THE REPUBLIC OF TURKEY

BAHÇEŞEHİR UNIVERSITY

A HEURISTIC FRAMEWORK FOR SOLVING TIME DEPENDENT VEHICLE ROUTING PROBLEM WITH TIME WINDOWS

Master's Thesis

ALPER YASİN SARICIOĞLU

İSTANBUL, 2014

THE REPUBLIC OF TURKEY
BAHÇEŞEHİR UNIVERSITY

GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
INDUSTRIAL ENGINEERING

A HEURISTIC FRAMEWORK FOR SOLVING TIME DEPENDENT VEHICLE ROUTING PROBLEM WITH TIME WINDOWS

Master's Thesis

ALPER YASİN SARICIOĞLU

Supervisor: ASST. PROF. İBRAHİM MUTER

İSTANBUL, 2014

THE REPUBLIC OF TURKEY
BAHCESEHIR UNIVERSITY

GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
INDUSTRIAL ENGINEERING

Name of the thesis: A Heuristic Framework for Solving Time Dependent Vehicle Routing Problem with Time Windows

Name/Last Name of the Student: Alper Yasin SARICIOĞLU
Date of the Defense of Thesis: 01.09.2014

The thesis has been approved by the Graduate School of Natural And Applied Sciences.

Assoc. Prof. Tunç BOZBURA
Graduate School Director
Signature

I certify that this thesis meets all the requirements as a thesis for the degree of Master of Arts.

Assoc. Prof. Barış SELÇUK
Program Coordinator
Signature

This is to certify that we have read this thesis and we find it fully adequate in scope, quality and content, as a thesis for the degree of Master of Arts.

Examining Committee Members	Signature
Thesis Supervisor Asst. Prof. İbrahim MUTER	----------------------------------
Member Asst. Prof. Demet ÖZGÜR ÜNLÜAKIN	----------------------------------
Member Assoc. Prof. Temel ÖNCAN	----------------------------------

ACKNOWLEDGEMENTS

Firstly, I thank to my father for his unquestioning support that he has always gave me.

I thank to my friend, Mustafa Gündoğar, who helped me with his magical touches whenever I stuck in codding.

And, of course, I thank to my supervisor Asst. Prof. İbrahim Muter for his guidance, patience and continuous support throughout this study.

İstanbul, 2014 Alper Yasin Sarıcıoğlu

CONTENTS

TABLES

FIGURES

ABBREVIATIONS

IP:	Integer Programming
LP:	Linear Programming
LPR:	Linear Programming Relaxation
RSCP:	Relaxation Set Covering Problem
SCP:	Set Covering Problem
TDVRP:	Time Dependent Vehicle Routing Problem
TDVRPTW:	Time Dependent Vehicle Routing Problem with Time Windows
VRP:	Vehicle Routing Problem
VRPTW:	Vehicle Routing Problem with Time Windows

SYMBOLS

Upper bound value obtained by metaheuristic at iteration k	:	$Z_{UB}(k)$
Potential lower bound obtained by the RSCP	:	$Z_{PLB}(k)$
Limit values of counters	:	$\bar{r}_1,\ \bar{r}_2$
Critical gap between $Z_{UB}(k)$ and $Z_{PLB}(k)$	:	α
The travel speed in time period T_k	:	ϑ_{cT_k}
Arrival time	:	t'
Time period	:	T_k
Upper and lower bounds of time period	:	$\underline{t}_k, \bar{t}_k$
The distance between i and j	:	d_{ij}
Random customer list size	:	q
Neigbour list size	:	h
The number of consecutive iterations	:	I_{tot}
The number of consecutive $I_{2opt}*$ iterations	:	$I_{2opt}*$
The number of consecutive $I_{or-relocate}$ iterations	:	$I_{or-relocate}$

ABSTRACT

A HEURISTIC FRAMEWORK FOR SOLVING TIME DEPENDENT VEHICLE ROUTING PROBLEM WITH TIME WINDOWS

Alper Yasin Sarıcıoğlu

Industrial Engineering

Thesis Supervisor: Asst. Prof. İbrahim Muter

September, 2014, 59 pages

In vehicle routing problem, most of the solution approaches deals with problems that the travel time between two demand points is assumed constant. While dealing with long distances like intercity transportation, variation in travel speed is negligible, it is essential when it comes to short distances and high traffic density. However as GPS technology got more common for civilian use in the last decade; it became possible to collect instant traffic data from drivers who carry a GPS application in their vehicle and share it with a geospatial database.

These technological developments intrigued researchers to study with more real-life problems such as time-dependent vehicle routing problem with time windows which is assuming the travel speed is perfectly correlated with time of the day and customers want their demand is satisfied within a predefined time period.

This study suggests a Tabu Search algorithm guided by MetaOpt framework for TDVRPTW problems. Optimizing departure times in vehicle routes are considered with diversification. Though some approaches are developed and introduced, they all focus on post-processing of the main problem. This study is aimed to search different-than-zero departure time in tabu search is processing.

Keywords: Time Dependency, Time Windows, Departure Time Optimization, Tabu Search, City Logistics

ÖZET

ZAMANA BAĞLI VE ZAMAN PENCERELİ ARAÇ ROTALAMA PROBLEMİ İÇİN BİR SEZGİSEL ÇÖZÜM YAKLAŞIMI UYGULAMASI

Alper Yasin Sarıcıoğlu

Endüstri Mühendisliği

Tez Danışmanı: Yrd. Doç. Dr. İbrahim Muter

Eylül, 2014, 59 sayfa

Araç rotalama problemde, çoğu çözüm yaklaşımı iki talep noktası arasındaki seyahat süresini sabit kabul eder. Seyahat hızının değişkenliği, şehirlerarası ulaşım gibi uzun mesafelerde görmezden gelinebilirken kısa mesafelerde ve yüksek trafik yoğunluğundan etkilenen problemler için önemlidir.

GPS teknolojisinin son on yılda sivil kullanım için yaygınlaşması ile aracında bir GPS uygulaması taşıyan ve bunu bir coğrafi veri tabanı ile paylaşan sürücülerin anlık trafik verilerini toplamak mümkün olmuştur.

Bu teknolojik gelişmeler pek çok araştırmacıyı zaman bağımlı hız ve zaman pencereli dağıtım gibi daha gerçekçi problemlere yönlendirmiştir.

Bu çalışma Zaman Bağımlı ve Zaman Pencereli Araç Rotalama Problemi için MetaOpt tarafından yönlendirilen bir Tabu Arama yaklaşımı önerir. Önemli bir değişken olan başlangıç zamanının değiştirilmesi literatürde rotalar belirli olduktan sonra ikinci bir çalışma ile çözülmeye çalışılmaktadır. Bu çalışma sıfırdan farklı başlangıç zamanlarını tabu arama algoritması devam ederken araştırmayı amaçlamaktadır.

Anahtar Kelimeler: Zaman Bağımlılık, Zaman pencereleri, Kalkış zamanı optimizasyonu, Tabu Arama, Şehir Lojistiği

1 INTRODUCTION

Logistics and its significance in business strategy have changed dramatically due to revolutionary experiences like Just-in-Time and Total Quality Management systems in the 1980s. As rapidly developing information and communication technologies implemented on different levels of supply chain, it gradually changed the way to manage production and distribution processes, such as, material handling, inventory control and internal and external communication (Persson 1991).

Fierce competition, expanded to global scale, altered what is understood and what is expected from logistics. Linking supply and demand nodes at profitable cost and service levels have become harder as new opportunities brings with it new demands.

Economical offers of technological developments are quickly adapted by private companies to gain advantages and having more shares in highly competitive markets. These information technologies boost managing capabilities due to its bidirectional communication capabilities and also open new aspects for urban governance to design transportation networks as well as it does it for private companies.

Since logistics operations require a number of performances within or between business partners, it is highly dependent on information and communication technologies more than any other sector in the business environment. This intense flow of information created a need to develop management systems based on information technology solutions, and this brought a high level of maturation in logistics in terms of implementation of complex information systems.

Although logistics systems comprise different types of activities, among them, physical distribution has a crucial role in steps of adding value to products. Transportation of goods to a demand point transforms it into a commodity for final customer.

Transportation operations can be simply defined as the process of moving goods from one point to another but it occupies a large part of a company’s overall logistics cost and determines the total success of supply chain of the company. Moreover, an efficient transportation operation allows companies to have shorter delivery times and fewer inventory costs which provide them a big advantage in competitive markets, besides savings in operational costs.

City logistics, on the other hand, is a developing concept of transportation management, targeting problems in major cities, caused by massive transportation needs in a limited environment. Highly complex nature of urban transportation operations and technological developments such as GPS technology created a new dimension in logistics. Serious problems like traffic congestions, environmental issues, human health problems and expanding urban areas and increasing urban population needed urgent measures and make this topic studied by not only private companies, but also governments.

As mentioned above, implementing solutions obtained by computer environment is specifically practical for transportation operations, many methods and approaches are proposed to seek out a solution or provide decision support over the years.

Combinatorial nature of logistics problems causes a heavy computational burden that makes exact methods impractical because of exponential explosion. Since heuristics provides fast and satisfactory solutions to problems, which require long computational time and memory need for an exact solution, they have an important place in logistics decision making. The performance advantages of heuristics allow to find optimal or near-optimal solutions for large scale problems with real life constraints (Ballou 1989).

Urban freight transportation had a great role in the evolution of Vehicle Routing Problems (VRP). Short distances and traffic congestions pushed researchers to study time dependency concept. Technological enablers such as GPS expanded coverage of real-time and historical traffic information in urban areas. Also, this practical availability led researchers to emphasize time dependency in the last decade.

Time Dependent Vehicle Routing Problem (TDVRP) is a variation of Vehicle Routing Problem (VRP), to model speed limitations dealt in urban transportation. Unlike VRP, travel speed is not constant and depends on time of day.

On the other hand, developments in e-commerce practices change the structure of goods deliveries and passenger transports in the cities. Highly customized product scale, lower volume and weight but higher value of the goods, increase the number of drops per tour in urban freight transportation (Huschebeck 2004).

One of the main operations the urban freight transportation is home delivery. It constitutes an important part of urban freight transportation operations. Customers who buy large goods that cannot be transported by their private vehicles or e-commerce customers, who gave their order online, require a transportation service by the supplier of the product. This gives dispatchers unique opportunities to schedule the fleet, owned by or dedicated to the supplier. Volume or weight of customer demand and location are known by the dispatcher prior to the operation. In these operations there is a rendezvous with customer within a time window.

This feature leads us to Vehicle Routing Problem with Time Windows (VRPTW) which is another variant of the Vehicle Routing Problem. It restricts arrival times of vehicles in a feasible time window and each customer requires a different volume to be delivered within a predefined deadline. These time windows vary from customer to customer. Since all demand nodes must be served before a predefined early and late time windows, these sequential constraints, which has low computation cost, can be used for pruning solution space.

Both hard time window, which does not allow vehicle to exceed late time window, and soft time window, which allows exceed but adds a penalty cost, are studied in the literature. In this study, time windows are cannot be exceeded.

In this study VRPTW and time dependency feature is held together. With these two merged features of VRP, the new problem is called Time Dependent Vehicle Routing Problem with Time Windows, which is studied in the field of combinatorial optimization extensively.

1.1 MOTIVATION

In vehicle routing problem, most of the models deals with problems assume that travel time between two demand points is constant. While dealing with long distances like intercity transportation, deviation in travel speed is negligible but it is crucial when it comes to short distances and high traffic density in urban transportation.

However as GPS technology got more common for civilian use in the last decade; it became possible to collect instant traffic data from drivers who carry a GPS application in their vehicles and share it with a geospatial database. Thus, instant traffic information is not covered for only main road segments which are monitored by speed radars or cameras but also for less major thoroughfares especially in urban areas.

Today this data is used for travel time calculations by popular map applications like Google and Yandex. As GPS applications will get common, coverage of these geospatial services will expand. These technological developments and urbanization trend make the time dependency a practical property for real life problems.

Today, urban population comprises 50% of global population. This rate is expected to increase to 60% in the next twenty years (World Health Organization 2009). Increasing urban population in limited geographical space, also, have environmental aspects of distribution operations in urban areas and will be crucial in the future more than it has ever been.

High correlation between GHG emission rate and travel speeds pointed out that traffic congestion in urban areas and environmental consequences of excessive fossil fuel consumption can be combined into one problem. Improving urban transportation operations and preventing traffic congestion yields the reduction of the emission (Boriboonsomsin & Barth 2008).

Traffic congestion causes excess consumption of fossil fuels beside productivity loss and increased emission of poisonous gasses. In Istanbul, the traffic congestion causes the loss of $3.12 billion per year regarding only fuel and time consumption (Ergün & Şahin 2006).

In this study, time dependency factor over travel speed is concerned only as a time cost. However, with small modifications on the objective function to calculate CO_2 emission cost can be added to the model easily, due to a strong correlation between GHG emission rate and travel speeds.

