12-2017

Methodologies for Solving Integrated Transportation and Scheduling Problems

Fereydoun Adbesh
University of Arkansas, Fayetteville

Methodologies for Solving Integrated Transportation and Scheduling Problems

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy in Engineering with a concentration in Industrial Engineering

by

Fereydoun Adbesh
Iran University of Science and Technology
Bachelor of Science inIndustrial Engineering, 2004
Mazandaran University of Science and Technology
Master of Science in Industrial Engineering, 2008

December 2017
University of Arkansas

This dissertation is approved for recommendation to the Graduate Council.

Dr. Chase Rainwater
Dissertation Director

Dr. Edward Pohl
Committee Member

Dr. Heather Nachtmann
Committee Member

Dr. John Kent
Committee Member

Abstract

This research proposes novel solution techniques to optimize two real-world problems in the area of scheduling and transportation. We first consider a model for optimizing the operations of dredges. In this problem, scheduling and assignment decisions are integrated across a finite planning horizon. Additional constraints and problem elements explicitly considered include, but are not limited, to environmental work window restrictions, budget limitations, dredge operation rates and schedule-dependent dredge availability. Our approach makes use of Constraint Programming (CP) to obtain quality and robust solutions within an amount of time small enough to be useful to practitioners. The expanded feature set of the methodology presented makes our solution tool the most comprehensive and flexible decision-making framework for dredge scheduling in existence.

The second transportation and logistics problem considered in this dissertation considers a unified variation of the Vehicle Routing Problem (VRP). This work offers a powerful yet flexible tool to model and solve real-world problems, each with their specifications, constraints, and requirements. We review existing VRP problems from the literature and propose new VRP variants that differ from the existing ones by the consideration of hours of service regulation on the active and drive hours of drivers in a single or multiple shifts. Real-world instances of these problems consist of thousands of customer locations and hundreds of vehicles. To ensure the quality of the solutions, we compare the performance of our approach with CPLEX on several benchmark instances from the literature.

Finally, the third chapter of this work focuses on a comprehensive analysis of the methodology presented in Chapter 4. Specifically, sensitivity analysis regarding the parameters driving the performance of the heuristics is performed. Also, we propose a Genetic Algorithm (GA) to solve the VRP variants in Chapter 3 and provide a computational study of its performance against CPLEX and the approaches in Chapter 3.

Acknowledgments

I would like to thank my advisor, Dr. Chase Rainwater, for all his guidance and support. The chance to work and study with him is truly a great privilege. I have been incredibly fortunate to have him as an advisor who challenged me, kept me enthusiastic and gave me just the right level of freedom and guidance I needed to explore on my own. I am forever grateful for his great mentorship, endless personal and professional support.

I also would like to thank my committee members, Drs. Heather Nachtmann, Edward Pohl and John Kent for their guidance, encouragements and invaluable advices throughout my doctoral studies. Their feedbacks have taught me great lessons and made this dissertation more solid. I am also grateful to all my professors for consistently sharing their knowledge without any hesitation to enable me succeed in my studies. I specifically thank Dr. Justin Chimka, our graduate program coordinator, for his great first impression.

I am thankful to my colleagues in ETP team of J.B. Hunt, Andy Edwardes, Becca Luetjen, Eric Schauer, Michael Herm and Ashly Artis, for the support and friendship provided by each one them. The few, the proud! I cannot thank enough my boss at J.B. Hunt, Douglas Mettenburg, for being such a great manager, mentor, leader and role model.

Finally, I am very grateful of my parents, the very first support of my life and my brothers and sister. My last appreciation goes to my true friend and wife, Sahar, for her immeasurable sacrifices and unfailing love in absence of my family while I pursued this final degree.

Dedication

To my loving wife, Sahar, without her endless patience and support, pursuing PhD would have stayed a dream for me.

تا در طلب گوهر کانی کانی　　تا در هوس لقمۀ نانی نانی

این نکتۀ رمز گر بدانی دانی　　هر چیز که در جستن آنی آنی

مولانا

You are the treasure, if the gems are your aim,

No more than a grain, if a loaf is your claim,

Recall this secret, when you play this game,

Whatever you pursued, is what you became.

Rumi (1207 1273),

A great Persian poet

Translation to English:

Maryam Dilmaghani

Contents

List of Figures

List of Tables

Disclaimer

The views expressed in this dissertation are those of the author and do not reflect the official policy or position of the United States Army Corps of Engineers.

1. Introduction

This research considers two classes of optimization problem that contribute to the scheduling and routing literature. Both problems seek to expand the abilities of practitioners in the transportation logistics field to more easily integrate operations research into their decision-making process. This integration is done by developing flexible, comprehensive and efficient approaches that better represent real-world transportation challenges. Both scheduling and routing problems have been widely studied over the last five decades. In this research, we contribute to these fields by considering new methods for determining the best utilization of available resources for realistic large-scaled problems, as well as introducing new models for the well-known Vehicle Routing Problem (VRP) with hours of service regulations.

Dredging is an excavation activity that usually happens at the bottom of shallow seas or waterways to remove the sediments, mud, and even soil. The primary purpose of dredging is to make the waterways navigable and creates an anti-sludge pathway for ships, boats and other sailing vessels. It also is used in eroded coastal beaches to replenish sand by scooping out the sand from water to shore. In the U.S. hundreds of maintenance dredging projects operate each year to provide safe, reliable and cost-efficient waterborne transportation systems with minimal impact on the environment. The transportation system mainly consists of the movement of commerce, national security needs, and recreation. The U.S. Army Corps of Engineers (USACE) is responsible for maintaining the channel depths at U.S. harbors and on inland waterways to keep them navigable. This mission includes nearly $12,000$ miles of inland and intra-coastal waterways. The dredging operations remove over 250 million cubic yards of material each year at an average annual cost of over $1.3 billion (see *Water Resources*). Each year, the decision of assigning individual dredging vessels (whether government or private industry) to navigation projects is made by the USACE to maximize the total cubic yards of dredging while not exceeding the total budget. The dredging projects must adhere to natural environmental windows to mitigate damage to the local environmental species which brings consequential limitations on how and when dredging

can occur. These limitations put temporal constraints on the conduct of dredging in order to protect biological resources or their habitats from potentially detrimental effects (Dickerson et al., 1998). In a part of Chapter 2, we used partial dredging during restricted periods as the relaxation and studied the impact of relaxing each window individually, as well as collectively. Partial dredging during a restricted period means a dredge vessel can work during the restricted periods, but with a slower operation rate. Having encountered a group of aquatic animals like turtles, fish and other wildlife during the restricted periods, dredging vessels might have to stop dredging to prevent causing harm. Restricted period relaxations can only be implemented in localized areas after extensive research has been conducted to pinpoint species migratory patterns and sensitivities to dredging activities.

In Chapter 2, we study the allocation and scheduling of available dredge vessels to dredging jobs under a heavily constrained environment. The constraints include environmental considerations, budget limitations, and resource availability. We build upon an existing Constraint Programming (CP) method to offer a robust and flexible decision tool that can be used by USACE to optimally utilize the available resources. This tool can handle different variations of the problem based on their needs that the previous work failed to address. The following capabilities are added to the tool based on the needs of the USACE in facing the real-world dredge operation.

1. Allow partial dredging during environmental windows.
2. Have variable job sizes.
3. Operate multiple dredges on the same job.
4. Make multiple trips to the same jobs.
5. Consider different operation rates for each job.
6. Set different unit costs of dredging for each job.
7. Simulate downtime for dredges.

8. Include Mob/Demob cost in the budget.

9. Check dredge capability to perform assigned projects.

The second problem of interest is one of the well-known problems in the area of transportation and logistics, the Vehicle Routing Problem (VRP). The goal of VRP is to obtain the best routes for a fleet of vehicles in which all locations are served and all vehicles return to the origin depot. The objective is to minimize the total cost of traveling between locations while adhering to each vehicle's capacity constraint. Over the past five decades, after Dantzig and Ramser (1959) introduced VRP as the truck dispatching problem to transport gas between stations using trucks, many researchers have studied several variations of this problem to model real-world routing scenarios. The appearance of different VRP variants is the reflection of the challenges faced by researchers as well as companies to model their transportation network.

In Chapter 3, we assume the role of a logistics professional in an attempt to consolidate and satisfy customer demand using a unified variant of the Vehicle Routing Problem (VRP) to reduce operation cost while maintaining system performance levels. We propose a comprehensive tool that integrates multiple algorithmic techniques into a platform to provide a high performance solution approach that addresses a wide range of VRP variants such as capacitated VRP (CVRP), VRP with time windows (VRPTW), heterogeneous fleet VRP (HVRP), open VRP (OVRP), multi-depot VRP (MDVRP), pickup and delivery VRP (PDVRP), simultaneous pick-drops VRP (SVRP) and VRP with backhauls (VRPB). This optimization tool is practical and appealing to industry tool that uses approaches which yield quality solutions in a short time and is flexible to a wide range of problems.

Moreover, a family of new variants of VRP is introduced which considers the hours of service regulations imposed by the U.S. Department of Transportation (DOT) to limit the daily and weekly driving and working hours of the truck drivers. Due to the huge additional complexity brought by imposing these regulations on the VRP models, these new variants are formulated in three different plan time horizons: i) daily, ii) weekly and iii) infinite time. Based on these governmental regulations, the drive, work, and rest hours of each driver must follow the rules on the

daily and weekly total hours of driving and the required amount of rest between them. These regulations usually reduces the drivers' working hours and ultimately the total number of shipments. Despite the fact that the routing and scheduling of trucks is currently being impacted by these regulations, to the best of our knowledge, there are no optimization solutions in the literature or practice that address this complication. In this research, we introduce a multi-phase heuristic solution approach to solve this class of problems and compare its performance with a black box solver. We also offer a realistic case study to illustrate the benefit of our model and tool on real-world logistics scenarios.

The final effort in this dissertation offers a comprehensive analysis of the heuristic method proposed in Chapter 4 and additional performance enhancements. In addition, a genetic algorithm is proposed which utilizes sub-procedures from Chapter 3. This approach is compared against the approach in Chapter 3 and CPLEX. The genetic algorithm offers improved solutions. Additional exploration of the algorithm parameters provide opportunities to retain these quality solutions with decreased computational effort.

Bibliography

Dantzig, G. B. and Ramser, J. H. (1959). The truck dispatching problem. Management science, 6(1):80–91.

Dickerson, D. D., Reine, K. J., and Clarke, D. G. (1998). Economic impacts of environmental windows associated with dredging operations. Technical report, ARMY ENGINEER WATERWAYS EXPERIMENT STATION VICKSBURG MS.

Water Resources. Inland waterway navigation: Value to the nation. http://www.corpsresults.us/. Accessed: July 2015.

2. Improvements and Enhancements to the Dredge Fleet Scheduling Problem

Abstract: The U.S. Army Corps of Engineers (USACE) annually spends more than 100 million dollars on dredging hundreds of navigation projects across more than $12,000$ miles of inland and intra-coastal waterways. In this study, we expand the logic-based mathematical programming solution approaches to address the real-sized dredge scheduling challenges faced by USACE. Each year USACE must determine how to allocate dredging resources to required jobs while adhering to various limitations such as budget, resource availability and so-called environmental restrictions that define when dredging cannot take place due to protecting wildlife.

In this study, we utilize constraint programming optimization capabilities to model and solve the dredge fleet scheduling problem (DFSP) to address USACE needs for each job. These requirements include having different equipment productivity rates on each dredging job, partial dredging during restricted periods, variable job sizes, multiple trips of dredges to the same jobs and multiple dredges working on the same jobs. The result of our research is to offer a robust decision tool that can be used by USACE to determine the appropriate dredge fleet and the operations and sequences associated with that fleet for a given set of jobs and their characteristics.

2.1 Introduction

In this chapter, we study the allocation and scheduling of an available fleet of dredge under a heavily constrained environment. Some constraints are the regulations imposed by the government to protect the wildlife in dredging locations and others are system constraints such as budget, availability and production rates of the resources. In general, CP captures the feasible solution space search with more constraints in the model with the ability to apply global (logical) constraints when the actual interest is to provide good feasible solutions (Jain and Grossmann, 2001). Also, so-called interval variables provide CP a competitive advantage when modeling a scheduling problem. We provide a powerful, yet flexible, tool to find robust solution to handle

different variations of the problem based on our client needs. We could maximize the total cubic yards of dredging by varying our CP model to use multiple resources and trips to the dredging location. Finally, performing a sensitivity analysis on partial operations during the environmental work windows provides useful insights into how these restricted periods affect the efficiency of dredges. Partial operations of dredges occur when we dredge at a slower rate during restricted periods.

2.1.1 Problem Description

The U.S. Army Corps of Engineers (USACE) conducts maintenance dredging at hundreds of navigation projects each year to provide safe, reliable and cost-efficient waterborne transportation systems with minimal impact on the environment to ensure movement of commerce, national security needs, and recreation. The primary purpose of dredging is to maintain navigable channel depths at U.S. harbors and on inland waterways across nearly $12,000$ miles of inland and intracoastal waterways and navigable channels in 192 commercial locks and dam sites in 41 states. The USACE dredges over 250 million cubic yards of material each year at an average annual cost of over $1.3 billion to keep the nation's waterways navigable (see Resources). Each year, the decision of assigning individual dredging vessels (whether government or private industry) to navigation projects is made by the USACE to maximize the total cubic yards of dredging while not exceeding the total budget.

To mitigate damage to the local environmental species, dredging jobs must adhere to natural environmental windows which brings consequential limitations on how and when dredging can occur. For example, west coast Ventura Harbor is not dredgable from mid-April to the end of August to avoid harming the California Grunion living in the area. It is worth mentioning that there is interest in studying how any future placement of new environmental windows as well as tightening of existing environmental windows could impact system cost efficiency.

In dredge fleet scheduling problem (DFSP) optimization, we offer a robust tool for the USACE to allocate one or more dredging jobs to each available dredging vessel and schedule the

dredging jobs about their environmental work windows. The tool must be powerful enough to find good quality solutions and at the same time flexible enough to deal with different operational needs of the Corps which will be explained in Sections 2.6.1 to 2.6.8.

In the DFSP, the following base assumptions are made according to how the jobs are assigned to the fleet of dredges in USACE. We refer to the environmental time windows which dredging is prohibited during their time interval the restricted periods (RPs). On the other hand, the work window in the time that dredging is allowed.

1. A dredge can work on at most one job at a time.
2. A job can be assigned to at most one dredge.
3. The travel time between the depot and the first dredging job and from the last job to the depot in each dredge schedule are not considered in the solutions.
4. A dredge must finish each assigned job in one visit.
5. Partial dredging is not allowed. On the other hand, dredging is completely prohibited in the restricted period(s) of each job.
6. The size of each job (cubic yards of material that needs to be removed from the bed of the waterway) is deterministic and constant.
7. The travel speed of each dredge is deterministic, constant, and is the same for all dredges. Thus, the travel time between each pair of job locations is the same for all dredges. For more information refer to Section 2.4.
8. The production rate (cubic yard of dredging per day) of each dredge is the same for all the jobs in any location add space. For more information refer to Section 2.4.

We refer to the model with all above assumption as our *base model* and relax some of these assumptions to make the tool more useful to USACE. These improvements will be discussed in detail in Section 2.6.

2.1.2 Problem Illustration

To illustrate the DFSP, we have shown the schedule of only one dredge among all available dredges in our fleet in Figure 2.1. All other dredges will have a similar set of choices for scheduling if they exist in the solution. In this example we have three jobs in three different locations and the time horizon is from t_1 to t_7. We assume that dredges are in their first job's locations at the beginning of the time horizon and they can perform dredging until the end of the time horizon. On the other hand, we do not consider the travel time of dredges from depot to their first job and from their last job to the depot.

