to deal with a new incoming real trip demand, a sequence of probabilistic demands is generated using random numbers. These probabilistic demands are incorporated into the solution search.

The most preferable method to use the generated probabilistic demands is through time horizons. Time horizon defines the future time period for which the algorithm uses probabilistic demands. The size of Time horizon is critical because this size practically defines how far in the future, the algorithm search uses probabilistic demands.

Generally speaking, it is expected that sometimes the produced solution could be worse in the short term, but eventually it becomes better.

5.2.2 Nomenclature

V = {1,2,3,   ,|V|} = set of vehicles

***EPT**$_i$**, LPT**$_i$* = Earlier and late pickup time of the *i-th* online trip demand

***t**$_c$* = is the current time as the online algorithm runs

THStep = is the time horizon step, Possible values are 60 minutes, or 120 minutes, or 180 minutes.

k =** is the time horizon index. Given the value of **THStep,** possible values of k are all integer values between 0 and 1440 with increasing step of **THStep. I.e. for ***THStep*** *equal to 180 minutes, k takes values between 0,180,360, 1280.*

TR(t**$_c$**) = is the procedure which returns the new trip request that showed up at time t_c

TH(TR(t**$_c$**)) = is the procedure which defines the time horizon index, based on *EPT$_i$, LPT$_i$, of the TR(t$_c$) demand.*

Possible values are : All time steps according to ***THStep,*** starting from 00:00 (0 minutes) ending to 24:00 (1440 minutes)

PD = the probabilistic demand

***PDem**$_k$* = set of all probabilistic demands for the *k*th time horizon index with occurrence probability larger than 0.

***SPDem**$_k$* = set of all randomly selected probabilistic demands from ***PDem**$_k$* set.

***APDem**$_k$* = set of assigned probabilistic demands from ***SPDem**$_k$* set.

InsertionTH (t) = is a modified version of the simple insertion algorithm InsertionH – presented in section 4.1 - which finds the solution of least cost only for those assigned demands where ***EPT**$_i$ **> t**$_c$*

InsertionProbH (TH(TR(t**$_c$**))) = is the insertion sub-algorithm which is used to insert probabilistic trip demands into vehicle routes. Those probabilistic trip demands selected randomly from those who belong to the same time horizon index as the requested real demand *TR(t_c).*

InsertionPRH (TR(t_c)) = is a modified version of the simple insertion algorithm *InsertionH* which finds the solution of least cost for the real demand ***TR(t_c).*** In case of an unsuccessful assignment, the algorithm removes the less probable assigned probabilistic demand in order to relax the solution space. After each removal this algorithm restarts.

RS(t_c) = is the procedure which returns the problem solution produced by *InsertionPRH* sub-algorithm at the time t_c

RegretPH (t_c) = is the online regret sub-algorithm that uses regret optimization in order to improve the *RS(t_c)* solution.

WithdrawProbH(.) = Is the procedure that withdraws all probabilistic demands that belong to set $APDem_k$

5.2.3 Algorithm Description

It is already noted that the crucial element for the OP-DARP algorithm is the probabilistic nature of the future demands.

In the beginning we define the appropriate horizon time step. The definition of the horizon time step is a very important issue. Wide time steps can overload the algorithm execution time. Narrow time steps offer little information and make the behavior of the online algorithm rather myopic. In our case we experimented with three horizon time steps of 60,120,180 minutes respectively. Probabilistic trip demands have the following features:

1. Typical Trip Demands features (origin, destination, EPTi, LPTi, *Maximum Ride Time)***)**

2. The occurrence probability – for a specific time horizon - as indicated by the probability distribution chosen for this particular application.

In the second phase, the OP-DARP algorithm uses the probability distribution as domain input, the Monte Carlo method for this domain in order to produce probabilistic demands, and the realized demand in order to produce the solution. Probabilistic demands are included in the solution as long as it is needed for the assignment of the real demand. After the assignment of the realized demand all probabilistic demands are withdrawn from the solution.

When a real demand occurs, we define the time horizon – called k - specified by this demand. Based on this time horizon, an initial set – called ***PDem$_k$* -** of probabilistic trip demands compliant to this time horizon is produced, by using the selected probability distribution. From this initial set of probabilistic demands, a second set – called ***SPDem$_k$* -** of the selected probabilistic demands is produced by using random numbers. This production of the second set is as follows. For each demand belonging to the ***PDem$_k$*** set, we generate a random number. If the produced random number is equal or larger to the occurrence probability then this the trip demand is eligible. This ***SPDem$_k$*** set of probabilistic demands is taken into account by the algorithm OP-DARP in order to find a feasible solution for the realized demand. The pure impact of probabilistic demands versus the regret optimization procedure is another interesting issue. For this problem we have developed two variations of the original OP-DARP algorithm. The first one is the OP-DARP-NoRegret algorithm which is the same as the OP-DARP algorithm without the use of regret optimization. The second one is the above referenced OR-DARP algorithm which is the same as the OP-DARP algorithm without the use of probabilistic demands.

The OP-DARP algorithm can be considered as a wrap algorithm that calls continuously other sub-algorithms. The basic task of OP-DARP is to assign the online real demand while extensively using probabilistic demands.

A more detailed description of OP-DARP follows:

Step 0:

1. Define: Time step horizons that will be used by the OP-DARP algorithm. *(The term "time step horizons" refers to time sub-divisions to which a full operational period (one day in our case) is split in. The size of these sub-periods practically defines the depth of the time-horizon in which our algorithm will search for probabilistic demands. i.e. 180 minutes time step horizon means that the day is divided into 8 sub periods and our algorithm always looks for probabilistic demands into a period of 180 mins)*
2. Build Empty routes for every vehicle

Step 1:

1. Read Current online real demand
2. Build time horizon for this demand. *(In this step we define the specific sub-period where the algorithm looks for probabilistic demands. We use the earlier and the late pickup time of the real demand to define this specific sub-period. I.e. if the earlier, late pickup time of the real demand is 680,695 minutes accordingly, and the time step - defined in step1- is 60 minutes, then the time horizon is 660-720 minutes. In this case the OP-DARP algorithm handles probabilistic demands in a sub-period 660-720 minutes)*
3. *Call the following:*
    a. Sub-algorithm ***InsertionProbH*** in order to assign the probabilistic demands included in the list $\boldsymbol{SPDem_k}$.
    b. Sub-algorithm ***InsertionPRH*** in order to assign the real demand
4. Run the regret optimization procedure in order to optimize the solution. *(in this step we use the regret optimization procedure for all assigned real demands with earlier pickup time larger than the start of the current time horizon)*

5. Jump to step 1

The **InsertionProbH** sub-algorithm works as follows:

Step 0:

1. Construct the ***PDem$_k$* ,*and* *SPDem$_k$*** *lists (PDem$_k$ list is created by using the specific time horizon index while SPDemk list, through random numbers)*
2. Sort ***SPDem$_k$*** list in descending order of occurrence probability *(the sorting is meaningful as we always use the most probable demands for assignment)*

Step 1:

*Repeat the following until you reach the end of **SPDem$_k$** list*

  1. Try to Assign every Probabilistic Demand included in ***SPDem$_k$*** set starting by those with the biggest show up probability
  2. Update ***APDem$_k$*** list with the current assigned probabilistic demand *(APDem$_k$ list initially is empty. Every time we add one more probabilistic demand this demand is added to APDem$_k$ list)*

The **InsertionPRH** sub-algorithm works as follows:

Step 0:

Sort ***APDem$_k$*** *list* in descending order of occurrence probability *(the sort procedure is meaningful in that we will always remove the less probable probabilistic demand from the solution)*

Step 1:

*Repeat the following until Real Demand has been assigned or **APDem$_k$** set is empty*

  1. Try to Assign the Real Demand
  2. If the assignment of real demand is not successful then remove the last probabilistic demand from the ***APDem$_k$*** set *(in this step we remove one by one the probabilistic demands –starting by with the on less probable to occur - if the real demand cannot be assigned. The purpose of this*

removal is to relax the solution as much as needed in order to assign the real demand)

A description of the OP-DARP algorithm in pseudo code is as follows:

```
Step0: For each vehicle v (v=1,2,3…..|V|)
        Build an empty route
 Step 1: Define THStep
Step2: For each time $t_c$ reading
     Step 2.1:
        call InsertionProbH((TH(TR($t_c$)))
        call InsertionRPH (TR($t_c$))
        WithdrawProbH(.)
     Step 2.2:
        Call  RegretPH($t_c$) for solution RS($t_c$)

Sub-algorithm InsertionProbH((TH(TR($t_c$)))
Step 0:   Build $PDem_k$ set for time horizon defined by TH(TR($t_c$))
          Build $SPDem_k$ set from $PDem_k$
          Sort $SPDem_k$ set in descending order concerning the show-up probability
          $APDem_k = \emptyset$
Step1: for every PD  in $SPDem_k$
           Call InsertionTH (TH(TR($t_c$)))  algorithm for PD
          if PD is assigned then $APDem_k = APDem_k \cup PD$

Sub-algorithm InsertionPRH (TR($t_c$))
Step0: Repeat the following until (TR($t_c$) is assigned or $APDem_k = \emptyset$)
              Step1.1 Call InsertionTH ($t_c$) algorithm to assign TR($t_c$)
              Step 1.2 If real demand TR($t_c$)  can't be assigned then $APDem_k = APDem_k \cap$
              (Probabilistic Demand with the less show-up probability in $APDem_k$)
Step 1: Evaluate solution RS($t_c$)
```

Computational effort of OP-DARP, given n demands, is the sum of the computational effort of:

1. **The InsertionProbH and the InsertionPRH** sub-algorithms we use in order to assign probabilistic demands and real demands into the solution. This sub algorithm is called whenever new real demand occurs. Suppose that the whole number of probabilistic demands during the operation period – one day in our case – is *pn* and the number of real demands is *n.* In the worst case for every real demand we may use the entire set of probabilistic demands. Because of this, the total number of possible searches is described by the following formula

$$\sum_{m=1}^{n}\sum_{i=m}^{m+p}\sum_{k=1}^{2i+1}k=\frac{1}{6}(17n+15n^{2}+4n^{3}+34np+21n^{2}p+4n^{3}p+21np^{2}+6n^{2}p^{2}+4np^{3}),($$

4-25). Where *n* is the number of real demands and *p* is the maximum number of selected probabilistic demands. The new formula is different from the (4-1) formula. This is due to the fact that for every real demand algorithm we should assign – in the worst case – *p* probabilistic demands. The term $\sum_{i=m}^{m+p}\sum_{k=1}^{2i+1}k$ expresses exactly this usage of probabilistic demands. By using 4-25 the complexity of this formula is estimated as *O(n^3+np^3+ n^2p^2)(4-26).*

2. The Procedure *RegretH*. For this procedure the complexity has been estimated at section 4.1.2 as rO(*mn^2*) where *n* is the number of real demands, r is the number optimization cycles (iterations in order to optimize the solution)

The total complexity is O(*n^3+np^3+ n^2p^2+)+rO(mn^2*)(4-27). The number *n* of trip demands and the number *p* of probabilistic demands clearly dominate the computational load.

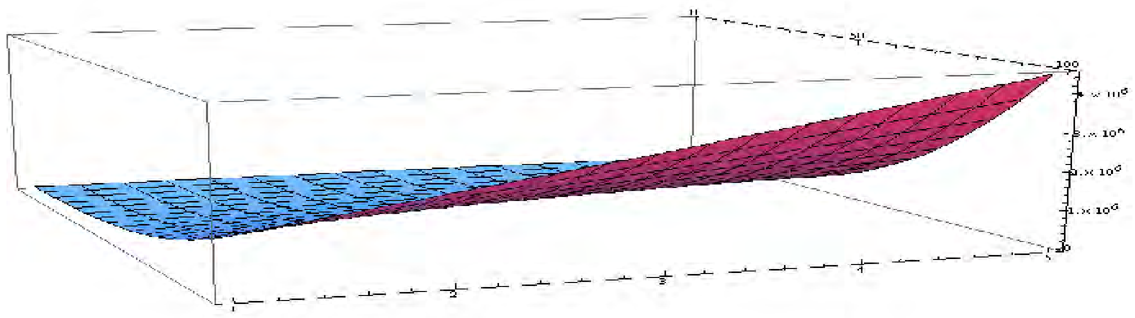


Figure 5-11.Computational effort of the OP-DARP. (x-axis describes the number of trip demands (1-100), y-axis (1-5) describes the number of probabilistic demands, and z-axis is the computational effort (0- 4E-6)

The above figure shows clearly the way that probabilistic demands heavily affect the computational effort in comparison with real demands.

5.2.4 Algorithm computational analysis

The proposed algorithm was implemented in C++ and tested on a Linux machine with a dual core processor. To evaluate the OP-DARP algorithm performance, real data were used, obtained from a pilot project in the municipality of Philippi. Based on these data we extracted an accurate number of every day real demands. For every real demand there is the exact time of the users' phone call. This information is valuable because it can be used in order to emulate the online operation. Since the pilot project provided data for 30 days of continuous operation, we can use these data to build an appropriate probability distribution regarding the occurrence probability for every trip demand presented during the pilot testing.

To properly conduct our experiments we need at least the following:

1. Real trip demands
2. Probability distribution which describes accurately the occurrence probability for all probabilistic demands.

Online trip demands: We chose to experiment with the most "loaded" days of the month. Wednesday was the most loaded day during a week period for all week periods. Based on these observations we experimented with 5 Wednesdays.

Probability distribution: Data from 30 days were analyzed in order to find the Probability distribution that describes accurately the occurrence probability for all trip demands that emerged during the pilot testing. Due to unsatisfactory fit for all known (Poisson etc.)probability distributions to our data, the empirical distribution was chosen.

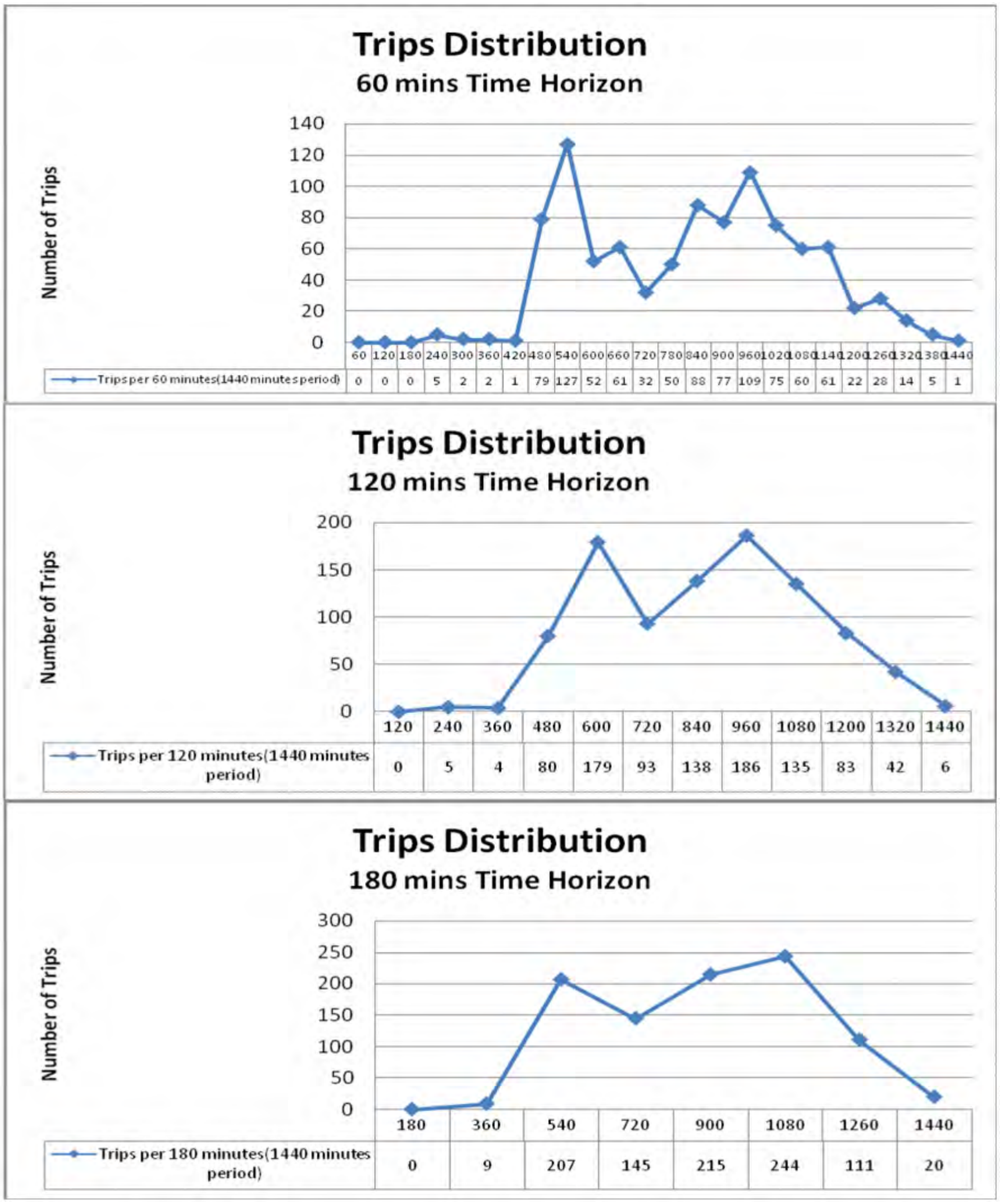
Table 5.11 represents the total number of trips emerged in a certain time horizon during the pilot operation. The use of this table is based on its entries; we can construct the set $\boldsymbol{PDem_k}$ of probabilistic trips by using a specific – empirical in our case - probability distribution.

- The 1st column describes the specific time horizons according to time step horizon

- The 2nd column describes the number of trip demands for each time fragment
- The 3rd and the 5th column are the same as the 1st column, while the 4th and the 6th are the same as the 2nd column

Table 5-11: Total Trip Demands Distribution for three time horizons with 60, 120, 180 minutes time step accordingly

Time Hor. 60 Mins		***Time Hor. 120 Mins***		***Time Hor. 180 Mins***	
Time	Demands	Time	Demands	Time	Demands
60	0	120	0	180	0
120	0	240	5	360	9
180	0	360	4	540	207
240	5	480	80	720	145
300	2	600	179	900	215
360	2	720	93	1080	244
420	1	840	138	1260	111
480	79	960	186	1440	20
540	127	1080	135		
600	52	1200	83		
660	61	1320	42		
720	32	1440	6		
780	50				
840	88				
900	77				
960	109				
1020	75				
1080	60				
1140	61				
1200	22				
1260	28				
1320	14				
1380	5				
1440	1				



	60	120	180	240	300	360	420	480	540	600	660	720	780	840	900	960	1020	1080	1140	1200	1260	1320	1380	1440
Trips per 60 minutes(1440 minutes period)	0	0	0	5	2	2	1	79	127	52	61	32	50	88	77	109	75	60	61	22	28	14	5	1

	120	240	360	480	600	720	840	960	1080	1200	1320	1440
Trips per 120 minutes(1440 minutes period)	0	5	4	80	179	93	138	186	135	83	42	6

	180	360	540	720	900	1080	1260	1440
Trips per 180 minutes(1440 minutes period)	0	9	207	145	215	244	111	20

Figure 5-12. Trips Distribution for time horizons of 60,120,180 time steps minutes accordingly

The evaluation of the OP-DARP performance was made through a series of specific experiments. Data for these experiments – as mentioned before – came through a pilot program in the municipality of Philippi. These experiments were designed in order to draw conclusions about:

1. The way that probabilistic demands affect solution quality
2. The way that regret optimization procedure in combination with probabilistic demands affect solution quality

Some details about the experiments environment are:

1. *Fleet size of 25 vehicles.*
2. *Vehicle Capacity 12 seats.*
3. *Maximum pickup wait time 15 minutes*
4. *Maximum trip demand ride time 1.5 time the absolute shortest path time*
5. *Work period was one day (or 1440 minutes)*
6. *Three time horizons 60, 120, 180 minutes*
7. *Monte Carlo Repetitions for each experiment was 100*

Computational results after the application of the OP-DARP algorithm are presented in the tables 5-12,5-14,5-16,5-18 5-20.

Where:

- Column C1 represents the number of experiment (1 to 5). Each number is associated with a specific number of real demands emerged in each one of the five Wednesdays during the pilot testing.

  1 =First Wednesday.

  2 = Second Wednesday.

  3 = Third Wednesday.

  4 = Fourth Wednesday.

  5 = Fifth Wednesday.
- Column C2 represents the number of real demands for the experiment number denoted by column C1
- Column C3 represents the time step horizon.
- Column C4 represents the average cost (the objective) produced by the repetitive runs of OP-DARP algorithm.