1.2 CONTRIBUTION

MetaOpt framework is an algorithmic framework which provides a decision support mechanism for diversification or intensification decisions in any metaheuristics. This study is the first known implementation of MetaOpt framework to improve the performance of a metaheuristic for a TDVRPTW problem.

Tabu search heuristic for the vehicle routing problem with time windows described by Potvin et al. (1996) is adopted as the core metaheuristic as a part of MetaOpt framework. Extensive tests carried on in order to evaluate framework's performance for every scenario proposed in the related literature.

Since there is not a widely accepted standard problem set for TDVRPTW, in this study Figliozzi's modification of Solomon instances is used to compare performances of algorithms. Figliozzi divides time span into five equal time period and changes travel speeds for each time period to test different scenarios.

2 LITERATURE REVIEW

This literature review summarizes the evolution of VRP from a basic generalization of TSP, with constant travel times and capacity without any other condition, to a more complicated version, based on time dependent structure and time window concept. After brief introduction about diversity in VRP variants, literature survey about exact and heuristic methods is given for TDVRP and VRPTW separately. Their consolidation as Time Dependent Vehicle Routing Problem with Time Windows and other versions of TDVRP is discussed in following topics.

2.1 VEHICLE ROUTING PROBLEM (VRP)

Vehicle Routing Problems (VRP) is introduced by Dantzig and Ramser (1959). They formulated it as a generalization of Traveling-Salesman Problem (TSP) and named the problem as Truck Dispatching Problem. Their aspect of generalization of problem was mainly focused on capacity as an additional condition. But in sake of simplicity of presentation they assumed there is only one type of product is to be delivered and that all vehicles have same capacity (Dantzig & Ramser 1959).

Vehicle routing problems studied extensively in the literature over the years. Since routing and scheduling of a fleet of vehicles to service customers has many different forms in distribution systems, many different models are introduced, vary on the different problem characteristics to model real-life problems.

Various combinations of different transpiration scenarios like time windows, periodic visits and different physical distribution models like pickup and deliveries, heterogeneous vehicle fleet is named and studied with different approaches like exact methods, heuristics, metaheuristics and simulation.

This immense diversity of VRP provides fertile study subjects for researchers. Numerous studies in literature focuses on VRP type combinatorial problems is rapidly increasing due to developments of computer processors that enabled relatively new approaches like tabu search, genetic algorithms to be applied on the VRP. Massive amount of VRP variants and exponentially growing literature created a need of the taxonomy for VRP classifications (Eksioglu et al. 2009).

This study can be classified as combination of two different properties which are hard time window, in terms of scenario characteristics and time dependent in terms of physical characteristics. Studies in literature about these two characteristics are vast and discussed in the following topics.

2.2 TIME DEPENDENT VEHICLE ROUTING PROBLEM (TDVRP)

In urban transportation with a congested environment, it is not accurate to assume that the travel time between two nodes can be obtained by the function of distance and constant travel speed. Changing traffic density causes fluctuations in travel speed that result in fluctuations in travel times (Malandraki & Daskin 1992).

Tough this time dependency concept was discussed before; the first solution approach for time dependent vehicle routing problem was introduced by Malandraki (1989). In this introduction, exact solutions like, cutting plane and branch and bound techniques and numerous simple heuristic algorithms were developed for the TDTSP and TDVRP. Later, Malandraki and Daskin proposed an exact solution (1992). In this study, they proposed a branch-and-cut algorithm and greedy heuristics.

In more realistic sense, travel speed not only depends on time of day but also the area in this particular time. Traffic density may vary in different parts of the city for different period of times. A model with travel speeds that are dependent to both time period and area is proposed by Park (2000). This model aims to minimize the total duration time and the total weighted tardiness and enables to test the clustered formations where travel times between nodes are relatively shorter the for each other than for the rest of the map (Park 2000).

If two vehicles leave the same node to arrive another node, the vehicle that leaves departure node later, should visit the arrival node later as well. This is called ‘‘first-in–first-out’’ (FIFO) property. However, in some cases, since travel speeds over time periods are discreet, a vehicle in a relatively faster time period can arrive a node earlier than a vehicle that is an in relatively slower period even if it leaves the node before.

One of the main turning points of time dependency concept was managing the FIFO property. All of the introduced models were incapable to satisfy the FIFO property due to their calculation methods for travel times crosses two consecutive time periods.

Ichoua et al. (2003) proposed a travel time function that adjusts the speed when the time period changes during a travel, instead of a stepwise travel time function which assume the travel time as a step function of time (Ichoua et al. 2003). This is the first study that satisfies FIFO rule that will be discussed later. This study directed literature to take FIFO rule account.

Fleischmann et al. (2004) used stepwise travel time function in their study. To ensure FIFO feature in model Smoothed Travel Time Function is carried on as a corrective method.

Haghani and Jung (2005) proposed a formulation for the dynamic vehicle routing problem with time-dependent travel times.

2.3 VEHICLE ROUTING PROBLEM WITH TIME WINDOWS (VRPTW)

The VRPTW deals with vehicle routing problems that a fleet of homogenous vehicles should visit customers whose demand is known beforehand, within a specific time range. It takes the minimization of the number of routes or vehicles as the primary objective while minimization of the total travel cost is the secondary objective.

In the literature VRPTW is studied extensively and due to its complexity, many different approaches have been proposed to find the best or near-best solution. The first exact method which is capable of solving large enough problems for real world implementation in the literature is the study of Desrochers et al. (1991). Their method is solving a LPR of the set partitioning formulation of the original problem by column generation in which feasible columns are generated by dynamic programming are added as needed. They solved Solomon instances for benchmarking computational results and succeeded to find optimal solution for 7 problems with 100 customers (Desrochers et al. 1992).

In this study a tabu search heuristic for the vehicle routing problem with time windows described by Potvin et al. (1996) is referenced as core of heuristic part of MetaOpt framework.

2.4 TIME DEPENDENT VEHICLE ROUTING PROBLEM WITH TIME WINDOWS (TDVRPTW)

One of the first vehicle routing problems that considers time window and time dependent travel time is discussed by Ahn and Shin (1991). They proposed the savings, insertion, and local improvement algorithms and considered the travel speed as a step function of time of day. Their non-passing property satisfies FIFO rule.

Though time dependency is always claimed to be applicable to real life problems, studies tested by with a real time problem is really scarce in the literature. Most of the researchers focus on standard problem sets due to the benchmark ability. In fact working with a real world problem with real traffic data requires a quietly different and complicated data structure. Since there are more than one possible path to go from one node to another in real traffic network there is a hidden shortest path problem to feed distance matrices. If we consider a path between two nodes that comprises different road fragments affected differently by traffic conditions then time-dependent travel time calculation concepts must be applied to the traverses of road fragments.

The travel times in depend on the departure times. In the literature the departure time is solved after all the routes are constructed. The vehicle departure time optimization problem (VDO), with time-dependent travel times is proposed by Kok et al. (2010). It was a post-processing solution method for the VDO.

Eglese et al. (2006) on the other hand, focused on a data structure that can adequately represent a natural network with one directional road fragments and junction elements. They proposed a data structure to store information of time-dependent data of the travel times for individual roads in the network to provide a realistic travel times between demand nodes and solved a scenario created with real traffic data with tabu search adopted from Ichoua. (Eglese et al. 2006).

If we assume that a vehicle can wait on any node or have the option to leave the depot at any time from the beginning, then it is critical to manipulate departure times. Because departure times directly affect the how much time will be spend in a particular time period for the vehicle. The main approach in the literature is to find optimal departure times with post-process methods changing departure times for fixed routes after problem is solved, due to practical reasons.

Yet, Dabia et al. not only proposed the only exact solution for TDVRPTW but also optimized departure time optimization problem concurrently with the original problem. They applied branch-and-cut-and price algorithm to minimize total travel time.

2.5 TDVRP EXTENSIONS

Inarguably the core objectives of fleet routing problems were always been economic concerns like shorted lead times and reduced variable costs. They are still the main objectives of vehicle routing problems. Nevertheless, after greenhouse gases have been denounced as a global threat in 1990’s and its emission is regulated by international policymakers to reduce its effects in global warming, GHG has been considered as a constraint along with fuel consumption.

The mutual benefit between environmental and economic concerns created a new problem type, EVRP. Simply it can be defined as the conversion of the time-dependent vehicle routing problem (TDVRP) into Emissions Vehicle Routing Problem (EVRP).

However, first researches about minimization of pollution created by fossil fuels only dating back to the first decade of 2000’s. Palmer has studied the connection between vehicle routing and emission but did not propose a specialized approach.

Figliozzi introduced Emissions Minimization Vehicle Routing Problem (EVRP) in which he derived an algorithm to optimize the departure time between nodes and used travel speed as a decision variable because the amount of emissions is a function of travel speed (2009).

Emissions-based Time-Dependent Vehicle Routing Problem (E-TDVRP) is introduced by Jabali et al. (2012). They modeled the traffic congestion with a two speed function: the congestion speed which is imposed by traffic and free flow speed. They proposed a tabu search procedure as a solution method.

The Time-Dependent Pollution-Routing Problem (TDPRP) is introduced by Franceschetti et al. (2003). The TDPRP aims to minimize the emissions and the driver costs. They benefit the same assumption of two speed function as in Jabali et al. (2012).

There are other extensions for different objective functions where the objective is to minimize total route duration, denoted the duration minimizing TDVRPTW (DM-TDVRPTW) and for different problem structures like real-time time- dependent vehicle routing problem with time windows (RT-TDVRPTW) where demands and time dependent travel times change over time and problem has to be solved dynamically.

3 PROPOSED APPROACH

3.1 TABU SEARCH

Tabu Search is a metaheuristic method proposed by Glover (1986) and the term coined by himself. In fact, some basic properties like short-term memory and avoiding recent moves of tabu search, dating back to early studies of Glover. It is regarded as a metaheuristics, a common guiding strategy, rather than a problem specific heuristic.

Tabu search method can be modified for specific needs of the problem at hand. Because of this, there are many different tabu search procedures for different variants of VRP in literature. This work is based on the modification of Potvin et al. (1996) tabu search procedure that is proposed to solve the Vehicle Routing Problem with Time Windows.

There are two complementary properties of Tabu Search method. The first is moving around in the solution space even if the solution is not improving. The second is not to be caught by a loop of feasible local optima by declaring backward moves as tabu moves and prohibit them to be visited for a limited time. This short-term memory, also called as tabu lists, which ensures not to cycling back to previously visited solutions, is not intended to avoid a repeating move but avoiding a reverse move that cancel a recently found solution. Such moves conflicts with the main strategy of tabu search that is always search undiscovered solution space. This feature insures the current solution not to go back the previous one.

Tabu moves can be recorded as complete solutions in tabu lists. Since this causes greater memory usage and more computation time to check if the move is in tabu list or not, tabu moves are recorded in the form of single changes of the current solution, performed in last few iterations.

The solution space is the subset of feasible solutions of the problem that can be revealed by neighborhood creating processes. Neighborhood is a subset of the solution space that is derived from the current solution by mutating iterations. In this work, local search heuristics are used to create a neighborhood. Local search heuristics are discussed in following topics.

Tabu Search method begins with a basic feasible solution. As solution improves with better outcomes in the solution space, it records the best solution obtained and update it whenever a new best emerges. This information is kept in long term memory in case of a diversification or intensification decision.

Diversification is one of the key properties of tabu search. Its aim is to jump to an undiscovered location in the solution space if the current location does not promise an improvement. This property distinguishes tabu search procedure from a classical local search technique. It enables procedure not to stop at a local optimum and expand solution space where more improvement can be found. The downside of this mechanism is it can skip a fertile search space where a better solution may be found, and go far away from the optimal point that is difficult to found back again. One of most common diversification method is restarting tabu procedure from modified best currently known solution that is not likely to be explored during the same search. Diversification method that is used in this work is discussed in following topics.

Intensification, on the other hand, aims to focus on a location that is considered as fertile neighborhood in the solution space. Procedure stays in promising neighborhood to investigate more locations. Restarting tabu procedure from best currently known solution is common method for intensification. Intensification is a powerful element of tabu search procedure. Yet it is not an urgent move in the search as diversification moves are. While diversification moves try to discover far spots in search space and may yield dramatic improvements, intensification moves try to go deeper in the current solution which its vicinity is explored already.

Termination criterion can be defined in different ways. It can be stopped when solution reach the best-ever value, if it is known beforehand. A number of consecutive iterations without an improvement in solution can be interpreted as the search space is consumed completely, and any improvement is not expected. CPU time of the execution can be limited by a threshold value.

3.2 LOCAL SEARCH HEURISTICS

Defining a proper local search heuristic, which is intended to create a fertile neighborhood to exploit, is the key factor for the success of a tabu search procedure. These heuristics modifies current solution to generate a neighborhood in the search space.