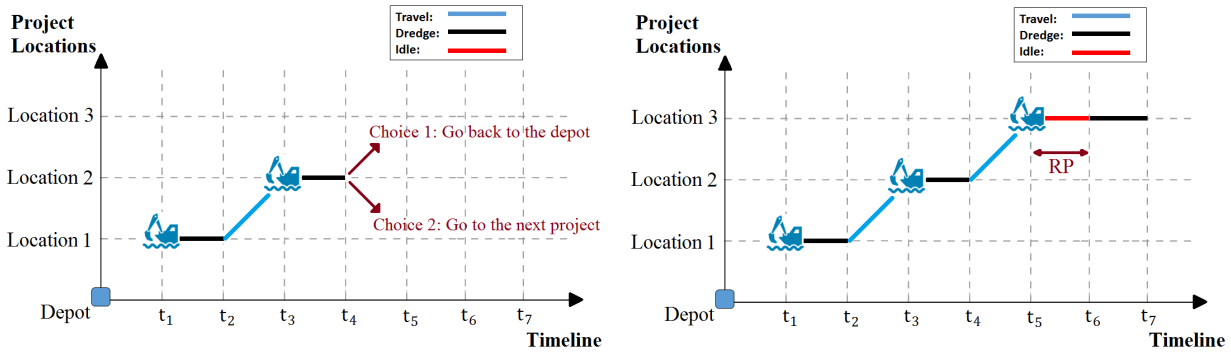


Figure 2.1: dredge fleet Scheduling Problem Illustration

In Figure 2.1 (a), the dredge starts its operation on job 1 at location 1 at time t_1 and finishes it at t_2. The dredging time is shown with a thick horizontal black line in the figure. After finishing job 1, the dredge travels from job location 1 to 2 in $t_3 - t_2$ units of time (shown with a thick blue inclined line) and starts working on job 2 at time t_3 and finishes it in t_4. At the end of each job, each dredge has two choices: i) it may terminate its operation by going back to the depot and ii) it can travel to another job and continue its dredging operations on another job. The latter choice is shown in Figure 2.1 (b) in which the dredge travels to location 3, but cannot start the dredging process immediately at time t_5 because job 3 has a restricted period (RP) that ends in t_6 (shown by a thick red line in the figure). Thus, the dredge remains idle at location 3 from time t_5 until time t_6 and starts working on jobs 3 at time t_6. As we mentioned before, after finishing job 3 at

time t_7, the dredge can go back to the depot or travel to another job's location to dredge another job.

In this illustration, we emphasized the scheduling part of the DFSP and the roll of restricted periods that might play in any solution. As mentioned in Section 2.1.1 there are other constraints and assumptions associated with the problem that needs to be fulfilled in any feasible solution.

2.2 Literature Review

Perhaps the closest problem to dredge scheduling problem in the literature is the machine scheduling problem and its variations. Gedik et al. (2016) represented the dredge scheduling problem as a case study for their study on the parallel machine scheduling problem with sequence dependent set-up times and job availability problem. They proposed a constraint programming (CP) model and logic-based Benders algorithms to find the best solutions for the scheduling of non-identical jobs on unrelated parallel machine with sequence dependent setup times and job availability intervals in a fixed planning horizon. The two logic-based benders decomposition algorithms used in their study can obtain near-optimal solutions compare with pure CP which yields good quality feasible solutions but not optimal.

A parallel scheduling problem was first described by Arkin and Silverberg (1987) and proven to be NP-complete. They tried to maximize the total value of a subset of processing jobs of interest each with a fixed start time, end time and an associated value. Rojanasoonthon and Bard (2005) studied a parallel machine scheduling problem with time windows on prioritized jobs. Their objective function maximized the total number of jobs scheduled. They proposed a greedy adaptive search to find quality solutions for large problem instances due to incapabilities of their ILP to find any feasible solution. In other research, Eliiyi and Azizoğlu (2009) developed a branch and bound algorithm for a fixed job scheduling problem with machine dependent job values to obtain near-optimal solutions for the problem instances up to 100 jobs. A comprehensive review is provided by Allahverdi et al. (2008) on the machine scheduling problems with different constraints and performance measures.

The machine scheduling problem has found interests in other industries aside from the production machine scheduling problems. Pearn et al. (2002) used a parallel machine scheduling problem with sequence dependent setup times to model the jobs in a water probing factory and transform it to the vehicle routing problem with time windows (VRPTW) due to the complexity of their ILP formulation. They applied three heuristic methods for VRPTW to find near-optimal solutions for a real-world instance. Many researchers used the parallel machine scheduling problem to semiconductor manufacturing operations. A comprehensive survey is presented in Mönch et al. (2011). For example, Cakici and Mason (2007) proposed a heuristic method which can help obtain near-optimal solutions with small optimality gap.

2.3 Foundational Work on Dredge Scheduling

In 2012, Nachtmann et al. (2014) began work on an optimization tool to improve dredge scheduling decision-making process. They introduced a mixed integer programming (MIP) formulation and developed a customized constraint programming (CP) approach that was shown to provide good quality solutions to the problem instances with 100+ jobs in reasonable time for the base model as defined in Section 2.1.1. Gedik et al. (2016) also modeled the base model of DFSP as a parallel machine scheduling problem with sequence dependent set-up times and job availability intervals and used CPO and logic-based benders decomposition algorithms to solve it. Their model had a better performance and could find near-optimal solutions comparing with pure CP used by Nachtmann et al. (2014) which yields good quality feasible solutions but not optimal.

However, there are significant opportunities for expansion and improvement of the developed optimization tool for the base model problem to be more robust and practicable. This tool can provide applicable solutions to various needs of USACE in different locations using several types of dredges in the dredging jobs. The limitations of the current tool do not address the following issues:

1. *Partial dredging during environmental windows:* Environmental restricted periods prevent any dredging from occurring over a specified horizon. That means partial dredging is not

allowed during the environmental windows. This restriction limits the capability of available resources to conduct their dredging jobs dramatically. For example according to a real problem instance in Section 2.4, 6 jobs out of 116 jobs cannot be dredged at all because their environmental restricted periods span the whole year.

2. *Variable job sizes:* Jobs must be fully completed by a dredge vessel before moving on to other jobs. This means partial jobs are not allowed, and the size of jobs cannot be varied. In practice, there is often a "minimum requirement" that really should attempt to be met as well as a "target requirement" that would be ideal to get if time/money allows.

3. *Multiple dredges on the same job:* Jobs must be satisfied by a single dredge vessel. Some jobs need to be done by different types of dredge vessels. For example, some dredging jobs (e.g., Cayuga Lake Watershed, NY) needs Yaquina, Essayons, and contract dredges typically.

4. *Multiple trips to the same jobs:* Jobs must be satisfied by a single dredge vessel trip. It means a job cannot start by a dredge vessel if it cannot be finished before the beginning of its environmental window(s) or the end of the overall time horizon. This restriction prevents us from dredging a job in multiple trips (e.g., some before the beginning time of the environmental window(s) of a particular job and after the end time of its environmental window(s)).

5. *Different operation rates for each job:* The operation rates of dredge vessels are currently constant in all dredging jobs. The cubic yards of dredging per hour for each dredge vessel can vary greatly from one job to the next due to wave conditions, weather, sediment types, etc. By allowing different operation rates of dredging for each job we make the model more realistic.

6. *Different unit costs of dredging for each job:* The dredging cost per cubic yard is constant in all jobs. Practically, the cost rate of dredging is different for each job-vessel combination

due to specifications of each job (e.g., the width and depth), type of sediments, and fuel cost.

7. *Stimulating downtime for dredges:* Dredge vessels typically have fixed amounts of time in the yards for repairs and consequently are not available.

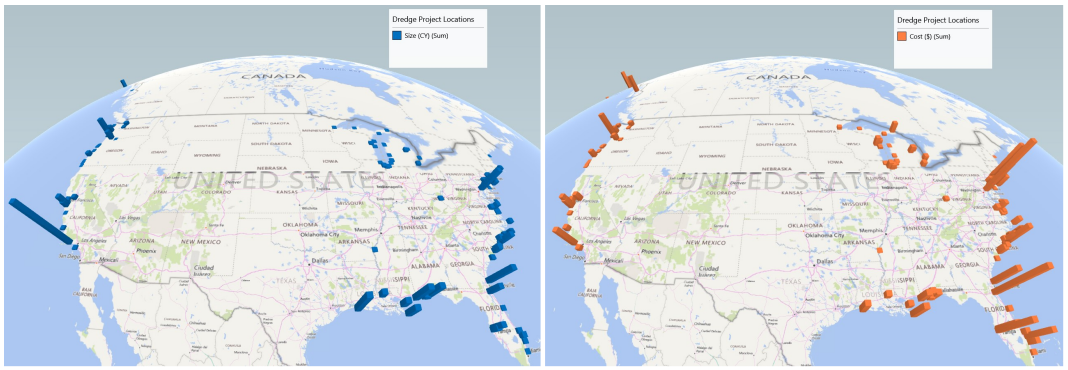
8. *Mob/Demob cost:* The mobilization cost is usually proposed to be split based on the cubic yardage dredged at each job or the travel time of each dredge vessel.

9. *Dredge Capability check on Jobs:* Some dredges cannot perform some of the jobs. Some dredges cannot physically perform the dredging job in some locations. Some of the dredges in the fleet is too big to maneuver for some small ports, and some of them do not have deep enough dragheads for some jobs.

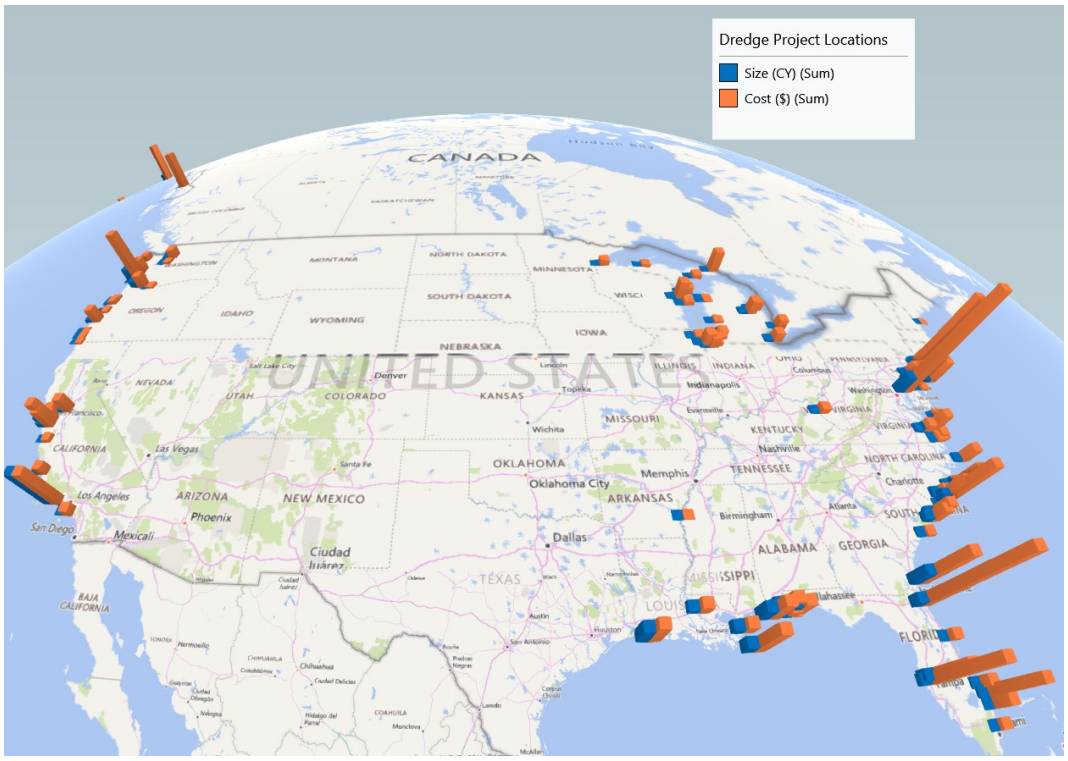
In Section 2.6, we address these weaknesses in details and show how to utilize the abilities of constraint programming optimization (CPO) to solve these issues and report the improvement by comparing with the base model.

2.4 Dredge Scheduling Requirements

In this section, the details of the requirements of the dredge scheduling problem are presented. The data used to establish the problem was provided by the USACE Dredging Information System (navigationdatacenter.us). A total of 116 unique navigation channel maintenance dredging jobs across the USA from east to west coast and Alaska with associated size and cost were identified and are shown in Figure 2.2 (a) and (b), respectively.



(a) i) Dredging Job Sizes (cubic yards) ii) Dredging Job Costs (US dollar)



(b) Dredging Job Sizes and Costs

Figure 2.2: 3-D Representation of 116 Dredge Projects Locations with Associated Size and Cost

The dredging jobs volumes and costs were calculated by averaging over the range of years for which DIS data was available for each job (Nachtmann et al., 2014). The information about the size and cost of each 116 dredging jobs are listed in the in Table 2.15 in the Appendix. The total size of the 116 jobs is $48,305,584$ cubic yards of dredging with the total cost of $\$223,012,020$. All type of costs associated with dredging jobs from start to finish, including the mob/demob,

fuel, labor, maintenance, etc., are considered in the calculation of each job cost. The size and cost of each dredging job are shown in Figure 2.2 (a). According to Figure 2.2 (b), the jobs in the east coast historically have more cost per cubic yards of dredging comparing with the dredging jobs in the west cost because of the different sediment type of the dredged waterways and weather conditions.

The DIS historical data was also used by Nachtmann et al. (2014) to gather information on the performance of the individual Corps-owned dredge vessels, as well as the contract dredges work for the USACE. They identified 30 different dredges with associated daily production rates as listed in Table 2.16 in the Appendix. The production rate of each dredge is the actual rate of the dredge and calculated by the total cubic yards dredged for each job by the dredge divided by the total number of days that the dredge worked on the job including the delays. The delays can be due to inclement weather, downtime of the dredge for maintenance and repair and any other problems delaying the finish time of the job. The range of production rate for dredge fleet is from 1,238 to 19,245 cubic yards per day.

The total number of 130 restricted periods are imposed to the 116 jobs and used as the parameter within the DFS optimization models. According to dredging job locations, each job can have more than one restricted period, which is identified using the Threatened, Endangered and Sensitive (TES) species program (fs.usda.gov). The summary of the restricted periods based on the type of endangered species is shown in Table 2.1.

Table 2.1: Summary of Restricted Periods (RPs) (duration: days)

Species Type	**Total Duration**	**Avg. Duration**	**No. of Jobs with RP**
Fishes	12,541	187	67
Marine Turtles	5,773	222	26
Birds	3,221	179	18
Marine Mammals	3,006	137	22
Crustaceans	1,496	150	10
Marine Mussels	832	104	8
TOTAL:	26,869	178	151

The final input parameter for our DFS optimization models is the travel time between each pair of job locations. A GIS layer is used to compute the distance on the waterways as the roads connecting all the locations. Ignoring the size and type of the dredges for simplicity, the average speed of 50 miles per day is assumed for all dredges.

2.5 Optimization Methodology

In this study, we used constraint programming optimization (CPO) to solve the DFSP and provide a powerful tool for USACE to find a good quality solution for their various needs. In the following two sections, first, we will discuss why we used CPO, and then we state the CP formulation of the base model which is the basis of our extensions and improvement of the model to address the USACE requirements.

2.5.1 Why Constraint Programming

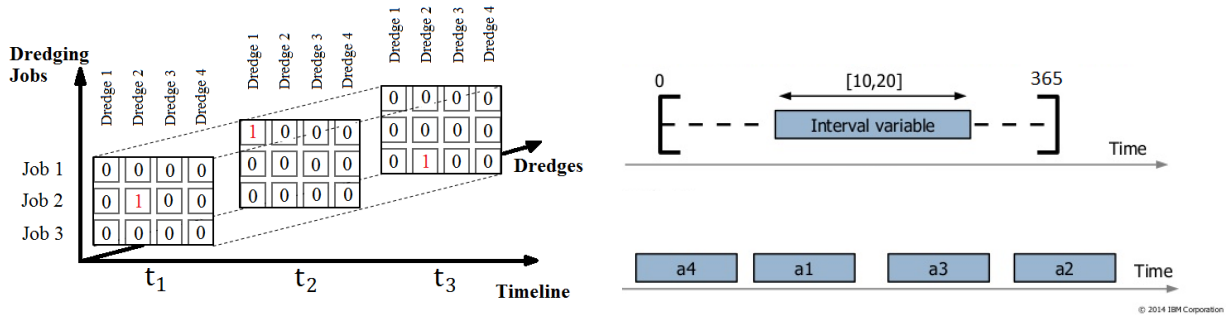
The primary motivation for constraint programming instead of solving the MIP formulation optimally is the size of the problems to be solved. The real data problem instance in Section 2.1.1 has 116 jobs with 30 dredges in the time horizon of 365 days with total 138 restricted periods associated with dredging jobs. DFSP with time windows can be converted to the parallel machine scheduling problem with sequence dependent setup times and job availability intervals (Gedik et al., 2016), which is known to be an NP-Complete problem. Table 2.2 shows the number of variables and constraints of the base model instance in both mathematical formulations of MIP and CP derived from ILOG CPLEX and ILOG CP solvers.

Table 2.2: Number of Variables and Constraints in MIP and CP Formulation

Model	**# of Var.**	**# of Con.**
MIP	1,273,680	1,741,276
CP	3,626	380

As shown in Table 2.2, the number of variables and constraints in MIP formulation are dramatically more than CP formulation which prevents CPLEX loading the data into its memory.