- Column C5 represents the average vehicles number produced by the repetitive runs of OP-DARP algorithm.
- Column C6 represents the average passengers number per distance unit produced by the repetitive runs of the OP-DARP algorithm
- Column C7 represents the standard deviation of all solution costs produced by the repetitive runs of OP-DARP algorithm.
- Column C8 represents the percentage difference between the cost of OP-DARP without probabilistic demands (OR-DARP) and the average solution costs of the OP-DARP algorithm
- Column C9 represents the confidence interval (95% confidence level) for all solution costs produced by OP-DARP algorithm.
- Column C10 represents the lower mean cost value produced by OP-DARP algorithm within the column C9 confidence interval
- Column C11 represents the upper mean cost value produced by OP-DARP algorithm within the column C9 confidence interval

Each grey row represents the solution produced by OR-DARP algorithm.

Every framed cell represents the solution cost which is less than the solution cost presented by the grey row.

Another variant of OP-DARP algorithm is the OP-DARP NoRegret algorithm that uses probabilistic demands but without Regret optimization. The basic parameters for those experiments are described in the tables 5-13,5-14,5-17,5-19,5-21 where:

- Column C21 represents the number of experiment (1 to 5).
- Column C22 represents the number of real demands for the experiment number denoted by column C1
- Column C23 represents the time step horizon.
- Column C24 represents the average cost (the objective) produced by the repetitive runs of the OP-DARP NoRegret algorithm.
- Column C25 represents the average vehicles number produced by the repetitive runs of the OP-DARP NoRegret algorithm.
- Column C26 represents the average passengers number per distance unit produced by the repetitive runs of OP-DARP NoRegret algorithm
- Column C27 represents the standard deviation of all solution costs produced by the repetitive runs of the OP-DARP NoRegret algorithm.

- C28 represents the percentage difference between costs of OR-DARP (grey row) and the average solution cost of the OP-DARP NoRegret algorithm
- Column C29 represents the confidence interval (95% Confidence Level) of all solution cost produced by the OP-DARP NoRegret algorithm.
- Column C30 represents the lower mean cost value produced by the OP-DARP NoRegret algorithm within the column C29 confidence interval
- Column C31 represents the upper mean cost value produced by OP-DARP NoRegret algorithm within the column C29 confidence interval

Table 5-12: Algorithm OP-DARP results for experiment #1

C1	*C2*	*C3*	*C4*	*C5*	*C6*	*C7*	*C8*	*C9*	*C10*	*C11*
1	76	NO(OR-DARP)	627500	6	1.134	0	0	0	0	0
	76	60	599150	10.5	1.252	2599.679	-4.52%	1611.268	597538.7	600761.3
	76	120	613100	11.5	1.238	2902.489	-2.29%	1798.948	611301.1	614898.9
	76	180	619090	11.5	1.257	7053.99	-1.34%	4372.028	614718	623462

Table 5-13: Algorithm OP-DARP NoRegret results for experiment #1

C21	*C22*	*C23*	*C24*	*C25*	*C26*	*C27*	*C28*	*C29*	*C30*	*C31*
1	76	NO(OR-DARP)	627500	6	1.134	0	0	0	0	0
	76	60	842360	22	1.070	20468.9	34.24%	12686.52	829673.5	924498.8
	76	120	902450	21.5	1.014	35575.1	43.82%	22049.27	880400.2	924498.8
	76	180	831950	22.5	1.131	6061.032	32.58%	3756.598	828193.4	835706.6

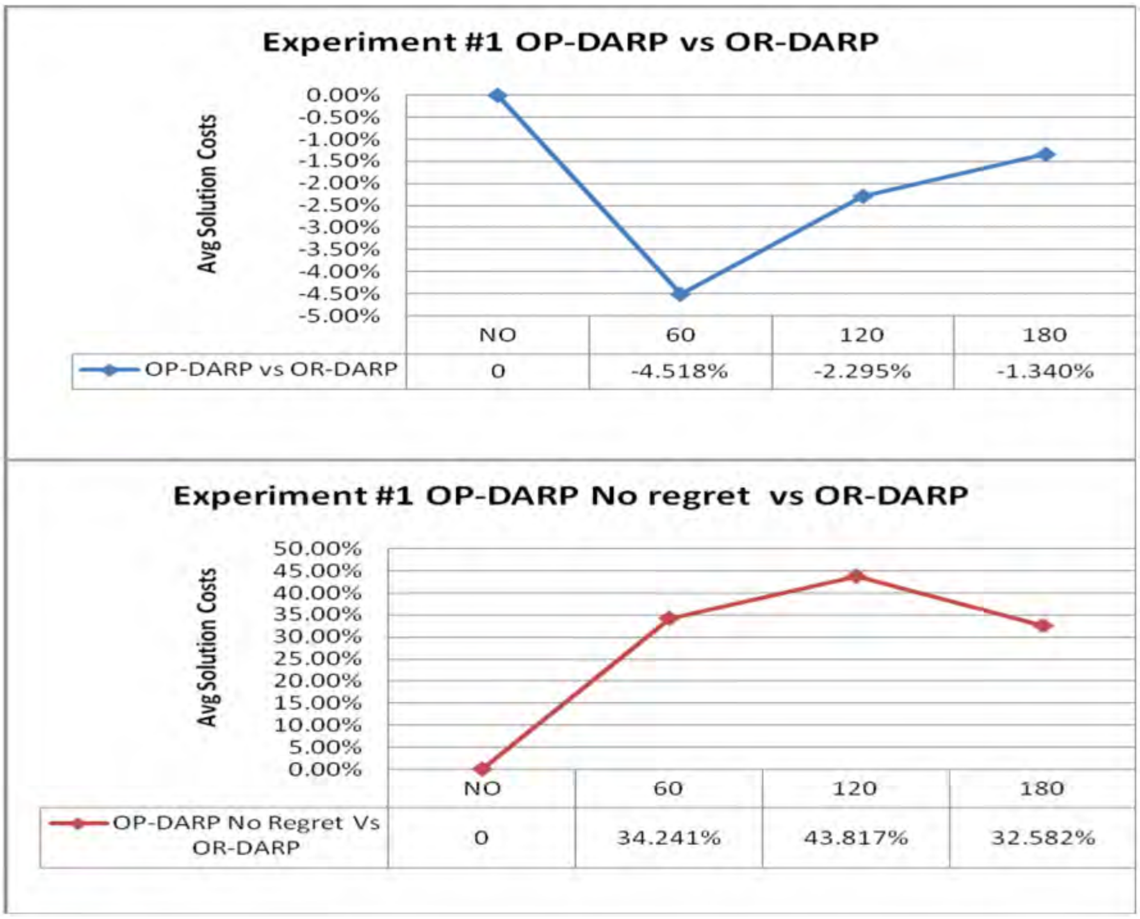


Figure 5-13. Comparison between the OP-DARP AND OP-DARP NoRegret solution costs (exper #1)

Table 5-14: Algorithm OP-DARP results for experiment #2

C1	*C2*	*C3*	*C4*	*C5*	*C6*	*C7*	*C8*	*C9*	*C10*	*C11*
2	70	NO(OR-DARP)	700400	6	1.157	0	0	0	0	0
	70	60	658300	10	1.325	737.8648	-6.01%	457.3249	657842.7	658757.3
	70	120	659300	11.5	1.320	1475.73	-5.87%	914.6499	658385.4	660214.6
	70	180	663900	10.5	1.320	5241.925	-5.21%	3248.919	660651.1	667148.9

Table 5-15: Algorithm OP-DARP NoRegret results for experiment #2

C21	*C22*	*C23*	*C24*	*C25*	*C26*	*C27*	*C28*	*C29*	*C30*	*C31*
2	70	NO(OR-DARP)	700400	6	1.157	0	0	0	0	0
	70	60	791400	21	1.206	17563.85	12.99%	10885.99	780514	802286
	70	120	829800	23.5	1.262	4427.189	18.48%	2743.95	827056.1	832543.9
	70	180	862150	21.5	1.156	24191.42	23.09%	14993.72	847156.3	877143.7

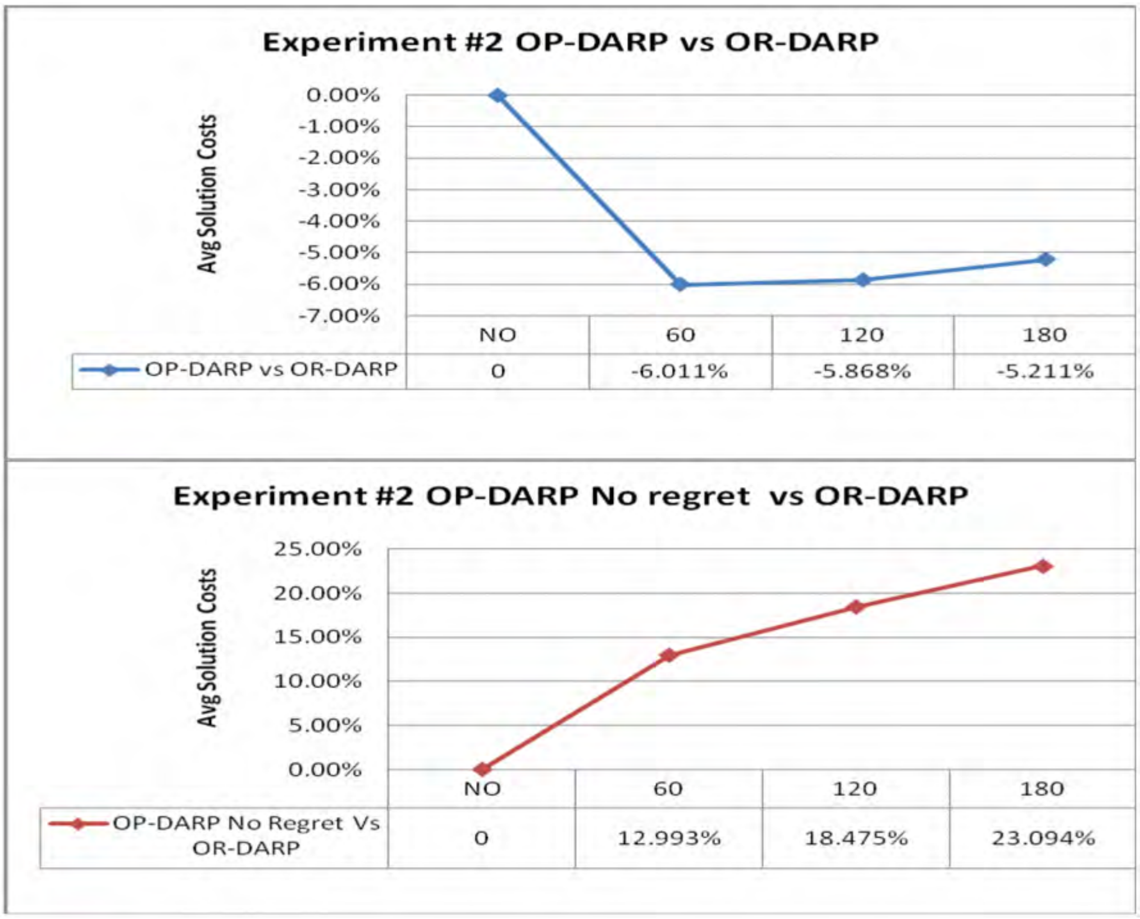


Figure 5-14. Comparison graph between the OP-DARP AND OP-DARP NoRegret solution costs (exper #2)

Table 5-16: Algorithm OP-DARP Details for experiment #3

C1	*C2*	*C3*	*C4*	*C5*	*C6*	*C7*	*C8*	*C9*	*C10*	*C11*
3	99	NO(OR-DARP)	656100	5	1.202	0	0	0	0	0
	99	60	665600	10.5	1.300	1054.093	1.45%	653.3213	664946.7	666253.3
	99	120	675300	12.5	1.325	421.637	2.93%	261.3285	675038.7	675561.3
	99	180	658550	12.5	1.367	3109.573	0.37%	1927.298	656622.7	660477.3

Table 5-17: Algorithm OP-DARP NoRegret results for experiment #3

C21	*C22*	*C23*	*C24*	*C25*	*C26*	*C27*	*C28*	*C29*	*C30*	*C31*
3	99	NO	656100	5	1.202	0	0	0	0	0
	99	60	937850	21.5	1.080	7747.58	42.94%	4801.912	933048.1	942651.9
	99	120	1006750	24	1.009	20080.46	53.44%	12445.77	994304.2	1019196
	99	180	966950	24	1.101	21767.01	47.38%	13491.09	953458.9	980441.1

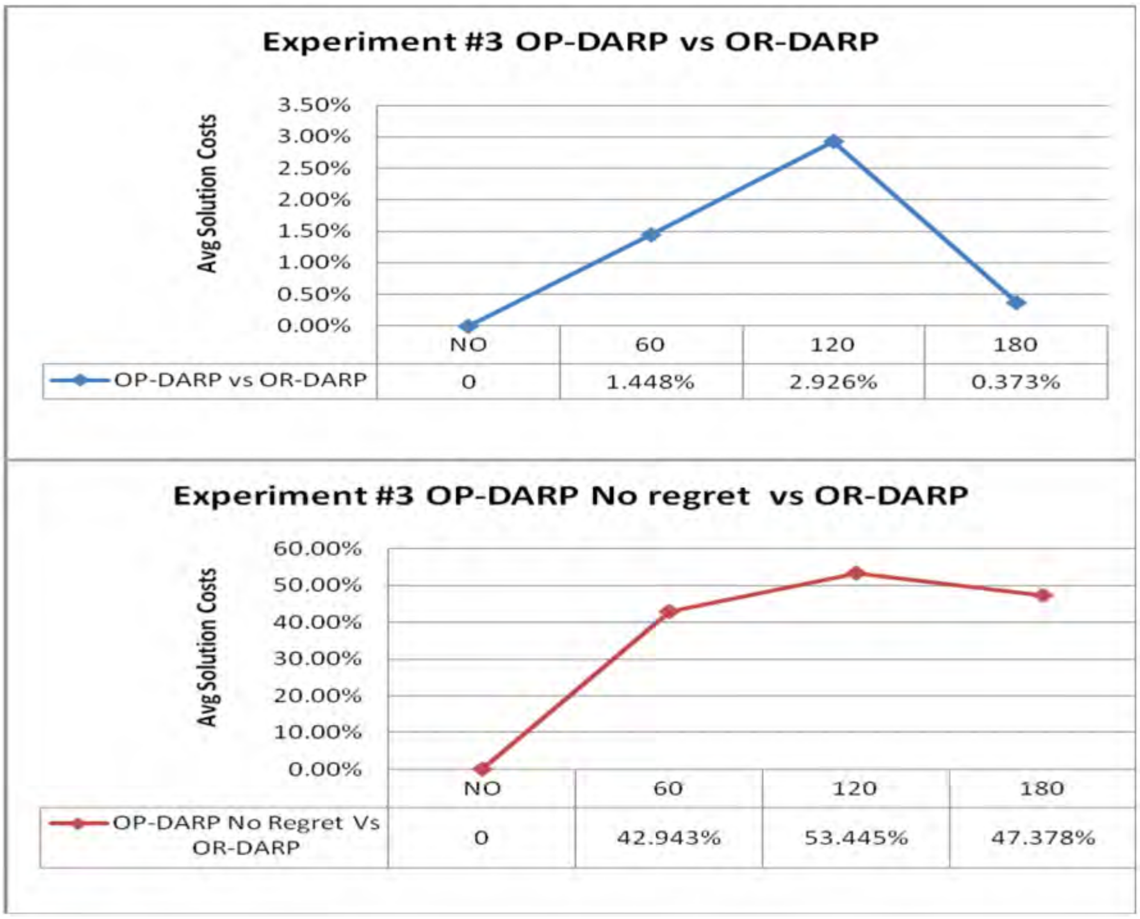


Figure 5-15. Comparison graph between the OP-DARP AND OP-DARP NoRegret solution costs (exper #3)

Table 5-18: Algorithm OP-DARP results for experiment #4

C1	*C2*	*C3*	*C4*	*C5*	*C6*	*C7*	*C8*	*C9*	*C10*	*C11*
4	101	NO(OR-DARP)	615000	6	1.420	0	0	0	0	0
	101	60	614390	10.4	1.568	2160.993	-0.10%	1339.372	613050.6	615729.4
	101	120	611300	11.8	1.675	994.4289	-0.60%	616.3421	610683.7	611916.3
	101	180	613790	11.9	1.695	619.0495	-0.20%	383.6838	613406.3	614173.7

Table 5-19: Algorithm OP-DARP NoRegret results for experiment #4

C21	*C22*	*C23*	*C24*	*C25*	*C26*	*C27*	*C28*	*C29*	*C30*	*C31*
4	101	NO(OR-DARP)	615100	6	1.420	0	0	0	0	0
	101	60	1073900	22.2	0.934	18529.26	74.59%	11484.34	1062416	1085384
	101	120	1119800	22.2	0.884	26267.85	82.05%	16280.68	1103519	1136081
	101	180	972950	21.9	1.032	6317.744	58.18%	3915.706	969034.3	976865.7

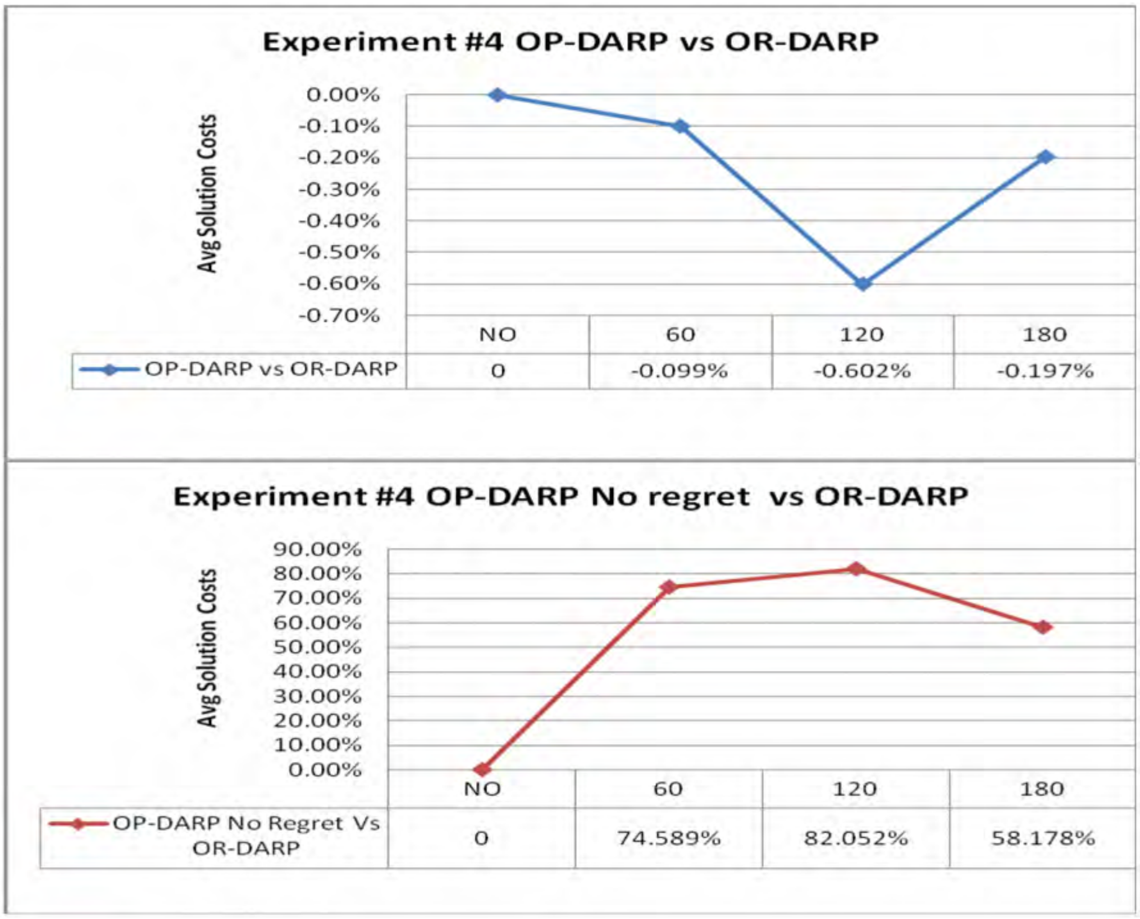


Figure 5-16. Comparison graph between OP-DARP AND OP-DARP NoRegret solution costs (exper #4)

Table 5-20: Algorithm OP-DARP results for experiment #5

C1	*C2*	*C3*	*C4*	*C5*	*C6*	*C7*	*C8*	*C9*	*C10*	*C11*
5	73	NO(OR-DARP)	531500	6	1.280	0	0	0	0	0
	73	60	555940	9.9	1.303	2416.241	4.60%	1497.574	554442.4	557437.6
	73	120	533441	10.4	1.425	4941.944	0.37%	3062.992	530378	536504
	73	180	529450	9.2	1.356	1423.025	-0.39%	881.9838	528568	530332

Table 5-21: Algorithm OP-DARP NoRegret results for experiment #5

C21	*C22*	*C23*	*C24*	*C25*	*C26*	*C27*	*C28*	*C29*	*C30*	*C31*
5	73	NO(OR-DARP)	531500	6	1.280	0	0	0	0	0
	73	60	833060	22.1	0.971	3850.31	56.74%	2386.403	830673.6	835446.4
	73	120	811940	22.7	1.067	7148.924	52.76%	4430.867	807509.1	816370.9
	73	180	854330	21.1	0.893	5378.155	60.74%	3333.354	850996.6	857663.4