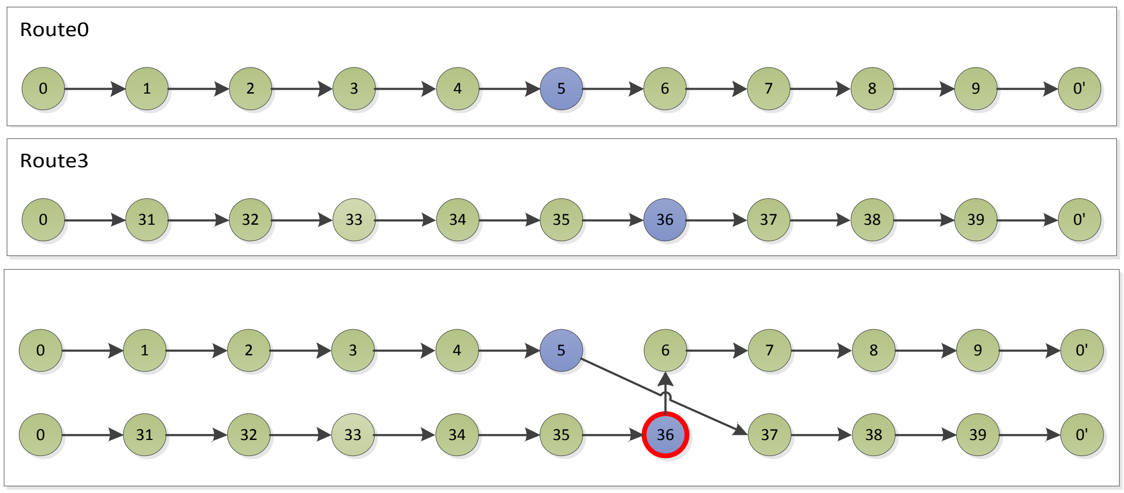
Although there are lots of successful local search heuristics such as k-opt and 2-opt exchanges for VRP type problems, Potvin et al. (1996) prefers and suggests to use 2-opt* and Or-opt exchange heuristics consecutively due to their performance in generating a fertile search space and preserving route orientation property (Potvin & Kervahut 1996). Local search heuristics used in this work are in following topics.

3.2.1 2-Opt* Exchanges

2-opt is an intra-route exchange, first proposed by Croes (1958). It breaks two arcs of a route and reshuffles two nodes that are not connected before to obtain a reordered new route. 2-opt exchange is also referred to as a 2-opt move.

2-opt* is a variant of 2-opt for inter-route exchanges. It exchanges two pairs of arcs between routes. 2-opt* move preserve orientation and introduce last part of the route on another route. Thus, it helps to keep feasibility of the time windows in the route.

Figure 3.1 Inter-route change (2 opt*)



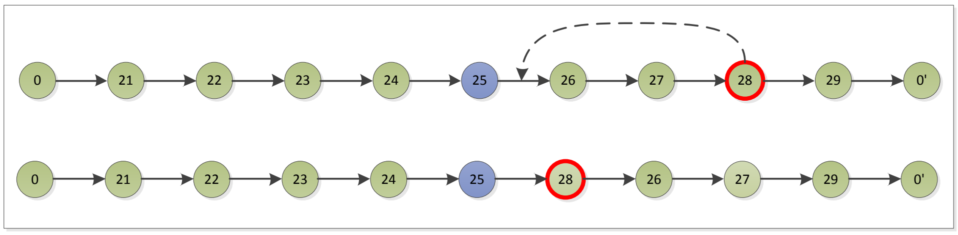
Source: Figure created by author.

Figure 3.1 illustrates a 2-opt* move. Node 36 is one of the h nearest neighbors in distance of node 5. 5, 6 and 36, 37 are replaced by links 5,37 and 36, 6.

3.2.2 Or-Opt Exchanges

Or-opt exchange was first proposed by Or (1976). Or Opt exchange changes the route by reordering sequences of one, two, three consecutive nodes within a route.

Figure 3.2 Intra-route change (Or-opt)



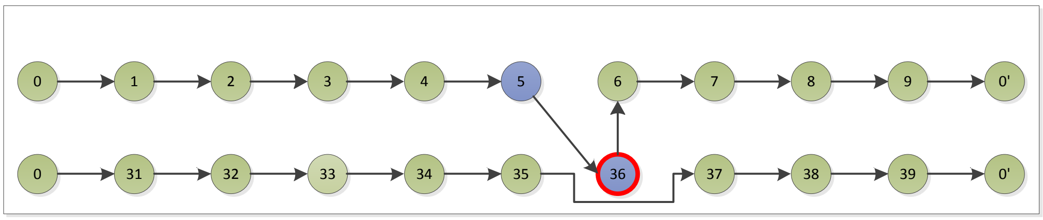
Source: Figure created by author.

Figure 3.2 illustrates an Or-opt move. Node 28 is one of the h nearest neighbors in distance of node 25. Link 25, 26 is broken and replaced by link 25, 28. Node 27 and node 29 are linked after node 28 removed. Yet or opt exchange can be executed for one or more consecutive nodes 27, 28 and 29 could be inserted between 25 and 26. However, in this study or opt exchanges is applied for only one node of the route.

3.2.3 Relocation

Relocation is inter-route exchange that tries to insert a node from one route into another. Relocation is also referred as relocate neighborhood (Savelsbergh 1992). It can be considered as inter-route version of or-opt swap without consecutive nodes. Sample relocation move is shown in Figure 3.3.

Figure 3.3 Inter-route change (Relocate)



Source: Figure created by author.

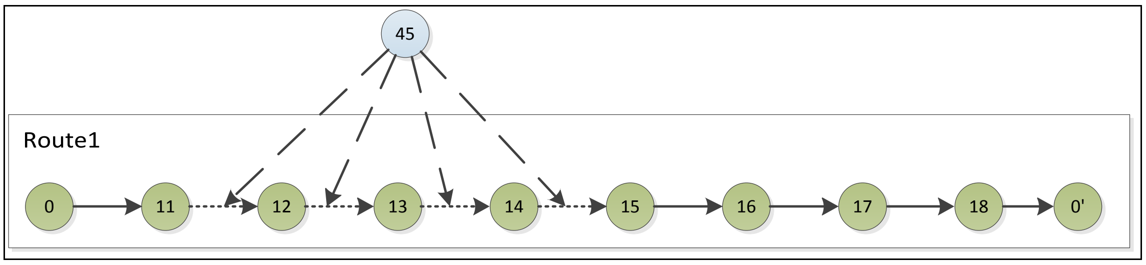
Figure 3.3 illustrates a relocation move. In this case node 36 is one of the h nearest neighbors from a different route in distance of node 5. Link 5, 6 and 36, 37 are replaced by link 5, 36.

3.2.4 Route Saving Reinsertion

Route saving reinsertion is a particular case of the relocation move. It aims to remove nodes of short routes and insert them into another route. It is not a greedy reinsertion procedure which seeks the cheapest insertion. Whenever it succeeds to insert a node in a new route from a short route, it executes the change in the current solution even if it increases the cost of the current solution. Figure 3.4 illustrates route saving reinsertion move.

Main difference between route saving reinsertion and relocation is in the second case every possible swap between a node and its neighbors are tried and recorded in a list. After iteration is done, the swap with the least cost become valid swap and updates the current solution. Since reinsertion procedure focus on consuming short in order to routes reducing total number of routes or vehicles, cost reduction is not considered. So it does not comprise a neighborhood structure and short-term memory to compare the costs of possible moves.

Figure 3.4 Route Saving Reinsertion



Source: Figure created by author.

3.3 METAOPT FRAMEWORK

Heuristics and exacts solvers have useful properties for each other in different ways. For instance, exact solvers can assist heuristic methods as subroutines to explore a neighborhood in a local search procedure. On the other side, heuristics has an essential role for exacts solvers. They provide high-quality primal values to help exact solvers in bounding solution space and preprocessing to reduce calculation time for large scale problems (Joncour et al. 2010).

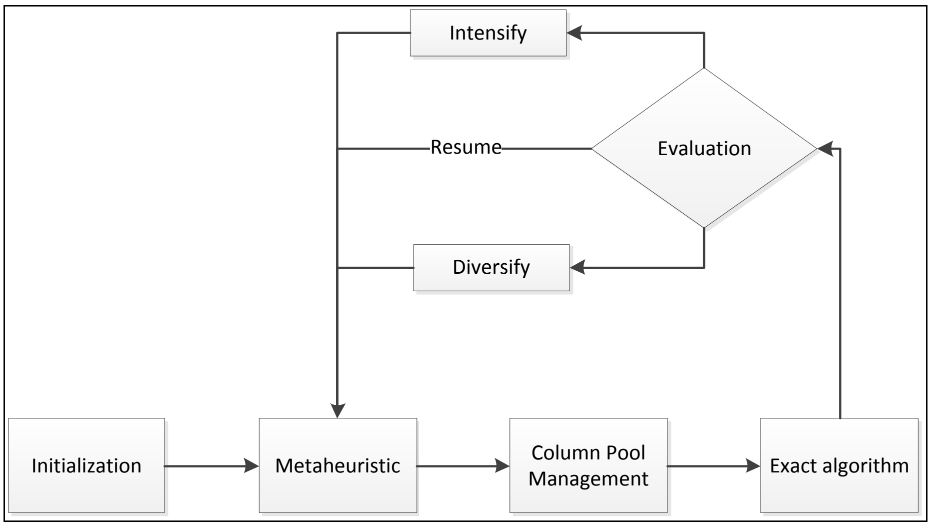
MetaOpt has a distinctive role in heuristics and exact solvers relation. It is a generic framework which provides a decision support mechanism for diversification or intensification decisions in combinatorial optimization problems. It combines metaheuristics that obtain approximate solutions with exact algorithms that can be modeled as a set covering problem (Muter et al. 2010).

3.3.1 The Framework

As it is illustrated in Figure 3.5, MetaOpt framework starts with the initialization step and follows metaheuristic, column pool management, exact algorithm and evaluation steps. More detailed diagram shows inner levels of tabu search in Figure 4.1.

First step of MetaOpt framework is finding an initial solution to provide a search space for the metaheuristic. This step is the only element that is not in the main loop except possible diversification decisions. Main loop comprises of four main elements, starting with metaheuristic. Metaheuristic generates new columns for the exact algorithm until the end of the main loop. Since their quality is questionable, all of the feasible moves and their columns that are produced by local search heuristics are not added to the column pool. Solemnly, the produced columns that are good enough to change the current solution are added to the column pool. Thus quality and manageable size of the solution pool is guaranteed.

Figure 3.5 The MetaOpt framework



Source: Muter et al. (2010) *Combination of Metaheuristic and Exact Algorithms for Solving Set Covering-Type Optimization Problems.* November 2010.

It is efficient and convenient to generate columns with heuristics. One of the renowned properties of metaheuristic algorithms is that they produce feasible solutions with excellent quality for combinatorial optimization problems with small computational burden.

New columns obtained by metaheuristic are handled by column management process. Column management, roughly speaking, sorts out the useless columns that are dominated by the existing columns in order to avoid exploded the number of columns but have an expanding solution space. Exact algorithm solves SCP and chooses the set of tours with minimal cost.

Evaluation step is the essential element of MetaOpt framework. It monitors changes in upper and lower bounds values and interpret the trend to guide metaheuristic. There are three possible outcomes of this step which are, diversify, intensify and resume. Evaluation step completes the loop by informing metaheuristic that way the search will continue.

This loop continues until the termination criterion that can be reaching a given number of iterations, or CPU time is satisfied

3.3.2 The Difference Between Column Generation and MetaOpt Framework

There is an analogy between MetaOpt framework and Column generation method. Column generation method is useful to solve a restricted master problem, where not all columns are in the tableau, in other words; not all variables are allowed to be basic. It benefits reduced cost which represents how much the objective function will change if you make a non-basic variable, a basic variable. New columns are generated by the pricing subproblem as they are needed or metaheuristic algorithms can be used to find a column with negative reduced cost.

Similarly, in MetaOpt framework columns are generated by metaheuristic algorithm and exact method solves restricted set covering problem (RSCP) where not all of the feasible columns are in the column pool.

However, in MetaOpt, the metaheuristic has more functions in the framework, beyond generating columns. Exact method and the metaheuristic algorithm are assigned to solve the original problem cooperatively. The quality of the current solution is evaluated by the comparison of the outcomes of metaheuristic and exact solution to guide the metaheuristic. Also, in MetaOpt, the exact method does not have to have the dual values (Muter et al. 2010).

3.3.3 Set Covering Problem

The set covering problem (SCP) chooses a set of tours with minimal cost subject to the defined constraints. The reformulation of a NP-hard combinatorial problem as an SCP yields to solve the original problem with an exact method. Integer programming formulation of the set covering problem is shown below:

$$\min \sum_{p \in P} c_p y_p \tag{1.1}$$

$$\text{subject to} \quad \sum_{p \in P} a_{ip} y_p \geq 1, \qquad i \in C, \tag{1.2}$$

$$y_p \in \{0, 1\}, \qquad p \in P, \tag{1.3}$$

This notation is reformulation of vehicle routing problem. C is a finite set and P is subset of C, $P \subset C$. The cost of a selected column $p \in P$ is shown by parameter c_p. The constraint (1.2) ensures that every customer is visited at least once. The parameter a_{ip} is equal to 1 if the customer i is in selected route $p \in P$. The decision variable y_p must satisfy the integrality conditions of (1.3).