In contrast, CP solver can find a quality solution for the base model in the reasonable amount of time. The main reason for the difference between the number of variables and constraints is using binary variables in MIP formulation versus interval variables in CP. In the MIP formulation (Nachtmann et al., 2014) the binary variables z_{djt} with value 1 if dredging vessel d begins work on job j in period t, $\forall d \in \{1,2,\ldots,30\}$, $j \in \{1,2,\ldots,116\}$, and $t \in \{1,2,\ldots,365\}$ is used. As a small example, consider a feasible solution with the binary variables z_{djt} with 3 jobs, 4 dredges and 3 periods of time in Figure 2.3 (a). In this solution, dredge 2 starts working on job 2 in t_1 (period 1) and then travels to job 3 and start working on it in t_3. Also, dredge 1 starts working on job 1 at t_2. The finish time of the jobs can be calculated by adding the required time to finish each job to the start time of each job. The required time for each dredge-job pair is calculated by dividing the size of the job to the production rate of the dredge.



(a) Binary Variables z_{djt}

(b) Interval Variables z_j (Source: 2014 IBM Corporation)

Figure 2.3: Binary Variable vs. Interval Variables

As shown in this small example, 36 (3 jobs $\times$ 4 dredges $\times$ 3 periods) binary variables exist in the MIP formulation, and just three of them will become equal to 1 in the final solution. Thus, we are considering variables with the value of 0 that consume computational memory. In contrast, in CP formulation of the problem contains only 3 interval variable, one for each job. An interval variable represents an interval of time during which something happens (*ibm.com*). More details on interval variables will be discussed in Section 2.6.1. A typical interval variable is shown in Figure 2.3 (b). Each interval variable includes a possible range, start and finish time

value and size of the variable. Moreover, interval variables can be set in any sequence for scheduling purposes.

2.5.2 Constraint Programming Formulation

In this section, we present the CP formulation studied by Gedik et al. (2016) for the base model of DFSP. In general, in CP scheduling the most complex and computationally expensive parts of the MIP model are handled by using *global constraints* and *interval variables*. The speed and effectiveness of a CP model are highly dependent on the variables domains and global constraints. CP Optimizer tool, a constraint programming solver engine developed by ILOG, solves a model using constraint propagation and constructive search with search strategies (*ibm.com*).

The notation of sets, parameters and decision variables used in developing the CP dredge fleet scheduling (CPDFS) formulation are shown in Table 2.3.

Table 2.3: Notation of CPDFS base model Formulation

Notation	Description
Sets:	
D	Set of dredging equipment resources available in each period.
T	Set of consecutive periods comprising the planning horizon.
J	Set of dredge jobs that need to be completed over the planning horizon.
W_j	Set of RPs applicable to dredging job j.
Parameters:	
b_w	The beginning of RP w, $w \in W_j, j \in J$.
e_w	The end of RP w, $w \in W_j, j \in J$.
r_d	The operation rate (cubic yards/day) of dredge equipment $d \in D$.
q_j	The dredging amount of job $j \in J$ (in cubic yards).
$t_{jd} = \lceil q_j / r_d \rceil$	The time (days) that it takes for dredge equipment piece $d \in D$ to complete job $j \in J$.
$t_{jj'}$	The time (days) that it takes to move a dredging equipment piece $d \in D$ from job site $j \in J$ to job site $j' \in J, j \neq j'$.
c_j	The cost for completing job $j \in J$.
B	The available budget for the planning horizon.
$I(j)$	The *Intensity Function* (pic.dhe.ibm.com) of job $j \in J$. That is $I(j) = 0\%$, if the job j is not allowed to be processed at time t such that $b_w \leq t \leq e_w$, and $I(j) = 100\%$ otherwise.
$TD_{\text{type}_j \text{type}_{j'}}$	The *Transition Distance* between job type $j \in J$ and $j' \in J$. It is used to inform other global constraints that the travel time between job pair j and j' should be at least $t_{jj'}$. Two jobs with the same location have the same type.
Decision Variables:	
y_{jd}	Optional interval variable when job $j \in J$ (with size q_j) is assigned to dredge vessel $d \in D$.
$Y_j = \{y_{j1}, y_{j2}, \ldots, y_{jD}\}$	Array of interval variables representing possible dredge equipment $d \in D$ that can be assigned to job $j \in J$.
$V_d = \{y_{1d}, y_{2d}, \ldots, y_{Jd}\}$	Array of interval variables representing possible jobs $j \in J$ that can be assigned to dredge vessel $d \in D$ (the *interval sequence variable* for d).
z_j	Optional interval variable associated with job $j \in J$.

The CPDFS formulation using the notation in Table 2.3 is presented as follows. Constraint programming formulations are more descriptive comparing with the mathematical formulation we usually see in the linear or mixed-integer programming:

$$\max \sum_{j \in J} q_j \times \text{PresenceOf}(z_j) \times z_j$$

$$\text{Alternative}\left(z_j, Y_j\right) \qquad j \in J \qquad (2.1)$$

$$\text{Cumulative}\left(z_j, c_j, B\right) \qquad (2.2)$$

$$\text{Cumulative}\left(z_j, 1, |D|\right) \qquad (2.3)$$

$$z_j.\text{StartMin} = 1 \qquad j \in J \qquad (2.4)$$

$$z_j.\text{EndMax} = |T| \qquad j \in J \qquad (2.5)$$

$$\text{ForbidExtend}\left(z_j, I(j)\right) \qquad j \in J \qquad (2.6)$$

$$\text{NoOverlap}\left(V_d, TD_{\text{type}_j \text{type}_{j'}}\right) \qquad d \in D \qquad (2.7)$$

The objective function of CPDFS maximizes the total amount of dredging by all the resources. Constraints (2.1) ensure that each job can only be assigned to at most one dredge vessel. Constraint (2.2) imposes that the total cost of dredging operations cannot exceed the total budget B. Constraints (2.3) to ensure that the total number of occupied dredge vessels at any time can not exceed the fleet size $|D|$. Constraints (2.4) and (2.5) set the minimum start time and maximum end time of each job to the first and last day of the planning horizon, receptively. The ForbidExtend Constraint (2.6) prevents job j to be performed at its restricted period(s) $I(j)$. Finally, the *NoOverlap* Constraints (2.7) ensure that if both jobs j and j' are operated by dredge vessel d then a minimum time $t_{jj'}$ (the travel time between jobs j and j') must be maintained between the end of interval variable y_{jd} and the start of the interval variable $y_{j'd}$.

2.6 Implementations and Results

In the following eight sections, the expansion and improvements to the base model of DFSP are discussed individually. In Section 2.6.9 a comprehensive model which incorporates all the modifications is presented. All computational experiments are conducted on real data test instances with the largest one containing 116 dredging jobs, 30 available dredge vessels, and 138 restricted periods over a one year (365 days) time horizon. All problems are modeled in IBM

ILOG CPLEX Optimization Studio 12.3 (pic.dhe.ibm.com) which uses IBM ILOG CPLEX 12.3 to solve MIP and IBM ILOG CP Optimizer 12.3 to solve CP models. All test problems are run on a Core(TM) i7 CPU @ 2.93 GHz, 8 GB RAM computer.

In this Section, in all the results reported in the tables, we have the following common features:

- The base model results are shown in gray highlighted color.
- The solution time is reported in seconds and the dredge, travel, and idle times are in days.
- The dredge time is the total time that dredging vessels spend on operating the jobs.
- Travel time is the sum of the time that each dredging vessel spend traveling between two consecutive jobs for all dredging vessels. The travel time between the origin location and each job's location is not considered in total travel time.
- Idle time is the total time of all dredging vessels being idle. A dredging vessel becomes idle when it finishes a job and travels to another job but cannot start dredging the job because the restricted period(s) of that job will not allow. Therefore, it will stay idle until the restricted period(s) has passed.

The result of implementing the base model for three different sizes of problem instances with real data as described in Section 2.4, are shown in Table 2.4. The small, medium, and large size problem instances have 32 jobs with 30 dredges, 57 jobs with 30 dredges, and 116 jobs and 30 dredges, respectively. These are the basis for comparing the improvement and extensions to the base models in the following sections.

Table 2.4: The Results of the Base Model for the Problem Instances with Real Data

Problem Size	**Instance**	**Obj. Function**	**Sol. Time**	**Dredge Time**	**Travel Time**	**Idle Time**
Small	$\|J\| = 32, \|D\| = 30$	8,413,704	601.1	1,229	335	93
Medium	$\|J\| = 57, \|D\| = 30$	17,090,811	602.2	2,565	961	513
Large	$\|J\| = 116, \|D\| = 30$	30,764,006	609.5	4,759	2,301	571

2.6.1 Partial Dredging

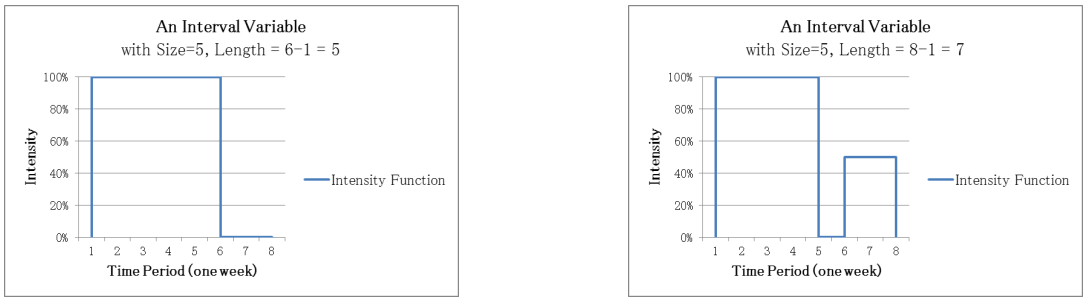
The Corps describes environmental windows as "temporal constraints placed upon the conduct dredging or dredged material disposal operations to protect biological resources or their habitats from potentially detrimental effect" (Dickerson et al., 1998). The scheduling of environmental work windows is intended to minimize environmental impacts by limiting the conduct of dredging activities to the periods when biological resources are not present or are least sensitive to disturbance. Surveys conducted by the Corps indicate that approximately 80% of all O&M dredging jobs are subject to some form of environmental work window constraint, with wide variations across Districts with the Atlantic and Pacific Coast Districts reporting the highest percent of jobs with restrictions (up to 100%) and the Districts in the Gulf of Mexico and Mississippi Valley regions reporting the lowest percentage (less than 20%) (Dickerson et al., 1998). Our data analysis on the restricted periods, summarized in Table 2.1, supports the results of this survey and further analysis shows that 6 dredging jobs out of 116 jobs cannot be conducted in any time of year because of the restricted periods associated with these jobs. For this reason, it is important to be able to consider so-called partial dredging. That is, only dredging a certain percentage of capacity during a restricted period.

In this research, we used partial dredging during restricted periods as the relaxation and studied the impact of relaxing each window individually, as well as collectively. Partial dredging during a restricted period means a dredge vessel can work during the restricted periods, but with a slower operation rate. Because of the possibility of encountering a group of aquatic animals like turtles, fish and other wildlife during the restricted periods, dredging vessels might have to stop dredging to prevent harming them.

To modify the model to allow partial dredging during the restricted periods, the intensity function of the interval variable z_j in Section 2.5.2 is changed and set in the range of $(0\%, 100\%)$ exclusive.

As mentioned in Section 2.5.1, an interval variable represents an interval of time during which something happens. An important feature of interval variables is the fact that they can

be optional, which means they can be *present* in the solution or *absent*. An interval is characterized by a possible *start* and *end* value and a *size*. The *length* of an interval variable is defined as its end value minus its start value (*ibm.com*) which is equal to the size of the interval variable if the intensity function associated with that variable is 100% between its start and end time. The size of an interval variable can be interpreted as the work requirements of the variable. For example, suppose an employee works for 5 days a week full time and does not work on weekends. As shown in Figure 2.4 (a) the intensity function of worker 1 is 100% for the first 5 days of the week and 0% for the weekends. The length of his work ($6 - 1 = 5$) is equal to the size of his work which is 5 man-days in one-week time horizon. On the other hand, worker 2 (Figure 2.4 (b)) works full time in first four days of a week and then take one day off and works half-time on the weekends. His work length is 7 ($8 - 1 = 7$) days, but he delivers 5 man-days as same as worker 1. Likewise, the intensity function of each dredging job can be manipulated to allow for the appropriate amount of partial dredging during the restricted periods for the interval variables z_j.



(a) Worker 1 (b) Worker 2

Figure 2.4: Two Interval Variables with the Same Sizes and Different Lengths

To study the impact of each restricted period on the total amount of dredging (the objective function of CP formulation), the intensity function of all interval variables associated with each restricted period is set to 50% functional. This means the interval variables can be dredged by any vessel at 50% operation rate instead of 100% during the normal working days.

The result of this relaxation is shown in Table 2.5. By allowing dredging in restricted pe-

riods, we should be able to improve the quality of our solution and increase the total cubic yards of dredging with the same available resources and budget restriction. However, this relaxation expands the solution space of the model and consequently might have a negative impact on the results, especially when we relax the restricted periods associated with the jobs that we already dredged in our base model. In Table 2.5, the percentage change from relaxing an individual restricted period versus the full restriction implemented in the base model is provided. As shown in Table 2.5, the largest improvement in objective function (8.7%) is obtained by allowing 50% of dredging during the restricted period number 17 with the range $[91, 334]$ (in the time horizon of a year $[1, 365]$), which is associated with the marine turtle restriction on the dredging job in the LA Calcasieu River Bar Channel.

Table 2.5: Individual Impacts of Restricted Periods Relaxation to 50% Dredging

Relaxed RP	**Obj. Function**	**Sol. Time**	**Dredge Time**	**Travel Time**	**Idle Time**	**Improv.**
5	30,632,870	609	3,217	2,257	934	−0.4%
23	30,632,870	609	3,717	2,213	765	−0.4%
50	30,632,870	609	4,728	1,859	790	−0.4%
32	30,745,673	608	3,807	2,217	767	−0.1%
71	30,760,420	608	3,988	3,210	982	0.0%
⋮	⋮	⋮	⋮	⋮	⋮	⋮
none	30,764,006	610	4,759	2,301	571	0.0%
36	31,325,902	607	5,031	1,738	1,258	1.8%
106	31,402,795	608	4,857	2,270	610	2.1%
58	31,662,392	608	4,977	2,234	961	2.9%
6	32,570,619	608	3,332	2,021	548	5.9%
17	33,433,044	607	5,162	2,464	987	8.7%

Notice, in some cases this additional flexibility improved the objective. In other cases (the first four cases), the flexibility resulted in a search space so large that an inferior solution is obtained in the amount of time allotted. As mentioned, the best solution of the base model with no relaxation is shown in highlighted gray in the table.

We know that the solution of the model with no dredging allowed during the restricted periods (the base model solution) is feasible to the relaxed models with restricted periods that can

be dredged. To overcome the negative impact of expanding our solution space on the objective functions of our relaxed model, we set the base solution as the starting point to the CPDFS with partial dredging model as follows:

Step 1. Solve the base model of CPDFS and store the solution in CP using the CPO built-in "*Store*" method.

Step 2. Set the stating point to the CPDFS with partial dredging during the RPs using the CPO built-in "*setStartingPoint*" method.

The results of setting the base model solution to the relaxed models as the starting point to search in CP is shown in Table 2.6. As we see in the table, there is no negative improvement (last column of Table 2.6) in any of the solutions and all of them are at least as good as the base solution.

Table 2.6: Individual Impacts of RPs Relaxation to 50% with Starting Point

Relaxed RP	**Obj. Function**	**Sol. Time**	**Dredge Time**	**Travel Time**	**Idle Time**	**Improv.**
5	30,764,006	606	4,759	2,118	948	0.0%
23	30,764,006	608	4,759	2,234	706	0.0%
50	30,764,006	609	4,757	2,292	701	0.0%
32	30,764,006	609	4,757	2,292	701	0.0%
⋮	⋮	⋮	⋮	⋮	⋮	⋮
none	30,764,006	610	4,759	2,301	571	0.0%
18	30,779,710	609	4,836	2,526	980	0.1%
71	30,779,710	610	5,184	2,256	685	0.1%
110	30,779,710	611	4,163	2,895	1,046	0.1%
35	30,883,336	608	4,256	2,691	1,089	0.4%
33	30,911,249	608	4,926	2,485	849	0.5%
36	31,255,264	610	4,445	2,462	978	1.6%
40	31,269,833	608	3,755	2,514	1,029	1.6%
37	31,346,116	609	4,028	2,944	765	1.9%
106	31,490,782	611	4,928	2,606	896	2.4%
58	31,550,350	609	4,191	2,606	835	2.6%
6	32,658,606	607	4,318	2,507	1,727	6.2%
17	33,433,044	609	3,935	2,536	690	8.7%

Similar to Table 2.5, the largest improvement in objective function in Table 2.6 is obtained by allowing 50% of dredging during restricted period number 17 by the same amount of 8.7%. Also, setting the base model result as the starting point to the model with relaxing the RP number 71, the objective function is improved by 0.1% which is 15,704 cubic yards.