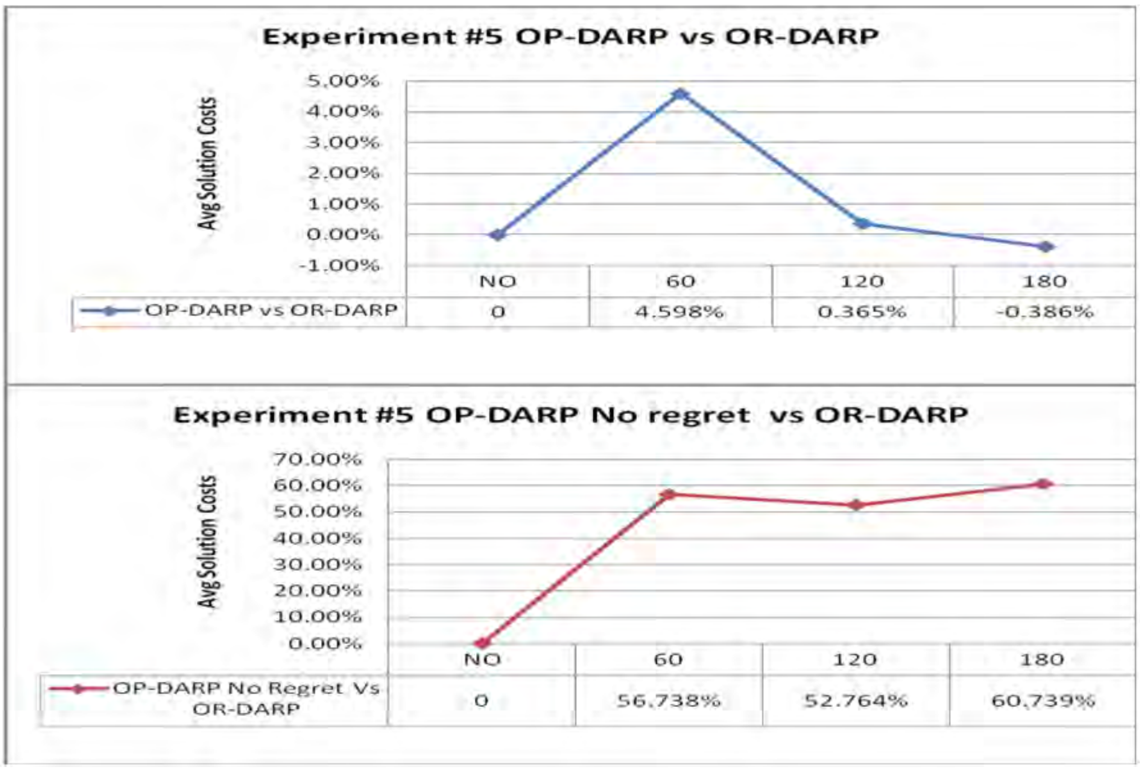


Figure 5-17. Comparison graph between OP-DARP AND OP-DARP NoRegret solution costs (exper #5)

Table 5-22: Comparison results for average solution costs between OR-DARP, OP-DARP algorithms

Average Number of Trip Demands	*Average Solution Cost produced by OR-DARP*	*Average Solution Cost produced by OP-DARP*	*Confidence Interval for OP-DARP Solutions(95% confidence level)*	*Solution Cost produced by OP-DARP (Lower Limit)*	*Solution Cost produced by OP-DARP (Uper Limit)*	*Imporvment Percentage*
83.80	626100	618040	7423	610617	625463	1.287%

By observing tables 5-12,5-14,5-16,5-18,5-20 we conclude:

1. The OP-DARP algorithm gave better solutions for four out of five experiments in comparison with the OR-DARP algorithm.
2. The maximum improvement percentage is 6.01%. However in experiment #3 OP-DARP produced no improvement at all.

By observing tables 5-13,5-15,5-17,5-19,5-21 we conclude:

1. The OP-DARP NoRegret algorithm gave worse solutions for all experiments in comparison with the OR-DARP algorithm and always worse solutions in comparison to OP-DARP algorithm.
2. The best solution obtained by OP-DARP NoRegret is by 12.99% worse in comparison with the solution obtained by OR-DARP algorithm. While the worst solution obtained by OP-DARP NoRegret is by 82% worse in comparison with the solution obtained by OR-DARP algorithm.

By observing table 5-22 we conclude that:

1. The average solution cost obtained by the OP-DARP algorithm is lower by 1.29% in comparison with the average solution cost obtained by the OR-DARP algorithm.
2. With 95% confidence level the average solution cost by the OP-DARP algorithm is always lower than the solution cost obtained by the OR-DARP algorithm.

A more detailed analysis is provided in table 5-23.

Column C101 represents the experiment number (1 to 5).

Column C102 represents the number of real demands for each experiment denoted by column C101.

Column C103 represents the cost (the objective) of the produced solution by the OP-DARP algorithm.

Column C104 represents the solution cost (the objective) produced by the OP-DARP No Regret algorithm.

Column C105 represents the solution cost (the objective) produced by the OR-DARP algorithm.

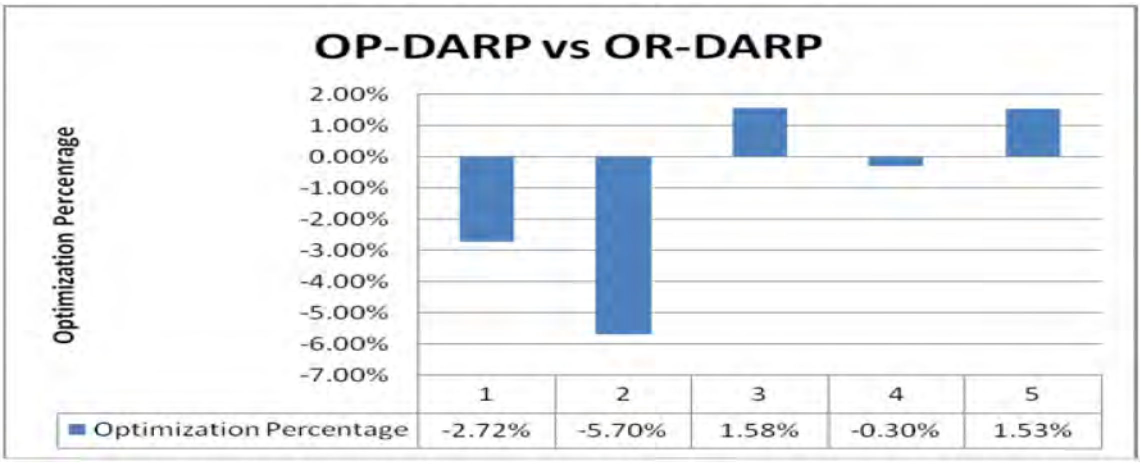
Column C106 represents the percentage difference between OR-DARP and OP-DARP algorithms solution costs (Column C103, Column C105).

Column C107 represents the percentage difference between OR-DARP and OP-DARP NoRegret algorithms solution costs (Column C104, Column C105).

Column C108 represents the percentage difference between OP-DARP and OP-DARP NoRegret algorithms solution costs (Column C103, Column C104).

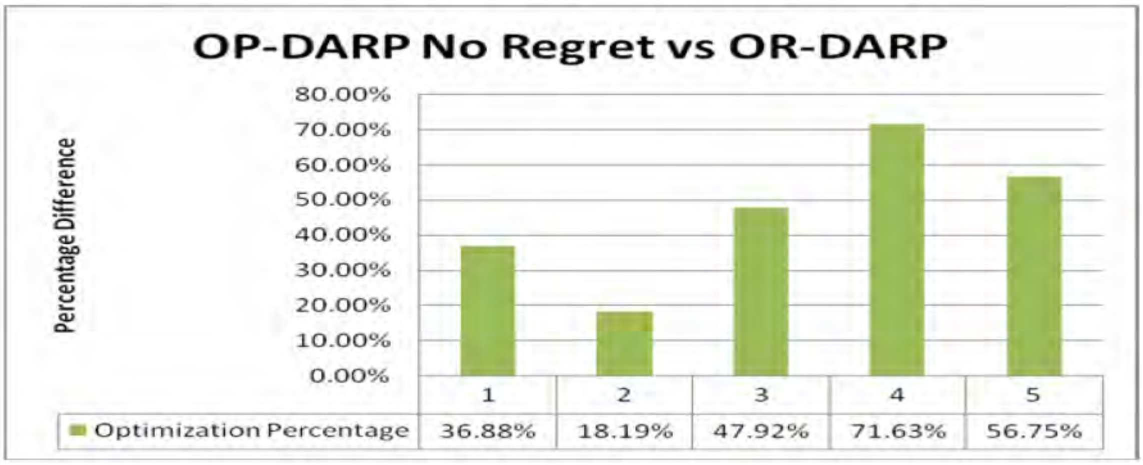
Table 5-23: Comparison results between OP-DARP, OP-DARP No Regret, OR-DARP algorithms

C101	*C102*	*C103*	*C104*	*C105*	*C106*	*C107*	*C108*
1	76	610447	858920	627500	-2.72%	36.88%	40.70%
2	70	660500	827783	700400	-5.70%	18.19%	25.33%
3	99	666483	970517	656100	1.58%	47.92%	45.62%
4	101	613160	1055550	615000	-0.30%	71.63%	72.15%
5	73	539610	833110	531500	1.53%	56.75%	54.39%



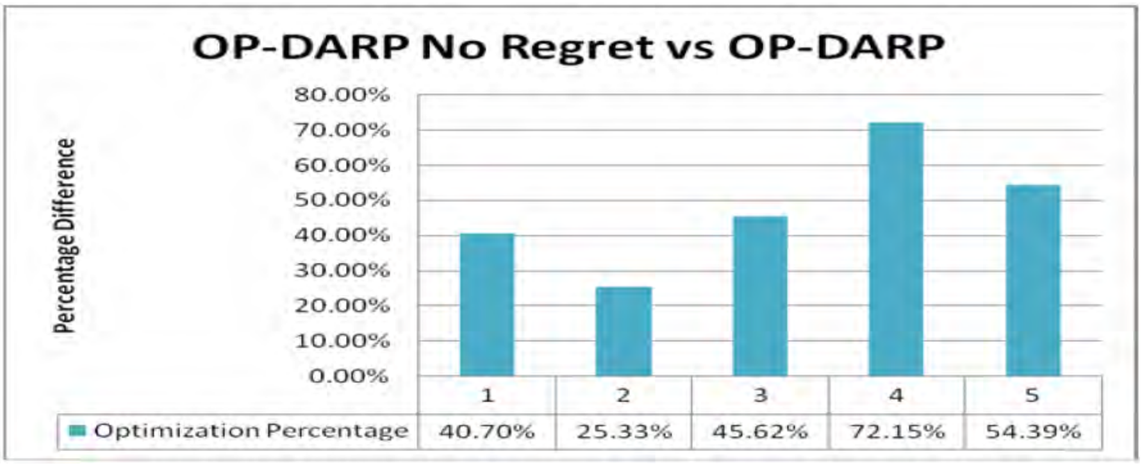
	1	2	3	4	5
■ Optimization Percentage	-2.72%	-5.70%	1.58%	-0.30%	1.53%

Figure 5-18. Comparison graph between OP-DARP , OR-DARP solution costs for all experiments(1 to 5)



	1	2	3	4	5
■ Optimization Percentage	36.88%	18.19%	47.92%	71.63%	56.75%

Figure 5-19. Comparison graph between OP-DARP No Regret ,OR-DARP solution costs for all experiments



	1	2	3	4	5
■ Optimization Percentage	40.70%	25.33%	45.62%	72.15%	54.39%

Figure 5-20. Comparison graph between OP-DARP No Regret , OP-DARP solution costs

By observing table 5-23 we conclude the following:

1. The usage of probabilistic demands combined with regret optimization procedure, gives better solutions than the usage of regret optimization alone. It is clear that the regret optimization procedure is critical for the solution quality, but the usage of probabilistic demands improves the solution further.
2. The usage of probabilistic demands did not provide better solutions than the regret optimization. Algorithm OP-DARP No Regret gave always worse solutions. We conclude that the usage of probabilistic demands without optimization cannot guarantee better solutions.

The number of vehicles used by the OP-DARP algorithm in comparison to the number of vehicles used by OR-DARP is presented in table 5-24.

Column C201 represents the number of experiment 1 to 5.

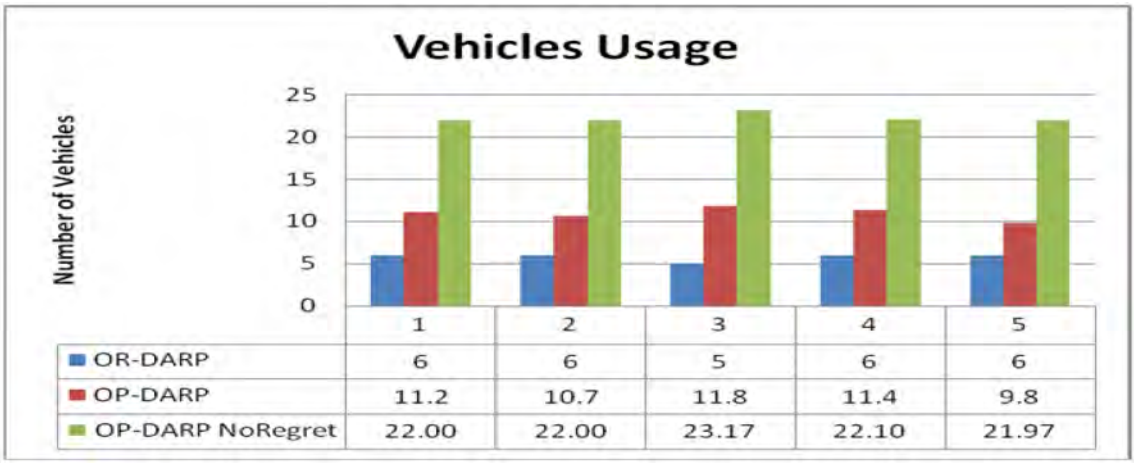
Column C202 represents the number of vehicles used by the OR-DARP algorithm.

Column C203 represents the average number of vehicles used by the OP-DARP algorithm.

Column C204 represents the average number of vehicles used by the OP-DARP No Regret algorithm

Table 5-24: Vehicle Number Comparison results between OP-DARP, OR-DARP algorithms

C201	*C202*	*C203*	*C204*
1	6	11.2	22.00
2	6	10.7	22.00
3	5	11.8	23.17
4	6	11.4	22.10
5	6	9.8	21.97



	1	2	3	4	5
🟦 OR-DARP	6	6	5	6	6
🟥 OP-DARP	11.2	10.7	11.8	11.4	9.8
🟩 OP-DARP NoRegret	22.00	22.00	23.17	22.10	21.97

Figure 5-21.Comparison graph between OR-DARP, OP-DARP, OP-DARP No Regret vehicles usage for all experiments

By observing table 5-24 we conclude that the number of vehicles used by the OP-DARP algorithm is significantly larger than the number used by OR-DARP algorithm. This can be explained as side effect of the probabilistic demands usage. Algorithm OP-DARP increases the number of vehicles required in order to satisfy all trip demands – stochastic and real ones. After the withdrawal of stochastic demands the number of vehicles remains high, although the total solution cost is decreased in comparison with the solution cost produced by the OR-DARP algorithm.

Table 5-25 presents monthly comparison results between the OR-DARP algorithm and the OP-DARP algorithm.

First column represents the day of the month

Second column represents the solution cost produced by OR-DARP algorithm

Third column represents the average (for all time step horizons) solution costs produced by OP-DARP

Fourth column represents the percentage difference between second and third column.

Table 5-25: One Month Comparison results between OR-DARP , OP-DARP algorithms

Month	*OR-DARP Cost*	*OP-DARP Average*	*% difference*
(Day)		*solution Cost*	
Day-02	397400	390800	-2%
Day-03	402700	407433	1%
Day-06	355800	350867	-1%
Day-07	331700	323633	-2%
Day-09	440900	424467	-4%
Day-10	570000	557567	-2%
Day-13	387600	384300	-1%
Day-14	422600	444967	5%
Day-16	552400	550833	0%
Day-17	821300	795467	-3%
Day-20	427100	418700	-2%
Day-21	529800	479017	-10%
Day-23	560100	556533	-1%
Day-24	554000	545000	-2%
Day-27	414000	405000	-2%
All Days	716740	703458	-2%

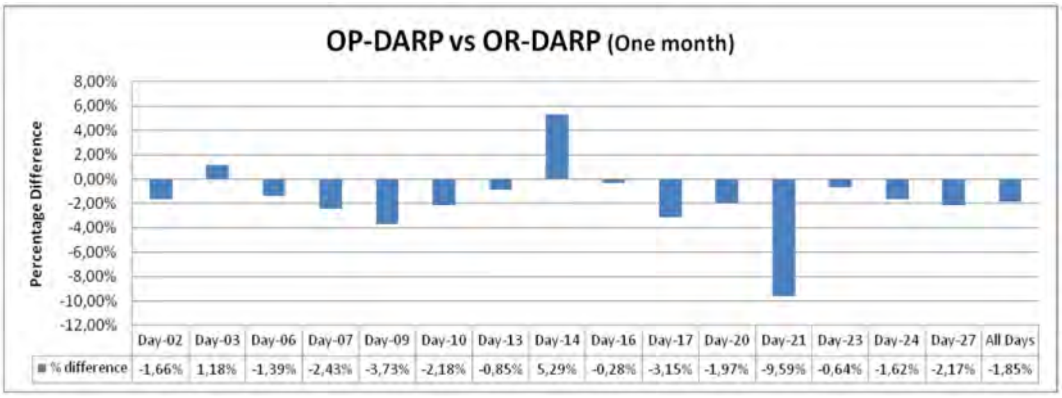


Figure 5-22. One Month Comparison graph between OR-DARP, OR-DARP

By observing table 5-25 we conclude that for two days (Day-14,Day-03), the OP-DARP algorithm gave worse results than the algorithm OR-DARP, while for all other days algorithm gave better results.

Table 5-25 shows clearly, that for the majority of the test days the solution was improved. The last line of Table 5-25 shows clearly that for the total monthly solution cost, the performance of the OP-DAR algorithm was better as well.

5.2.5 Online Regret Algorithm with Probabilistic Demand: Analysis and Conclusions

The proposed OP-DARP algorithm is expected to produce better solutions than the online regret algorithm that does not take into account the probability distribution of the demands. The usage of probabilistic demands offers to any online dial-a-ride algorithm the possibility to reduce the myopic behavior because of the “advanced knowledge” of the future incoming trip demands. In fact, the complete knowledge of the future alters the nature of the algorithm from online to static. Three main questions should be answered prior to the use of an online algorithm that uses historical data. The first question is how far in the future the algorithm should look. The second one is how much history for a specific future the algorithm should examine. The third question is how important is the optimization procedure in comparison with the use of probabilistic demands.

Our experiments showed a behavior for the OP-DARP algorithm that can be concluded to the following list:

1. OP-DARP algorithm gave more optimized solutions (on average) than the OR-DARP Regret algorithm when applied to our problems.
2. OP-DARP algorithm uses always a larger number of vehicles in comparison with the OR-DARP algorithm.
3. The usage of probabilistic demands improves the solution quality, although the main improvement factor is always the optimization procedure, in our case being the regret method.

Chapter 6 : The use of the static heuristic regret to determine profitability of a proposed Demand Responsive Transportation system

6.1 Introduction

In order to achieve an economically viable DRT system, it is important to assess whether the proposed transportation system could be profitable. On-demand transportation services provided by private enterprises should be profitable so as to be attractive to investors. In order to investigate if an investment of this kind is profitable or not, a specific methodology that could provide the appropriate information is necessary. To our knowledge, no specific studies have been presented addressing this issue. On the other hand, simulation tools have been used extensively in order to provide some knowledge concerning the various parameters that affect the operation of on-demand transportation systems. These tools are very powerful in evaluating system performance and have been extensively utilized in the literature in a variety of fields, including transportation. Wilson et al [143] pioneered the use of simulation to compare different heuristics to assess the influence of the service area, the demand density and the service quality on the fleet size requirements. Ali Haghani et al [144] presented a study about a real problem concerning bus transit vehicle scheduling for the Mass Transit Administration (MTA) in Baltimore USA. He showed that the proposed models for Multiple Depot Vehicle scheduling problems (MDVS) and Multiple Depot Vehicle scheduling problems with route time constrains (MDVSRTC) offer improvement over MTA schedules by decreasing operational costs 5.77%. In another study, Sinoda [145] focused on the usability of DRT systems in large towns, especially in comparison with fixed route transportation systems. Luca Quadrifoglio et al [146][147](Quadrifoglio, Dessouky [146]; Quadrifoglio, Dessouky [147]) used simulation in order to propose an innovative concept that merges the flexibility of DRT systems with the low cost operability of fixed-route systems. The same author (Quadrifoglio, Dessouky, Ordonez)[148] used simulation tools to study the impact specific operating practices currently used by DRT providers have on productivity. They investigated the effect of using a zoning vs a no-zoning strategy in Los Angeles County and time-window settings on performance variables such as total trip miles, deadhead miles and fleet size. Simulation has been used in the above studies in order to present the usefulness of the proposed algorithms in terms of cost reduction or solution improvement. In the above referenced literature we notice that most of the studies use some kind of simulation techniques in order to prove that the algorithms they propose give better results in terms of cost. They are based on existing operational transportation systems and apply DARP algorithms in order to reduce it. None of them provides a methodology of how we can use the simulation based on Dial-a-ride algorithms in order to determine if a proposed DRT system is profitable or not.