The restricted problem that have columns in the pool that is corresponding to the decision variable, greatly reduce the computational cost of the problem; compared to the original model that have decision variables for every arc between two nodes and every route that might include them. Finding exact integer solution for SCP in every iteration of the evaluation step may yield a great computational burden. To overcome this drawback, a relaxation of set covering problem (RSCP) is proposed to solve the problem by relaxing the integrality constraints (1.3).

3.3.4 Lower and Upper Bounds

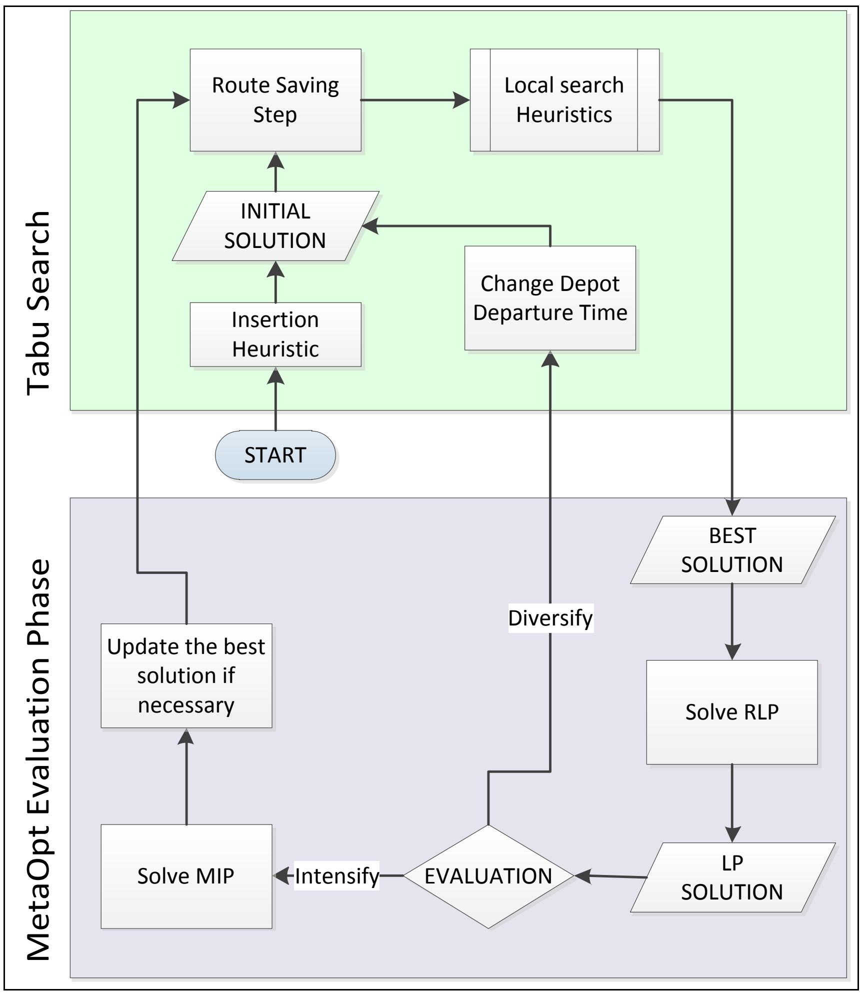
The optimal solution of RSCP at any time provides a lower bound for RSCP. Since restricted problem does not have every possible column produced by metaheuristic, lower bound found by RSCP cannot be referred as lower bound of the original problem. If C presents the all the columns in the pool then P represents the selected columns from the pool, subset of C (1.1).

Nevertheless generated columns are a portion of the solution space revealed by the metaheuristic algorithm. Thus lower bound of the restricted problem presents the solution quality of metaheuristic, and it is referred to as potential lower bound. Generally speaking, the upper bound is the best ever solution of original problem found by the metaheuristic where the potential lower bound is the RLP solution of restricted problem.

4 APPLICATION TO TDVRPTW

TDVRPTW can be described as a vehicle scheduling problem with homogenous vehicles with the same capacity. Each vehicle serves a number of customers with known demands, within a specific time range for each customer. The total demand of customers in a route must be less than or equal to the capacity of the vehicle. Vehicles arrive at the customer before its early time window is reached is not penalized but waits until the early time window. Travel speeds are not constant and depend on the time of the day. Travel time between two nodes varies for different time periods. All routes begin and end at the central depot. The primary objective of TDVRPTW is minimization of the total distance where the secondary objective is minimization of the number of vehicles.

Figure 4.1 Tabu Search MetaOpt Diagram



Source: Figure created by author.

4.1 INITIALIZATION

As Figure 4.1 shows a detailed illustration of MetaOpt framework modified for a tabu search heuristic, the first step of MetaOpt is finding an initial solution. In this study insertion heuristic of Solomon's I1 heuristic (Solomon 1987) is attached to the main framework. Insertion Heuristics is class of sequential, tour-building heuristics which varied in three different criteria.

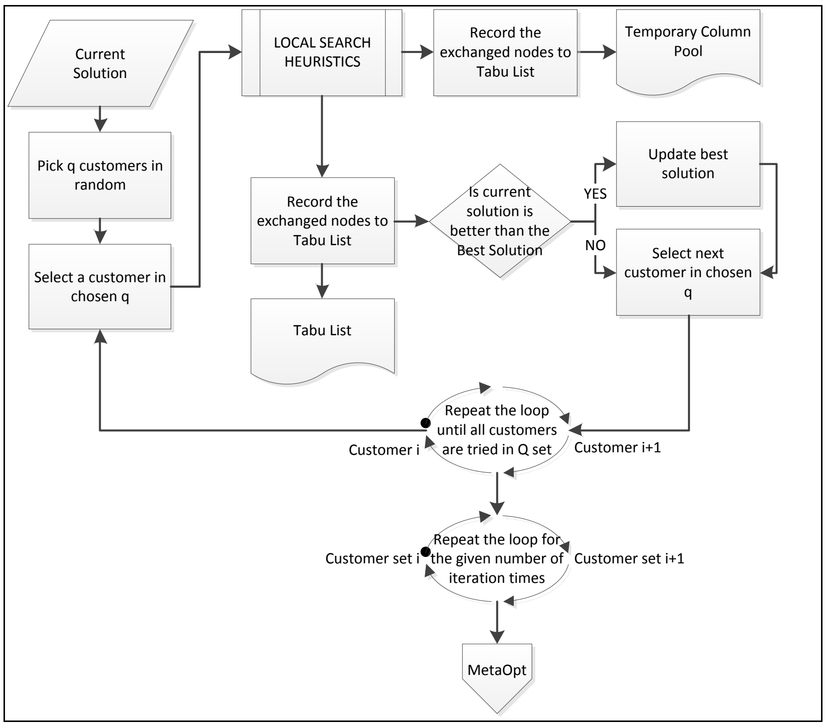
The algorithm starts with a seed solution which is composed of the depot and a customer. In this study, seed solution constructed by inserting the un-routed customer with earliest late time window into a blank route. Then algorithm tries to insert un-routed customers between adjacent nodes (customer or the depot) and calculate cost of this temporary route, if capacity and time window constraints are satisfied. It resembles the relocation move in Figure 3.3 in terms of insertion geometry. At each iteration cost of all of the feasible insertions between every adjacent node in the route is stored, and the node that is in the insertion with minimum cost is awarded to be placed in the route permanently. After a customer is inserted to the route, iterations continues to try out rest of the un-routed customers to insert into the route until there is not a feasible insertion left for the route. Then another seed route is generated, and procedure continues to build new routes sequentially until all of the un-routed nodes are consumed.

4.2 TABU SEARCH

In this study metaheuristic element of MetaOpt, framework is adopted from tabu search algorithm of Potvin et al. (1996). It has been extensively altered in terms of parameter configuration and the mix of local search heuristics.

Tabu Search method begins after the basic feasible solution obtained by Solomon I1 insertion heuristic. Then initial solution is modified by local search heuristics to generate a search space. Local search heuristics used in this study are, 2-opt*, Or Opt, Relocation and Route Saving Reinsertion as introduced above.

Figure 4.2 Tabu Search Diagram


Current Solution
Pick q customers in random
Select a customer in chosen q
LOCAL SEARCH HEURISTICS
Record the exchanged nodes to Tabu List
Temporary Column Pool
Record the exchanged nodes to Tabu List
Tabu List
Is current solution is better than the Best Solution
YES
NO
Update best solution
Select next customer in chosen q
Repeat the loop until all customers are tried in Q set
Customer i
Customer i+1
Repeat the loop for the given number of iteration times
Customer set i
Customer set i+1
MetaOpt

Source: Figure created by author.

The most major difference between this study and study of Potvin et al. (1996) in tabu search methods is, whereas 2-opt* and Or-Opt exchanges are used together by Potvin et al. (1996), relocation and route saving reinsertion moves are not included in their tabu search method. Another major difference is, in Potvin et al. (1996) Or-opt exchanges perform for one, two and three consecutive nodes in the route. In this study, or-opt exchanges for two and three consecutive nodes are not performed. The saving of computational time from this reduction is spend for another local search heuristic, relocation move, which is an inter-route, unlike or-opt exchange.

In Potvin's algorithm, local search heuristics are performed until any improvement in the best-known solution does not occur for the given number of iterations. Since the tabu search is executed several times in MetaOpt framework, the number of iterations are reduced ($I_{tot} = 10,\ I_{2opt} * = 25,\ I_{or-relocate} = 25$) compared with Potvin's algoritm and local search heuristics are performed until a certain number of iteration is executed regardless the improvement trend.

Steps of the modified tabu search heuristic of Potvin et al. (1996) is shown below.

Step 1. Initialization step.

Populate the nearest neighbors list for each customer. The parameter h determines the size of neighbors list. Run Solomon's I1 heuristic to find the initial solution. Set the current and best known solutions to the initial solution.

Repeat Step 2 until I_{tot} iterations are performed.

Step 2. Tabu search step

Step 2.1 Route saving phase

Execute Step 2.1 once before 2-opt* iteration

Find routes with three customers or less in the current solution. Try to remove customers of short routes and insert them into any another route with route saving reinsertion. Update both current and best known solution if a route is consumed or shortened for next iterations.

End Step 2.1.

Step 2.2. Tabu phase with 2-opt*

Repeat Step 2.2 until $I_{2opt} *$ consecutive iterations are performed.

Populate the random customers list with q customers. For each customer in the list execute the 2-opt* exchanges. The best solution is updated to the current solution. Add reverse moves of last five iterations in tabu list. Add the two new routes of best solution in column pool.

End Step 2.2.

Step 2.3 Route saving phase

It is the same as Step 2.1.

End Step 2.3.

Repeat Step 2.4 until $I_{or-relocate}$ consecutive iterations are performed.

Step 2.4. Tabu phase with Or-opt and Relocation

Populate the random customers list with q customers. For each customer in the list, if the the random customer is in the same route with its neighbor execute the or-opt exchanges otherwise execute Relocation exchange. The best solution is updated to the current solution. Add reverse moves of last five iterations in tabu list. Add the one or two new routes of best solution in column pool, regarding which local search heuristic is performed.

End Step 2.4.

End Step 2.

The nearest neighbors list size h, and random customers list size q parameters can be set to different values to obtain different performances. Two different configurations are proposed by Potvin et al. (1996). Each has some advantages and disadvantages in the trade-off between more stable results and more computational time. In this study Setting B ($h = 35$, $q = 15$) is preferred due to its stability. Larger size of nearest neighbors list, h, visits more feasible exchanges for each customer. Random customers list size, q, is reduced from 25 to 15 to avoid increased computational time.

4.3 CALCULATING TRAVEL TIME

Travel time calculation method is an essential element to satisfy FIFO rule, which is mentioned in Chapter 2. In this study, stepwise calculation is adopted exactly as Ichoua et al proposes it. (2003). The procedure is described in below.

```
1.  set $t$ to $t_0$ ,
    set $d$ to $d_{ij}$ ,
    set $t'$ to $t + (\mathrm{d}/\vartheta_{cT_{k+1}})$ ,
2.  while ($t' > \bar{t}_k$) do
    2.1. $d \leftarrow d - \mathrm{d}/\vartheta_{cT_{k+1}}$ ,
    2.2. $t' \leftarrow \bar{t}_k$ ,
    2.3. $t' \leftarrow t + (\mathrm{d}/\vartheta_{cT_{k+1}})$
    2.4. $k \leftarrow k + 1$.
3.  return $(t' - t_0)$
```

Departure time of the vehicle from customer i is t_0 in period T_k which is between $\underline{t}_k$ and $\overline{t}_k$. The parameter d_{ij} is the distance between i and j, t is current time, t' arrival time and ϑ_{cT_k} is the travel speed in time period T_k .