Another interesting question in studying the impact of restricted periods on the solutions is what happens when we allow partial dredging during all restricted periods. The impacts of all RPs of dredging jobs by allowing to dredge at a different percent of relaxation from 0% to 50% with 5% increment is shown in Table 2.7. In implementing the problem instances, the budget limitation has been removed from the model to focus on the impacts of restricted periods. The total cubic yards of dredging jobs is 48,305,584 cubic yards, which can be obtained by allowing 51% of dredging in all RPs as shown in the last row of Table 2.7 in highlighted green color by removing the budget constraint from the model.

Table 2.7: All RPs Relaxation Impacts on the Objective Function without Budget Limits

Relax.	**Obj. Function**	**Sol. Time**	**Dredge Time**	**Travel Time**	**Idle Time**	**Improv.**
0%	31,145,977	606.5	4,143	1,319	1,233	0.0%
5%	34,504,954	607.2	7,605	2,104	148	10.8%
10%	35,173,379	608.8	8,137	1,848	6	12.9%
15%	39,008,287	608.8	8,195	2,094	0	25.2%
20%	39,585,711	608.0	8,976	1,442	0	27.1%
25%	40,419,016	608.2	9,534	886	11	29.8%
30%	40,419,016	608.4	8,440	962	0	29.8%
35%	40,419,016	610.2	9,170	839	0	29.8%
40%	42,891,619	609.2	9,290	676	161	37.7%
45%	42,891,619	609.3	9,649	642	0	37.7%
50%	42,891,619	608.9	7,748	722	0	37.7%
51%	48,305,584	610.2	9,276	1,075	25	55.1%

Similar to the results in Table 2.6, we set the base model solution as the starting point for the solutions obtained in Table 2.7. However, the results were almost the same as the problem without starting solutions.

Figure 2.5 demonstrates the increment in total cubic yards of dredging from the base model with no dredging to partial dredging of restricted periods at 51%. The improvement in objective

function of total cubic yards of dredging in comparison with the base model (first row of the table) is shown in the last column of the previous table.

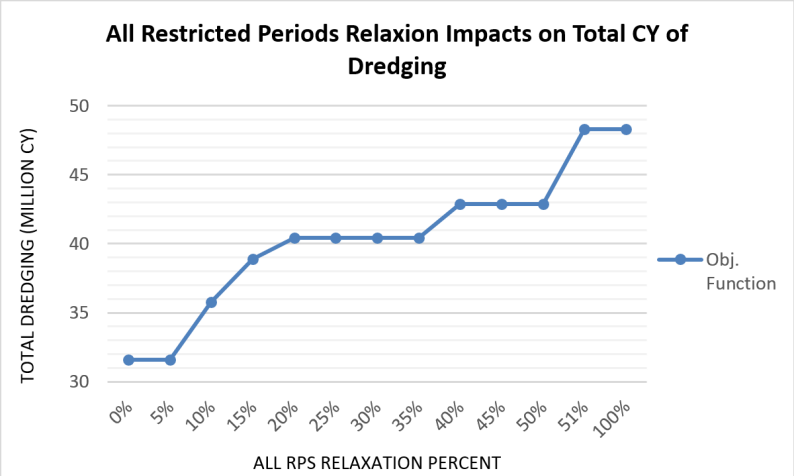


Figure 2.5: All Restricted Periods Relaxation Impacts on Total Cubic Yards of Dredging

Figure 2.6 shows the increment in total dredging time and decrement in total travel time and idle time by increasing the relaxation of dredging in restricted periods from 0% (base model) to 51%.

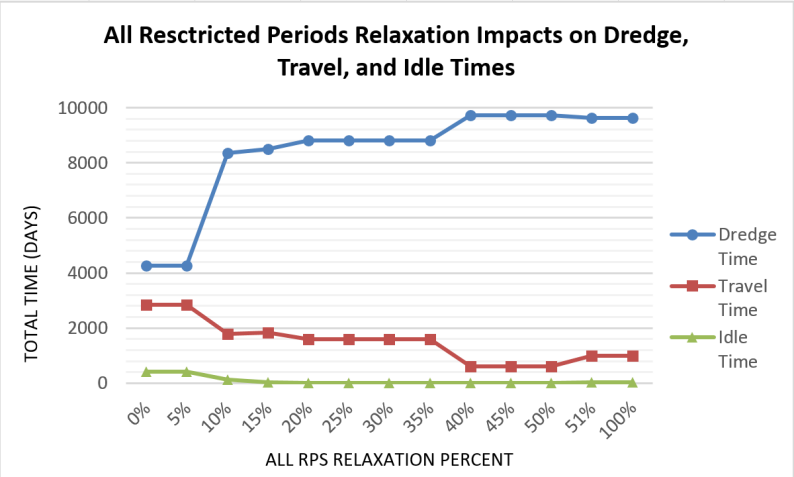


Figure 2.6: All Restricted Periods Relaxation Impacts on Total Dredge, Travel, and Idle Time

As shown in Figure 2.6, by allowing more dredging in the restricted periods the total amount of travel and idle time will decrease and the vessels spend more time on dredging their assigned jobs.

2.6.2 Variable Job Sizes

The CPDFS model algorithm attempts to complete a job before moving on. However, in reality, there is often a "minimum requirement" of dredging that must be met as well as a "target requirement" that would be ideal to achieve if time/money allows. In this context, the size of the jobs is now variable and may be chosen from the range [minimum requirement, target requirement] inclusive. To modify the CPDFS model to have the variable job sizes we add these two parameters and constraints to the model in Section 2.5.2:

Additional parameters:

- h_{jd}, the target requirement of dredging job $j \in J$ using vessel $d \in D$,
- m_{jd}, the minimum requirement of dredging job $j \in J$ using vessel $d \in D$.

Additional constraints:

$$\text{SizeOf}(y_{jd}) \leq h_{jd} \qquad j \in J, d \in D \tag{2.8}$$

$$\text{SizeOf}(y_{jd}) \geq m_{jd} \times \text{PresenceOf}(y_{jd}) \qquad j \in J, d \in D \tag{2.9}$$

Alternative additional constraints:

$$y_{jd}.\text{setSizeMax} = h_{jd} \qquad j \in J, d \in D \tag{2.10}$$

$$y_{jd}.\text{setSizeMin} = m_{jd} \qquad j \in J, d \in D \tag{2.11}$$

Constraints (2.8) ensure that the size of all dredging job $j \in J$ conducted by dredge vessel $d \in D$ remains less than or equal to the target size of the jobs. Similarly, Constraints 2.9 make sure that the size of each dredging job $j \in J$ is greater than or equal to the minimum job size for all available dredge vessels $d \in D$ if the variable y_{jd} is present in our solution. The Constraints (2.10) and (2.10) are alternatives to Constraints (2.10) and (2.10), respectively and impose the same limitations on the size of the jobs in the model. However, the model converges faster with the Constraints (2.8) and (2.9) comparing with (2.10) and (2.11).

As we discussed in Section 2.5.1, an interval variable can be optional. Being optional means these variables can be *present* in the solution or *absent*. If we do not include the PresenceOf(y_{jd}) term in Constraints 2.9, the optional interval variables y_{jd} for all $j \in J, d \in D$ will assign a positive size. If all of the interval variables be presented in the solution, each job $j \in J$ will be conducted by all available dredge vessel $d \in D$ which makes the problem infeasible.

Table 2.8 shows the total amount of dredging and total dredge, travel, and idle time for our base model (gray highlighted row) with the original size of jobs equal to the target requirements and compares it with four other test instances with different ranges of [minimum requirement, target requirement] for all jobs. In the first column of Table 2.8, h is the original size of the jobs in our base model with constant job sizes. In all four test instances, we set the target size of jobs equal to their original size and a fraction of the target size as the minimum requirements of the jobs. For example, the job size range $[0, h]$ means the minimum requirements of all jobs are equal to 0 and their target requirements are equal to the original size of the jobs. The range $[0.25h, h]$ means the minimum requirements of all jobs and their target requirements are equal to 25% of the original sizes (h) and the original size of the jobs, respectively.

Table 2.8: Impact of Having Variable Job Sizes within the Range [min. req., target req.]

Job Size Range	**Obj. Function**	**Sol. Time**	**Dredge Time**	**Travel Time**	**Idle Time**	**Gap%**
h	30,764,006	609.5	4,759	2,301	571	0%
$[0, h]$	12,628,669	611.5	1,376	2,954	2,364	−59%
$[0.25h, h]$	19,188,239	612.4	1,768	1,994	910	−38%
$[0.50h, h]$	22,562,323	610.6	2,734	3,209	1,130	−27%
$[0.75h, h]$	28,508,171	611.6	2,934	2,846	1,096	−7%

As we can see in the Table 2.8, allowing variable job sizes has a negative impact on the objective function of total cubic yards of dredging. This is because the solution space grows significantly. According to the "Gap%" column in the table, the tighter the range of job requirement, the better the solution that can be found by CP Optimizer. As shown in Table 2.8, the gap between the objective function of our base model and the variable job size model decreases from −59% to −7% if we can assess the dredging jobs requirement more precisely. This result sug-

gests that (i) decision-makers should be precise in determining the range of job production and (ii) the optimizer should establish a baseline solution (using the base model) from which CP can begin its search with the constraints added in this section.

To apply the above suggestion (ii) to the CPDFS with variable size model, we can set the best solution of the base model as an initial solution to the model to start the optimization. As we mentioned in Section 2.6.1, first we solve the CPDFS base model and then set it as a starting point to the CPDFS with variable size model to neutralize the negative impact of having variable job size on the objective function as the feasible region expands.

The percentage of the gap between the objective function (total cubic yards of dredging) of the base model and the CPDFS with variable job size and starting point model is compared with the CPDFS with and without this starting point is shown in Figure 2.7. The same problem instances as in Table 2.8 are used in this comparison.

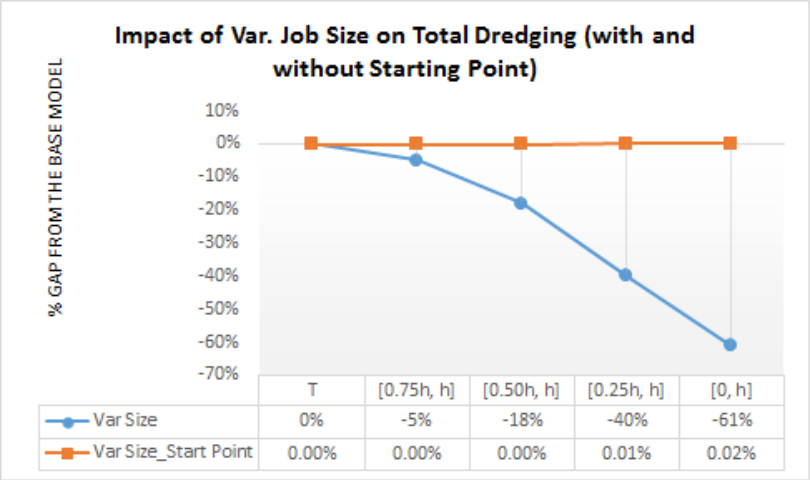


Figure 2.7: Impact of Allowing Variable Job Sizes with and without Starting Point

According to Figure 2.7, setting the base model as the starting point of CP search compensates the negative impact of the feasible region expansion. According to Figure 2.7, a slight improvement in the objective function of the model with start point from the base model is seen (0.02% improvement in job size range $[0, h]$).

2.6.3 Multiple Trips to the Same Job

In the base model, each job must be finished in one visit before encountering its restricted period(s). In practice, some jobs cannot be done within one environmental work window and must be visited again to finish. To modify the base model to allow for multiple trips to the same jobs with the same or different dredge vessels, each job is split into sub-jobs according to the restricted periods of the job and the sub-jobs will be added to the model instead. For example, if job_1 has the restricted period of $I(1) = [152, 274]$, it will be split to two sub-jobs in such a way that sub-job_1 can only take place in the range $[1, 151]$ and sub-job_2 in the range $[275, 365]$, which are the allowable ranges of dredging for job_1. To modify the CPDFS base model in Section 2.5.2, additional sets, variables and constraints are used. Note that similar to Section 2.6.2, the job sizes are also variable in CPDFS with multiple trips model and the CP optimization tool will decide the amount of dredging for each sub-job. We use the following additional sets, variables, and constraints for the CPDFS with multiple trips model.

Additional sets:

- $i \in I_j^c$, set of possible sub-jobs of each dredge job $j \in J$. These sub-jobs can start and finish outside of job $j \in J$ restricted periods, W_j, wherever $I(j) \neq 0$.

Modified variables:

- y_{jid}, optional interval variable when sub-job i of job $j \in J$ is assigned to dredge vessel $d \in D$.
- $Y'_{ji} = \{y_{ji1}, y_{ji2}, \ldots, y_{jiD}\}$, array of interval variables representing possible dredge vessel d that can be assigned to sub-job i of job $j \in J$.
- $V'_d = \{y_{11d}, y_{12d}, \ldots, y_{21d}, y_{22d}, \ldots, y_{J1d}, y_{J2d}, \ldots\}$, array of interval variables representing possible sub-job i of job $j \in J$ that can be assigned to dredge vessel $d \in D$.
- x_{ji}, optional interval variable associated with sub-job i of job $j \in J$.

After we split each job into sub-jobs in the allowed range of dredging for the job, we do not need the ForbidExtend$(z_j, I(j)), \forall j \in J$ constraints because we already set the start and end time of each sub-job to a period of time in which dredging is allowed.

Removed constraints:

$$\text{ForbidExtend}\left(z_j, I(j)\right) \qquad j \in J \tag{2.12}$$

In addition to the Constraints (2.8) and (2.9) in the CPDFS with variable job size model, Constraints (2.13), (2.14) and (2.15) are added to this model.

Modified constraints:

$$\text{Span}(z_j, x_{ji}) \qquad j \in J \tag{2.13}$$

$$\text{Alternative}(x_{ji}, Y'_{ji}) \qquad j \in J, i \in I_j^c \tag{2.14}$$

$$\text{NoOverlap}\left(V'_d, TD_{\text{type}_j \text{type}_{j'}}\right) \qquad d \in D \tag{2.15}$$

Constraints (2.13) state that each interval variable $z_j, j \in J$ spans over all present interval variables from the set $\{x_{j1}, x_{j2}, \ldots\}$. The interval variable z_j starts with the first present interval variable from $\{x_{j1}, x_{j2}, \ldots\}$ and ends with the last in the time horizon. Note that, similar to Section 2.6.2, we still have the constraints for target requirement and minimum requirement of all jobs. The sum of all sub-jobs of job $j \in J$ is still between the range of $[m_j, h_j]$. Similar to Constraints (2.1) and (2.7) of CPDFS base model in Section 2.5.2, Constraints (2.14) and (2.15) are for the assignment of available dredge vessels and the travel time between jobs, respectively. Constraints (2.14) make the CP choose exactly one possible assignment for x_{ji} from all possible assignments. In the scheduling of the dredge fleet, we need to consider the time to travel between two consecutive jobs locations operated by the same dredge, which is handled by Constraints (2.15).

Table 2.9 shows the computational results from allowing multiple trips to the same job model in comparison with the variable job sizes model in Section 2.6.2 for four problem instances with a different size range.

Table 2.9: Impact of Allowing Multi Trips to the Same Job vs. the Base and Var. Job Size Model

Model	Job Size Range	Obj. Function	Sol. Time	Dredge Time	Travel Time	Idle Time	Improv. from Base Model	Improv. from Var. Jobs
Base Model	h	30,764,006	609.5	4,759	2,301	571	0%	-
Multi. Trips	$[0,h]$	25,849,481	624.8	2,842	2,407	2,797	-16%	105%
Multi. Trips	$[0.25h,h]$	29,915,071	622.9	4,124	3,414	2,378	-3%	56%
Multi. Trips	$[0.50h,h]$	27,316,269	615.7	3,631	2,043	2,624	-11%	21%
Multi. Trips	$[0.75h,h]$	29,594,587	618.5	3,725	2,591	2,035	-4%	4%

In in the last two columns of Table 2.9, the results of the CPDFS with multiple trips model are compared with both the base model and the CPDFS with variable job sizes model. As we can see in Table 2.9, the total cubic yards of dredging by allowing multiple trips to the same jobs is about 16% less than our base model in which a dredge vessel could not start a job without finishing it in one visit in its allowed working time windows. The reason of this increment in total cubic yards of dredging is that in the multiple trips model, the size of each job is variable which causes expansion in the solution space. However, we could improve the objective function from 4% up to 105% compared with the variable job sizes model utilizing the job sizes.