In this chapter we propose a new methodology for defining and evaluating critical operational parameters in order to formulate a profitable transportation system, using the "Convergence Algorithm". This algorithm utilizes data and conclusions drawn from user surveys and their subsequent analysis in order to find a critical point where the system becomes profitable. The algorithm starts from an initial parameter set, drawn from the survey analysis, and works towards a final set where the system is profitable. User demands are projected using a model where demand is a monotonic cumulative increasing function as we move towards a specific direction through this projection. The process uses the static regret heuristic algorithm presented already.

6.2 *METHODOLOGY AND THE CONVERGENCE PROCCES*

6.2.1 Methodology in General

Basic elements of our methodology are: trip data, definition of key service parameters and the Converge Process. First we gather survey data which we then analyze in order to infer the population's transportation needs and service-quality related factors. The final step is the convergence process, where we make a guided search in order to define the critical point where the system becomes profitable. In more detail:

Step 1: *Production of trip demand data. A survey is carried out in order to draw the population's habits and preferences.*

Step 2: *Definition of key parameters. The most important factors, - as they are derived (in our case) by the survey- defining service quality and affecting profitability are:*

Charging price. *Or ticket price, represents the charging costs that users are willing to pay for a specific trip. It can either be proportional to the distance covered, or fixed for each specific trip or fixed for all trips. In our case the price is proportional to the shortest path distance*

Travel time. *Travel time represents how much time customers spend in-vehicle, in comparison with the time the customer would spend by using taxi if his travel path between pickup and delivery points was the shortest path. It is quite obvious that if the travel time approximates the SP travel time, it affects heavily the number of required vehicles – and thus the cost - to satisfy the demand.*

Pickup (deliver) time windows*. Pickup (deliver) Time windows represent the typical time windows for the pickup and delivery time. Tighter window times translate to more vehicles needed*

Step 3: *Run the Convergence Process. Static Regret algorithms were used to derive the exact system state in terms of costs and benefits. A convergence process is used to define the critical point at which the system reaches profitability.*

6.2.2 Data Sources

The most widely used method in order to identify the market potential is to conduct a survey by questionnaire. The survey can be formatted in many ways. It depends heavily on the information we like to extract from its analysis. Studies concerning survey formation can be used in order to assist in its construction. In our case, the survey technique used was stratified sampling where two sample size criteria are used. The first relates to the level of precision (or sampling error) and is set to be approximately 7%. The second criterion, the confidence level, equal to 95%. In accordance with the above criteria, that are often used to produce surveys concerning public opinion, the sample size in was 210 questionnaires, which was approximately 2% of the population (25,26,27). We used the survey to identify:

Daily or weekly trip habits (origin, destination, time of transit) in order to define a trip number for every week day

The user perception of the most interesting trip features for our model:

Charging Price, in relation to existing means of transport for a specific set of service quality parameters

Waiting time (pickup or deliver time windows)

Maximum trip travel time in relation to the shortest path distance (taxi time) travel time.

6.2.3 Data Projection

The model we use in order to obtain a visual perspective of the system parameters is a 3-d model where the x-axis represents the charging price range, the y-axis represents the travel time range (in relation to the absolute shortest path travel time) and the z-axis represents the pickup time window range. That model was chosen because it allows us to study profitability for each of the above mentioned key parameters while keeping the other two constant. The charging price was the target parameter for us.

Looking at the example in Figure 6-1, we see a number of cubes. Each cube contains a name (A..D) and a set of three numbers, which correspond to price-per-km, acceptable ride-time extension and acceptable time window.

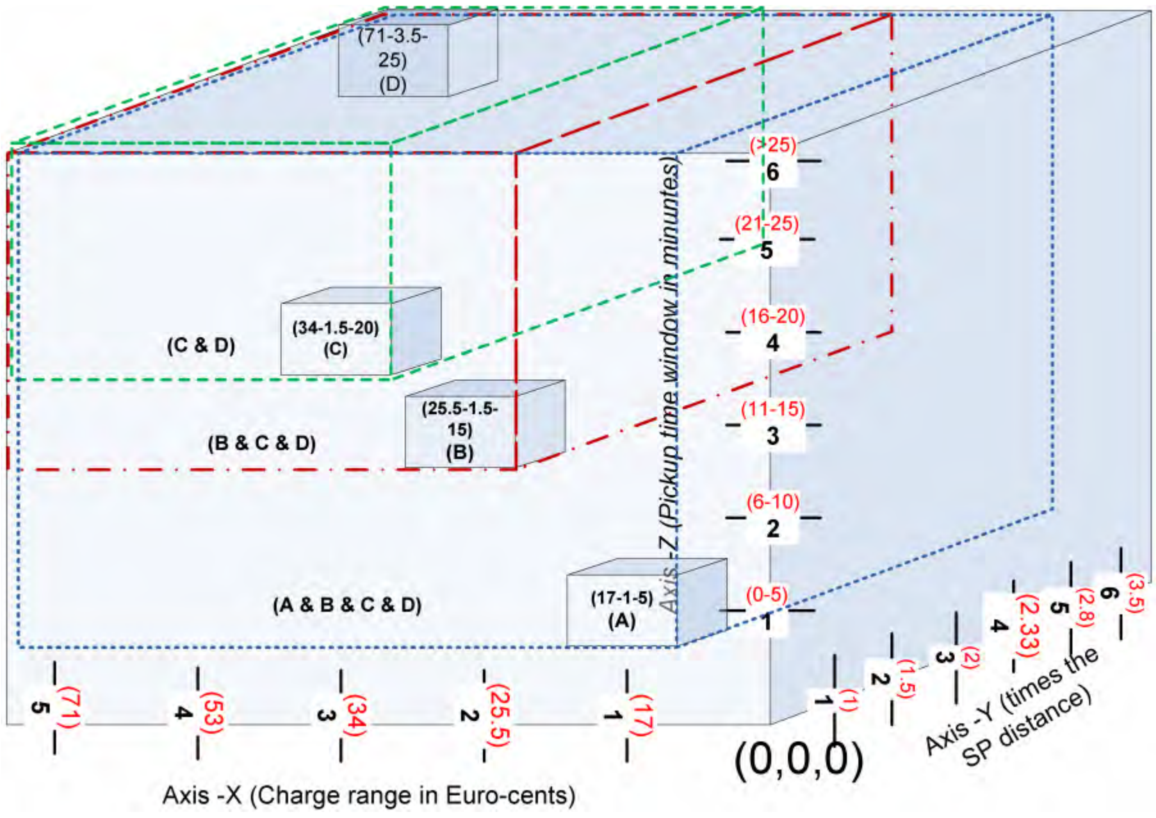
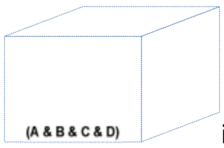


Figure 6-1.An example of Data Projection

Cube D represents the trip demands where customers are willing to pay the maximum price (71 euro-cents per Km), use the widest time window (>25 minutes) and tolerate the maximum ride time (3.5 times the time for the shortest path). Cube A represents the trip demands where customers that are willing to pay the minimum price (17 euro-cents), to use the minimum pickup/deliver window and tolerate a ride time equal to the minimum ride time (the shortest path time). We can observe that if a transportation system satisfies the user demands at B, then that system can satisfy every point inside the cube

(i.e. C and D). The total number of demands of every wrapping cube is the sum of all included sub-cubes. For example, the number of demands of the A 

is the sum of cubes (A), (B), (C), (D).

6.2.4 Nomeclature

X_i *(i=1,2,…M)…* represents the distinct points for axis *X (charging price)*

Y_j *(j=1,2,3…N)…* represents the distinct points for axis *Y (max trip time)*

Z_k *(k=1,2,3…K)…* represents the distinct points for axis *Z (time windows)*

TN_{ijk}…represents the total trips number included inside the wrapping cube X_i-Y_j-Z_k

TD_{ijk}… is the total distance covered by all vehicles when all trips inside the wrapping cube X_i-Y_j-Z_k are satisfied

TPD_{ijk}… is the total passengers distance – the sum of all SP trip distances - when all trips inside the cube X_i-Y_j-Z_k are satisfied

$f(TD_{ijk})$… is the function that returns the total cost when all trips inside the wrapping cube X_i-Y_j-Z_k are satisfied

TSC_{ijk} … is the total cost equal the result of ***$f(TD_{ijk})$*** function

$g(TPD_{ijk})$… is the function that returns total charging fees when all trips inside the wrapping cube X_i-Y_j-Z_k are satisfied

TCF_{ijk} … is the total cost charging fees equal to the result of ***$g(TD_{ijk})$*** function

P_{ijk} … is the profit/loss equal to ***TCF_{ijk}*** - ***TSC_{ijk}*** when all trips inside the wrapping cube X_i-Y_j-Z_k are satisfied

CoDU … is the real system cost per unit distance.

ChDU … is the customers charging price per unit distance.

Note that functions ***f(TD**$_{ijk}$**)** , ***g(TPD**$_{ijk}$**)*** are not always linear, meaning that they do not always have to follow the form "ax+b". Many factors affect their linearity, such as the type(s) of vehicles (whether the fleet is homogeneous or not), the type(s) of trips (urban, suburban) etc. In our case, the functions ***f(TD**$_{ijk}$**)** , ***g(TPD**$_{ijk}$**)*** were considered completely linear:

1. $f(TD_{ijk})=TD_{ijk} * CoDU$
2. $g(TPD_{ijk})=TPD_{ijk} * ChDU$

6.2.5 The convergence algorithm

In order to reach the desirable point where the system becomes profitable, we conduct an extensive guided search sequence by running multiple iterations of various dial-a-ride algorithms. The Convergence algorithm is designed to reach that point while maintaining the values of the service quality parameters unchanged. The first course of action is to decide the desired service quality parameter values (in our case: waiting time and maximum trip travel time). These values can be derived from the survey analysis, or they can be defined by the system provider. In our scenario, we used the survey data to derive these values by the use of the following rule:

- Find where the biggest "concentration" (cube with the largest number of trip demands) is, in terms of time window time and maximum travel trip time.
- If there are other points (cubes) with a similar number of trip demands (i.e. at least 80% the number of the "average" number of trip demands), we choose the one that has the tightest restrictions in terms of time window and maximum travel trip time. Average is defined as the sum of all trip demands in each cube, divided by the number of cubes.

The scale of time windows begins at 5 minutes and ends at >25 minutes. The scale of maximum travel trip time begins at 1 and ends at 3.5, and is based on the taxi SP time.

In the next step, we start the search by using the following rule:

We define a wrapping cube (X_i - Y_j - Z_k) where $j=\beta$, $k=\gamma$ represent the values, drawn from the previous step, of time windows and maximum trip travel time, as the start point. The initial charging price is the lowest possible (*i=1*). We run the dial-a-ride algorithms for that point ($X_{\alpha=1}$ - Y_β - Z_γ) for all trips $\boldsymbol{TN_{ijk}}$ included in that wrapping cube, without any restrictions on the number of vehicles. After the run has been executed, the algorithm returns the number of vehicles used in order to satisfy all trip demands, the total vehicles distance ($\boldsymbol{TD_{ijk}}$) and the total passengers distance ($\boldsymbol{TPD_{ijk}}$) where ($i=\alpha=1$, $j=\beta$, $k=\gamma$). By using these results we can easily calculate:

1. $\boldsymbol{TSC_{ijk} = f(TD_{ijk})}$
2. $\boldsymbol{TCF_{ijk} = g(TPD_{ijk})}$
3. $\boldsymbol{P_{ijk} = TCF_{ijk} - TSC_{ijk}}$

If the system produces (financial) loss for that X_i ($i=\alpha=1$), then we recalculate the charging price per distance unit by using the following formula: $\boldsymbol{ChDU = TSC_{ijk} / TPD_{ijk}}$ **.** After that, we select a point X_i ($i=p$) (where $p > \alpha$) such that its value is the first higher than the current ***ChD.*** Subsequently, the value of ***ChDU*** is updated to the new X_p. By using this new point X_i, we have to deal with a new wrapping cube (X_p - Y_β - Z_γ). It should be obvious that at that new point (X_p - Y_β - Z_γ), the wrapping cube contains less trip demands that the previous one.

On the other hand, if the system produces profit for that X_i ($i=\alpha$) then we follow the same procedure mentioned above, but in this case we select a point X_i ($i=p$) (where $p< \alpha$) with the first value lower than the new calculated ***ChDU*** value. It should be obvious that for that new point (X_p - Y_β - Z_γ), the wrapping cube contains more trip demands than the initial one.

The next step is to run Dial-a-Ride algorithms again. That process continues until we find the point where the calculation produces neither profit nor loss. If we identify that point, then we know that this point is the systems' *point of equilibrium*. Otherwise, if the algorithm oscillates infinitely between two points on the *x-axis* (charging prices), where the first one produces profit and the second one produces loss, then we select the one that produces profit as the equilibrium point.

The *"Convergence algorithm"* is presented in pseudo code and a chart as follows:

```
Step 1: Define service quality indicators (time windows & ride time) & their values either
  through data analysis or manually. Define the lowest ChDU value for those quality in-
  dicators
Step 2: Calculate the total number ($TN_{ijk}$) of demands included in the appropriate wrap-
  ping cube
Step 3: Run Dial-a-Ride algortihms and calculate total operating costs ($TSC_{ijk}$), total
  charging fees ($TCF_{ijk}$) and profit/loss ($P_{ijk}$)
Step 3: Calculate the new charge value by using the formula $ChDU= TSC_{ijk}./ TPD_{ijk}$
Step 4: If profits are negative ($P_{ijk}<0$) then {
          1. Identify the point on axis-x that has the first bigger charging value close to
               the previous calculated new charge value, which produces profit if applied
               to the current itinerary
          2. Update ChDU to that value
     } Else (i.e. if there is profit $P_{ijk}>0$) then {
          1. Identify the point on axis-x that has the first lower charging value close to
               the previous calculated new charge value, which produces loss if applied to
               the current itinerary
          2. Update ChDU  to that value
     } Else if ($P_{ijk}\cong 0$) {
        Goto Step 5
     }
     If (ChDU has a value that has not been found before at least once) {
        Goto Step 2
     } else {
        Goto Step 5
     }
Step 5: Exit
```

6.2.6 Test Case – The municipality of Phillipi

The rural Municipality of Philippi, Greece, was chosen for the test case. The Municipality consists of 19 villages with a total population of 10,827 inhabitants, most of which are elderly. The purpose of the survey was to investigate the market potential and the potential acceptance of the system in a rural community in Greece.

The questionnaire was specially designed to capture the potential acceptance of the inhabitants of the area towards the proposed system. Data was collected through was face-to-face interviews carried out by trained people that had a good knowledge of the area, the available modes of transport and the population’s mobility needs. In order to ensure that the results of the survey would be reliable, stratified sampling of approximately 2% (220 questionnaires) of the population was used. The questionnaire consisted of two parts:

The first part included general questions for the respondents (such as demographics).

The second part included questions that were meant to derive the basic design parameters of the proposed DRT system that would affect its profitability. In order to make the questionnaire more understandable to the people, some questions had predefined answers (multiple choices). The most important of these are presented in Appendix C, along with their answers:

For all possible combinations of charging price, travel time (as a multiple of the SP time) and pickup time window for those users who wish to use the system we have the cumulative number of projected trips for that wrapping cube that is described by those specific coordinates, according to the 3-d model.

In the Table 6-1 we present in detail the way in which the data concluded by the survey are organized.

Table 6-1: Cumulative number of trips (according to the 3-d model)

Charging price(cents/km) (x-Axis)	**Times the SP time (y-Axis)**	**Time window In minutes (z-Axis)**	**Wrapping cube (cumulative) number of trip demands**

1. (17)
2. (25.5)
3. (34)
4. (53.25)
5. (71)
	1. (1)
	2. (1.5)
	3. (2)
	4. (2.33)
	5.(2.8)
	6.(3.5)
		1.(0–5)
		2.(6–10)
		3.(11 – 15)
		4.(16 – 20)
		5.(21 – 25)
		6.(> 25)	

1	1	1	**1112**
	1	2	**954**
	1	3	**528**
	1	4	**38**
	1	5	**0**
	1	6	**0**
	2	1	**838**
1	2	2	**773**
	2	3	**477**
	2	4	**38**
	2	5	**0**
	2	6	**0**
..	..	..	**..**
2	2	2	**525**
..	..	..	**..**
3	2	2	**185**
..	..	..	**..**
..	..	..	
4	2	2	**38**
..	..	..	**..**
..	..	..	**..**
5	6	1	**0**
		2	**0**
		3	**0**
		4	**0**
		5	**0**
		6	**0**

Let us present a short example. When the charging cost is the lowest – (answer 1), the travel time is lowest (answer 1) and time windows are the narrowest (answer 1), then the cumulative number of demands (wrapping cube X1 - Y1 - Z1) contains all trip demands. This solution can be explained as follows: If our system provides transportation service at the lowest cost with a travel time equal to that of taxis and a time window between 0 and 5 minutes, then every trip request that has at least one parameter more "relaxed" (i.e. worse for the customer) than the above parameters, that trip can also be satisfied. The wrapping cube (X1 - Y1 - Z2) contains 954 demands,158 less trip demands than the previous cube. Notice that sub-cube (X1 - Y1 - Z1) contains 158 trip demands. On the other hand, wrapping cube (X5 - Y6 - Z6) contains no trip demands at all. This is quite obvious, as no user would want to get that worst service quality at the highest price.

Computational Results

Using the above statistical analysis, we can design our experiments (see Table 2). Before we proceed it is necessary to define the real operational cost per km for a DRT transportation system. To the knowledge of the authors, there is no such study that calculates costs like that in a generally applicable way. This seems quite logical, taking into consideration the variance of the different factors that affect cost. Due to the lack of such information, we used as operational cost the operational cost of taxis, since it is the most expensive (due to the nature of the offered transportation service). We asked the taxi union of Philippi to provide us their best estimate of cost-per-km, based to their long experience in the field. This cost has been estimated by taxi drivers as:

0.34 euro for distances above 300 km / day

0.4 euro for distances between 200-300 km / day

0.45 euro for distances below 200 km / day

Our next step was to define the desired service quality parameter values. By data analysis we found that the most desired service quality parameter values were:

Waiting Time Window: 6-10 minutes

Max Ride Time in comparison to SP Travel Time: 1.5

The experiment's design also included the vehicle capacity definition. For our experiments we chose three different potential vehicle capacities:

8-seat Vehicles

12-seat Vehicles

16-seat Vehicles

The Table 6-2 presents the convergence algorithm's behavior during various tests. While we consider most table headings easily comprehensible, special notes are given for:

- Column 13, which is the converge algorithm correction value, concerning the new charging cost per KM produced by the type ***ChDU= TSC_{ijk}. / TPD_{ijk}***

- Column 14, which is the converge algorithm next value (according to 3-d model) which is the first lower (or higher) value than the new 13th column

Table 6-2: Experiment Set Results (Vehicle capacity 8, 12, 16 seats)

#of Exper-iment (vehicle Capacity)	Number of accumulated (projected) trip de-mands	harging Cost in Euros per KM	eal Vehicle Cost Per KM in euros	Number of Vehicles	Total Vehicle Distance in KMs	Total Passengers Distance in KMs	Total vehicles Cost (Euros)	Total Charging Cost (Euros)	Profit (Euros)	Profit(Loss) percentage to real vehicle cost	Times the Max trip Ride Time(Actual Value)	Converge Algorithm Correction Value	Converge Algorithm Next Value	Average Number of passengers per KM	Critical Point for “break even”
1(8-seat) vehicle	773	0.17	0.34	13	4997	8539	1498.98	1451.63	-47.35	-3.1%	1.3	0.19896709	0.255	2.008	No
	525	0.255	0.34	9	3407	5933	1158.38	1512.92	354.54	30.61%	1.38	0.19524355	0.17	2.18	Yes
	185	0.34	0.4	4	1171	1738	468.40	590.92	122.52	26.16%	1.4	0.26950518	never reach that point	1.83	No
	38	0.5325	0.45	2	369	671	166.05	357.31	191.26	115.18%	1.16	0.24746647	0.17	1.81	No
		0.71													
2(12-seat) vehicle	773	0.17	0.34	10	4477	8539	1522.18	1451.63	-70.55	-4.63%	1.41	0.17826209	0.255	2.42	No
	525	0.255	0.34	9	3281	5933	1115.54	1512.92	397.38	35.62%	1.39	0.18802292	0.17	2.25	Yes
	185	0.34	0.4	4	1172	1738	468.80	590.92	122.12	26.05%	1.4	0.26973533	never reach that point	1.83	No
	38	0.5325	0.45	2	369	671	166.05	357.31	191.26	115.18%	1.16	0.24746647	0.17	1.82	No
		0.71													
3(16-seat vehicle)	773	0.17	0.34	10	4403	8539	1497.02	1451.63	-45.39	-3.03%	1.41	0.17531561	0.255	2.45	No
	525	0.255	0.34	9	3276	5933	1113.84	1512.92	399.08	35.83%	1.39	0.18773639	0.17	2.26	Yes
	185	0.34	0.4	4	1172	1738	468.80	590.92	122.12	26.05%	1.4	0.26973533	never reach that point	1.83	No
	38	0.5325	0.45	2	369	671	166.05	357.31	191.26	115.18%	1.16	0.24746647	0.17	1.82	No
		0.71													

From table 6-2 we can derive that when the charging cost is the minimum (17cents), the system is always unprofitable, while it is profitable for all other charging price levels and reaches high profits for the highest charging price (53cents). We found that the highest real profit is found where the charging cost is 25.5 euro cents per km. From table 6-2 we can conclude that increasing the vehicle capacity increases profits and puts a burden on the average duration of a single trip. However this increase in duration is insignificant compared to the profit increase. The columns 13 and 14 also show us that regardless of the starting point of the algorithm, it always converges at a single point where the charging cost is 25.5 euro cents. In any case, the system's charging policy remains a strategic decision, depending on whether the operator's goal is high profitability percentages or high profitability in absolute figures.