4.4 COLUMN POOL MANAGEMENT

New columns produced by local search heuristics are accumulated in column pool as tabu search iterations continues. Where an inter-route exchange generates two new feasible columns, an intra-route exchange generates one feasible solution. Although all of the generated columns are feasible and expand solution space for RSCP some of them are not allowed to enter the pool. Duplicate columns (routes with same customers in the same order) and dominated columns (routes with same customers in a different order and have higher cost than existing one), are sorted out by column management controls.

Since comparing two different vectors is computationally expensive, an additional control step is executed before original comparison step in order to reduce computational time. This step checks if two vectors have the same size. If comparison fails in this step it skips the next iteration to avoid unnecessary comparison.

Thanks to its selective structure, column pool management ensures that the search space for exact algorithm extends with promising columns as metaheuristic runs. Thus the pool size does not get too large to increase computational time.

4.5 EXACT ALGORITHM

The columns, which are generated by local search heuristics and stored in the pool, are corresponded to the decision variables to build the LP relaxation (LPR) of the set covering problem (SCP). Then, built model is imported to CPLEX environment to be solved by simplex method.

Since integrality constraint is relaxed, most of the time LPR of SCP produces fractional solutions that are infeasible for TDVRPTW. Nevertheless at some iteration, the solution found by exact algorithm can satisfy the integrality condition and other feasibility constraints. In these cases, LP relaxation solution is also optimal for the Restricted SCP. At the same time set covering property may cause the solution have customers in more than one column. This solution is infeasible for TDVRPTW repeated customers can be easily sorted out by a simple method and solution turns into a feasible solution.

This method is initiated whenever LP relaxation solution satisfies integrality condition. In this method, repeated customers are located in the solution and their costs are compared with each other. Route with least cost kept its original order and repeated customer is removed from other routes. Then total travel times of reduced routes are recalculated. The solution time decrease if an over covered customer is reduced but possibly total duration time of the solution may remain unchanged due to time window property.

Since LP relaxation of the set covering is known to provide an excellent primal lower bound (Desrochers et al. 1992), lower bound obtained by LPR is used to interpret the quality of search space that tabu search is in.

$$(Z_{UB}(k) - Z_{PLB}(k))/Z_{PLB}(k) > \alpha \qquad (2.26)$$

The gap between the upper bound and the potential lower bound shows that the columns in the pool possibly improve the solution of original problem. Therefore when the gap between these two bounds reached a critical level (α), Integer Programming is solved instead of RLP. Warm up solution does not have to be a feasible solution of original problem, the initial solution of IP is set to the best tabu solution to provide a warm start to IP.

To ensure that exact algorithm does not find a solution with more vehicles than tabu search has already found, an additional constraint is added the exact model. The number of vehicles in the exact solution cannot exceed the number of vehicles in the current tabu solution. Otherwise, when exact algorithm found a better solution with more vehicles it cancels the outcomes of route saving phase. However after the diversification steps, where the number of vehicles is expected to increase in new solution, this constraint is updated to new limit. This may cause to lose fewer number of vehicles previously found.

4.6 EVALUATION

Detailed steps of evaluation of upper and lower bounds is given with a pseudo-code below $Z_{UB}(k)$ is upper bound value obtained by metaheuristic at iteration k, whereas $Z_{PLB}(k)$ is potential lower bound obtained by the RSCP. Parameters r_1 and r_2 are incremental variables that monitors upper and potential lower bounds. If a significant change has not occurred in both upper and potential lower bound, r_1 increases. If upper bound remains same but potential lower bound reduces, r_2 increases. The limit values of these counters determine the timing of intensification or diversification decisions and shown as $\bar{r}_1$ and $\bar{r}_2$.

```
**Input:** $Z_{UB}(k), Z_{PLB}(k)$                                   (2.1)
**if** Improve $(Z_{UB}(k), Z_{UB}(k-1))$ **then**                    (2.2)
    $r_1$= 0, $r_2$= 0                                               (2.3)
    **if** solution is feasible and $Z_{PLB}(k) < Z_{UB}(k)$          (2.4)
    **then**
        Update()                                                     (2.5)
    **end**                                                          (2.6)
**else if** Improve $(Z_{PLB}(k), Z_{PLB}(k) - 1)$ **then**           (2.7)
    $r_1$ = 0                                                        (2.8)
    **if** solution is feasible **then**                             (2.9)
    Update()                                                         (2.10)
    **else**                                                         (2.11)
```

```
                $r_2 \leftarrow r_2 + 1$                                (2.12)
                **if** $r_2 = \bar{r}_2$ **then**                       (2.13)
                Intensify()                                             (2.14)
                $r_2 = 0$                                               (2.15)
                **end**                                                 (2.16)
        **end**                                                         (2.17)
**else**                                                                (2.18)
        $r_1 \leftarrow r_1 + 1$, $r_2 = 0$                             (2.19)
        **if** r1 = $\bar{r}_1$ **then**                                (2.20)
                Diversify()                                             (2.21)
                $r_1 = 0$                                               (2.22)
        **end**                                                         (2.23)
**end**                                                                 (2.24)
$k \leftarrow k$ +1                                                     (2.25)
```

If upper bound, the solution of metaheuristic, improves then both r_1 and r_2 is reset to zero and metaheuristic algoritm resumes its process (2.1), (2.3). If the solution of RSCP is better than the solution of metaheuristic and satisfies the integrality condition then upper bound is updated with potential lower bound (2.4), (2.6).

If upper bound has not improved, but potential lower bound has, then and r_1 is reset to zero. At this point if RSCP solution is integer then upper bound is updated as in (2.5). Otherwise r_2 is increased by one and checked if it has exceed the limit $\bar{r}_2$ (2.13). If it reaches then metaheuristic is directed for intensification procedure and and r_2 is reset to zero (2.15).

If there is not an improvement in both RSCP and metaheuristic solution then r_1 is increased with 1, r_2 is reset to zero (2.19), (2.22) and if $r_1$exceed the limit $\bar{r}_1$ (2.20) metaheuristic is directed for diversification procedure (2.21).

Improvement in potential lower bound in consecutive iterations shows the quality of the columns added at current iteration. It can be interpreted as the metaheuristic algorithm is in a promising search space but upper bound is not improving. Thus when r_2 reached $\bar{\mathrm{r}}_2$ intensification procedure is called to exploit the vicinity of current search space.

Improvement in upper bound in consecutive iterations means that the portion of the current solution space is promising and metaheuristic should continue to search the vicinity of current search space for further improvements.

If the upper bound and the potential lower bound have not changed for $\bar{\mathrm{r}}_1$ consecutive iterations intensification procedure is called to explore different points in the solution space and add more varied columns.

As it is mentioned above, the evaluation step is the key element of MetaOpt framework. It monitors improvements in the upper-bound and the potential lower-bound values and interpret outcomes to give decisions of diversification, intensification and resuming tabu search with the current solution.

Intensification is triggered in three cases. The first is when upper bound does not improve while potential lower bound does for certain times (2.14). Whenever this condition is satisfied the current solution is updated to the best solution, and the tabu search continues from the new solution.

$$Z_{UB}(k) \leftarrow Z_{PLB}(k), \tag{2.27}$$

$$Z_{current}(k) \leftarrow Z_{PLB}(k), \tag{2.28}$$

The second occurs when the gap between these two bounds reaches a critical level and problem is solved as IP. Since IP solution provides a feasible solution for the original solution, upper bound is updated to IP solution. In the worst case the solution of IP is equal to the current upper bound. In this case the tabu search continues unaffectedly.

In the third case intensification occurs without screening upper and potential lower bound. Whenever LP solution satisfy the integrality condition and is better than upper bound, upper bound is updated to current lower bound.

4.7 DIVERSIFICATION

The main purpose of diversification is to reveal an undiscovered location, away from the not promising solution space. Introducing a diverse solution which is unlikely to be found in current search space and restarting the search from this solution is a common method for diversification strategy.

In this study, diversification method combines its original aim with optimal departure time problem. Departure times are optimized after routes have been constructed with post-process methods. On the other hand, considering departure times as decision variables is computationally expensive to obtain an exact solution. Though, Dabia et al. (2013) propose a branch-and-cut-and price algorithm to determine the departure times for each customer. However, proposed exact algorithm is capable to solve 38% of the instances with 50 customers and 15% of the instances with 100 customers. (Dabia et al. 2013).

It is hard to turn back to a feasible solution from an infeasible one for tabu search of TDVRPTW, due to time window constraints. Because of this, in tabu search method feasibility of the current solution is kept at any iteration and under any circumstances. Since the change in departure time of one customer affect succeeding customers, searching optimal departure times for every customer without violating feasibility requires intense search for local search heuristics. Besides, waiting at customer for a faster time period is not a practical decision for real life problems but departure time from the depot can be determined.

Thus, in this solution, departure times of customers are not intervened directly. Instead, departure time of vehicles from depot is altered to shift entire route to find better solutions. Intuitively traveling long distances in faster periods and short distances in slower periods decrease total travel time, since capacity constraint evens the number of customers in each route most of the time.

A general notation about time windows is given below for further expressions.

Let dt_{ik} is the departure time of vehicle k from customer i, e_i is the early time window of customer i, l_j is the late time window of customer j, wt_{ik} is the waiting time if arrival of vehicle k to customer i (a_{ik}) is smaller than and t_{ij} is the travel time between i and j.

Time window constraints can be defined as below:

$$dt_{ik} + t_{ij} \leq l_j \tag{3.1}$$

$$wt_{ik} = e_i - a_{ik}, \qquad e_i \geq a_i \tag{3.2}$$

Diversification is performed with Solomon's I1 heuristic. As mentioned before Solomon's I1 heuristic constructs tours sequentially and starts with a seed solution. In diversification step customer with earliest late time window is used to construct seed routes as it is preferred in the initialization step. The shift in departure time of customer i is denoted by θ_i. It is a uniformly distributed random variable, between zero and the late time window of selected customer (3.8). In Solomon instances departure time at the depot for all vehicles is zero ($dt_{0k} = 0$).

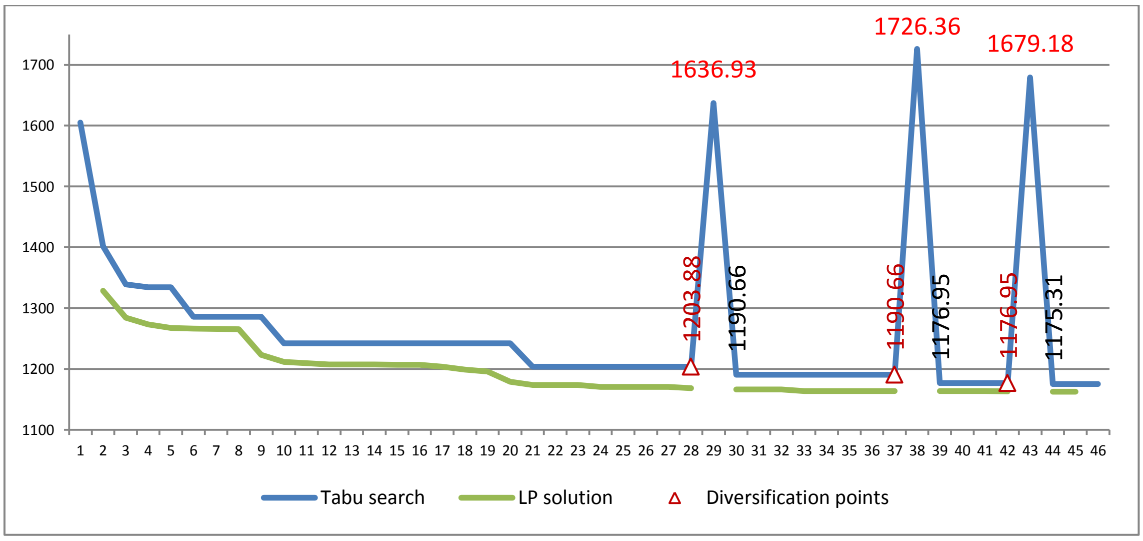
$$\theta_0 \sim uniform(0, l_i) \tag{3.3}$$

$$0 \leq dt_{0k} + \theta_0 \leq l_i \tag{3.4}$$

Slackness of each customer (difference between the arrival time and early time window) could be analyzed for descriptive statistics and mean and standard deviation of this slackness array could be used to generate a random variable. However, in computational experiments it is observed that adding random variables with these parameters can cause infeasible start times for some customers. Instead, using late time window of customers with earliest late time as an upper limit for random variable ensures that all of the customers will be inserted in the initial solution and prevent this drawback.

Insertion heuristic starts with a seed solution, a start time, or waiting time, is added to departure time of vehicle k from the depot ($i = 0$) (3.9). Nevertheless, guaranteeing every customer to insert initial solution may cause increasing number of vehicles due to decline in the number of possible insertion for successive customers. But this outcome is welcomed, since finding a worse solution is acceptable as long as it provides diversity in solution regarding diversification strategy.