Table 2.10 shows the result of the CPDFS with multiple trips model for our three different size problem comparing with the base model by increasing the run time CPO to 30 minutes. In all three sizes of the problem instance, we used the $[0.25h,h]$ for the range of the size of the dredging jobs because it has the best performance for multiple trips among other ranges in Table 2.9. As we can see in the Table 2.10, increasing the CPO run time from 10 minutes to 30 not only overcomes the downside of expanding the feasible solution space, but also gives us better solutions than the base model in small and medium-size problem, with the improvement of 83% and 41%, respectively. In the large size problem with 116 jobs, the results are very close to the base model, and the improvement is noticeable.

Table 2.10: Impact of Allowing Multiple Trips to the Same Job Comparing with the Base Model in 30 minutes Run Time

Model	Instance	Obj. Function	Sol. Time	Dredge Time	Travel Time	Idle Time	Improv.
Base Model	$\|J\| = 32, \|D\| = 30$	8,413,704	1801	1,229	335	93	0%
Multi. Trips	$\|J\| = 32, \|D\| = 30$	15,427,550	1,803	1,577	537	1,307	83%
Base Model	$\|J\| = 57, \|D\| = 30$	17,090,811	1802	2,565	961	513	0%
Multi. Trips	$\|J\| = 57, \|D\| = 30$	24,067,043	1,807	2,377	1,440	1,091	41%
Base Model	$\|J\| = 116, \|D\| = 30$	30,764,006	1809	4,759	2,301	571	0%
Multi. Trips	$\|J\| = 116, \|D\| = 30$	30,380,849	1,821	4,189	3,427	2,242	-1%

The improvement in objective function from Table 2.9 to Table 2.10 by increasing the run time from 10 minutes to 30 minutes motivated us to increase the CPO run to 4 hours. The result is reported in Table 2.11 in which we can see the improvement in all of the problem instances from 7% up to 86% comparing to the base model.

Table 2.11: Impact of Allowing Multiple Trips to the Same Job Comparing with the Base Model by increasing the Run Time to 4 Hours

Model	Instance	Obj. Function	Sol. Time	Dredge Time	Travel Time	Idle Time	Improv.
Base Model	$\|J\| = 32, \|D\| = 30$	8,413,704	1801	1,229	335	93	0%
Multi. Trips	$\|J\| = 32, \|D\| = 30$	15,648,372	14,402	1,888	486	1,216	86%
Base Model	$\|J\| = 57, \|D\| = 30$	17,090,811	1802	2,565	961	513	0%
Multi. Trips	$\|J\| = 57, \|D\| = 30$	24,375,873	14,406	2,646	1,247	1,832	43%
Base Model	$\|J\| = 116, \|D\| = 30$	30,764,006	1809	4,759	2,301	571	0%
Multi. Trips	$\|J\| = 116, \|D\| = 30$	32,866,861	14,425	4,171	3,091	2,218	7%

The results of increasing the run time from 10 to 30 minutes and up to 4 hours on the total cubic yards of dredging in CPDFS with multiple trips model from Table 2.10 and Table 2.11 are summarized in Figure 2.8.

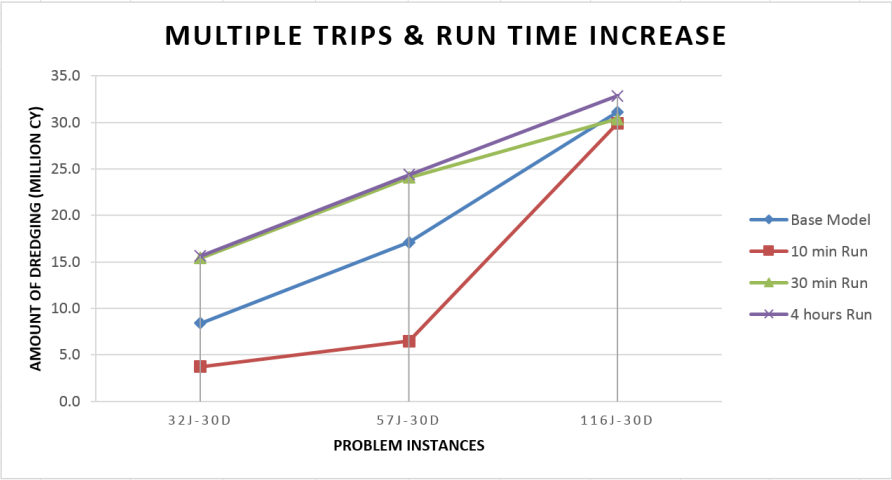


Figure 2.8: Impact of Increasing Run Time on Dredging in Multiple Trips to the Same Jobs Model

As we can see in Figure 2.8, by running the model for 10 minutes the amount of dredging for all problem instances are below the base model. Increasing the run time to 30 minutes allows more dredging than the base model in the small and medium size instances with 32 jobs and 30 dredges and 57 jobs and 30 dredges respectively, but not in the large instance with 116 jobs and 30 dredges. Put simply, the ability to visit sites multiple times yields extensive benefit to the US-ACE especially when variable job size is not included.

2.6.4 Multiple Dredges on the Same Job

The ability to dredge a job with multiple dredges in the model can speed up the dredging process of jobs and maximize the utilization of available dredge vessels. Moreover, in practice, some jobs need to be dredged by multiple dredge vessels depending on the geographic characteristics of the job's location, depth, and wideness of the waterways. For example, Clearwater Harbor, Florida typically needs Yaquina, Essayons, and contract dredges.

To model the multiple dredge extension, we consider three scenarios. In all three the size of the problem will increase comparing to the base model and accordingly solving the problem become harder and more time-consuming. In the first scenario, the dredges can work separately on the job. This scenario would be easy to handle. Such dredging jobs can be split into additional

jobs with the associated location, size, and compatible dredge vessel type. Second scenario happens if all different types of dredge vessels can work together simultaneously i.e., start and end the dredging job at the same time. In this situation, we can modify the Constraints (2.1) in the CPDFS base model (Section 2.5.2) to make them compatible. The modified constraints are as follows:

Modified constraints:

$$\text{Alternative}(z_j, Y_j, c) \qquad j \in J \tag{2.16}$$

Constraints (2.16) enforce each job $j \in J$ be assigned to exactly c dredge vessels from all dredge vessels compatible with the job. The additional parameter c in *Alternative* global constraints is needed to be sure that, if an interval decision variable z_j is present in the solution, then exactly c elements of Y_j must be presented in the solution and all of them start and end together.

The third scenario and the most challenging form of multiple dredges working on the same job is when dredging can happen by different type of dredges with different start and finish times. To address this case in our model, similar to Section 2.6.3, each job is divided into some sub-jobs with variable job sizes. In the most flexible and computationally expensive case when any type of dredge can work on any dredging jobs, each job is divided to the number of available dredge vessels ($|D|$) to maintain the possibility of working all dredge vessels on each job. After solving the problem with the CP optimization tool, the start and end time, size of each sub-job and the dredge vessel assigned to each sub-job will be determined. Similar to our base model (with one dredge vessel on each job), there is a travel distance between each sub-job and it is equal to 0 if two consecutive sub-jobs are dredged by the same dredge vessel. The total size of each job, which is the summation of all its sub-jobs, must be between the minimum and the target requirement size of each job. As we divide each job into the number of available dredge vessels sub-jobs, the following modifications to the sets, variables, and constraints need to be made (similar to Section 2.6.3):

Additional Set:

- $k \in D$, set of dredging sub-jobs that need to be completed over the planning horizon (each

job is divided to $|D|$ sub-jobs).

Modified and Additional Variables:

- y_{jkd}, optional interval variable when sub-job $k \in D$ of job $j \in J$ is assigned to dredge vessel $d \in D$.
- $Y'_{jk} = \{y_{jk1}, y_{jk2}, \ldots, y_{jkD}\}$, the array of interval variables representing possible dredge vessel d that can be assigned to sub-job $k \in D$ of job $j \in J$.
- $V'_d = \{y_{11d}, y_{12d}, \ldots, y_{1kd}, \ldots, y_{1Dd}, y_{21d}, y_{22d}, \ldots, y_{2kd}, \ldots, y_{2Dd}, \ldots, y_{J1d}, y_{J2d}, \ldots, y_{Jkd}, \ldots, y_{JDd}\}$, the array of interval variables representing possible sub-job $k \in D$ of job $j \in J$ that can be assigned to dredge vessel $d \in D$.
- x_{jk}, optional interval variable associated with sub-job $k \in D$ of job $j \in J$.

In addition to the Constraints (2.8) and (2.9) in Section 2.6.2, the following constraints are added to this model.

Modified and Additional Constraints:

$$\text{Span}(z_j, x_{jk}) \qquad j \in J, k \in D \qquad (2.17)$$

$$\text{Alternative}(x_{jk}, Y'_{jk}) \qquad j \in J, k \in D \qquad (2.18)$$

$$\text{NoOverlap}\left(V'_d, TD_{\text{type}_j \text{type}_{j'}}\right) \qquad d \in D \qquad (2.19)$$

$$\sum_{k \in D} \text{SizeOf}(x_{jk}) \leq h_j \qquad j \in J \qquad (2.20)$$

$$\sum_{k \in D} \text{SizeOf}(x_{jk}) \times \text{PresenceOf}(x_{jk}) \geq m_j \qquad j \in J \qquad (2.21)$$

Constraints (2.17) state that each interval variable $z_j, j \in J$ spans over all present intervals variables from the set $\{x_{j1}, x_{j2}, \ldots, x_{jD}\}$. As mentioned in Section 2.6.3, the interval variable z_j starts with the first present interval from $\{x_{j1}, x_{j2}, \ldots, x_{jD}\}$ and ends with the last one. Similar to Constraints (2.14) and (2.15), Constraints (2.18) and (2.19) are for the available dredge vessels assignment to the jobs and travel time between jobs enforcement, respectively. Constraints

(2.20) impose a limitation on the total size of each job, which is the summation of the sizes of its sub-jobs. Note that, similar to Section 2.6.2, the size of each sub-job is variable and between the minimum and target requirements.

The ability of multiple dredges to work on the same job in our model is the most flexible yet complex improvement in our base model. In this model, the dredge vessels can have multiple trips to the same jobs and dredge any portion of the total size of each job during each visit. By splitting each job into the number of available dredge vessels, the number of variables and constraints increases enormously and accordingly the solution space will expand dramatically. To help the CP optimization tool to find feasible solutions, we removed the budget constraint and set the range of $[0.25h, h]$ for the job sizes. Without this, CP could not find any feasible solution even for the problem instance with 10 jobs and 5 dredge vessels after 4 hours time limit. An instance with the largest number of jobs and dredges that could be solved by CP on a Core(TM) i7 CPU @ 2.93 GHz, 8 GB RAM desktop computer was with 57 jobs and 15 dredges. A problem instance with 57 jobs and 20 dredges ran for 4 hours on the same computer without finding any feasible solutions.

Table 2.12 shows the computational result of allowing multiple dredge vessels working on the same job in comparison with our base model (with at most one dredge assigned to each job). The test problems have 10, 32, and 57 jobs and 5, 10 and 15 dredge vessels with associated RPs. All problem instances ran for 4 hours or 14,400 seconds.

Table 2.12: Impact of Allowing Multiple Dredges Working on the Same Jobs.

Model	Instance	Obj. Function	Sol. Time	Dredge Time	Travel Time	Idle Time	Improv.
Base Model	$\|J\| = 10, \|D\| = 5$	2,487,676	0.2	347	137	82	0%
Multi. Dredges	$\|J\| = 10, \|D\| = 5$	6,743,828	61	735	543	313	**171%**
Base Model	$\|J\| = 32, \|D\| = 10$	8,413,704	3,819	864	780	595	0%
Multi. Dredges	$\|J\| = 32, \|D\| = 10$	18,922,824	14,409	1,873	1,356	398	125%
Base Model	$\|J\| = 32, \|D\| = 15$	8,413,704	3,672	909	588	369	0%
Multi. Dredges	$\|J\| = 32, \|D\| = 15$	18,150,677	14,451	1,667	2,134	1,119	116%
Base Model	$\|J\| = 32, \|D\| = 20$	8,413,704	3,645	482	836	336	0%
Multi. Dredges	$\|J\| = 32, \|D\| = 20$	10,890,124	14,535	1,114	2,832	1,625	29%
Base Model	$\|J\| = 32, \|D\| = 30$	8,413,704	3,621	1,101	399	156	0%
Multi. Dredges	$\|J\| = 32, \|D\| = 30$	9,469,173	16,401	1,004	2,467	1,251	13%
Base Model	$\|J\| = 57, \|D\| = 10$	13,044,882	3,647	1,176	1,026	351	0%
Multi. Dredges	$\|J\| = 57, \|D\| = 10$	21,371,547	14,439	2,191	1,086	257	64%
Base Model	$\|J\| = 57, \|D\| = 15$	18,093,069	3,621	1,834	717	978	0%
Multi. Dredges	$\|J\| = 57, \|D\| = 15$	12,536,703	14,599	1,327	2,204	1,499	**-31%**
Base Model	$\|J\| = 57, \|D\| = 20$	18,093,069	3,611	1,932	1,066	989	0%
Multi. Dredges	$\|J\| = 57, \|D\| = 20$	no solution	14,400	-	-	-	-%

In this model, by having variable job sizes and multiple dredges working on same jobs, there is a trade-off between having more flexibility in assigning dredge vessels to dredging jobs and expanding the solution space enormously. Flexibility gives us more opportunity to dredge more jobs and for each job more cubic yards.

As we can see in the Table 2.12, allowing multiple dredges to work on the same jobs has positive impacts (up to 171%) on the objective function (total cubic yards of dredging) for small instances and negative impacts (-31%) on the large instances (unlike the base model, in the expanded model the instance with 57 jobs and 15 dredges is considered to be a large instance instead of a medium size instance because the number of jobs that need to be scheduled is $57 \times 15 = 855$ using 15 dredges). In Figure 2.9, the impact of having multiple dredges on the same job model on the total cubic yards of dredging as reported in Table 2.12 is shown.

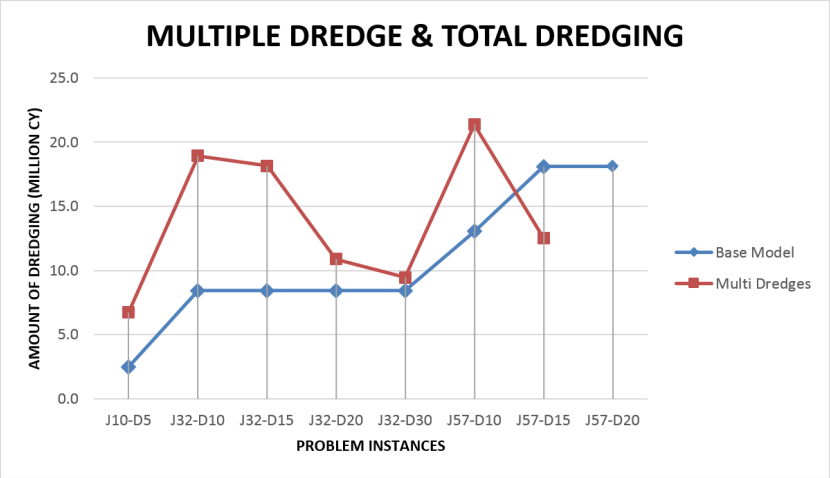


Figure 2.9: Impact of Having Multiple Dredge on the Same Job Model

As shown in Figure 2.9, having multiple dredges working on the same job increases the total cubic yards of dredging in the small and medium size instances from the problems with 10 jobs and 5 dredges to the problem with 57 jobs and 10 dredges.

2.6.5 Different Operation Rates and Unit Cost of Dredging

The operation rate of each dredge vessel can vary greatly from one job to another due to wave conditions, weather and sediment types. To modify the base model to work with different operation rates for each vessels in each job, the parameter r_d in Section 2.5.2 is changed to $r_{jd}, \forall j \in J, d \in d$ and accordingly the parameter t_{jd}, the time (days) required for dredge vessel d to complete job j, will be changed to $t_{jd} = \lceil q_j / r_{jd} \rceil, \forall j \in J, d \in D$. The variables and constraints of the base model will remain unchanged.