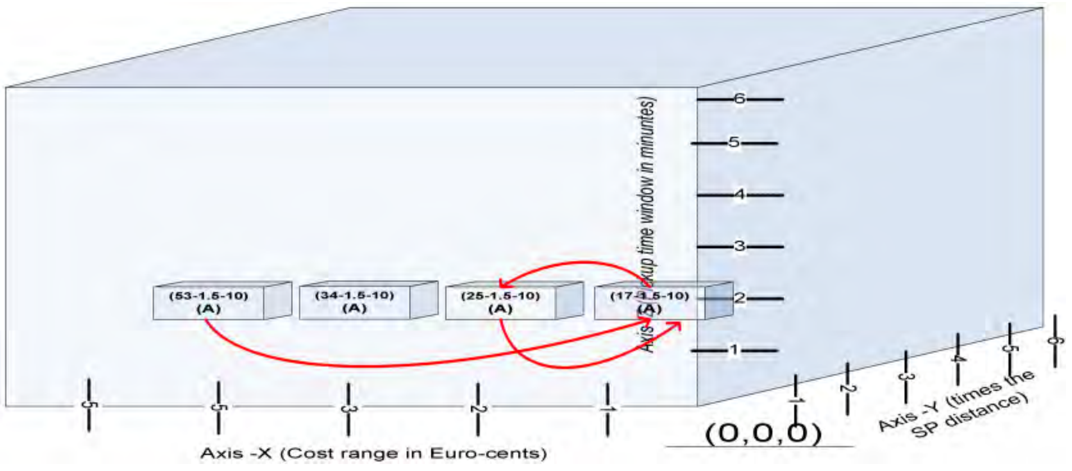

(53-1.5-10) (A)
(34-1.5-10) (A)
(25-1.5-10) (A)
(17-1.5-10) (A)
Axis (pickup time window in minuntes)
Axis -X (Cost range in Euro-cents)
Axis -Y (times the SP distance)
(0,0,0)

Figure 6-2.An example of convergence process sequence

Figure 6-2 presents the convergence sequence for the case of 8-seat vehicles. We start from the initial point (X1 - Y2 - Z2)= (17-1.5-10) where the system operates at a loss. The converge algorithm correction value for the new charging cost-per-km produced by ***ChDU= TSC_{ijk}./ TPD_{ijk}*** is 0.19896709 euro-per-km. The first bigger value – according to the projection model - is 0.255 euro-per-km. We move to the next point, which is (X2 - Y2 - Z2)=(25.5-1.5-10). For that point we run the Dial-a-Ride algorithms again and the system operates at a profit. The converge algorithm correction value for the new charging cost per km produced by ***ChDU= TSC_{ijk}./ TPD_{ijk}*** is now 0.19524355 euro-per-km. The first lower value – according to the projection model - is 0.17 euro-per-km. The system circles again to the same values. At that point the convergence process ends. The first point that produces profit is (X2 - Y2 - Z2). If we start the converge process from the last point (X4 - Y6 - Z6)=(53-1.5-10), it finally leads the system to the point (X2 - Y2 - Z2).

Chapter 7 : Conclusions and Future Work

7.1 Conclusions

This work contributed to the Dial-a-Ride Problem by offering six algorithms (offline and online) and in addition , an interesting contribution regards the profitability of a proposed but not yet existing DRT system, presented in chapter 6. For these algorithms the complexity was studied offering useful information regarding the execution time. For the VLSN algorithm the size of the neighborhood was identified in comparison with the total solution space. For the OR-DARP algorithm we also proposed a linear approach to describe the relationship between full static solutions and full online solution.

For the static Dial-a-Ride problem we developed:

1. The simple insertion InsertionH implementation. This implementation is not new and innovative since it has been introduced years ago. The main need for the implementation of this algorithmic was the use of the initial solutions it produced for other optimization algorithms. The computational effort has been calculated in relation to the number of possible searches in order to assign trip demands. This number has been estimated as $1+\frac{1}{6}n(17+15n+4n^2)$, where *n* is the number of trip demands.

   DP Exact implementation. The proposed algorithm has been compared with the Mixed Integer Linear Program (MILP) implementation of the DARP problem. The mathematical formulation presented in Chapter 3 and has been used to implement the MILP in AMPL. Both implementations DP and MILP gave always the same results concerning objective function and trips pickup and delivery sequence. Although execution times clearly indicates that the DP implementation is faster than MILP implementation and this is the main reason that makes this implementation the best candidate for use as sub-problems. The total computational effort is estimated by formula $\sum_{L=1}^{v}\sum_{d=1}^{vd}\frac{vd!}{(vd-d)!d!}(d!(d!+\sum_{x=1}^{d-1}\sum_{y=1}^{d-x}\sum_{z=1}^{d-1}(d-z)!))$ that clearly shows the NP-Nature of this DARP problem.

2. A “hybrid” implementation called VLSN that uses a heuristic part and an exact part in order to search a much broader area concerning the solution space. For the purpose of evaluating the VLSN algorithm, table 4-3 presents a comparative evaluation between the VLSN algorithm and the pure exact algorithm. Although there is no methodology proposed up to now to estimate the closeness of any solution produced by a heuristic to optimal solution, table 4-3 provides computational results that show the solution produced by that algorithm and the full solution provided by the exact algorithm. Problem size is of course limited to small instances because of the limited ability of the exact algorithm to solve large problem sizes. However, it is worthwhile to present those results. We see that the algorithm provides solutions close enough to the optimal solution. The ratio of the space size investigated by VLSN algorithm in comparison with the total solution space size has been estimated as

$$\frac{\frac{v!}{(v-2)!2!}2^{vd}-2\frac{v!}{(v-2)!2!}+v}{v^{vd}}.$$

3. The RegretH heuristic algorithm that uses regret technique as an optimization engine. Tables 4-6 to 4-10 present an extensive testing of the Regret heuristic on real and artificial data. The main conclusion of these tables is that the proposed Regret Heuristic improves at much better ratios the solutions for the real data. The execution times are comparable to the execution times of the simple insertion algorithm and that makes the regret algorithm attractive for large scale problem instances.

For the online Dial-a-Ride problem we developed:

1. The online Regret algorithm ("OR-DARP"), that consists of two sub-algorithms. The first sub-algorithm inserts the trip demand immediately. Time responses for this trip demand cannot exceed a specific time threshold. The second sub-algorithm focuses on optimization. We use regret techniques for optimization. The most important feature of the online regret algorithm is that optimization works continuously. There is no need for time horizons. The algorithm takes advantage of the small time periods that the system is in idle state and optimizes the solution. Table 7-1 indicates that the maximum response time for the insertion of a new trip request is below 15 seconds.

Table 7-1: Max Response Time and Max Optimization for the Online Regret Algorithm

#of exper	*Trip Requests Assigned*	*Max Ins Time*	*Max Reg Time*
1	94	0.410	3.330
2	226	2.57	39.24
3	414	2.82	196.83
4	622	3.51	343.3
5	735	7.160	790.9
6	727	11.41	909.
7	873	9.2	768.0
8	890	12.37	709.7
9	1012	14.1	1207

Another important result presented in table 5-6, is the existence of a critical limit where the regret optimization affects the solution quality negatively in terms of profitability. After that point the usage of the regret optimizations operates in such a way that the profits we lose are more than the cost reduction.

Table 7-2: Online Regret without Optimization Vs Online Regret with Optimization

#of exper	*Trip Demands serviced by online regret algorithm without optimization*	*Trip Demands serviced by online regret algorithm with optimization*	*Cost Diff%*	*Profit Diff %*
1	94	94	-1.14%	3.20%
2	226	226	-1.87%	3.63%
3	415	414	-3.59%	3.42%
4	624	622	-2.24%	1.34%
5	803	735	-9.92%	-7.13%
6	949	737	-16.02%	-29.26%
7	1151	873	-16.74%	-29.60%
8	1370	890	-26.11%	-40.99%
9	1619	1012	-31.44%	-40.65%

The experiments, based on real data, indicate the critical number of trip demands for which the system starts to produce more losses (because of the denied trip demands) than profits because of the optimization. This result could be valuable for the setup of an online DRT system.

2. The Online Regret algorithm with probabilities (OP-DARP) was developed as an extension of the OR-DARP algorithm in order to anticipate the knowledge we have, for the selection of the trip assignments in a more intelligent way. The critical factor of this algorithm is the integration of the historical data. The OP-DARP algorithm utilizes real historical trip data and gives more optimized solutions than the OR-DARP algorithm, in the majority of cases we have tested. For the handling of probabilistic demands the OP-DARP algorithm requires large amounts of processor power. If our processing system is powerful enough then the usage of OP-DARP is a necessity in order to optimize it as much as possible.

For the Profitability Study of a proposed but not existing DRT system:

A new methodology has been presented regarding the way we define profitability of a proposed but not yet deployed system. This methodology utilizes data and conclusions drawn from user surveys and their subsequent analysis to find a critical point where the system becomes profitable. The convergence algorithm starts from an initial parameter set, drawn by the survey analysis, and works towards a final set where the system is profitable. This methodology can also be implemented for real time DRT systems, given the appropriate historical data concerning users' behavior.

7.2 Future Work

Fixed conventional bus services are more appropriate in areas and time periods with high demand densities, which can sustain high network densities and service requests, while flexible route DRT services are suitable for suburban areas or time periods with low demand densities. In this dissertation we have developed four static DARP algorithms, two online DARP algorithms, and one methodology with a convergence algorithm.

This research can be extended to investigate the performance of other heuristics or even more exact approaches concerning the static DARP to the online algorithms. Issues like:

1. The queuing time. Response time is a critical factor for the Online DRT systems. Tight response times make the solution worse. On the other hand long response times may affect heavily the customer's satisfaction when they call in a trip demand. The use of internet adds one more factor concerning the response time.

2. The optimization procedures. The usage of optimization procedures in online algorithms could be an interesting field of study. The way we embed optimization into online algorithms, the execution times for one step optimization, the usability of op-

timization in comparison with rejected (if any) of requested trips are some interesting issues concerning the optimization procedures

3. The forecast of the incoming demands. The utilization of historical data regarding users' mobility habits and the duration of the forecasting period is an issue concerning the online Dial-a-Ride algorithms. Many parameters concerning forecasting (time horizons, demand occurrence probabilities, evaluation of probabilistic demands concerning this “importance”) can be a possible research subject.

4. The computing an optimal waiting strategy

References

[1] G. Andreatta and G. Romanin-Jacur. The ow management problem. Transportation Science, 21(4):249-253, 1987.

[2] J. Aronson. A survey of dynamic network ows. In P. Hammer, editor, Annals of Operations Research, pages 1{66. J.C. Baltzer AG, 1989.

[3] J. Aronson and B. Chen. A forward network simplex algorithm for solving multiperiod network flow problems. Naval Research Logistics Quarterly, 33(3):445-467,1986.

[4] M. Ball and A. Roberts. A graph partitioning approach to airline crew scheduling. Trans. Sci, 19:107-126, 1985.

[5] J. C. Bean and R. L. Smith. Optimal capacity expansion over an infinite horizon. Management Science, 31(12):1523-1532, 1985.

[6] M. Bellmore, W. Ecklof, and G. Nemhauser. A decomposable transshipment algorithm for a multiperiod transportation problem. Naval Research Logistics Quarterly, 16:517-524, 1969.

[7] O. Berman. Dynamic repositioning of indistinguishable service units on transportation networks. Transportation Science, 15(2):115-136, 1981.

[8] O. Berman and B. LeBlanc. Location - relocation of mobile facilities on a stochastic network. Transportation Science, 18(4):315-330, 1984.

[9] O. Berman and D. Simchi-Levi. Minisum location of a traveling salesman. Networks, 16:329-354, 1986.

[10] O. Berman and D. Simchi-Levi. Finding the optimal a priori tour and location of a traveling salesman with non-homogeneous customers. Transportation Science, 22:148-154, 1988.

[11] O. Berman and D. Simchi-Levi. A heuristic algorithm for the traveling salesman location problem on networks. Operations Research, 36:478-484, 1988.

[12] D. Bertsimas. Probabilistic combinatorial optimization problems. Technical Report 194, Operations Research Center, MIT, 1988.

[13] D. Bertsimas. A vehicle routing problem with stochastic demand. Operations Research,, 1993.

[14] D. Bertsimas and G. van Ryzin. Stochastic and dynamic vehicle routing problem in the Euclidean plane with multiple capacitated vehicles. 1991.

[15] M. Bielli, G. Calicchio, B. Micoletti, and S. Ricciardelli. The air traffic flow control problem as an application of network theory. Computers and Operations Research, 9(4):265-278, 1982.
[16] J. Blum. Approximation methods which converge with probability one. Ann. Math. Stat., 25:382-386, 1954.
[17] E. Bowman. Production scheduling by the transportation method of linear programming. Operations Research, 4:100-103, 1956.
[18] G. Brown, G. Graves, and D. Ronen. Scheduling ocean transportation of crude oil. Mgmt. Sci., 33:335-346, 1987.
[19] R. Burness and J. White. The traveling salesman location problem. Transportation Science, 10:348-360, 1976.
[20] R. Busacker and P. Gowen. A procedure for determining a family of minimal cost network flow patterns. Operations Research Office, Tech. Paper 15, 1961.
[21] K. Chih. A real time dynamic optimal freight car management simulation model of the multiple railroad, multicommodity, temporal spatial network flow problem. Ph.d. dissertation, Department of Civil Engineering and Operations Research, Princeton University, 1986.
[22] K. Chih, M. Hornung, M. Rothenberg, and A. Kornhauser. Implementation of a real time locomotive distribution system. In T. Murthy, R. Rivier, G. List, and J. Mikolaj, editors, Computer Applications in Railway Planning and Management, pages 39-50. Computational Mechanics Puublications, 1990.
[23] N. Christofides and P. Brooker. Optimal expansion of an existing network. Mathematical Programming, 6:197-211, 1967.
[24] K. Cooke and E. Halsey. The shortest route through a network with time- dependent internodal transit times. J. Math. Anal. and Appl., 14:493-498, 1966.
[25] T. Crainic, J. Ferland, and J.-M. Rousseau. A tactical planning model for rail freight transportation. Transportation Science, 18:165-184, 1984.
[26] T. Crainic, M. Gendreau, and P. Dejax. Dynamic stochastic models for the allocation of empty containers. Operations Research, 41:102-126, 1993.170
[27] T. Crainic and J. Roy. Or tools to the tactical planning of freight transportation.European Journal of Operations Research, 1987.
[28] G. Dantzig and A. Ferguson. The allocation of aircrafts to routes: An example of linear programming under uncertain demand. Management Science, 3:45-73,1956.
[29] G. Dantzig and D. Fulkerson. Minimizing the number of tankers to meet a fixed schedule. Naval Research Logistics Quarterly, 1:217-222, 1954.

[30] P. Doulliez and M. R. Rao. Optimal network capacity planning: A shortest path scheme. Operations Research, 23:811-818, 1975.

[31] S. Dreyfus. An appraisal of some shortest-path algorithms. Operations Research, 17:395-412, 1969.

[32] D. Erlenkotter. Preinvestment planning for capacity expansion: A multi-location dynamic model. Ph.d. dissertation, Graduate School of Business, Stanford University, Stanford, CA, 1969.

[33] D. Erlenkotter. Capacity planning for large multi-location systems: Approximate and in-complete dynamic programming approaches. Management Science, 22:274-285, 1975.

[34] D. Erlenkotter. A comparative study of approaches to dynamic location problems. European J. Oper. Res., 6:133-143, 1981.

[35] Y. Ermoliev, T. Krivets, and V. Petukhov. Planning of shipping empty seaborne containers. Cybernetics, 12:664, 1976.

[36] J. Farvolden and W. Powell. Subgradient optimization for the service network design problem. Transportation Science, 28:256-272, 1994.

[37] G. Feeney. Controlling the distribution of empty cars. In Proc. 10th National Meeting, Operations Rese arch Society of America, 1957.

[38] M. L. Fisher and M. B. Rosenwein. An interactive optimization system for bulk cargo ship scheduling. Naval Research Logistics, 36:27-42, 1989.

[39] H. Florez. Empty-container repositioning and leasing: An optimization model. Ph.D. dissertation, Polytechnic Institute of New York, 1986.

[40] C. Fong. The multi-region dynamic capacity expansion problem: Exact and heuristic approaches. Ph.d. dissertation, Graduate School of Management, University of Rochester, 1974.

[41] C. Fong and V. Srinivasan. The multiregion dynamic capacity expansion problem . part ii. Operations Research, 29:800-816, 1981.

[42] C. Fong and V. Srinivasan. The multiregion dynamic capacity expansion problem part i. Operations Research, 29:787-799, 1981.

[43] C. Fong and V. Srinivasan. The multiregion dynamic capacity expansion problem: An improved heuristic. Management Science, 32(9):1140-1152, 1986

[44] C. Fong and V. Srinivisan. Multiperiod capacity expansion and shipment planning with linear costs. Naval Research Logistics Quarterly, 23(1):255-260, 1964.

[118-] L. Ford and D. Fulkerson. A suggested computation for maximal multicommodity network flows. Management Science, 4(5):97-101, 1958.

[45] K. Fox, B. Gavish, and S. Graves. An n-constraint formulation of the (time dependent) traveling salesman problem. Operations Research, 28:1018-1021, 1980.

[46] L. Frantzeskakis and W. Powell. A successive linear approximation procedure for stochastic dynamic vehicle allocation problems. Transportation Science, 24(1):40-57, 1990.

[47] T. Friesz, D. Bernstein, T. Smith, R. Tobin, and B. Wie. A variational inequality formulation of the dynmamic network user equilibrium problem. Operations Research,, 1992.

[48] T. Friesz, J. Luque, R. Tobin, and B. Wie. Dynamic network traffic assignment considered as a continuous time optimal control problem. To appear in Operations Research, 1989.

[49] T. Glickman and H. Sherali. Large-scale network distribution of pooled empty freight cars over time, with limited substitution and equitable benefits. Trans.Res., 19:85-94, 1985.

[50] F. Glover, G. Jones, D. Karney, D. Klingman, and J. Mote. An integrated production, distribution and inventory planning system. Interfaces, 9(5):72-86, 1979.

[51] S. Gorenstein, S. Poley, and W. White. On the scheduling of the railroad freight operations. technical report 320-2999, ibm philadelphia scientific center, IBM,1971.

[52] H. H. The distribution of empty wagons by means of computer: An analytical model for the swiss federal railways (ssb). Rail International, 4(1):1005-1010,1973.

[53] H. H. Computer controlled empty wagon distribution on the ssb. Rail International, 8(1):25-32, 1977.

[54] A. Haghani. Formulation and solution of a combined train routing and makeup, and empty car distribution model. Transportation Research, 23B(6):433{452, 1989.

[55] M. Haimovich, A. R. Kan, and L. Stougie. Analysis of heuristics for vehicle routing problems. In B. Golden and A. Assad, editors, Vehicle Routing: Methods and Studies. North Holland, Amsterdam, 1988.

[56] P. Hammer. Time-minimizing transportation problems. Naval Research Logistics Quarterly, 16:345-357, 1969.

[57] P. Hammer. Communication on `the bottleneck problem' and 'some remarks on the time transportation problem'. Naval Research Logistics Quarterly, 18:487-490,1971.

[58] J. L. Higle, J. Bean, and R. L. Smith. Capacity expansion under stochastic demands. Tech. report 84-28, Dept. of Systems and Industrial Engineering, U. of Arizona, 1984.