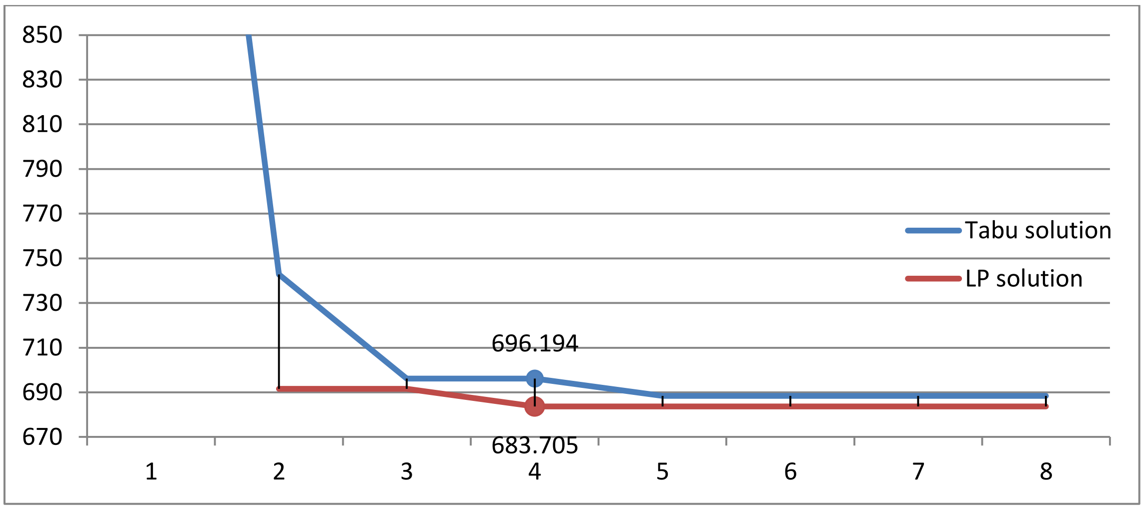
Figure 4.3 Improvements in the Upper Bound and the Potential Lower Bound



Source: Figure created by author from data obtained in computational experiments.

As it is shown in Figure 4.3 diversification dramatically increase the value of objective function. Then, one iteration later, tabu search induce a sudden drop in the solution value, back to the previous level of upper bound, even lower. After every diversification step, IP is solved to reduce the upper bound. In Figure 4.5 tabu search line represents the tabu best solution value, except diversification steps. It is showed to emphasize increased tabu search value.

Figure 4.4 Gap Between Upper and Lower Bound



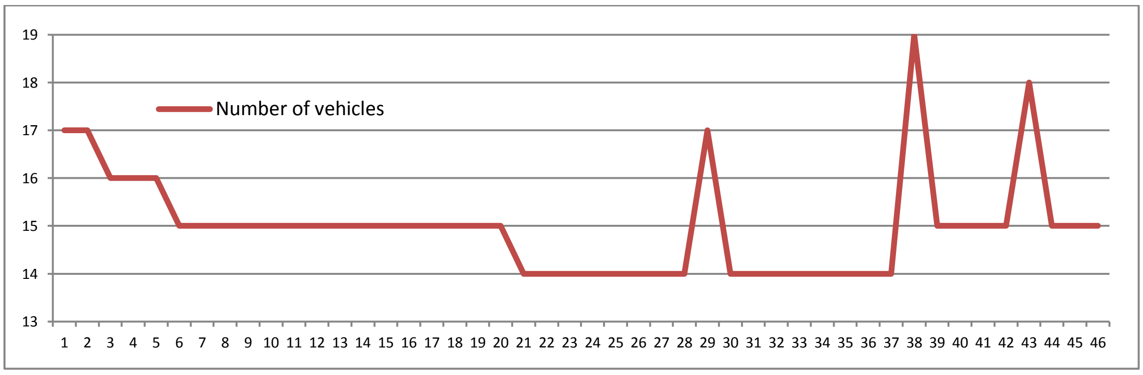
Source: Figure created by author from data obtained in computational experiments.

In Figure 4.4 as the gap between the upper bound and the potential lower bound exceed the critical level (α) the intensification condition (2.26) is triggered immediately. Consequently IP is solved and reduced the upper bound.

Termination condition is modified not to stop the algorithm if a diversification occurs in the last iteration for certain times more tabu search to exploit the possible improvement with this diversification. To prevent an end to end diversification loop it is limited for three times, equal with parameter $\bar{r}_1$, triggers diversification procedure.

Since the pool has the same columns beside new columns added after diversification, the LP solution is not affected by diversification instantaneously. At this point, solving LPR of SCP and column pool management has a regulator role. If a route with the initial departure value is disadvantageous then, it is sorted out by column pool management. Also, integer-feasible solutions (see. Appendix A.2.). of LP solution and IP solution overlook these low-quality columns in intensification moves. Besides, diversified columns with good quality are promoted when upper bound is updated with potential lower bound.

Figure 4.5 The number of vehicles at iterations



Source: Figure created by author from data obtained in computational experiments.

It is can be seen in Figure 4.5 that when the first diversification occurs total number of vehicles increase abruptly then tabu search manages to reduce it back to its previous number. Nevertheless after second diversification total number of vehicles lost its best value found at previous iterations.

The secondary objective of this problem is the minimization of the total number of vehicles. While tabu search minimizes the total travel time, in route saving phases total number of vehicles are reduced in expense of increasing total travel time. It can be recalled that route saving reinsertion remove customers in short routes and insert them into another route regardless of cost of the new route. In computational experiments, it is observed that the total travel time increases as the total number of vehicles decrease (see. Appendix A.1.). The trade-off between total travel time and the total number of vehicles is discussed in computational experiments.

5 COMPUTATIONAL EXPERIMENTS

Solomon problem set is used to test the algorithm as a benchmarking problem in the literature. Solomon instances are proposed to benchmark VRPTW problems. It comprises of six sets of problems. Each set emphasize different elements of the problem to evaluate the capabilities of algorithms which are distribution pattern of nodes is coordination plane, the capacity of a vehicle and constriction and placement of the time windows.

Since there is not a widely accepted standard problem set for TDVRPTW, in this study Figliozzi's modification of Solomon instances will be used to compare performances of algorithms. Figliozzi diverses time span into five equal time period and changes travel speeds for each time period to test different scenarios.

Travel speed distributions vary for different scenarios. To test his algorithm Figliozzi (2012) introduced four distinctive speed distribution pattern with different characteristics to emphasize different real life conditions and generated three distributions with different amplitudes for each of them (Figliozzi 2012).

All 12 different scenarios have been tested and compared with Figliozzi's results for different objectives.

All individual problems in a problem set are limited to 12 minutes of CPU time. However since diversification at last iteration prolongs the run, the runtime for all 56 problems average 12.86 minutes.

Table 5.1 Comparison of IRCI and MetaOpt Tabu Search for results type (a)

Travel time distribution	R1		R2		C1		C2		RC1		RC2	
	IRCI	MetaOpt Tabu	IRCI	MetaOpt Tabu	IRCI	MetaOpt Tabu	IRCI	MetaOpt Tabu	IRCI	MetaOpt Tabu	IRCI	MetaOpt Tabu
Average number of vehicles												
(1) TD1	11.67	13.0	2.82	3.8	10	10.0	3	3.5	11.38	12.9	3.25	3.9
(2) TD2	10.75	12.2	2.55	3.7	10	10.0	3	3.8	10.5	11.9	2.88	3.8
(3) TD3	9.92	11.7	2.27	3.5	10	10.0	3	3.5	10	11.3	2.75	3.9
Average distance												
(1) TD1	1295	1239	1216	973	879	842	657	677	1405	1397	1444	1329
(2) TD2	1258	1215	1244	972	864	841	654	702	1395	1399	1454	1399
(3) TD3	1237	1196	1269	983	880	860	697	682	1362	1376	1434	1141
Average travel time												
(1) TD1	1080	991	990	737	729	683	563	568	1164	1109	1177	1018
(2) TD2	897	802	861	593	644	610	495	511	989	914	993	893
(3) TD3	793	652	774	511	608	566	485	455	860	773	867	629

Source: Figure created by author from data obtained in computational experiments and Figliozzi (2012).

Table 5.2 Comparison of IRCI and MetaOpt Tabu Search for results type (b)

Travel time distribution	R1		R2		C1		C2		RC1		RC2	
	IRCI	MetaOpt Tabu	IRCI	MetaOpt Tabu	IRCI	MetaOpt Tabu	IRCI	MetaOpt Tabu	IRCI	MetaOpt Tabu	IRCI	MetaOpt Tabu
Average number of vehicles												
(1) TD1	12.42	13.7	3	3.5	10	10.0	3	3.8	12.13	13.6	3.38	4.5
(2) TD2	11.5	13.3	2.73	3.5	10	10.0	3	3.6	11.25	12.8	3.25	3.9
(3) TD3	11.42	12.4	2.73	3.4	10	10.0	3	3.6	11	12.5	3	4.0
Average distance												
(1) TD1	1289	1314	1212	1397	892	841	670	709	1454	1545	1403	1101
(2) TD2	1279	1219	1218	998	883	849	667	695	1429	1434	1433	1368
(3) TD3	1265	1209	1245	1040	866	864	714	709	1442	1453	1483	1386
Average travel time												
(1) TD1	1064	1068	1027	1183	732	667	545	572	1180	1243	1200	912
(2) TD2	905	826	893	667	650	582	467	476	1010	974	1053	943
(3) TD3	808	720	831	583	584	529	446	428	916	863	981	841

Source: Figure created by author from data obtained in computational experiments and Figliozzi (2012).

Table 5.3 Comparison of IRCI and MetaOpt Tabu Search for results type (c)

Travel time distribution	R1		R2		C1		C2		RC1		RC2	
	IRCI	MetaOpt Tabu	IRCI	MetaOpt Tabu	IRCI	MetaOpt Tabu	IRCI	MetaOpt Tabu	IRCI	MetaOpt Tabu	IRCI	MetaOpt Tabu
Average number of vehicles												
(1) TD4	11.67	12.8	2.73	3.8	10	10.0	3	3.9	11.5	12.6	3.25	3.9
(2) TD5	10.83	12.8	2.55	3.4	10	10.0	3	3.6	10.75	11.6	2.75	3.6
(3) TD6	10.17	11.8	2.36	3.2	10	10.0	3	3.8	10.13	11.8	2.75	3.5
Average distance												
(1) TD4	1302	1212	1245	961	865	832	683	675	1435	1416	1407	1365
(2) TD5	1266	1348	1238	1277	863	846	658	702	1413	1305	1472	1188
(3) TD6	1272	1214	1243	950	862	844	665	657	1409	1350	1438	1165
Average travel time												
(1) TD4	1066	966	1003	738	697	669	573	551	1186	1138	1147	1075
(2) TD5	881	896	843	813	618	595	483	493	1012	887	1027	791
(3) TD6	801	702	760	537	565	538	451	411	904	810	886	694

Source: Figure created by author from data obtained in computational experiments and Figliozzi (2012).

Table 5.4 Comparison of IRCI and MetaOpt Tabu Search for results type (d)

Travel time distribution	R1		R2		C1		C2		RC1		RC2	
	IRCI	MetaOpt Tabu	IRCI	MetaOpt Tabu	IRCI	MetaOpt Tabu	IRCI	MetaOpt Tabu	IRCI	MetaOpt Tabu	IRCI	MetaOpt Tabu
Average number of vehicles												
(1) TD4	12.25	13.6	3	3.8	10	10.0	3	3.6	12	13.1	3.38	4.1
(2) TD5	11.58	13.5	2.73	3.6	10	10.0	3	3.8	11.25	12.8	3.25	4.1
(3) TD6	11.08	13.3	2.64	3.5	10	10.1	3	3.9	10.75	12.5	3.25	4.0
Average distance												
(1) TD4	1311	1275	1218	957	872	834	666	685	1425	1407	1394	1271
(2) TD5	1272	1334	1216	1124	856	838	679	761	1404	1429	1412	1365
(3) TD6	1293	1330	1215	1318	867	835	690	733	1436	1515	1424	1296
Average travel time												
(1) TD4	1114	1079	1045	823	731	699	552	566	1192	1172	1192	1081
(2) TD5	943	984	915	831	652	632	494	545	1035	1039	1053	982
(3) TD6	871	899	846	859	612	588	461	473	964	999	975	852

Source: Figure created by author from data obtained in computational experiments and Figliozzi (2012).