Separately, the cost of dredging per cubic yard can vary from one dredge vessel to another due to the size of dredge vessels, different types of equipment they use and the crew size of each dredge vessel. Similar to having different operation rates for each job, we can change the parameter c_j, the cost for completing job j, in Section 2.5.2 to $c_{jd}, \forall j \in J, d \in D$, the cost for completing job j by dredge vessel d.

Finally, in addition to the overall budget constraint on the total cost of all jobs in the base model, we impose a limitation on the cost of each job to not exceed the available budget for each

job. The budget parameters and constraints for each job are as follows:

Additional parameters:

- b_j, the available budget for the job $j \in J$.

Additional constraints (individual job budget constraints):

$$\text{PresenceOf}(y_{jd}) \times q_j \times c_{ij} \leq b_j \qquad j \in J, d \in D \tag{2.22}$$

Constraints (2.22) ensure that if job j is performed by dredge vessel d, the cost of dredging ($q_j \times c_{ij}$) will not exceed the available budget for job j. The result of having different operation rates and cost of dredging for each job-dredge combination is shown in Table 2.13.

Table 2.13: Impact of Having Different Operation Rates and Cost of Dredging for each Job

Model	Instance	Obj. Function	Sol. Time	Dredge Time	Travel Time	Idle Time
Different Rates	$\|J\| = 32, \|D\| = 30$	8,413,704	601.7	1,101	399	156
Different Rates	$\|J\| = 57, \|D\| = 30$	14,907,474	603.0	2,334	810	812
Different Rates	$\|J\| = 116, \|D\| = 30$	25,141,975	605.8	4,498	1,104	1,520

In Table 2.13, the result of running the base model and the model with different rates of operation is reported. In Table 2.13, we are not comparing the base model with the model with different rates because the data structure has been changed and operation rates of each dredge vessel are different from one job to another.

2.6.6 Stimulating Downtime for Dredges

To stimulate the downtime in the model for each dredge on the assigned job(s), as mentioned in Section 2.6.1, the intensity function of the interval variable associated with the specified dredge and all the jobs is changed and set equal to 0%. The interval variables associated with each job-dredge pair exist in the set of variables Y_j in the CP formulation presented in Section 2.5.2. By setting the intensity function of these variables equal to 0%, we tell the CP model that during the

downtime period the dredge cannot work at all. We add the following parameters to the model to specify the begin and end of the downtime for each dredge.

Additional parameters:

- a_d, the begin of downtime of dredge d.
- b_d, the end of downtime of dredge d.

All that remains is to set the intensity function of the interval variable y_{jd} between the period $[a_d, b_d]$ equal to 0%, $\forall j \in J$ using the CPO build-in "*setIntensity*" method.

2.6.7 Mob/Demob Cost

The two standard ways of calculating the mobilization and demobilization costs that actually used in the Corps is to split the cost among all jobs who use the contracted dredge. They prorate the costs according to the respective cubic yard of dredging at each job or the amount of travel time/distance for each dredge. Each approach is explained in the following sections.

2.6.7.1 Mob/demob cost based on cubic yards of dredging

In this case we must track the size of all jobs that are performed by a dredge d during the optimization process of CPO. This is done by using SizeOf(y_{jd}) as discussed in Section 2.6.2. We can add the following constraint to the model to record the total amount of dredging performed by dredge d on all the jobs.

Additional Constraints:

$$\text{Cumulate}_{j \in J}(\text{SizeOf}(y_{jd})) \qquad d \in D \tag{2.23}$$

After calculating the total amount of dredging for each dredge, note as CY_d, we can easily specify the proportion of the mob/demob cost for each dredge d, mob_d, using the following equation:

$$\text{mob}_d = \text{mob}_{\text{total}} \times \frac{CY_d}{CY_{\text{total}}} \qquad d \in D \tag{2.24}$$

2.6.7.2 Mob/demob cost based on travel time/distance

In the second case, in which we are splitting the mob/demob cost among contract dredges, we need to know how much each dredge travels between jobs. This is possible by taking advantage of the CP optimizer's ability to formulate a sequence depending setup cost using the TypeOfNext() built-in function. We must add Constraint 2.25 to get the total travel time/distance of each dredge d.

Additional Constraints:

$$\text{Cumulate}_{j \in J}(TD[\text{Type}(j), \text{TypeOfNext}(Y_d, j)]) \qquad d \in D \tag{2.25}$$

Similar to the first case, after calculating the total travel time/distance for each dredge, TR_d, we use the following formula to get the proportion of the mob/demob cost for each dredge d, mob_d.

$$\text{mob}_d = \text{mob}_{\text{total}} \times \frac{TR_d}{TR_{\text{total}}} \qquad d \in D \tag{2.26}$$

2.6.8 Dredge Capabilities to Perform Jobs

In some cases, a particular dredge may not be able to operate on some jobs. To prevent assigning dredge d to job j, we can use two different methods. The first method is to set the intensity function associated with interval variable y_{jd} equal to 0% as we mentioned in Sections 2.6.1 and 2.6.6. In the second method, we set the operation rate $r_{jd} = 0$ as shown in Section 2.6.5. This means that there is not any increase in objective function, so the CP optimizer will not consider

such an assignment.

2.6.9 Comprehensive Model

In this section, the modified CPDFS comprehensive model, MCPDFS, in which all the modifications that we discussed in Section 2.6 are presented in a comprehensive formulation. In addition to the notation in Table 2.3 in Section 2.5.2, the following parameters and variables shown in Table 2.14 are used in the MCPDFS formulation. The modified comprehensive model referred to as MCPDFS is formulated using the notation in Table 2.14.

Table 2.14: Notation of MCPDFS Formulation in Addition to CPDFS in Table 2.3

Notation	**Description**
Sets:	
$k \in K$	Set of dredging sub-jobs that need to be completed over the planning horizon (each job is divided into $\|K\|$ sub-jobs).
Parameters:	
h_j	The target requirement of dredging job $j \in J$.
m_j	The minimum requirement of dredging job $j \in J$.
h_{jd}	The target requirement of dredging job j using vessel d.
m_{jd}	The minimum requirement of dredging job j using vessel d.
r_{jd}	The operation rate (cubic yards/day) of dredge equipment $d \in D$ conducting job $j \in J$.
t_{jd}	The time (days) required for dredge vessel d to complete job j.
c_{jd}	The cost for completing job $j \in J$ by dredge vessel $d \in D$.
Decision Variables:	
y_{jkd}	Optional interval variable when sub-job $k \in D$ of job $j \in J$ is assigned to dredge vessel $d \in D$.
$Y'_{ji} = \{y_{ji1}, y_{ji2}, \ldots, y_{jiD}\}$	The array of interval variables representing possible dredge vessel $d \in D$ that can be assigned to sub-job i of job $j \in J$.
$V'_d = \{y_{11d}, y_{12d}, \ldots, y_{21d}, y_{22d}, \ldots, \vdots, y_{J1d}, y_{J2d}, \ldots, y_{JKD}\}$	The array of interval variables representing possible sub-job i of job $j \in J$ that can be assigned to dredge vessel $d \in D$.
x_{jk}	Optional interval variable associated with sub-job $k \in D$ of job $j \in J$.

$$\max \sum_{j \in J} q_j \times \text{PresenceOf}(z_j) \times z_j$$

$$\text{Span}(z_j, x_{jk}) \qquad j \in J, k \in D \qquad (2.27)$$

$$\text{Alternative}(x_{jk}, Y'_{ji}) \qquad j \in J \qquad (2.28)$$

$$\text{NoOverlap}\left(V'_d, TD_{\text{type}_j \text{type}_{j'}}\right) \qquad d \in D \qquad (2.29)$$

$$\text{Cumulative}\left(z_j, c_{jd}, B\right) \qquad (2.30)$$

$$\text{Cumulative}\left(z_j, 1, |D|\right) \qquad (2.31)$$

$$z_j.\text{StartMin} = 1 \qquad j \in J \qquad (2.32)$$

$$z_j.\text{EndMax} = |T| \qquad j \in J \qquad (2.33)$$

$$\sum_{i \in D} \text{SizeOf}(x_{ji}) \leq h_j \qquad j \in J \qquad (2.34)$$

$$\text{SizeOf}(y_{jd}) \leq h_{jd} \qquad j \in J, d \in D \qquad (2.35)$$

$$\text{SizeOf}(y_{jd}) \geq \text{PresenceOf}(y_{jd}) \times m_{jd} \qquad j \in J, d \in D \qquad (2.36)$$

Constraints (2.27) state that each variable $z_j, j \in J$ spans over all present intervals variables from the set $\{x_{j1}, x_{j2}, \ldots, x_{jK}\}$. The interval variable z_j starts with the first present interval from $\{x_{j1}, x_{j2}, \ldots, x_{jK}\}$ and ends with the last one. Constraints (2.28) are included for the assignment of jobs to available dredge vessels. Constraints (2.29) are in the model for setting the travel time, $TD_{\text{type}_j \text{type}_{j'}}$, between two consecutive sub-jobs j and j' that are conducted by the same dredge vessel d. Constraint (2.30) imposes that the total cost of dredging job $j \in J$ by dredge vessel $d \in D$ with the cost of c_{jd} cannot exceed the total budget B. Also, Constraint (2.31) makes sure that the total number of occupied dredge vessels at any time does not exceed the fleet size $|D|$. Constraints (2.32) and (2.33) set the minimum start time and maximum end time of each job to the first and last day of the planning horizon, respectively. Constraints (2.34) impose a limitation on the total size of each job, which is the summation of the sizes of its sub-jobs. Note that the

size of each sub-job is variable and is less than or equal to the target size (cubic yards of dredging) and greater than or equal to the minimum size of each job. Constraints (2.35) and (2.36) ensure that the size of all dredging job $j \in J$ conducted by dredge vessel $d \in D$ remain between the minimum and the target size of the job jm if the variable y_{jd} is present in the solution.

2.7 Impacts of Implementation

The impact of the implementations in this work can be measured quantitatively, as was shown in the chapter. However, of equal importance is the impact of this work on the future of decision analysis within USACE. After initial success with the base model presented at the beginning of this report (see Nachtmann et al. (2014)), maritime professionals were intrigued by the use of operations research to aid in their decision process. However, the potential of the initial tool was met with concern over the fact that many realistic components were not considered. For example, in the newly developed tool in this chapter we can now model different operation rates and unit costs for each project and dredging vessel pair, simulation of downtime for dredges, inclusion of Mob/Demob cost in the budget, dredge capability to perform assigned projects and multiple dredges/trips on/to the same job. The main contribution of this chapter is that several concerns presented by USACE have now been addressed from a modeling perspective. The decision makers now understand that optimization tools can be flexible and with the appropriate amount of attention, complex challenges can be modeled.

2.8 Conclusion

This work has offered a highly generalized dredge scheduling optimization framework for use by dredge planners. The work has already been transferred to USACE computing systems and various versions of the developed model have been utilized in support of planning efforts on the West and East coast. The results of the job show that partial dredging, dredge maintenance, modified mob/demob costs/budgets, operations rates, multiple dredges per job and multiple visits to jobs can all be allowed for in a constraint programming platform. Using this platform, feasible

solutions can be obtained to this complex model in a matter of minutes or hours. Evaluating the potential benefit on cubic yards dredged by considering each model enhancement suggests that these new flexibilities are significant for guiding practitioners to solutions. That is, adding the discussed flexibilities to the models makes a significant difference in the solutions obtained.

With a more flexible model and the increased potential for significant cubic yards dredged gains comes a new set of computational challenges. In addition to revealing how to model additional problem features, this chapter has revealed a number of new methodological challenges that need to be explored. That is, with increased flexibility comes a much larger solution space for any optimization methodology to explore. While one solution to this problem is to use the solution from a simplified model as a seed solution to the more complex model, more sophisticated approaches are certainly worthy of exploration. In the course of studying these issues, the investigators note that many aspects of the expanded problem formulation (e.g., schedule of an individual dredge) decompose nicely. That is, there are components of the scheduling problem that can be thought of in separate pieces. The acknowledgment of this fact leads the investigators to believe that opportunities to implement the existing constraint programming approach in a parallel computing system could yield immediate solution improvements. Moreover, the complexities of the new problem suggest that it is now appropriate to formally study the parameters utilized in the constraint programming search.

Bibliography

Allahverdi, A., Ng, C., Cheng, T. E., and Kovalyov, M. Y. (2008). A survey of scheduling problems with setup times or costs. European journal of operational research, 187(3):985–1032.

Arkin, E. M. and Silverberg, E. B. (1987). Scheduling jobs with fixed start and end times. Discrete Applied Mathematics, 18(1):1–8.

Cakici, E. and Mason, S. (2007). Parallel machine scheduling subject to auxiliary resource constraints. Production Planning and Control, 18(3):217–225.

Dickerson, D. D., Reine, K. J., and Clarke, D. G. (1998). Economic impacts of environmental windows associated with dredging operations. Technical report, DTIC Document.

Eliiyi, D. T. and Azizoğlu, M. (2009). A fixed job scheduling problem with machine-dependent job weights. International Journal of Production Research, 47(9):2231–2256.

fs.usda.gov. The threatened, endangered and sensitive (tes) species program. http://www.fs.usda.gov/detail/r1/plants-animals/?cid=stelprdb5130525. Accessed: October 2015.

Gedik, R., Rainwater, C., Nachtmann, H., and Pohl, E. A. (2016). Analysis of a parallel machine scheduling problem with sequence dependent setup times and job availability intervals. European Journal of Operational Research, 251(2):640–650.

Jain, V. and Grossmann, I. E. (2001). Algorithms for hybrid milp/cp models for a class of optimization problems. INFORMS Journal on computing, 13(4):258–276.

Mönch, L., Fowler, J. W., Dauzère-Pérès, S., Mason, S. J., and Rose, O. (2011). A survey of problems, solution techniques, and future challenges in scheduling semiconductor manufacturing operations. Journal of Scheduling, 14(6):583–599.

Nachtmann, H., Mitchell, K., Rainwater, C., Gedik, R., and Pohl, E. (2014). Optimal dredge fleet scheduling within environmental work windows. Transportation Research Record: Journal of the Transportation Research Board, (2426):11–19.

navigationdatacenter.us. Usace dredging information system. http://www.navigationdatacenter.us/dredge/dredge.htm. Accessed: July 2015.

Pearn, W., Chung, S., Yang, M., et al. (2002). The wafer probing scheduling problem (wpsp). Journal of the Operational Research Society, 53(8):864–874.

pic.dhe.ibm.com. Ibm ilog cplex optimization studio v12.3, 2011. http://pic.dhe.ibm.com/infocenter/cosinfoc/v12r3/index.jsp. Accessed: October 2013.

Resources, W. Inland waterway navigation: Value to the nation. http://www.corpsresults.us/. Accessed: July 2015.

Rojanasoonthon, S. and Bard, J. (2005). A grasp for parallel machine scheduling with time windows. INFORMS Journal on Computing, 17(1):32–51.

ibm.com. Ibm ilog cplex optimization studio cp optimizer user manual version 12 release 6. http://www-01.ibm.com/support/knowledgecenter/SSSA5P_12.6.1/ilog.odms.studio.help/pdf/usrcpoptimizer.pdf. Accessed: January 2015.