[59] P. Jaillet. Stochastic routing problems. In G. Andreatta, F. Mason, and P. Sera_ni, editors, Stochastics in Combinatorial Optimization. World Scienti_c, Singapore, 1987.

[60] P. Jaillet. Probabilistic routing problems in the plane. In H. Bradley, editor Operational Research 90. Pergamon Press, London, 1991.

[62] P. Jaillet and A. Odoni. Probabilistic vehicle routing problems. In B. Golden and A. Assad, editors, Vehicle Routing: Methods and Studies. North Holland, Amsterdam, 1988.
[63] W. Jordan and M. Turnquist. A stochastic dynamic network model for railroad car distribution. Transportation Science, 17:123-145, 1983.
[64] U. Karmarkar. Convex/stochastic programming and multilocation inventory problems. Naval Reseach Logistics Quarterly, 26(1):1-19, March 1979.
[65] U. Karmarkar. The multiperiod, multilocation inventory problem. Operations Research, 29(2):215-228, March-April 1981.
[66] U. Karmarkar. Policy structure in multistage production/inventory problems :An application of convex analysis. TIMS Studies in the Management Sciences,16:331-352, 1981.
[67] U. Karmarkar. The multilocation, multiperiod inventory problem: Bounds and approximations. Management Science, 33(1):86-94, January 1987.
[68] D. Kaufman and R. L. Smith. Fastest paths in time-dependent networks for intelligent vehicle highway systems application. IVHS Journal, 1:1-11, 1993.
[69] D. Klingman and J. Mote. A multi-period production, distribution and inventory planning model. Advances in Management Studies, 1(2):56-76, 1982.
[70] D. Kraay, P. Harker, and B. Chen. Optimal pacing of trains in freight railroads: Model formulation and solution. Operations Research, 39(1):82-99, 1991.
[71] G. Laporte and P. J. Dejax. Dynamic location-routeing problems. J. Opl.Res. Soc, 40(5):471-482, 1989.
[72] C. Leddon and E.Wrathall. Scheduling empty freight car eets on the louisville and nashville railroad. In Second International Symposium on the Use of Cybernetics on the Railways, October, pages 1-6, Montreal, Canada, 1967.
[73] C. Malandrak and R. B. Dial. A dynamic programming heuristic algorithm for the time dependent traveling salesman problem. Unpublished technical report, Roadnet Technologies, Inc., 1992.
[74] C. Malandraki. Time dependent vehicle routing problems: Formulations, solution algorithms and computational experiments. Ph.d. dissertation, Department of Civil Engineering, Northwestern University, 1989.
[75] V. Mendiratta. A dynamic optimization model of the empty car distribution process. Ph.D. Dissertation, Department of Civil Engineering, Northwestern University, 1981.
[76] V. Mendiratta and M. Turnquist. A model for the management of empty freight cars. Trans. Res. Rec., 838:50-55, 1982.

[77] M. Minoux. Network synthesis and dynamic network optimization. Annals of Discrete Mathematics, 31:283-324, 1987.

[78] P. Mirchandani and R. Francis, editors. Discrete Location Theory. John Wiley and Sons, New York, 1990.

[79] S. Misra. Linear programming of empty wagon disposition. Rail International, 3:151-158, 1972.

[80] J. Mulvey and S. Zenios. Real-time operational planning for the u.s. air traffic system. Applied Numerical Mathematics, 3:427-441, 1987.

[81] A. Orda and R. Rom. Shortest-path and minimum-delay algorithms in networks with time-dependent edge-length. Journal of the Association for Computing Machinery, 37:607-625, 1990.

[83] A. Perakis and N. Papadakis. Minimal time vessel routing in a time-dependent environment. Transportation Science, 23(4):266-276, 1989.

[84] J. Picard and M. Queyranne. The time dependent travelling salesman problem and its application to the tardiness problem in one-machine scheduling. Operations Research, 26(2):86-110, 1978.

[85] W. Powell. A stochastic model of the dynamic vehicle allocation problem. Transportation Science, 20:117-129, 1986.

[86] W. Powell. An operational planning model for the dynamic vehicle allocation problem with uncertain demands. Transportation Research, 21B:217-232, 1987.

[87] W. Powell. A comparative review of alternative algorithms for the dynamic vehicle allocation problem. Vehicle Routing: Methods and Studies, pages 249-292, 1988.

[88] W. Powell and R. Cheung. A network recourse decomposition method for dynamic networks with random arc capacities. Networks, 24:369-384, 1994.

[89] W. Powell and Y. She_. Design and implementation of an interactive optimization system for network design in the motor carrier industry. Operations Research, 37(1):12-29, 1989.

[90] W. Powell, Y. She_, K. Nickerson, K. Butterbaugh, and S. Atherton. Maximizing pro_ts for north american van lines' truckload division: A new framework for pricing and operations. Interfaces, 18:21-41, 1988.

[91] W. Powell, Y. She_, and S. Thiriez. The dynamic vehicle allocation problem with uncertain demands. In Ninth International Symposium on Transportation and Traffic Theory, pages 357-374, 1984.

[92] H. Psaraftis. Dynamic vehicle routing problems. In B. Golden and A. Assad, editors, Vehicle Routing: Methods and Studies, pages 223-248, Amsterdam, 1988. North Holland.
[93] H. Psaraftis, J. Orlin, D. Bienstock, and P. Thompson. Analysis and solution algorithms of sealift routing and scheduling problems. Sloan working paper 1700 85, Sloan School of Management, MIT, 1985.
[94] B. Ran, D. Boyce, and L. LeBlanc. Toward a new class of instantaneous dynamic user-optimal traffic assignment models. Operations Research, 1992.
[95] L. Ratcli_e, B. Vinod, and F. Sparrow. Optimal prepositioning of empty freight cars. Simulation, pages 269-275, 1984.
[96] U. K. S. Agnihothri and P. Kubat. Stochastic allocation rules. Operations Research, 30(3):545-555, May-June 1982.
[97] Y. Shan. A dynamic multicommodity network flow model for real-time optimal rail freight car management. Ph.d. dissertation, Princeton University, 1985.
[98] J. Shapiro. Mathematical programming models and methods for production planning and scheduling. In A. R. K. S.C. Graves and P. Zipkin, editors, Logistics of Production and Inventory, pages 371-444, 1993.
[99] S. Smith and Y. She_. Railroad car distribution: A fast network model with multiple car types and delays. Cts working paper, M.I.T., 1992.
[100] W. Szwarc. The time transportation problem. Zastowania Matematyki, Warsaw, 8:231-242, 1966.
[101] W. Szwarc. The dynamic transportation problem. Naval Research Logistics Quarterly, 13:335-345, 1971.
[102] C. Tapiero. Transportation over time: A system theory approach. In Proceedings {XX International TIMS Meeting, Tel-Aviv, Israel, Vol 1, pages 239-243,Jerusalem, 1975.
[103] C. Tapiero and M. Soliman. Multicommodities transportation schedules over time. Networks, 2:311-327, 1972.
[104] M. Turnquist. Mov-em: A network optimization model for empty freight car distribution. School of Civil and Environmental Engineering, Cornell University,1986.
[105] M. A. Turnquist and B. P. Markowicz. An interactive microcomputer-based planning model for railroad car distribution. For the cors/orsa/tims national meeting,Cornell University, 1989.
[106] T. Van Roy and D. Erlenkotter. A dual-based procedure for dynamic-facility location. Working paper 80-31, International Institute for Applied Systems Analysis,Laxenburg, Austria, 1980.

[107] W. White. Dynamic transshipment networks: An algorithm and its application to the distribution of empty containers. Networks, 2(3):211-236, 1972.
[108] W. White and A. Bomberault. A network algorithm for empty freight car allocation. IBM Systems Journal, 8(2):147-171, 1969.
[109] N. Zadeh. On building minimum cost communication networks over time. Networks, 4:19-34, 1974.
[110] B. Fleischmann, S. Gnutzmann, Elke Sandvoss Dynamic Vehicle Routing on Online Traffic Information. Tans Science Vol 38, No. 4, P 420-433, 2004.
[111] H. Psaraftis, Dynamic vehicle routing: Status and prospects. Anals of Oper Research 61(1995)143-164.
[112] B. Bagchi and B. Nag. Dynamic vehicle scheduling: An expert systems approach. International Journal of Physical Distribution and Logistics Management, 21(2):10-18, 1991.
[113] A. Odoni. The flow management problem in air traffic control. In L. B. A.R. Odoni and G. Szego, editors, Flow Control of Congested Networks, pages 269-288, New York, 1986. Springer-Verlag.
[114] Psaraftis. An Exact algorithm for the single Vehicle many to many Dial –a – Ride Problem with time With Time Windows. Transportation Science Vol 13. No 3, August 1983.
[115] Sexton, Bodin. Optimizing Single Vehicle many to many operations with Desired Delivery Times I Scheduling. Transportation Science Vol 19. No 4, November 1985.
Optimizing Single Vehicle many to many operations with Desired Delivery Times
II Routing. Transportation Science Vol 19. No 4, November 1985.
[116] A heuristic Algorithm for The multi-vehicle advance Request Dial – a – Ride Problem with time windows. Transportation Res. –B Vol 20B No 3, pp 243-257 , 1986
[117]Fisher,MarshallL.,Jaikumar,Ramchandran. Generalized assignment heuristic for vehicle routing. 1981) *Networks*, 11 (2), Pages 109-124.
[118] A heuristic algorithm for a dial – a – ride problem with time windows, multiple capacities, and multiple objectives. Annals of Operations Research 60 pp 193-195, 1995
[119] Yvan Dumas, Jacques Desrosiers and Francois Sumis. The pickup and delivery problem with time windows. Eropean Journal of Operational Research 53 pp 7-22, 1991
[120] Paolo Toth and Daniel Vigo. Heuristic Algorithms for the handicapped Persons Transportation Problem. Transportation Science Vol 31, No 1, February 1997.
[121] Liping Fu. Scheduling Dial – a – Ride paratransit under time varying, stochastic congestion. Transportation Research Part B 36, pp 485-506, 2002.

[122] Cordeau, J.-F., & Laporte, G. The dial-a-ride problem (DARP): variants, modeling issues and algorithms. 4OR: A Quarterly Journal of Operations Research, 1, 89–101, 2003b.

[123] Andrea Attanasio a, Jean-Francois Cordeau b,Gianpaolo Ghiani c,*, Gilbert Laporte b. Parallel Tabu search heuristics for the dynamic multi-vehicle dial-a-ride problem. Parallel Computing 30 , 377–387, 2004.

[124] Jacques Desrosiers, Yvan Dumas, Francois Soumis. A dynamic programming solution of the large –scale single vehicle Dial-a-Ride problem with time windows. American Journal of mathematical and management sciences, Vol 6 pp 301–325, 1986.

[125] Kozanidis G, Ziliaskopoulos A. Dynamic Programming Strategies for the Dial-a-Ride Problem with Time Window Constraints, Transportation Research Board 85th Annual Meeting ,2006.

[126] Wilson, N. H. M. and Weissberg, H. Advanced Dial-a-Ride Algorithms Research, 1976.

[127]Wilson, N. H. M. and Colvin, N. H. Computer Control of the Rochester Dial-a-Ride System. Technical Report R77-31, Department of Civil Engineering,Massachusetts Institute of Technology, Cambridge, MA, 1977.

[128] Cordeau, J.-F., & Laporte, G.The dial-a-ride problem: models and algorithms Ann Oper Res ,153: 29–46, 2007.

[129] Psaraftis, H. N. A dynamic programming approach to the single-vehicle, many-to-many immediate request dial-a-ride problem. Transportation Science, 14, 130–154, 1980.

[130] Cordeau, J.-F. A Branch-and-cut algorithm for the dial-a-ride problem. Operations Research, 54,573–586, 2006.

[131] Ropke, S., Cordeau, J.-F., & Laporte, G. Models and branch-and-cut algorithms for pickup and delivery problems with time windows. Networks, 49, 258–272, 2007.

[132] Jaw, J., Odoni, A. R., Psaraftis, H. N., &Wilson, N. H. M. A heuristic algorithm for the multi-vehicle advance-request dial-a-ride problem with time windows. Transportation Research B, 20, 243–257, 1986.

[133]Ioachim, I., Desrosiers, J., Dumas, Y., & Solomon, M. M. A request clustering algorithm for door-todoor handicapped transportation. Transportation Science, 29, 63–78, 1995.

[134]Cordeau, J.-F., & Laporte, G. A tabu search heuristic for the static multi-vehicle dial-a-ride problem.Transportation Research B, 37, 579–594, 2003.

[135]Diana, M., & Dessouky, M. M. A new regret insertion heuristic for solving large-scale dial-a-ride problems with time windows. Transportation Research Part B, 38, 539–557, 2004.

[136]Teodorovic, D., & Radivojevic, G. A fuzzy logic approach to dynamic dial-a-ride problem. Fuzzy Sets and Systems, 116, 23–33, 2000.

[137]Colorni, A., & Righini, G. Modeling and optimizing dynamic dial-a-ride problems. International Transactions in Operational Research, 8, 155–166, 2001.

[138]Mitrovic-Minic, S., Krishnamurti, R. and Laporte, G. Double-horizon based heuristics for the dynamic pickup and delivery problem with time windows.Transportation Research, 38B, 669-685, 2004.

[139]Coslovich, L., Pesenti, R., & Ukovich, W. A two-phase insertion technique of unexpected customers for a dynamic dial-a-ride problem. European Journal of Operational Research, 175, 1605–1615, 2006.

[140]Xiang, Z., Chu, C., & Chen, H. The study of a dynamic dial-a-ride problem under time-dependent and stochastic environments. European Journal of Operational Research, 185(2),pp. 534-531,2008

[141] Feuerstein,E Stougie,L On-line single-server dial-a-ride problems Theoretical Computer Science 268, 91–105, 2001.

[142] Fink,I. Krumke,O. Westphal,S New lower bounds for online k-server routing problems. Information Processing Letters (ARTICLE IN PRESS), 2009

[143] Wilson, N.H.M., Sussman, J.M., Higonnet, B.T., Goodman, L.A.“Simulation of a computer-aided routing system” (CARS). Highway Research Record, 318, pp. 66–76, 1970

[144] Haghani, A., Banihashemi, M.“Heuristic approaches for solving large-scale bus transit vehicle scheduling problem with route timeconstraints. Transportation Research A , 36A, pp. 309–333, 2002.

[145] Shinoda K., Noda I., Ohta M., Kumada Y., Nakashima, H. “Is dial-a-ride bus reasonable in large scale towns? Evaluation of usability of dial-a-ride systems by simulation”. Lecture Notes in Computer Science, 3012, pp. 105–119, 2004.

[146] Quadrifoglio L., Dessouky, M. “Insertion heuristic for scheduling Mobility Allowance Shuttle Transit (MAST) services”. Journal of Scheduling, 10, pp. 25–40, 2007.

[147] Quadrifoglio, L., Dessouky, M.”Insertion heuristic for scheduling Mobility Allowance Shuttle Transit (MAST) services: sensitivity to service area. In: Computer-Aided Systems in Public Transport, Springer Series, Lecture Notes in Economics and Mathematical Systems 600, 2007.

[148] Luca Quadrifoglio, Maged M. Dessouky , Fernando Ordonez,”A simulation study of demand responsive transit system design”,2008, Transportation Research Part A 42 , pp. 718–737,2008.

[149] Ravindra K. Ahujaa; , Ozlem Ergunb, James B. Orlinc,Abraham P. Punnend “A survey of very large-scale neighborhood search techniques” Discrete Applied Mathematics 123, 75 – 102, 2002.

[150] Gendreau, M., F. Guertin, J.-Y. Potvin, and R. S´eguin. “Neighborhood Search Heuristics for a Dynamic Vehicle Dispatching Problem with Pick-ups and Deliveries.” Technical Report CRT-98-10,.Centre de recherche sur les transports, Universit´e de Montr´eal. ,1998.

[151] Kikuchi S Scheduling of demand-responsive transit vehicles. J Transp Eng, 110:511–520, 1984.

[152] Kikuchi S, Rhee J Scheduling algorithms for demand-responsive transportation system. J Transp Eng, 115:630–645,1989.

[153] J.-Y. Potvin, J.-M. Rousseau, A parallel route building algorithm for the vehicle routing and scheduling problem with time windows, European Journal of Operational Research 66, 331-340,1993.

[154] Alexandre Beaudry , Gilbert Laporte, Teresa Melo · Stefan Nickel, Dynamic transportation of patients in hospitals, OR Spectrum, 32:77–107,2010.

[155] Greg N. Frederickson and D. J. Guan. Nonpreemptive ensemble motion planning on a tree, J. Algorithms,15(1):29–60, 1993.

[156] Benjamin Hiller, Bad Guys are Rare: Probabilistic Analysis of an Elementary Dial-a-Ride Problem, Phd Thesis, 05/07/2004.

[157] Horn MET, Fleet scheduling and dispatching for demand-responsive passenger services. Transport Res C-Emer, 10:35–63,2002

[158] Horn MET, Multi-modal and demand-responsive passenger transport sys systems: a modeling framework with embedded control systems. Transport Res A-Pol, 36:167–188,2002b.

[159] Diana M, Dessouky M, Xia N A model for the fleet sizing of demand responsive transportation services with time windows. Transport Res B-Meth, 40:651–666,2006.

[160] Diana M. The importance of information flows temporal attributes for the efficient scheduling of dynamic demand responsive transport services. J Adv Transport 40:23–46,2006

[161] Ascheur N.Krumke S., Rambau . Online dial-a-ride problems: Minimizing the completion time. In STACS 2000 Lecture notes in computer science. Springer, Berlin,1770:639-650,2000

[162]Hauptmeier D., Krumke S., Rambau J. The online dial-a-ride problem under reasonable load. In Proceedings of Algorithms and Complexity, 4th Italian Conference,Lecture Notes in Computer Science. Springer, Berlin, 1767:125-136,2000.

[163]Madsen G., Ravn H. , Rygaad J. A heuristic algorithm for a dial-a-ride problem with time windows , multiple capacities, and multiple objects. Anals of Operations Research, 60:193-208,1995