Table 5.5 Percentage Comparison of IRCI and MetaOpt Tabu Search

	TDVRPTW results type (a).						TDVRPTW results type (b).					
Travel time distribution	**R1**	**R2**	**C1**	**C2**	**RC1**	**RC2**	**R1**	**R2**	**C1**	**C2**	**RC1**	**RC2**
	Gap	Gap	Gap	Gap	Gap	Gap	Gap	Gap	Gap	Gap	Gap	Gap
Average number of vehicles												
(1) TD1	-11%	-35%	0%	-17%	-13%	-19%	-10%	-18%	0%	-25%	-12%	-33%
(2) TD2	-13%	-46%	0%	-25%	-13%	-30%	-15%	-27%	0%	-21%	-13%	-19%
(3) TD3	-18%	-52%	0%	-17%	-13%	-41%	-9%	-23%	0%	-21%	-14%	-33%
Average distance												
(1) TD1	4%	20%	4%	-3%	1%	8%	-2%	-15%	6%	-6%	-6%	22%
(2) TD2	3%	22%	3%	-7%	0%	4%	5%	18%	4%	-4%	0%	5%
(3) TD3	3%	23%	2%	2%	-1%	20%	4%	16%	0%	1%	-1%	7%
Average travel time												
(1) TD1	8%	26%	6%	-1%	5%	13%	0%	-15%	9%	-5%	-5%	24%
(2) TD2	11%	31%	5%	-3%	8%	10%	9%	25%	10%	-2%	4%	10%
(3) TD3	18%	34%	7%	6%	10%	27%	11%	30%	9%	4%	6%	14%

	TDVRPTW results type (c).						TDVRPTW results type (d).					
Travel time distribution	**R1**	**R2**	**C1**	**C2**	**RC1**	**RC2**	**R1**	**R2**	**C1**	**C2**	**RC1**	**RC2**
	Gap	Gap	Gap	Gap	Gap	Gap	Gap	Gap	Gap	Gap	Gap	Gap
Average number of vehicles												
(1) TD4	-10%	-40%	0%	-29%	-10%	-19%	-11%	-27%	0%	-21%	-9%	-22%
(2) TD5	-18%	-32%	0%	-21%	-8%	-32%	-17%	-33%	0%	-25%	-13%	-27%
(3) TD6	-16%	-35%	0%	-25%	-16%	-27%	-20%	-34%	-1%	-29%	-16%	-23%
Average distance												
(1) TD4	7%	23%	4%	1%	1%	3%	3%	21%	4%	-3%	1%	9%
(2) TD5	-7%	-3%	2%	-7%	8%	19%	-5%	8%	2%	-12%	-2%	3%
(3) TD6	5%	24%	2%	1%	4%	19%	-3%	-9%	4%	-6%	-5%	9%
Average travel time												
(1) TD4	9%	26%	4%	4%	4%	6%	3%	21%	4%	-3%	2%	9%
(2) TD5	-2%	4%	4%	-2%	12%	23%	-4%	9%	3%	-10%	0%	7%
(3) TD6	12%	29%	5%	9%	10%	22%	-3%	-2%	4%	-3%	-4%	13%

Source: Figure created by author from data obtained in computational experiments and Figliozzi (2012).

As opposed to MetaOpt with Tabu Search, the primary objective of Figliozzi's the Iterative Route Construction and Improvement approach (IRCI) is the minimization of the fleet size. Moreover, the secondary objective is minimization of the total distance (Figliozzi 2012). Thus, performances of these two approaches for two different objectives should be interpreted accordingly (Table 5.6 to Table 5.4). Difficulty levels of problem sets are different from each other. Naturally the gap between performance scores, grows for more difficult solution space and manifest strong and weak parts of approaches.

In computational experiments, columns with different-than-zero departure time are encountered (see. Appendix A.3.). This finding can be interpreted as altering departure time in the diversification step can produce advantageous columns.

In gap analysis table (Table 5.7), referenced values are set to Figliozzi's results. In other words, percentages show how much the proposed approach improved IRCI approach. Negative values mean that the proposed approach underperformed IRCI. It can be seen that there is a strict trade-off between total vehicle number and total travel time. Since C1 set problems are is close the optimality results are close to each other with a slight difference in favor of the proposed approach. In C2 set, vehicles have more capacity, and time span is wider compared the C1 set. As a result of this, C2 set produce longer routes with fewer vehicles. It also has greater solution space which makes it harder to find a near optimal solution. IRCI approach has better performance for C2 set for both objectives. Since nodes in C sets have a clustered form in the two-dimensional plane, it can be said that IRCI approach is capable of exploiting clustered forms.

On the other hand, in R and RC sets which are scattered randomly in the plane, the proposed approach shows better results for total travel time especially for R2 and RC2 sets. The positive response of the proposed approach to the random problem geometry relatively clustered geometry can interpret as arbitrary nature of tabu search is more capable of searching solution space.

The trade-off between fleet size and total travel time, or in other words, fixed costs and variable cost can be advantageous or disadvantageous for different scenarios. It is impossible to determine that which objective is exactly more advantageous for reducing total cost, unless certain figures about transportation scenarios are known. However, either way the proposed approach is successful to reduce fleet size as a secondary objective.

6 CONCLUSIONS AND FUTURE RESEARCH

This study is the first known implementation of MetaOpt framework on a metaheuristic for a TDVRPTW problem. MetaOpt framework is an algorithmic framework which provides a decision support mechanism for diversification or intensification decisions in any metaheuristics. A tabu search procedure in the literature is slightly modified, and a new diversification approach is introduced to search solution space for routes starts with different departure times. Extensive tests have been carried out, and results are compared with another solution approach for TDVRPTW.

MetaOpt framework is implemented to TDVRPTW successfully. The results show that the proposed approach is capable of exploiting the search space where departure time in the depot is different-than-zero. Compared the benchmarking data, better solutions are found in terms of total travel time in expense of total vehicle number.

The proposed approach can be modified with different local search heuristics to respond different problem structures, like asymmetric and site dependent travel times which are more realistic for urban transportation.

Since the amount of carbon emission and fuel consumption of a motor vehicle is a function of travel speed, the proposed approach can be modified to minimize the environmental effects of fossil fuels.

Data structure of the problem can be modified time-dependent data of the travel times for individual roads in the network to provide a realistic travel times between demand nodes to represent a more realistic traffic environment.

REFERENCES

Ballou, R., 1989. Heuristics: rules of thumb for logistics decision making. *Journal of Business Logistics*, 10 (1) pp.112-132.

Boriboonsomsin, B. & Barth, M., 2008. Real-World CO 2 Impacts of Traffic Congestion. , (951), pp.1–23.

Dabia, S. et al., 2013. Branch and Price for the Time-Dependent Vehicle Routing Problem with Time Windows. *Transportation Science,* 47(3), pp.380–396.

Dantzig, G. & Ramser, J., 1959. The truck dispatching problem. *Management science*, 6(1), pp.80–91.

Desrochers, M., Desrosiers, J. & Solomon, M., 1992. A new optimization algorithm for the vehicle routing problem with time windows. *Operations Research,* 40(2), p.13.

Eglese, R., Maden, W. & Slater, A., 2006. A Road Timetable to aid vehicle routing and scheduling. *Computers & Operations Research*, 33(12), pp.3508–3519.

Eksioglu, B., Vural, A.V. & Reisman, A., 2009. The vehicle routing problem: A taxonomic review. *Computers & Industrial Engineering*, 57(4), pp.1472–1483.

Ergün, G. & Şahin, N., 2006. Development of Traffic Congestion Management Strategies: Strategic Plan Studies.

Figliozzi, M.A., 2012. The time dependent vehicle routing problem with time windows: Benchmark problems, an efficient solution algorithm, and solution characteristics. *Transportation Research Part E: Logistics and Transportation Review*, 48(3), pp.616–636.

Huschebeck, M., 2004. BEST Urban freight solutions. *Deliverable D1*, (February), pp.1–27.

Ichoua, S., Gendreau, M. & Potvin, J.Y., 2003. Vehicle dispatching with time-dependent travel times. *European Journal of Operational Research*, 144(2), pp.379–396.

Joncour, C. et al., 2010. Column Generation based Primal Heuristics. *Electronic Notes in Discrete Mathematics*, 36, pp.695–702.

Malandraki, C. & Daskin, M.S., 1992. Time Dependent Vehicle Routing Problems: Formulations, Properties and Heuristic Algorithms. *Transportation Science*, 26(3), pp.185–200.

Muter, I., Birbil, S.I. & Sahin, G., 2010. Combination of Metaheuristic and Exact Algorithms for Solving Set Covering-Type Optimization Problems. *INFORMS Journal on Computing*, 22(4), pp.603–619.

Park, Y.-B., 2000. A solution of the bicriteria vehicle scheduling problems with time and area-dependent travel speeds. *Computers & Industrial Engineering*, 38(1), pp.173–187.

Persson, G., 1991. Achieving competitiveness through logistics. *International Journal of Logistics Management* 2(1), pp.1-11.

Potvin, J. & Kervahut, T., 1996. The Vehicle Routing Problem with Time Windows Part I: Tabu Search. *INFORMS Journal on Computing* 8(2), pp.158-164.

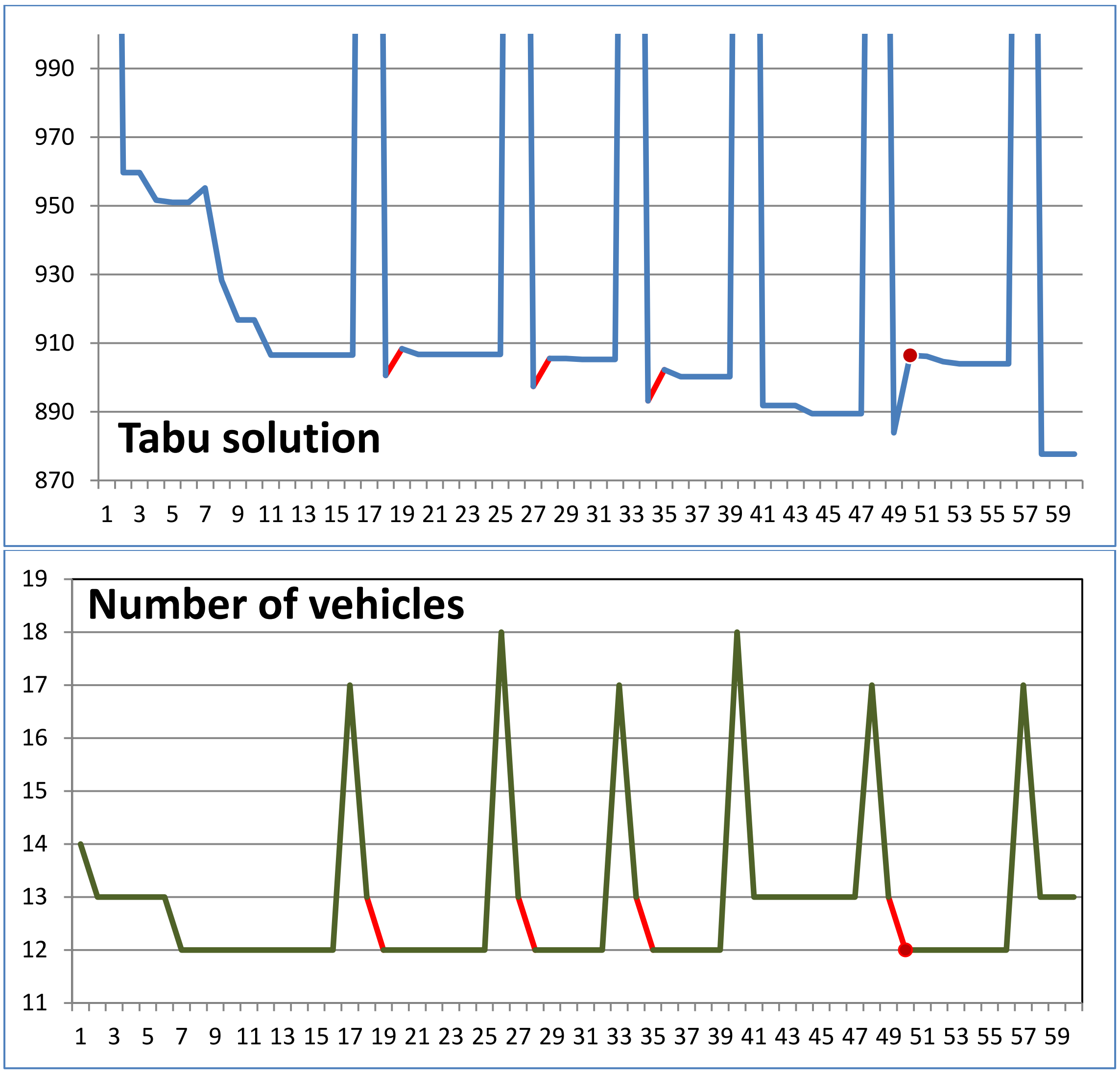
Savelsbergh, M., 1992. The vehicle routing problem with time windows: Minimizing route duration. *ORSA journal on computing*, 4(2), pp.146-154.

Solomon, M.M., 1987. Algorithms for the Vehicle Routing and Scheduling Problems with Time Window Constraints. *Operations Research*, 35, pp.254–265.