Appendix

Appendix 2.A Dredging Projects and Vessels Characteristics

Table 2.15: 116 Project Properties (volumes: CY, costs: USD)

Job ID	Volume	Cost	Job ID	Volume	Cost
000030	439,726	3,201,839	011810	577,711	2,972,600
000360	900,709	5,533,068	011860	156,607	1,104,938
046063	4,376	46,441	011880	30,523	420,827
074955	2,267,192	14,477,345	012030	544,338	2,338,424
000950	466,950	2,989,574	012550	123,064	9,739,760
001120	2,001,129	2,523,736	008190	174,603	998,309
088910	39,308	1,016,772	072742	26,937	644,784
010222	178,088	791,822	012801	67,578	318,000
076060	451,796	1,261,920	012990	217,888	967,081
080546	6,723	275,719	073567	34,637	302,055
002080	2,472,603	6,685,844	013080	723,937	2,628,970
002250	102,032	1,242,273	013330	44,401	334,654
041015	85,093	2,409,673	013590	119,668	1,891,959
003630	277,836	786,758	013680	1,193,406	2,009,923
002440	2,890,491	3,793,482	013880	252,670	251,296
002410	179,782	1,612,871	013940	192,277	980,108
002620	116,357	2,307,509	014310	82,949	748,816
002640	396,079	909,977	076031	46,686	481,990
014360	5,413,965	5,452,500	014370	4,510	102,371
008160	67,221	1,231,600	021530	26,009	144,042
003130	13,252	226,709	014760	59,003	690,963
076106	35,672	321,356	015100	572,395	2,405,442
022140	45,533	142,900	015280	95,491	723,544
003600	808,778	1,502,833	015600	21,003	178,236
003840	397,516	1,745,287	087072	83,378	146,508
004550	243,898	1,489,330	087455	32,688	453,483
004610	38,598	306,499	015870	295,967	1,881,768
004710	201,116	1,122,792	057420	231,639	1,709,816
004800	117,090	719,437	016130	833,305	2,509,084
005050	80,528	733,469	076063	120,808	900,546
005220	191,015	1,708,370	074709	145,537	942,239
005700	261,440	1,058,165	016550	261,985	1,363,696
005880	1,117,205	9,124,564	067318	127,064	310,965
041016	63,380	2,260,932	073644	572,249	4,008,166
006260	186,551	1,183,650	016800	216,709	864,890
006480	668,425	2,073,745	016860	47,674	284,901
006670	41,563	311,454	017180	22,153	159,881
006770	577,424	1,543,516	017370	306,546	5,944,930

Continued on the next page

Table 2.15 – continued from previous page

Job ID	Volume	Cost	Job ID	Volume	Cost
006910	147,811	2,153,095	074390	633,833	8,574,738
007150	1,038,304	1,534,705	017300	64,118	1,162,671
007610	42,408	283,559	017350	42,577	389,861
007810	167,704	1,416,099	017380	49,558	2,497,492
007860	1,494,596	4,048,374	017760	64,262	950,325
008410	1,189,684	12,991,774	017720	212,214	1,588,367
054000	225,664	1,427,334	017960	1,037,987	4,895,841
008430	283,367	1,151,256	073598	229,090	456,000
010020	67,571	380,810	018710	55,762	326,262
010040	80,000	1,579,250	018750	105,955	443,959
074719	122,930	864,000	024190	1,086,812	1,486,174
010310	102,424	751,304	019550	97,935	442,630
010490	74,288	519,202	019560	50,777	331,749
010580	261,769	1,845,812	039023	9,868	66,150
011060	59,190	419,900	019990	53,971	258,289
011270	40,729	530,127	020040	323,758	1,262,279
000068	681,961	1,419,778	020030	1,171,297	6,527,537
011410	944,417	1,496,737	072852	33,939	4,687,087
000063	1,505,100	5,388,149	020290	75,373	468,695
011670	1,282,956	2,509,501	073803	561,192	2,499,452
			Total:	48,305,584	223,012,020

Table 2.16: Production Rates of Dredge Vessels (cubic yards/day)

Dredge ID	Rate
01	1,238
02	1,301
03	1,637
04	1,962
05	1,989
06	2,296
07	2,375
08	2,709
09	2,855
10	3,311
11	3,481
12	3,728
13	3,941
14	4,532
15	5,941
16	6,837
17	6,965
18	8,332
19	8,443
20	9,007
21	10,436
22	10,478
23	10,959
24	12,347
25	12,882
26	15,556
27	17,080
28	17,282
29	17,537
30	19,245

Appendix 2.B Certification of Student Work

UNIVERSITY OF ARKANSAS

College of Engineering
Department of Industrial Engineering

Date: December 5, 2017

Graduate School
University of Arkansas

Dear Dr. Needy:

I am writing to verify that Fereydoun Adbesh completed more than 51% of the work for the chapter titled “Improvements and Enhancements to the Dredge Fleet Scheduling Problem” in his dissertation.

Sincerely,

Chase Rainwater
cer@uark.edu
479-575-2687
Associate Professor
Department of Industrial Engineering
University of Arkansas

3. Vehicle Routing Problems with Hours of Service Regulations for Trucking Industry

Abstract: In this research, we study variants of the Vehicle Routing Problem (VRP) facing trucking providers that serve the transportation logistics industry. We formulate a new VRP variant that adheres to working hour regulations for truck drivers in both single and multiple work shift settings. Solution techniques are proposed that are presented as a single-point decision tool called the Enterprise Transportation Planning (ETP). We perform computational experiments on the pickup and delivery capacitated VRP with backhauls, time windows, and working hour regulations. The results confirm the strength of ETP in finding quality solutions in a very short amount of time in comparison with those obtained via a black box solver.

3.1 Introduction

Each day transportation logistics providers seek to deliver thousands of shipments to their destinations using hundreds of trucks. This problem is complicated by the facts that trucks have different capacities and each destination can be bound by differing delivery time windows restrictions. Also, truck drivers needed to fulfill these deliveries must be scheduled in accordance with work hour restrictions imposed through government regulations. Companies, like J.B. Hunt, have provided services under constraints similar to these for thousands of customers over the last decades. However, the decisions made to satisfy customer demand have either utilized disaggregated quantitative models or qualitative approaches guided by experiential expertise. This chapter seeks to answer delivery decision to this complicated problem via the study of a generalized variant of the so-called Vehicle Routing Problem (VRP).

The goal of VRP is to obtain the best routes for a fleet of vehicles in which all locations are visited and all vehicles return to the origin depot. The objective is to minimize the total cost of traveling between locations while adhering to the vehicles' capacity constraints. VRP was first introduced by Dantzig and Ramser (1959). Over the past five decades, many researchers have

studied several variations of this problem to model real-world routing scenarios. Several variants and families of this problem have been defined in the literature (please see the review articles of Braekers et al. (2016); Kumar and Panneerselvam (2012); Montoya-Torres et al. (2015)). The appearance of different VRP variants is the reflection of the challenges that researchers and companies have to face when they model their transportation network, find practical high-quality solutions and ultimately reduce the cost of their logistic systems. The difficulties in solving these combinatorial optimization problems and their practical relevance in the real-world have attracted many researchers in both academia and industry (Toth and Vigo, 2014).

The most practical and appealing optimization tools in industry are those that use approaches which yield quality solutions in a short time, are flexible to a wide range of problems and are easy to make a use of. Academic researchers continue to develop sophisticated optimization techniques as alternatives to more simple, but widely utilized, techniques (Reimann et al., 2003). A resolution as to which approaches are most beneficial was considered formally by Cordeau et al. (2002) who argued that the performance of an algorithm must be measured by four characteristics: i) quality, ii) speed, iii) flexibility and iv) simplicity. Quality is judged by the best optimal solution obtained, while speed is determined by the computational time of an algorithm. Flexibility of an algorithm is measured by its ability to solve a broad and diverse family of problems. Finally, simplicity is determined by the level of automation incorporated into the algorithm.

In this chapter we propose a comprehensive tool that integrates multiple algorithmic techniques into a platform that we refer to as Enterprise Transportation Planning (ETP). The underlying motivation behind ETP is to provide a high-performance solution approach that addresses the interests of both academic and industry parties. ETP offers an optimization framework for the family of VRP and its variants. Early implementations of ETP proposed by J.B. Hunt operations researchers offered solutions for the VRP variants listed in Table 3.1. The work in this chapter expands on these efforts and considers an integration of the problems shown in the table and with the newly introduced variants.

Table 3.1: Existing VRP Variants in ETP

VRP Variant	Brief Explanation
Capacitated (CVRP)	The vehicles have limited capacity.
Time windows (VRPTW)	In each location, the pickup/delivery time is restricted.
Heterogeneous (HVRP)	The vehicles are different.
Open (OVRP)	The vehicles do not need to back to a depot.
Multi-depot (MDVRP)	There is more than one depot in the network.
Pickup and delivery (PDVRP)	Each shipment has a pickup and delivery location.
Simultaneous pick-drops (SVRP)	Each customer location can have both pickup and delivery.
Backhauls (VRPB)	The customers are divided into two linehaul and backhaul sets which one of them must be visited first.

Several solution techniques, primarily based on customized heuristic methods, are incorporated into the optimization engine of ETP. The optimization procedure in ETP has five phases: i) grouping, ii) merging, iii) improving, iv) filtering, and v) forcing (see Section 3.4 for more details).

One of the modeling contributions of this chapter is the consideration of a new variant of VRP with respect to driver working hours regulations imposed by the United States Department of Transportation (DOT) over single and multiple shifts of work. We formulate a general variant of VRP by combining these regulatory restrictions with problem components common to VRP. Specifically, we propose the pickup delivery, heterogeneous fleet VRP with time windows and DOT regulations in an infinite planning time horizon (PHTD). We focus our computational testing on a specific scenario modeled by PHTD that considers the single shift pickup and delivery, heterogeneous fleet vehicle routing problem with backhauls, and DOT regulations (PHBD1) for the computational experiments. The PHBD1 is a special case of PTWDOT with the following extra restrictions: i) the pickup points needs to be visited first before the delivery points by the vehicle(s), ii) the time horizon is limited to a single shift and iii) the pickup/delivery time windows is

relaxed. We restricted the PTDOT general model to PBTDOT1 because the CPLEX solver could not find feasible solutions of the general model for the modified benchmark instances. Also, the PBTDOT1 model is the most frequent VRP variant that is considered in practice.

The performance of ETP is compared with CPLEX 12.6 regarding the objective function and running time on a several benchmark instances which each include a network of 100 nodes. We modified the benchmark instances by cutting the number of nodes to half to help the CPLEX solver find feasible solutions. The results show ETP can provide quality solutions in a very short amount of times (seconds). At the end of this chapter, to further emphasize the strength of ETP in solving real-world problems, a case study with realistic datasets from the transportation logistics industry is presented.

The remainder of the chapter is organized as follows. Section 3.2 discusses the relevant literature. In Section 3.3, the new variants of VRP are presented. In Section 3.4, heuristic methods used in ETP to optimize the proposed class of VRP models are described. In Section 3.5, the computational experiments of the proposed approaches are discussed with respect to VRP benchmark instances in the literature. This section also includes a case study illustrating the usage of the models introduced and the ETP framework. Finally, conclusions are offered in Section 3.6.

3.2 Literature Review

In this section, we provide a concise literature review on the different variants of VRP that are studied in this research. These variants are Capacitated VRP, VRP with time windows, heterogeneous fleet VRP, pickup delivery VRP, open VRP and VRP with Backhauls.

The most studied version of VRP is the *Capacitated* Vehicle Routing Problem (CVRP), which is the basis of all other variants of VRP (Toth and Vigo, 1997). Different exact algorithms, mostly focused on efficient enumeration (branch and bound, branch and price, and branch and cut) and relaxation (Lagrangian relaxation) methods (Fisher, 1994; Miller, 1995), have been proposed. Also, *Set Partitioning* (Balinski and Quandt, 1964), *Spanning Tree* and the *Shortest Path* (Christofides et al., 1981) representations of the problem have been developed to solve the prob-

lem to optimality. For more additional information, please see the study of Toth and Vigo (2014).

Also, several types of heuristic algorithms have been used to solve the problem to near optimality. These algorithms range from matching-based heuristics (Desrochers and Verhoog, 1989; Wark and Holt, 1994) to constructive and saving approaches such as Clarke-Wright (Clarke and Wright, 1964) and Petal algorithms (Renaud et al., 1996; Ryan et al., 1993). Also, many improvement algorithms such as several λ-opt exchange-based methods introduced by Lin (1965), granular searches (Johnson and McGeoch, 1997; Toth and Vigo, 2003), and large neighborhood searches (Pisinger and Ropke, 2007) for the large size VRPs have been developed to increase the quality of the solutions. The contemporary meta heuristic algorithms, such as Simulated Annealing (Osman, 1993), Tabu Search (Cordeau et al., 2001; Zachariadis and Kiranoudis, 2010) and Genetic Algorithms (Nagata and Bräysy, 2009; Prins, 2004), have been found to be very beneficial in finding high-quality solutions. Finally, hybrid algorithms that combine various types of mentioned algorithms have proposed (Rochat and Taillard, 1995; Tarantilis, 2005; Vidal et al., 2012, 2013).

A wide variety of extensions and variations of the VRP have been introduced by researchers during the last five decades. The VRP with time windows (VRPTW) is an extension to CVRP in which there is an allowable interval of time associated with each pickup/delivery location (see Golden et al. (2008), chapter 5). Several exact and heuristic methods have been developed and applied to solve VRPTW (Bräysy and Gendreau, 2005a,b; Nagata et al., 2010; Potvin and Bengio, 1996; Spoorendonk and Desaulniers, 2010). The heterogeneous fleet VRP (HVRP) is a CVRP variant with a fleet of vehicles of the different type, cost, and capacity. This extension of VRP has gotten many researchers and small logistic companies attention during the past decades. For more details on the HVRP, please see the work of Baldacci et al. (2008).

Pickup and delivery VRP (PDVRP) is an extension of the CVRP in which there is a set of pickup and delivery customer requests that must be satisfied by a fleet of vehicles. The PDVRP was first introduced by Hernández-Pérez and Salazar-González (2004a) for the TSP and extended to the VRP afterward. The PDVRP itself is categorized into three different categories based on

the type of demand and route structures: i) many-to-many (M-M) PDVRP, ii) one-to-many-to-one (1-M-1) PDVRP, and iii) one-to-one (1-1) PDVRP according to the study of Golden et al. (2008). The 1-1 PDVRP is the most comprehensive model and serves as a generalization to the M-M and 1-M-1 models. Several methods and algorithms are presented in the studies of the Hernández-Pérez and Salazar-González (2004b), Hernández-Pérez and Salazar-González (2007), and Hernández-Pérez and Salazar-González (2009) for M-M PDVRP, Berbeglia et al. (2007) and Min (1989) for 1-M-1 PDVRP, and Psaraftis (1983), Ascheuer et al. (2000), Ropke et al. (2007), and Ruland and Rodin (1997), Nanry and Barnes (2000), Lau and Liang (2002), Bent and Van Hentenryck (2006), and Cordeau et al. (2008) for 1-1 PDVRP.

Open VRP is another variant of CVRP in which the vehicles are not required to return to the depot after visiting the customers. The application of this problem is observed in the home delivery of packages and newspapers, as well as hiring third-party contractors with their own vehicles. Effective methods for solving this problem can be found in Brandão (2004), Li et al. (2007), Sariklis and Powell (2000) and Fleszar et al. (2009).

VRP with backhauls is an extension of CVRP in which the delivery customers (linehaul) must be visited before the pickup customers (backhaul) by each vehicle. Several exact and heuristic methods have been developed in the past decades. Notable amongst these are Toth and Vigo (1997), Goetschalckx and Jacobs-Blecha (1989), Duhamel et al. (1997) and Zhong and Cole (2005).

3.3 VRP with Drive Service Regulations

In this section, a new variant of VRP that has not been addressed in the literature is presented, VRP with DOT regulations (VRPDOT). In this variant, the hours of service (HOS) by commercial motor vehicle (CMV) drivers are restricted according to the regulations imposed by the Federal Motor Carrier Safety Administration (FMCSA) of the U.S. Department of Transportation (DOT). These regulations restrict the hours of work and drive of truck drivers on their daily and weekly shifts as well as the amount of rest between them (please see Section 3.3.1 for more de-

tails). Transportation companies, such as J.B. Hunt, are responsible for implementing these regulations. In J.B. Hunt, drivers utilize the electronic hour logging system to record hours driven. In this system, the drive, work, and rest hours of each driver are precisely recorded so the drivers cannot exceed their hours of service. Exceeding the hours of service has historically been the practice to increase their daily working hours and compensation. In short terms, reducing the working hours of drivers can reduce the profit margins of transportation companies. On the other hand, in the long terms, it can prevent the surcharges of involvement in critical events such as crashes or crash-relevant conflicts resulting from drivers fatigue.

The Regulatory Impact Analysis (RIA) by Analysis Division, Federal Motor Carrier Safety Administration (2011) provides an assessment of the costs of operational changes and safety benefits of final rule changes DOT HOS regulations. In this RIA, the costs of operational changes are estimated by following the chain of consequences caused by changes in HOS on existing work patterns in terms of work and driving hours per week. These costs are estimated using impacts on industry productivity. However, they would most likely be passed along as increases in freight transportation rates of goods that are transported by truck for consumers. Safety benefits are estimated by counting the changes in hours worked that results in a reduction in expected fatigue-related crashes. The changes in crash risks were monetized by measuring the losses of life, medical costs for injuries, lost time due to the congestion effects of crashes and property damage caused by the crashes themselves. According to the results of this analysis, net benefits (i.e., benefits minus costs) are likely to be positive. It could range from a negative $250 million to more than a positive $550 million depending on the baseline level of fatigue involvements in crashes (7%, 13% and 18%) and the percentage of discount in future health benefits (7% and 3%).