Appendix A

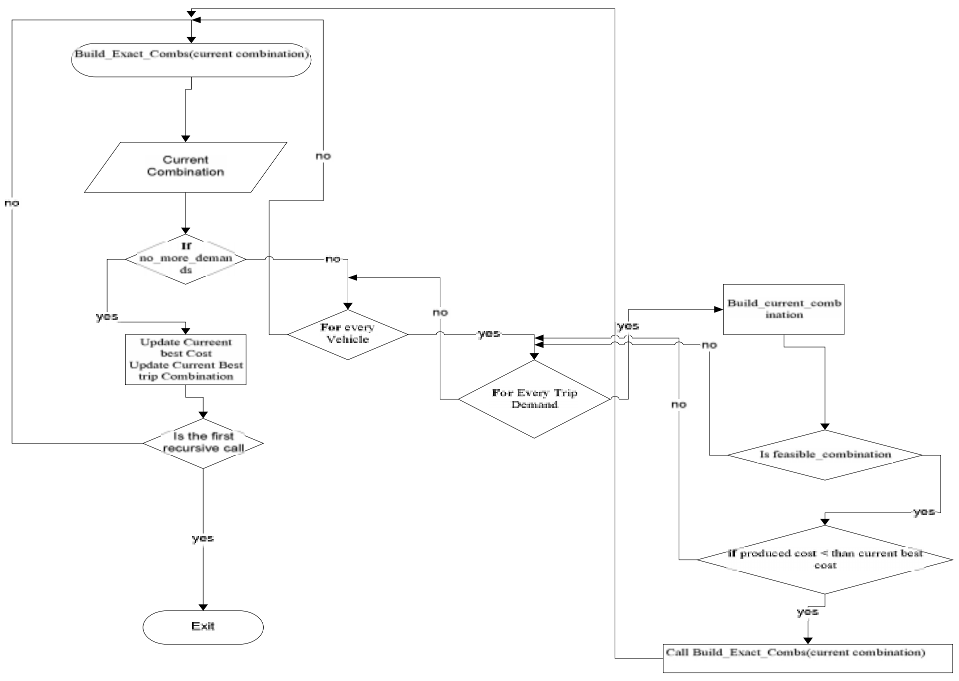
For the formulation mentioned in chapter 3, the following implementation in AMPL language was created for one vehicle.

```
set VEHICLES; # vehicles
param vehicles > 0; # vehicles number
set VSD;      # start depot
param vsd > 0; # start depots number
set VED;      # end depot
param ved > 0; # end depots number
set VSED := VSD union VED; # start and end depots
param Q >0; # vehicle capacity
param MVRT >0; # vehicle ride time


param NtPickup > 0;
param Nt = NtPickup*2;
param NtVSD = Nt + vsd;
param NtVED = Nt + ved;
param NtVSED = Nt + vsd + ved;

set Pplus;
set Pminus;
set P := Pplus union Pminus;
set PVSD := VSD union P;
set PVED := P union VED;
set PVSDplus := VSD union Pplus;
set PVEDplus := Pminus union VED;
set PVSED := PVSDplus union PVEDplus;

param earlier{PVSED};
param later{j in PVSED}> earlier[j];
param drt{P};
param travel_time{PVSED,PVSED};
param travel_cost{PVSED,PVSED};
param service_time{PVSED};
param load{PVSED};
var xvij{i in PVSED, j in PVSED} binary;
var Tvi{i in PVSED} >= earlier[i];
var Lvi{i in PVSED}>=0, <= (Nt div 2);

minimize total_cost_darp : sum {i in PVSED,j in PVSED} travel_cost[i,j]*xvij[i,j];

#ensure flow constrains
sum {j in PVSED} xvij[j,i+NtPickup] = 0;
subject to network_flow1 {i in Pplus} : sum {j in PVSED} xvij[i,j] = 1;
subject to network_flow11 {i in VSD union P}  : sum {j in PVSED} xvij[i,j] = 1;
subject to network_flow12 {j in P union VED}  : sum {i in PVSED} xvij[i,j] = 1;
subject to network_flow13 {i in PVSED} : xvij[i,i] = 0;
subject to network_flow14 {j in VSD} : sum {i in P union VED} xvij[i,j] = 0;
subject to network_flow15 {i in VED} : sum {j in P union VSD} xvij[i,j] = 0;
subject to network_flow2 {i in P} : sum {j in PVSED}  xvij[i,j] - sum {j in PVSED} xvij[j,i] = 0;
subject to network_flow3 {i in VSD} : sum {j in Pplus} xvij[i,j] =1;
subject to network_flow4 {j in VED} : sum {i in PVSED} xvij[i,j] =1;
subject to network_flow5 {i in Pplus} : sum {j in PVSED} xvij[i,j] - sum {j in PVSED} xvij[i+NtPickup,j] = 0;

#ensure time constrains
subject to time_service1  {i in Pplus} : Tvi[i]+service_time[i]+travel_time[i,NtPickup+i] <= Tvi[NtPickup+i];
subject to time_service2  {j in P, i in P} : Tvi[j]-Tvi[i]-service_time[i]-travel_time[i,j] >= (xvij[i,j]-1)*9999999999;
subject to time_service3  {i in VSD, j in Pplus} : Tvi[j]-Tvi[i]-travel_time[i,j] >= (xvij[i,j]-1)*9999999999;
subject to time_service4  {i in Pminus, j in VED} : Tvi[j]-Tvi[i]-service_time[i]-travel_time[i,j] >= (xvij[i,j]-1)*99999999;

# ensure time windows constrains
subject to time_windows1 {i in P} : earlier[i]<= Tvi[i] <= later[i];
subject to time_windows2 {i in VSD} : earlier[i]<= Tvi[i] <= later[i];
subject to time_windows3 {i in VED} : earlier[i]<= Tvi[i] <= later[i];
```

```
#ensure load constrains
subject to load_balance1 {i in Pplus union VSD,j in Pplus} : Lvi[j]-Lvi[i]-load[j]>=(xvij[i,j]-1)*999999999;
subject to load_balance2 {i in Pplus,j in Pminus} : Lvi[j]-load[j]-Lvi[i]>=(xvij[i,j]-1)*999999999; # j to i
subject to load_balance3 {i in Pminus,j in Pminus} : Lvi[j]-load[j]-Lvi[j]>=(xvij[i,j]-1)*99999999;# j to i
subject to load_balance4 {i in Pminus,j in Pplus} : Lvi[j]-Lvi[i]-load[j]>=(xvij[i,j]-1)*99999999;
subject to load_balance5  {i in VSD,j in Pplus} : Lvi[j]-Lvi[i]-load[j]>=(xvij[i,j]-1)*999999999;

#ensure customers max ride time;
subject to max_cust_drive_time {i in P} : Tvi[i] <= drt[i];

#ensure vehicle max ride time;
subject to exceed_vehicle_ride_time  {i in PVSED} : Lvi[i]<= MVRT;

#ensure vehicle max load;
subject to max_vehicle_load  {i in PVSED} : Lvi[i]<= Q;
```

Appendix B

A. DP Exact algorithm Chart



Build_Exact_Combs(current combination)
Current Combination
If no_more_demands
yes
no
Update Current best Cost Update Current Best trip Combination
Is the first recursive call
yes
Exit
no
For every Vehicle
no
yes
For Every Trip Demand
yes
Build_current_combination
no
Is feasible_combination
yes
no
if produced cost < than current best cost
yes
Call Build_Exact_Combs(current combination)

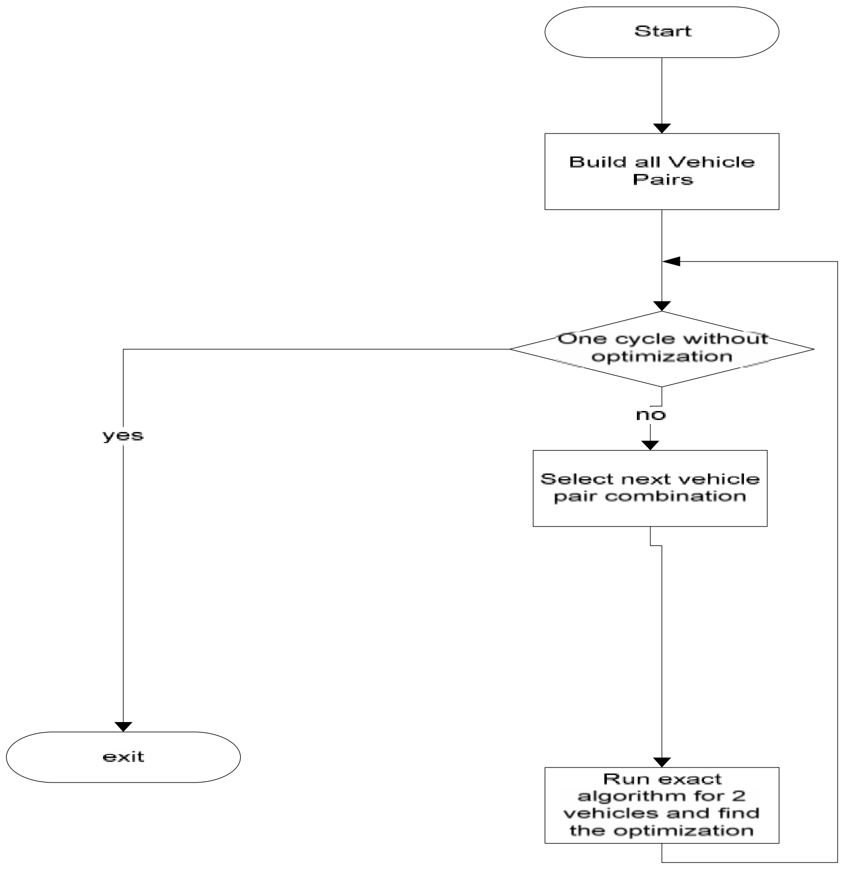
B. DP Exact algorithm Pseudocode

```
Function Build_Exact_Combs(current combination)
If no_more_requests then
Update Current Best Cost
Update Current Best Trip Combination
Return from Build_Exact_Combs (recursive return)
Else
        For every Vehicle
                For Every Trip Request
                        Build_current_combination
                        If not a feasible_combination then
                                Continue next comb
                        Else if not produced cost < than current best cost then
                                Continue next comb
                        Else
                                Call Build_Exact_Combs(current combination)// recursive
                                Return from Build_Exact_Combs (recursive return)
                        End if
                End for
        End for
End if
End
```

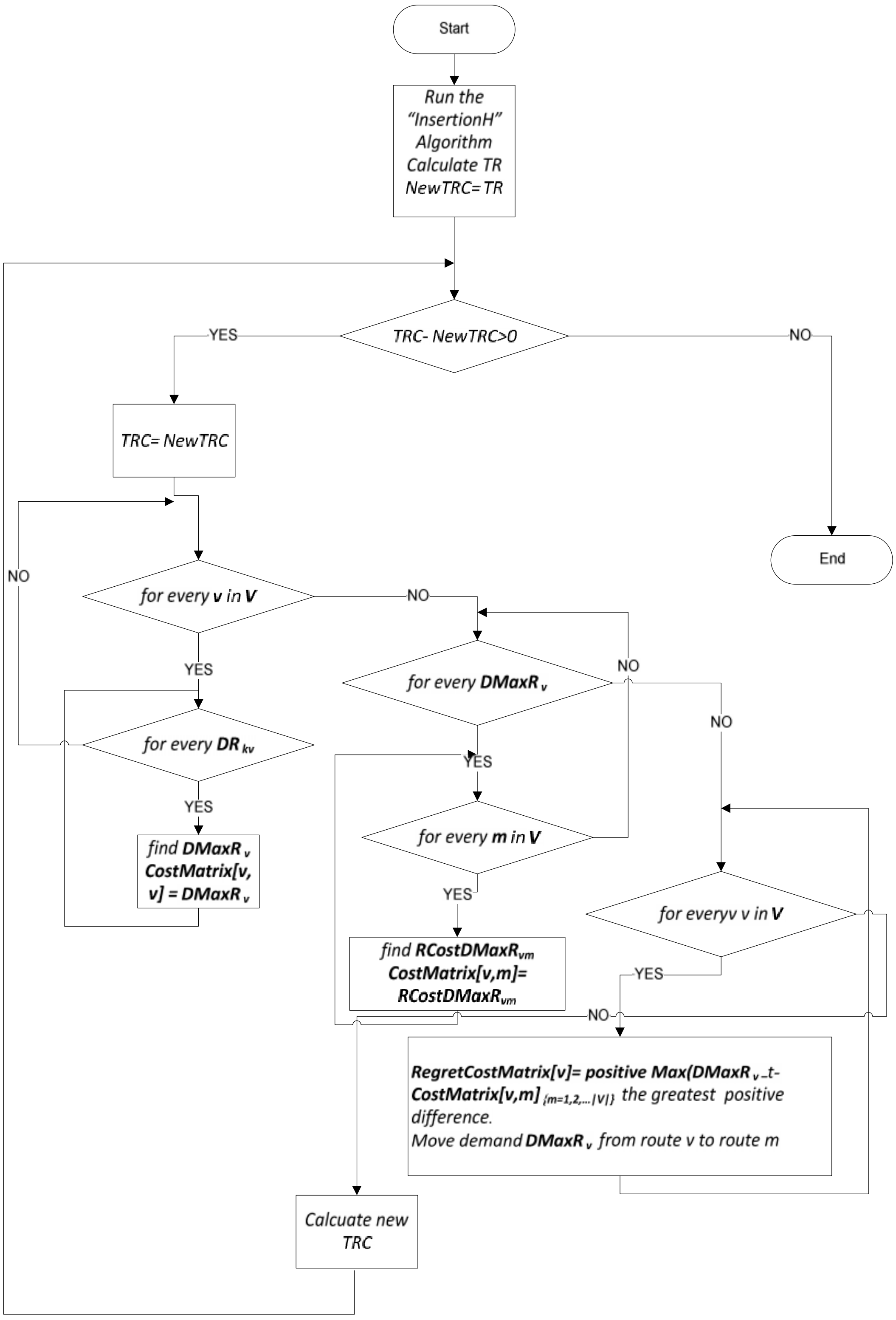
C. *DP Exact algorithm Samples Execution Sequence*

```
Combination : V1P1
Combination : V1P1V1D1
Combination : V1P1V1D1V1P2 Not feasible solution. (further search is discontinued)
Combination : V1P1V1D1V2P2
Combination : V1P1V1D1V2P2V2D2 Feasible solution but over current best cost (further search is dis-
continued): 193.861
Combination : V1P1V1P2 Not feasible solution. (further search is discontinued)
Combination : V1P1V2P2
Combination : V1P1V2P2V1D1
Combination : V1P1V2P2V1D1V2D2 Feasible solution but over current best cost (further search is dis-
continued): 193.861
Combination : V1P1V2P2V2D2 Feasible solution but over current best cost (further
search is discontinued): 193.861
Combination : V2P1
Combination : V2P1V2D1
Combination : V2P1V2D1V1P2 Feasible solution but over current best cost (further
search is discontinued): 193.861
Combination : V2P1V2D1V2P2 Not feasible solution. (further search is discontinued)
Combination : V2P1V1P2
Combination : V2P1V1P2V2D1 Feasible solution but over current best cost (further
search is discontinued): 193.861
Combination : V2P1V1P2V1D2
Combination : V2P1V1P2V1D2V2D1 Feasible solution but over current best cost (further search is dis-
continued): 193.861
Combination : V2P1V2P2 Not feasible solution. (further search is discontinued)
Combination : V1P2
Combination : V1P2V1P1
Combination : V1P2V1P1V1D1
Combination : V1P2V1P1V1D1V1D2 Not feasible solution. (further search is discontinued)
Combination : V1P2V1P1V1D2 Not feasible solution. (further search is discontinued)
Combination : V1P2V2P1
Combination : V1P2V2P1V2D1 Feasible solution but over current best cost (further
search is discontinued): 193.861
Combination : V1P2V2P1V1D2
Combination : V1P2V2P1V1D2V2D1 Feasible solution but over current best cost (further search is dis-
continued): 193.861
Combination : V1P2V1D2
Combination : V1P2V1D2V1P1
Combination : V1P2V1D2V1P1V1D1
Combination : V1P2V1D2V1P1V1D1 New best cost found :191.941
Combination : V1P2V1D2V2P1
Combination : V1P2V1D2V2P1V2D1 Feasible solution but over current best cost (further search is dis-
continued): 191.941
Combination : V2P2
Combination : V2P2V1P1
Combination : V2P2V1P1V1D1
Combination : V2P2V1P1V1D1V2D2 Feasible solution but over current best cost (further search is dis-
continued): 191.941
Combination : V2P2V1P1V2D2 Feasible solution but over current best cost (further
search is discontinued): 191.941
Combination : V2P2V2P1
Combination : V2P2V2P1V2D1
Combination : V2P2V2P1V2D1V2D2 Not feasible solution. (further search is discontinued)
Combination : V2P2V2P1V2D2 Not feasible solution. (further search is discontinued)
Combination : V2P2V2D2
Combination : V2P2V2D2V1P1 Feasible solution but over current best cost (further
search is useless): 191.941
Combination : V2P2V2D2V2P1
Combination : V2P2V2D2V2P1V2D1 Feasible solution but over current best cost (futher search is dis-
continued): 191.941
 Algorithm Ended : (MIN)V1P2V1D2V1P1V1D1 With Cost : 191.941
 Time Elapsed in secs : 0.094
Total Combs : 46 Reject Combs : 21
```

D. VLSN algorithm Chart


Start
Build all Vehicle Pairs
One cycle without optimization
no
yes
Select next vehicle pair combination
exit
Run exact algorithm for 2 vehicles and find the optimization

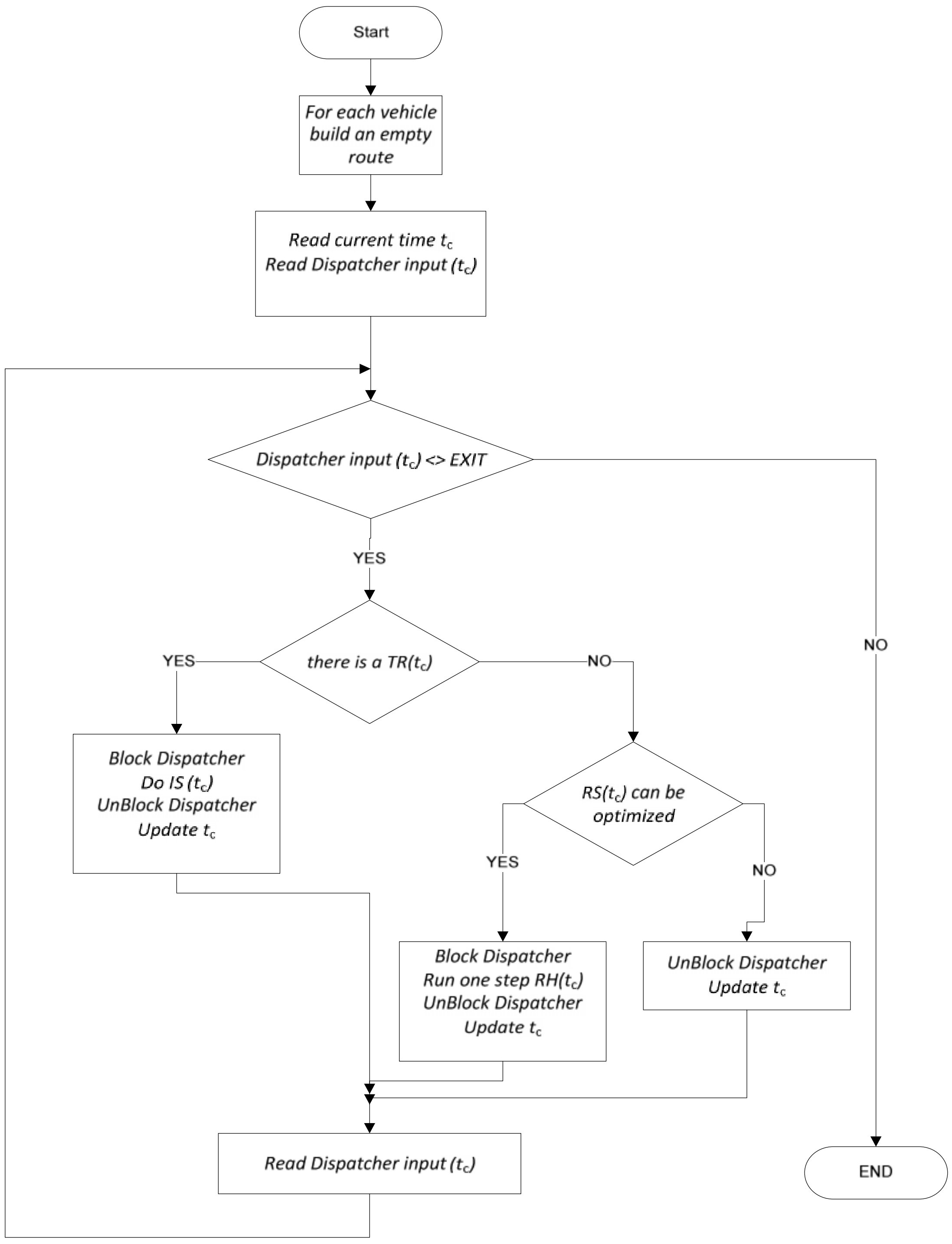
E. RegretH Algorithm Chart


Start
Run the "InsertionH" Algorithm Calculate TR NewTRC= TR
TRC- NewTRC>0
YES
NO
TRC= NewTRC
End
for every v in V
NO
YES
for every DR kv
YES
find DMaxR v CostMatrix[v, v] = DMaxR v
for every DMaxR v
NO
YES
for every m in V
YES
find RCostDMaxR vm CostMatrix[v,m]= RCostDMaxR vm
for everyv v in V
YES
NO
RegretCostMatrix[v]= positive Max(DMaxR v –t- CostMatrix[v,m] {m=1,2,...|V|} the greatest positive difference. Move demand DMaxR v from route v to route m
Calcuate new TRC