World Health Organization, 2009. Urban population growth. *Global Health Observatory (GHO).*

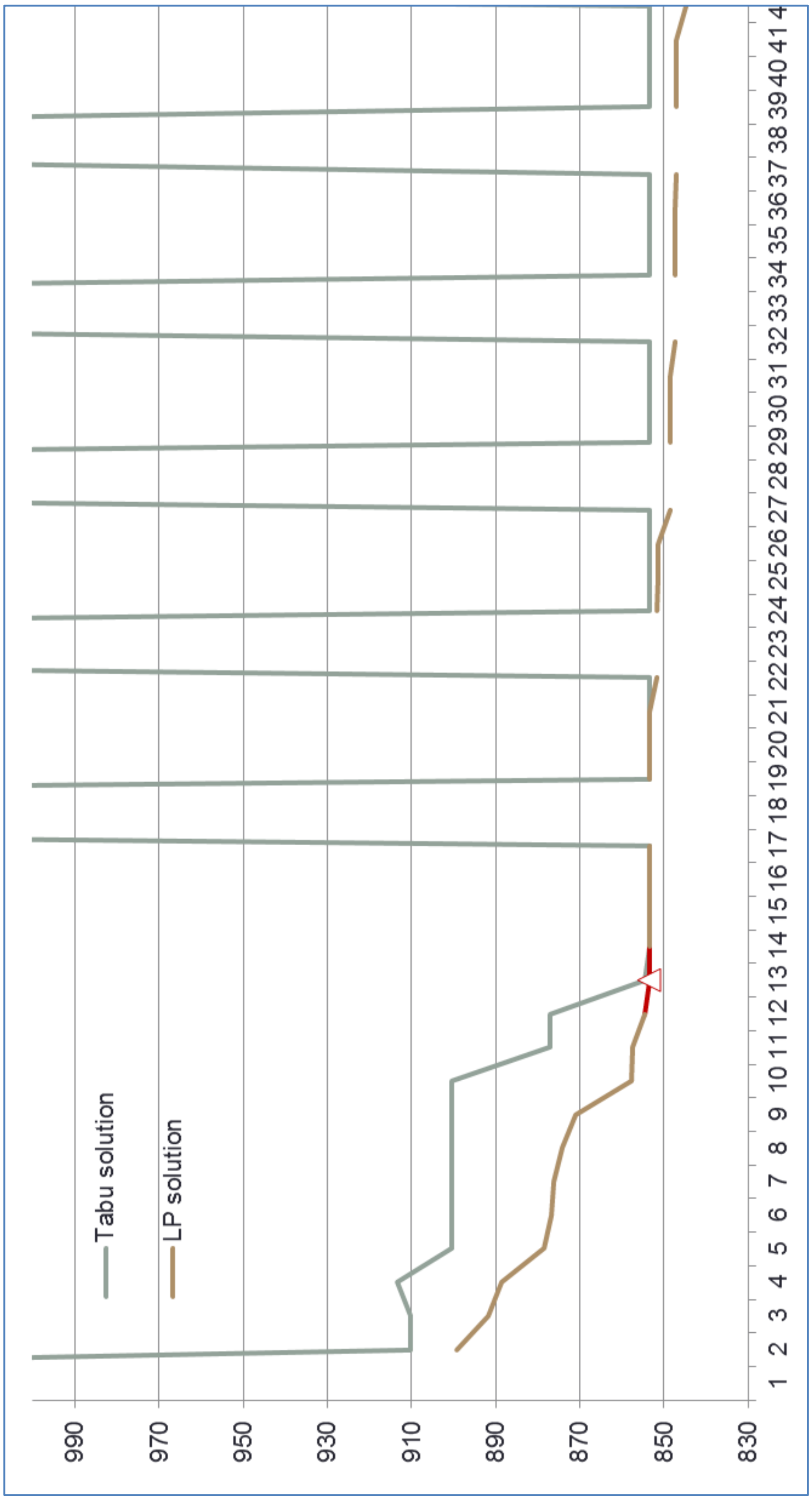
APPENDICES

Appendix A.1. Effects of Route Saving Phase



Source: Figure created by author from data obtained in computational experiments.

Appendix A.2. Integer Feasible Solutions



Source: Figure created by author from data obtained in computational experiments.

Appendix A.3. Route with Different-than-zero Departure Time

Customer	Arrival Time	Deptarture Time	Early Time Window	Late Time Window	Random Start Time	Solution Time
0	0.0	0.0	0.0	230.0	0.0	4.3
53	95.0	105.0	95.0	105.0	0.0	4.3
101	216.1	216.1	0.0	230.0	0.0	4.3
0	0.0	0.0	0.0	230.0	0.0	90.5
51	10.8	20.8	0.0	193.0	0.0	90.5
20	24.0	34.0	0.0	188.0	0.0	90.5
65	51.0	61.0	51.0	61.0	0.0	90.5
71	71.3	81.3	0.0	180.0	0.0	90.5
9	97.0	107.0	97.0	107.0	0.0	90.5
66	127.0	137.0	127.0	137.0	0.0	90.5
35	152.4	162.4	143.0	153.0	0.0	90.5
34	172.6	182.6	0.0	183.0	0.0	90.5
81	187.4	197.4	0.0	192.0	0.0	90.5
50	201.7	211.7	0.0	203.0	0.0	90.5
101	216.1	216.1	0.0	230.0	0.0	90.5
0	0.0	0.0	0.0	230.0	0.0	30.1
40	85.0	95.0	85.0	95.0	0.0	30.1
22	104.0	114.0	97.0	107.0	0.0	30.1
74	149.0	159.0	149.0	159.0	0.0	30.1
72	162.2	172.2	0.0	197.0	0.0	30.1
73	175.3	185.3	0.0	199.0	0.0	30.1
21	186.6	196.6	0.0	201.0	0.0	30.1
101	216.1	216.1	0.0	230.0	0.0	30.1
0	**0.0**	**0.0**	**0.0**	**230.0**	**31.0**	**68.3**
39	**44.6**	**54.6**	**44.0**	**54.0**	**31.0**	**68.3**
23	**68.0**	**78.0**	**68.0**	**78.0**	**31.0**	**68.3**
67	**90.0**	**100.0**	**83.0**	**93.0**	**31.0**	**68.3**
55	**136.0**	**146.0**	**136.0**	**146.0**	**31.0**	**68.3**
25	**172.0**	**182.0**	**172.0**	**182.0**	**31.0**	**68.3**
24	**189.2**	**199.2**	**0.0**	**190.0**	**31.0**	**68.3**
26	**207.4**	**217.4**	**0.0**	**208.0**	**31.0**	**68.3**
101	**216.1**	**216.1**	**0.0**	**230.0**	**31.0**	**68.3**
0	**0.0**	**0.0**	**0.0**	**230.0**	**4.0**	**39.3**
31	**11.0**	**21.0**	**0.0**	**202.0**	**4.0**	**39.3**
62	**58.0**	**68.0**	**58.0**	**68.0**	**4.0**	**39.3**
88	**74.3**	**84.3**	**74.0**	**84.0**	**4.0**	**39.3**
6	**100.2**	**110.2**	**99.0**	**109.0**	**4.0**	**39.3**
101	**216.1**	**216.1**	**0.0**	**230.0**	**4.0**	**39.3**
0	0.0	0.0	0.0	230.0	0.0	56.8
3	8.9	18.9	0.0	197.0	0.0	56.8
33	37.0	47.0	37.0	47.0	0.0	56.8
76	73.0	83.0	73.0	83.0	0.0	56.8
79	92.6	102.6	92.0	102.0	0.0	56.8
78	105.8	115.8	96.0	106.0	0.0	56.8
29	121.8	131.8	0.0	190.0	0.0	56.8
68	142.0	152.0	142.0	152.0	0.0	56.8
80	182.0	192.0	182.0	192.0	0.0	56.8
12	194.5	204.5	0.0	205.0	0.0	56.8
101	216.1	216.1	0.0	230.0	0.0	56.8
0	0.0	0.0	0.0	230.0	0.0	53.0
94	4.8	14.8	0.0	207.0	0.0	53.0
98	18.5	28.5	0.0	198.0	0.0	53.0
14	33.2	43.2	0.0	187.0	0.0	53.0
44	69.0	79.0	69.0	79.0	0.0	53.0
38	89.8	99.8	83.0	93.0	0.0	53.0
86	107.3	117.3	0.0	184.0	0.0	53.0
16	120.9	130.9	0.0	190.0	0.0	53.0
91	133.8	143.8	0.0	194.0	0.0	53.0
100	185.0	195.0	185.0	195.0	0.0	53.0
37	196.1	206.1	0.0	198.0	0.0	53.0

101	216.1	216.1	0.0	230.0	0.0	53.0
0	0.0	0.0	0.0	230.0	0.0	95.3
82	9.3	19.3	0.0	196.0	0.0	95.3
48	21.5	31.5	0.0	192.0	0.0	95.3
47	34.0	44.0	0.0	185.0	0.0	95.3
36	48.3	58.3	41.0	51.0	0.0	95.3
64	79.7	89.7	73.0	83.0	0.0	95.3
49	108.0	118.0	108.0	118.0	0.0	95.3
10	132.5	142.5	124.0	134.0	0.0	95.3
63	151.9	161.9	0.0	185.0	0.0	95.3
90	166.4	176.4	0.0	187.0	0.0	95.3
32	180.9	190.9	0.0	186.0	0.0	95.3
1	199.2	209.2	0.0	204.0	0.0	95.3
101	216.1	216.1	0.0	230.0	0.0	95.3
0	0.0	0.0	0.0	230.0	0.0	77.9
2	7.2	17.2	0.0	202.0	0.0	77.9
57	19.5	29.5	0.0	196.0	0.0	77.9
42	32.7	42.7	31.0	41.0	0.0	77.9
43	46.9	56.9	0.0	185.0	0.0	77.9
15	64.2	74.2	61.0	71.0	0.0	77.9
41	97.0	107.0	97.0	107.0	0.0	77.9
75	111.6	121.6	0.0	192.0	0.0	77.9
56	130.0	140.0	130.0	140.0	0.0	77.9
4	149.0	159.0	149.0	159.0	0.0	77.9
54	168.2	178.2	0.0	197.0	0.0	77.9
77	187.5	197.5	179.0	189.0	0.0	77.9
28	202.9	212.9	0.0	213.0	0.0	77.9
101	216.1	216.1	0.0	230.0	0.0	77.9
0	**0.0**	**0.0**	**0.0**	**230.0**	**36.0**	**35.4**
27	**38.0**	**48.0**	**37.0**	**47.0**	**36.0**	**35.4**
69	**55.3**	**65.3**	**50.0**	**60.0**	**36.0**	**35.4**
30	**78.6**	**88.6**	**71.0**	**81.0**	**36.0**	**35.4**
70	**182.0**	**192.0**	**182.0**	**192.0**	**36.0**	**35.4**
101	**216.1**	**216.1**	**0.0**	**230.0**	**36.0**	**35.4**
0	0.0	0.0	0.0	230.0	0.0	31.4
95	5.9	15.9	0.0	205.0	0.0	31.4
92	18.0	28.0	18.0	28.0	0.0	31.4
59	28.9	38.9	0.0	202.0	0.0	31.4
99	83.0	93.0	83.0	93.0	0.0	31.4
87	99.5	109.5	93.0	103.0	0.0	31.4
97	111.9	121.9	0.0	202.0	0.0	31.4
13	159.0	169.0	159.0	169.0	0.0	31.4
58	200.0	210.0	200.0	210.0	0.0	31.4
101	216.1	216.1	0.0	230.0	0.0	31.4
0	0.0	0.0	0.0	230.0	0.0	68.0
52	4.5	14.5	0.0	208.0	0.0	68.0
7	18.5	28.5	0.0	198.0	0.0	68.0
11	67.0	77.0	67.0	77.0	0.0	68.0
19	84.1	94.1	0.0	187.0	0.0	68.0
8	104.2	114.2	95.0	105.0	0.0	68.0
46	119.6	129.6	0.0	184.0	0.0	68.0
17	157.0	167.0	157.0	167.0	0.0	68.0
5	177.0	187.0	0.0	199.0	0.0	68.0
60	188.8	198.8	0.0	201.0	0.0	68.0
89	202.4	212.4	0.0	211.0	0.0	68.0
101	216.1	216.1	0.0	230.0	0.0	68.0
0	0.0	0.0	0.0	230.0	0.0	32.9
18	6.3	16.3	0.0	204.0	0.0	32.9
83	19.0	29.0	0.0	198.0	0.0	32.9
45	32.2	42.2	32.0	42.0	0.0	32.9
84	101.0	111.0	101.0	111.0	0.0	32.9
61	115.0	125.0	0.0	194.0	0.0	32.9
85	127.6	137.6	0.0	196.0	0.0	32.9
93	188.0	198.0	188.0	198.0	0.0	32.9
96	200.0	210.0	0.0	204.0	0.0	32.9
101	216.1	216.1	0.0	230.0	0.0	32.9