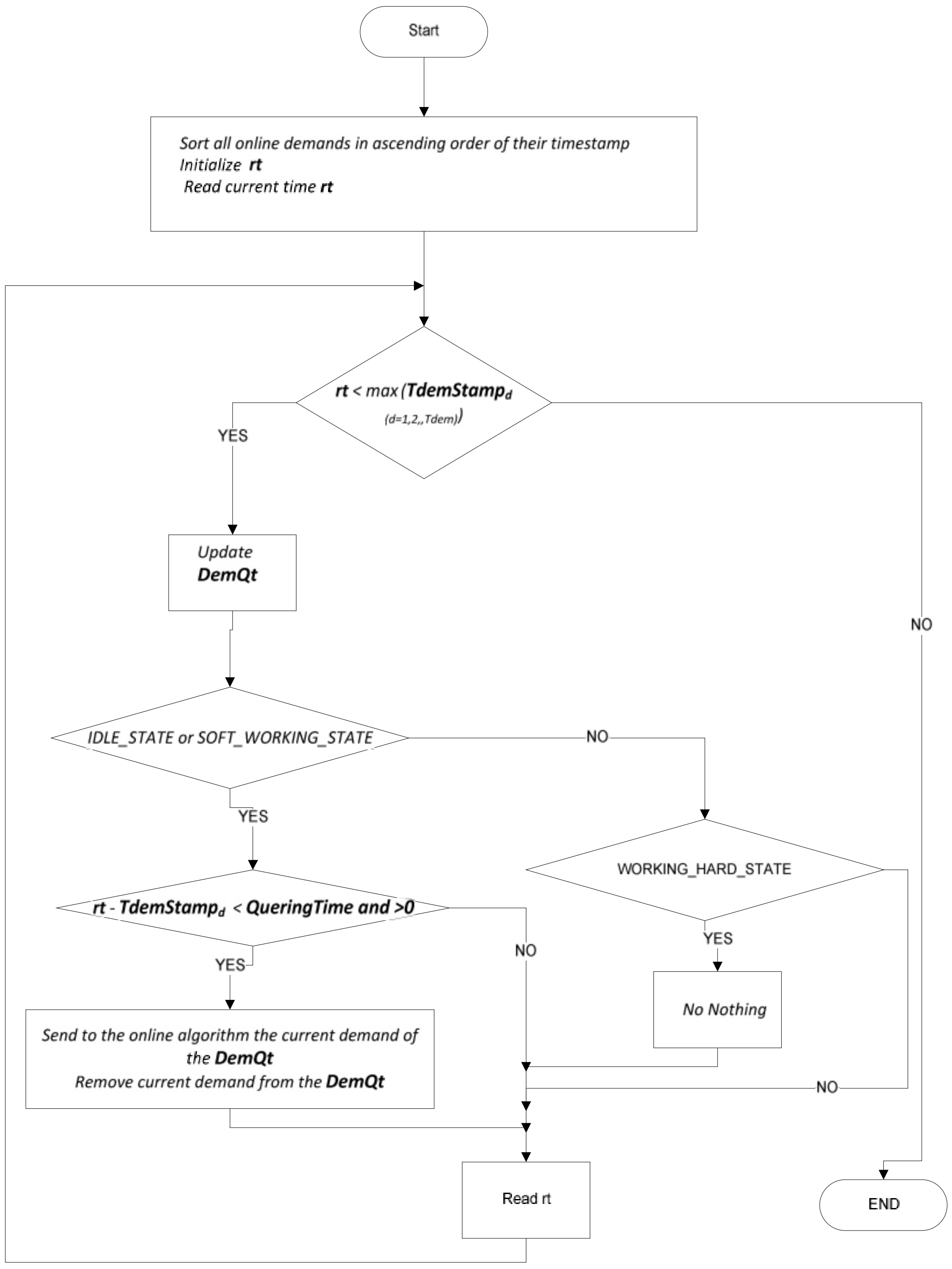
F. OR-DARP Algorithm Pseudocode

```
For each vehicle v (v=1,2,3…..|V|)
        Build an empty route
Read current time $t_c$
Define $maxRT$
Read Dispatcher input ($t_c$)
While Dispatcher input ($t_c$) <> EXIT do
    If there is a TR($t_c$)
            Block  Dispatcher (signal HARD_WORKING_STATE)
            Do InsertionH ($t_c$)
            UnBlock Dispatcher (signal SOFT_WORKING_STATE)
            Update $t_c$
    Else if RS($t_c$) can be optimized more
            Block  Dispatcher (signal HARD_WORKING_STATE)
            Run one step RegretH($t_c$)
            UnBlock Dispatcher (signal SOFT_WORKING_STATE)
            Update $t_c$
    Else
            UnBlock Dispatcher (signal IDLE _STATE)
            Update $t_c$

    End if
    Read Dispatcher input ($t_c$)
End while
```

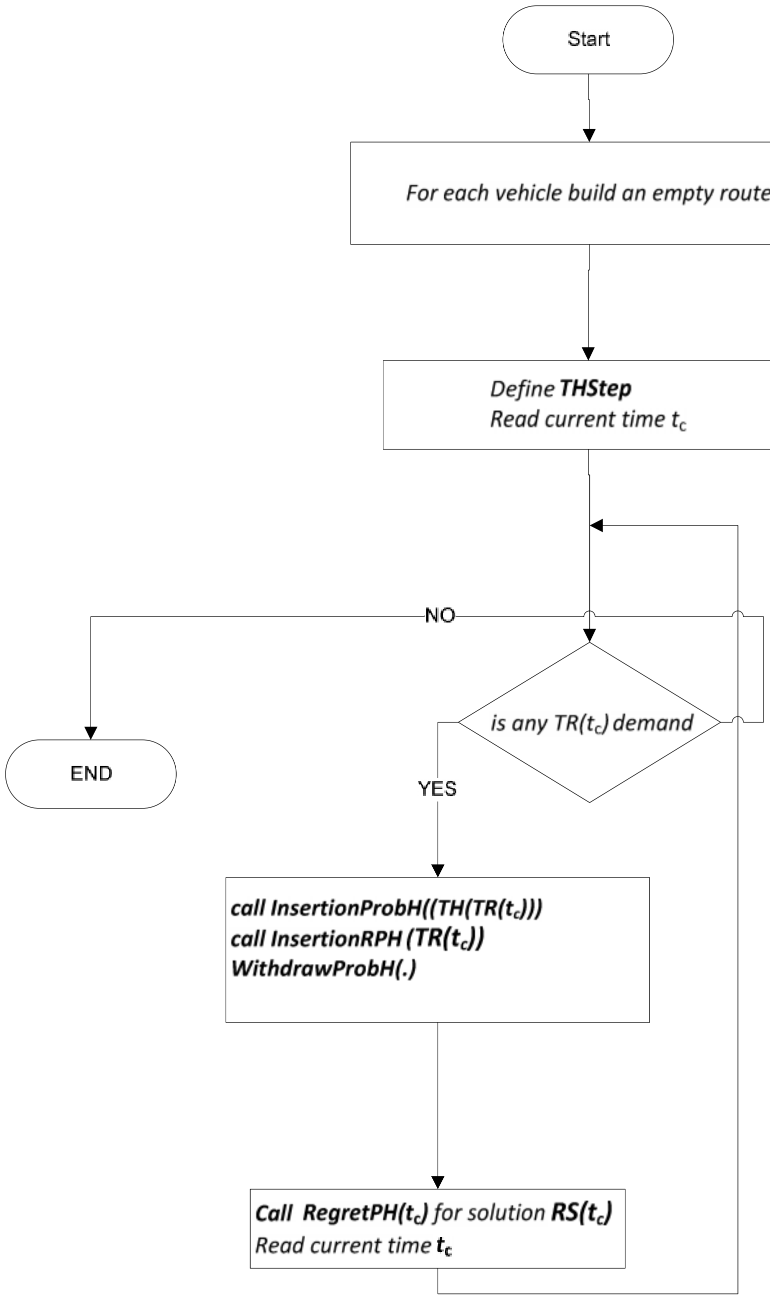
G. OR-DARP Algorithm Chart

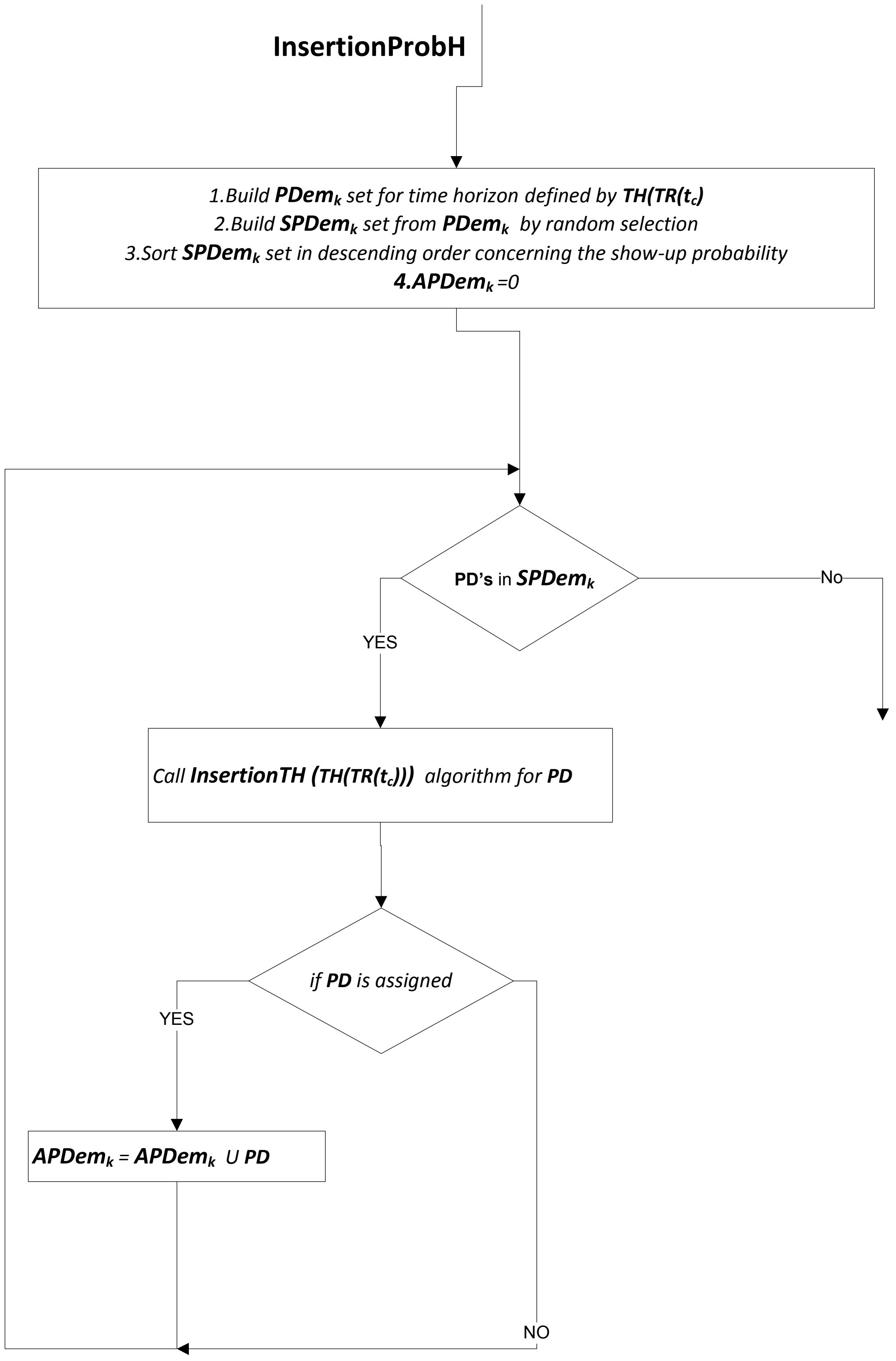


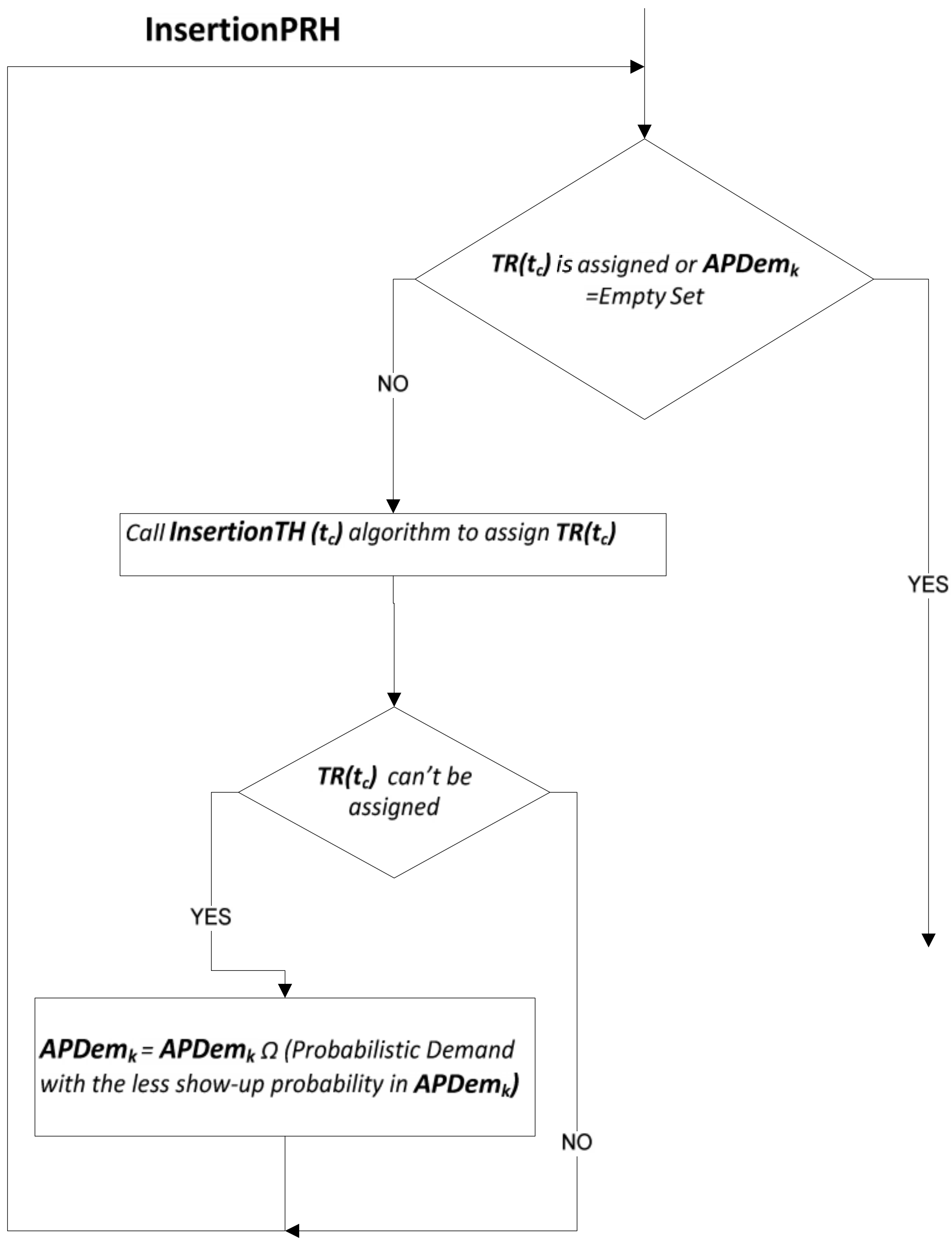
H. OR-DARP Algorithm Dispatcher Module Chart


Start
Sort all online demands in ascending order of their timestamp
Initialize rt
Read current time rt
rt < max (TdemStamp_d (d=1,2,,Tdem))
YES
NO
Update DemQt
IDLE_STATE or SOFT_WORKING_STATE
NO
YES
WORKING_HARD_STATE
YES
rt - TdemStamp_d < QueringTime and >0
NO
YES
No Nothing
Send to the online algorithm the current demand of the DemQt
Remove current demand from the DemQt
NO
Read rt
END

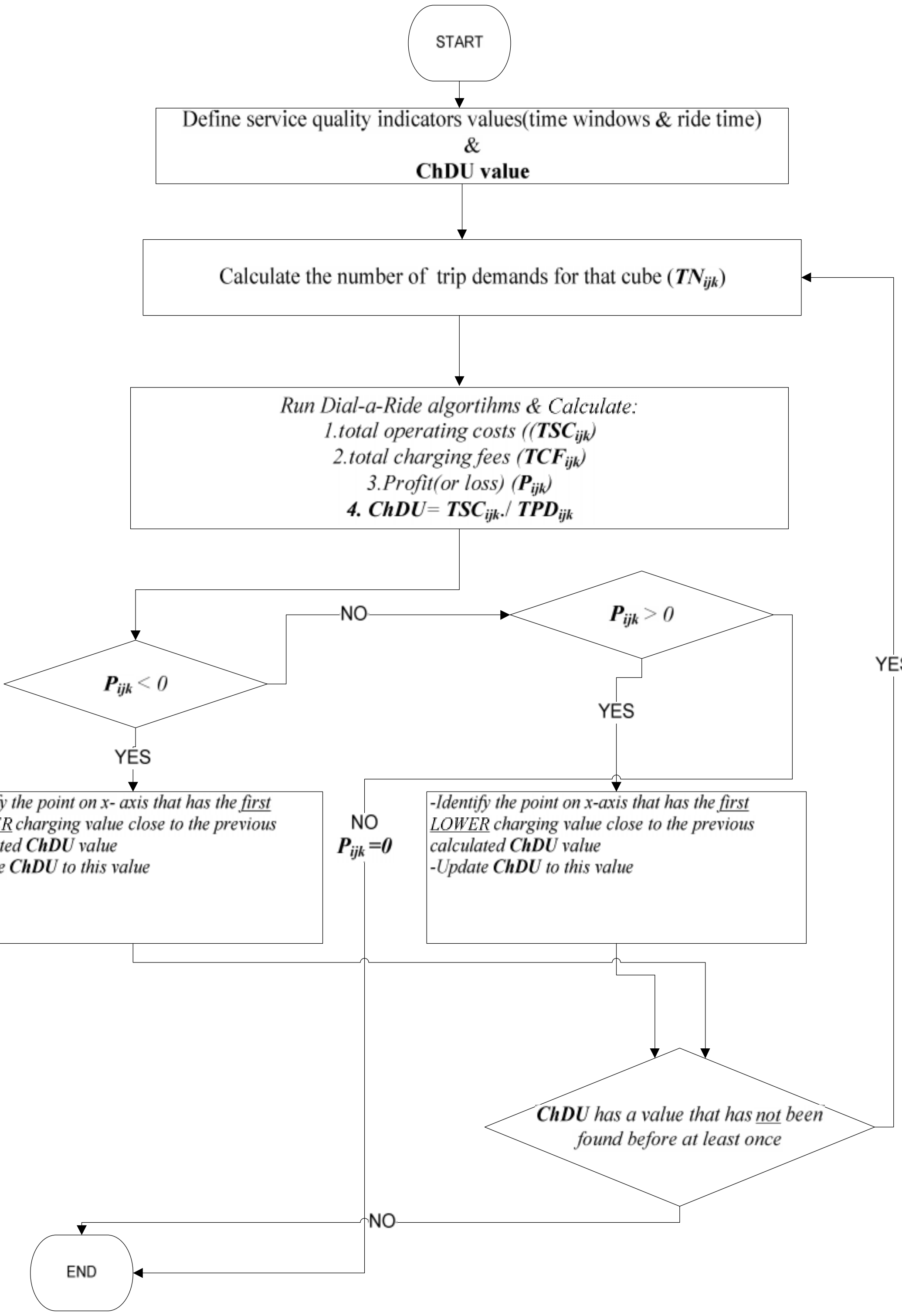
I. *OP-DARP Algorithm Chart*


Start
For each vehicle build an empty route
Define ***THStep***
Read current time t_c
NO
is any TR(t_c) demand
YES
END
call InsertionProbH((TH(TR(t_c)))
call InsertionRPH (TR(t_c))
WithdrawProbH(.)
Call RegretPH(t_c) *for solution* ***RS(t_c)***
Read current time $\mathbf{t_c}$


InsertionProbH
1.Build PDem_k set for time horizon defined by TH(TR(t_c)
2.Build SPDem_k set from PDem_k by random selection
3.Sort SPDem_k set in descending order concerning the show-up probability
4.APDem_k =0
PD's in SPDem_k
No
YES
Call InsertionTH (TH(TR(t_c))) algorithm for PD
if PD is assigned
YES
APDem_k = APDem_k U PD
NO


InsertionPRH
TR(t_c) is assigned or APDem_k =Empty Set
NO
YES
Call InsertionTH (t_c) algorithm to assign TR(t_c)
TR(t_c) can't be assigned
YES
NO
APDem_k = APDem_k Ω (Probabilistic Demand with the less show-up probability in APDem_k)

J. Convergence Algorithm Chart


START
Define service quality indicators values(time windows & ride time) & ChDU value
Calculate the number of trip demands for that cube (TN_{ijk})
Run Dial-a-Ride algortihms & Calculate:
1.total operating costs ((TSC_{ijk})
2.total charging fees (TCF_{ijk})
3.Profit(or loss) (P_{ijk})
4. ChDU= TSC_{ijk}./ TPD_{ijk}
$P_{ijk} < 0$
NO
$P_{ijk} > 0$
YES
YES
-Identify the point on x- axis that has the first BIGGER charging value close to the previous calculated ChDU value
-Update ChDU to this value
NO
$P_{ijk} = 0$
-Identify the point on x-axis that has the first LOWER charging value close to the previous calculated ChDU value
-Update ChDU to this value
YES
ChDU has a value that has not been found before at least once
NO
END

Appendix C

1. The time they are willing to wait for pick-up on the average
(Possible answers)
   1. 0 – 5 minutes
   2. 6 – 10 minutes
   3. 11 – 15 minutes
   4. 16 – 20 minutes
   5. 21 – 25 minutes
   6. > 25 minutes

2. The amount of money that they are willing to pay in order to travel faster
(Possible answers)
   1. Bus Price (17 euro cents / per KM)
   2. 1.5 more than bus price (25.5 euro cents / per KM)
   3. Twice the bus price (34 euro cents / per KM)
   4. 0.75 taxi price (53.25 euro cents / per KM)
   5. Taxi price (71 euro cents / per KM)

3. The maximum travel time that they are willing to spend travelling in comparison of taxi travel time for the same distance.
   1. Same as taxi *(1 time the SP travel time)*
   2. 1.5 the taxi time*(1.5 times the SP travel time)*
   3. Twice the taxi time*(2 times the SP travel time)*
   4. 0.5 Bus Time*(2.33 times the SP travel time)*
   5. 0.75 Bus Time *(2.8 times the SP travel time)*
   6. Bus time *(3.5 times the SP travel time)*

4. If they were willing to use the new DRT system.
   1. Yes
   2. No