The quantitative analysis of the impact of the DOT HOS rules is followed by the two studies of the American Transportation Research Institute (ATRI) in 2013. In ATRIs first study (Short, 2013a), the assessment of the direct and indirect impacts of HOS rules is surveyed from more than 500 motor carriers and 2,000 commercial drivers prior to the implementation of the rules changes. ATRIs analysis resulted in the net industry cost from the new restart provisions (one of

the DOT rule changes) that ranged from $95 million to $376 million. This result differs significantly from FMCSA's RIA identifying $133 million net benefits, annually. The second study of ATRI (Short, 2013b) discusses actual industry operational impacts post-rules change. The results show the increment in fatigue levels, decrement in quality of life and negative impacts on payment of commercial drivers. In 2015, ATRI conducted another study (Murray and Short, 2015) of post driver data analysis which shows overall crashes increased from 2013.

An internal study in J.B. Hunt showed that imposing these regulations reduced the profit margins up to 3.5%, as it reduces the working hours of drivers. Young (2013) studied a simulation modeling approach on the preliminary impacts of the 2013 HOS regulation changes on a large random over-the-road (OTR) trucking fleet. This model quantifies the impact of the regulation changes to guide the company to minimize the impact to high-risk customers and provide a foundation for proactive customer communications. The differences between the models with and without the changes in DOT regulations are compared using Single-Factor ANOVA. Results show a driver can expect to lose 22 miles and available work hours of 0.68 or 41 minutes per week, on average.

In contrary to these analytical approaches to analyze the impact of changes in HOS by DOT regulations, this research provides an Operation Research (OR) framework for modeling and optimizing the fleet of vehicles considering these regulations. From the modeling point of view, each of these regulations (five regulations that are explained in details in the following section) forces a series of constraints to any VRP variant of interest. Despite the fact that the routing and scheduling of trucks are currently being impacted by these regulations, to the best of our knowledge, there are no optimization solutions in the literature or practice that address this complication.

In the following sections, we state the DOT regulations and then introduce a model that accounts for these restrictions. Based on this model, we define and formulate a new variant of VRP that we solve with ETP.

3.3.1 Overview of DOT Regulations

In this study, we refer to the regulations on the HOS for CMV drivers by FMCSA within the U.S. DOT, as the "DOT" regulations. The newest version of DOT regulations released in 2014 (Please see *fmcsa.dot.gov*), consist of the following five items:

1. Active Hours in a Work Shift (**DOT_ACTIVE**): The number of total consecutive active hours allowed for a driver. As of 2014, a driver can be active for 14 hours before he has to take a DOT_REST. Active hours of a driver is the total time of his work shift including the drive, load and unload, and refueling time.

2. Drive Hours in a Work Shift (**DOT_DRIVE**): The number of drive hours allowed for a driver during a shift. As of 2014, a driver can drive 11 hours during DOT active hours.

3. Rest Hours Between Consecutive Drives (**DOT_REST**): The number of hours required between two consecutive shifts. As of 2014, a driver has to wait for 10 hours before DOT active hours resets.

4. Weekly Drive Hours (**DOT_WEEKLY_DRIVE**): The number of total driving hours in a 7/8 day period. As of 2014, a driver can drive up to 60 hours in a 7 consecutive day period and up to 70 hours in an 8 consecutive day period. After that, a DOT weekly rest is required.

5. Weekly Reset Hours (**DOT_WEEKLY_REST**): Total break time required between two consecutive weekly limits. As of 2014, a driver has to take a 34-hour break before DOT Weekly Drive Hours resets.

To consider these regulations in the VRP variants, we start from less complicated models to the most ones by limiting the planning time horizon. The first two regulations are associated with the daily driving restrictions. That means, if we limit our planning time horizon to only one day, the last three regulations become irrelevant as they affect the hours of service of the drivers

in more than one day working shifts. This model is suitable for short-haul transportation problems (see *Federal Highway Administration* (2012)). Note that the modeling of any VRP variant's actions taken with respect to drivers and trucks has direct implications on routes. As we limit the working hours of each truck, we are actually limiting the duration of the associated route. Thus, the first regulation (DOT_ACTIVE) restricts the total duration of each route to 14 hours in the VRP. The second item (DOT_DRIVE) places an 11 hour limitation on the traveling (driving) time of each truck between the nodes in the associated route. The service time at each node, if there is any, is not included in this 11 hours of driving. Note that if the service time is zero for all the nodes in the network the two constraints associated with DOT_ACTIVE and DOT_DRIVE are the same. These two regulations impact the daily planning in the VRP variant. A shift is often represented by a day. However, the start and end date of a shift may vary between drivers. It does not matter what time of a day we start a shift as long as we limit the total duration of the shift. We can start a shift at 8 AM and end it at 10 PM (14 hours of working) in the same day, or we can start the shift at 2 PM and end it at 4 AM on the next day.

If we expand our planning time horizon from a single day to a week, we make the problem more complicated regarding the number of constraints and variables in the mathematical formulation. In the weekly model, regulations 3 and 4 are included, in addition to the first two regulations. Regulation 3 (DOT_REST) necessitates that drivers take a 10-hour rest between two consecutive shifts. Regulation 4 (DOT_WEEKLY_DRIVE) limits the total weekly driving hours of each driver to 60 and 70 hours in a 7 and 8 days period, respectively.

The most general and complicated model of the VRP with DOT is for any planning horizon beyond one week. In this model, all the DOT regulations become relative and turn into active constraints in the associated mathematical formulations. Regulation 5 ensures that appropriate extended breaks are taken between weeks of work. A driver must take a 34-hour break before the start of the next week. The most general VRP variant with DOT regulations includes all the five regulations. If we limit the time horizon to one week, the fifth regulation (DOT_WEEKLY_REST) become irrelevant. Similarly, the single shift (day) time horizon model only includes the first two

regulations. Shrinking the planning time horizon makes the associated variant simpler, with fewer constraints and variables. The single shift VRP variant with DOT regulations is the only model that CPLEX solver can find feasible solutions for, even if we reduce the number of customer points in the benchmark instances. More details on the benchmark instances are presented in Section 3.5.

It is a common practice in the VRP literature to use the distance and the time between locations, interchangeably. As the DOT regulations limit the hour of service of drivers, not the traveling distance, we must convert the distance to travel hours. We used a function derived from an internal study in J.B. Hunt on the average speed of dedicated fleet of trucks to several customers over a ten year period (see Table 3.2).

Table 3.2: Distance to Time Conversion

Distance Range (miles)	**Calculated Time (hours)**
0	0.00
$(0, 5]$	0.23
$(5, 10]$	(miles $-$ 5.0) / 30.0 $+$ 0.23
$(10, 15]$	(miles $-$ 10.0) / 33.33 $+$ 0.4
$(15, 20]$	(miles $-$ 15.0) / 37.5 $+$ 0.55
$(20, 40]$	(miles $-$ 20.0) / 48.0 $+$ 0.68
$(40, 60]$	(miles $-$ 40.0) / 54.54 $+$ 1.1
$(60, \infty)$	(miles $-$ 60.0) / 60.0 $+$ 1.47

In the above table, the travel distance is divided into seven disjoint ranges with different average speeds for trucks to calculate the hours of travel. According to this calculation, if the distance is less than 5 miles, it takes 15 minutes to travel. This function returns 16 minutes of travel between two points with the distance of 6 miles. It takes 14 minutes for traveling 5 miles and 2 minutes to travel the remaining 1 mile.

In the following section, we formulate a general VRP variant with the DOT regulations for an infinite time horizon. This variant is the pickup delivery, heterogeneous fleet VRP with time windows and DOT regulations (PHTD). Following the general model, we discuss how the

inclusion of different subsets of the regulatory constraints can affect various real-world scenarios.

3.3.2 PHTD with Infinite Time Horizon

As mentioned in Section 3.2, in the one-to-one pickup delivery VRP each shipment has an origin and a destination. This model is the most flexible and general PDVRP. In this work, all the pickup delivery problems are modeled as the 1-1 PDVRP. However, for brevity, we refer to the 1-1 PDVRP as simply PDVRP through the remainder of the chapter. An example of the PDVRP is illustrated in Figure 3.1.

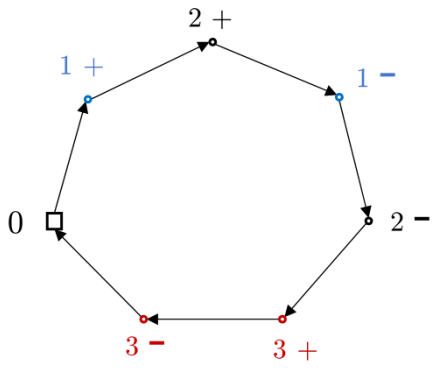


Figure 3.1: An example of the PDVRP (Toth and Vigo, 2014)

In Figure 3.1, three shipments, 1, 2, and 3, must be transported between their pickup and delivery points. The "+" sign indicates the pickup point and the "−" sign the delivery point. In each route, the associated vehicle must pickup each shipment before it is delivered.

To formulate PHTD with infinite time horizon, all DOT regulations that are mentioned in Section 3.3.1, are imposed to the model. The notations of the mathematical formulation for the sets, parameters, and decision variable are stated in Tables 3.3 to 3.5.

Table 3.3: Sets of PHTD with Infinite Time Horizon

Notation	Description
• $V = \{0, 1, \ldots, n, n+1, \ldots, 2n, 2n+1\}$	The depot and customer locations (points).
◦ $\{0\}$	The start depot.
◦ $P = \{1, \ldots, n\}$	The pickup points.
◦ $D = \{n+1, \ldots, 2n\}$	The delivery points.
◦ $\{2n+1\}$	The end depot.
• $K = \{1, \ldots, l\}$	The vehicles.
• $W = \{1, 2, \ldots, f\}$	The weeks of the planning time horizon. The time horizon is f weeks.
• $S = \{1, 2, \ldots, d, d+1, \ldots, 2d, \ldots, (f-1)d, \ldots, fd\}$, $d = 7$ or 8	The shifts (days) of the planning time horizon. If $d = 7$ then DWD $= 60$, otherwise DWD $= 70$.
◦ $S_w = \{(w-1)d, \ldots, wd\}$; $\forall w \in W$	The shifts of the week w.

Table 3.4: Parameters of PHTD with Infinite Time Horizon

Notation	Description
• s_i; $\forall i \in V$, $s_0 = s_{2n+1} = 0$	The service time in location i. $s_0 = s_{2n+1} = 0$.
• $[b_i, e_i]$; $\forall i \in V$	The time windows of node i.
• c_{ij}; $\forall i, j \in V$	The cost of traveling between nodes i and j.
• h_{ij}	The approximate hours of travel between node i and j.
• q_i; $\forall i \in V$, $q_0 = q_{2n+1} = 0$, $q_{n+i} = -q_i$	The demand of each customer. $q_i > 0$ means pickup and $q_i < 0$ means delivery.
• C_k; $\forall k \in K$	The capacity of vehicle k.
• DA $= 14$	DOT_ACTIVE, see Section 3.3.1.
• DD $= 11$	DOT_DRIVE, see Section 3.3.1.
• DR $= 10$	DOT_REST, see Section 3.3.1.
• DWD $= 60$ or 70	DOT_WEEKLY_DRIVE, see Section 3.3.1. If $d = 7$, then DWD $= 60$. Otherwise, DWD $= 70$
• DWR $= 34$	DOT_WEEKLY_REST, see Section 3.3.1.

Table 3.5: Decision Variables of PHTD with Infinite Time Horizon

Notation	**Description**
• T_{ik}; $\forall i \in V, k \in K$	The beginning of service time at location i by vehicle k.
• Q_{ik}; $\forall i \in V, k \in K$	The load of vehicle k after visiting node i.
• $x^t_{ijk} \in \{0,1\}$; $\forall i, j \in V, k \in K, t \in S$	Equals 1 if vehicle k moves between location i and j in shift (day) t; 0, otherwise.

$$\min \sum_{t\in S}\sum_{k\in K}\sum_{i\in V}\sum_{j\in V} c_{ij}x^t_{ijk} \quad \text{(PHTD)}$$

$$\sum_{t\in S}\sum_{k\in K}\sum_{j\in V} x^t_{ijk} = 1 \quad \forall i \in P \tag{3.1}$$

$$\sum_{t\in S_w}\sum_{j\in V} x^t_{ijk} - \sum_{t\in S}\sum_{j\in V} x^t_{n+i,j,k} = 0 \quad \forall i \in P, k \in K, w \in W \tag{3.2}$$

$$\sum_{t\in S_w}\sum_{j\in V} x^t_{jik} - \sum_{t\in S}\sum_{j\in V} x^t_{ijk} = 0 \quad \forall i \in P\cup D, k \in K, w \in W \tag{3.3}$$

$$\sum_{t\in S}\sum_{j\in P} x^t_{0jk} = 1 \quad \forall k \in K \tag{3.4}$$

$$\sum_{t\in S}\sum_{i\in V} x^t_{i,2n+1,k} = 1 \quad \forall k \in K \tag{3.5}$$

$$Q_{ik} - Q_{jk} + q_j \le M(1 - x^t_{ijk}) \quad \forall i, j \in V, k \in K, t \in S \tag{3.6}$$

$$\max\{0, q_i\} \le Q_{ik} \le \min\{C_k,\ C_k + q_i\} \quad \forall i \in V, k \in K \tag{3.7}$$

$$\sum_{i\in V}\sum_{j\in V} (h_{ij}x^t_{ijk} + s_j x^t_{ijk}) \le \text{DA} \quad \forall k \in K, t \in S \tag{3.8}$$

$$\sum_{i\in V}\sum_{j\in V} h_{ij}x^t_{ijk} \le \text{DD} \quad \forall k \in K, t \in S \tag{3.9}$$

$$\sum_{t\in S_w}\sum_{i\in V}\sum_{j\in V} h_{ij}x^t_{ijk} \le \text{DWD} \quad \forall k \in K, w \in W \tag{3.10}$$

$$T_{ik} - T_{jk} + s_i + h_{ij} \le \text{DR}\left(1 - \left|x^t_{ijk} - \textstyle\sum_{v\in V} x^{t+1}_{jvk}\right|\right) \quad \forall i \in V;\ j \in P\cup D\cup\{2n+1\};\ k \in K;\ t \in S \tag{3.11}$$

$$T_{ik} - T_{jk} + s_i + h_{ij} \le \text{DWR}\left(1 - \left|x^t_{ijk} - \textstyle\sum_{v\in V} x^{t'}_{jvk}\right|\right)$$

$$\forall j \in P\cup D\cup\{2n+1\};\ i \in V;\ k \in K;\ t \in S_w;\ t' \in S_{w+1};\ w \in W \tag{3.12}$$

$$T_{ik} - T_{jk} + s_i + h_{ij} \le M(1 - x^t_{ijk}) \quad \forall i \in V;\ j \in P\cup D\cup\{2n+1\};\ k \in K;\ t \in S \tag{3.13}$$

$$T_{n+i,k} - T_{ik} - s_i - h_{i,n+i} \ge 0 \quad \forall i \in P;\ k \in K \tag{3.14}$$

$$b_i \le T_{ik} \le e_i \quad \forall i \in V, k \in K \tag{3.15}$$

$$x^t_{ijk} \in \{0,1\} \quad \forall i, j \in V, k \in K, t \in S \tag{3.16}$$

The objective function seeks to minimize the total traveling cost. M is a very large num-

ber. Constraints (3.1) are the degree constraints and make sure that each pickup point is visited exactly once in the time horizon. Constraints (3.2) and (3.3) are the transition constraints that restrict the number of the ingoing vehicle(s) to each point equal to the outgoing. Constraints (3.4) and (3.5) impose the departure and return of the set of available vehicles from the start to the end depot. Constraints (3.6) and (3.7) are the capacity constraints which limit the cumulative orders of customers and load of each vehicle to the capacity of each vehicle, respectively. Constraints (3.8) limits the summation of drive time and service time of each vehicle to DA = 14 hours according to the DOT_ACTIVE regulation. Similarly, constraints (3.9) and (3.10) restrict the daily and weekly driving hours of each vehicle to DD = 11 and DWD = 60 or 70 hours according to the DOT_DRIVE and DOT_WEEKLY_DRIVE regulations, respectively. Constraints (3.11) force a DR = 10 hour gap to the travel and service time between the visit time of two points in two consecutive shifts served by the same vehicle according to the DOT_REST regulation. Similarly, Constraints (3.12) impose a DWR = 34 gap between the two points in two consecutive weeks according to the DOT_WEEKLY_REST. Finally, Constraints (3.13 - 3.15) are the time window constraints which make sure each customer be visited within its allowed time windows.

The PHTD with infinite time horizon is a general variant of VRP which can be converted to some interesting special cases of VRP by relaxing specific constraints. For example, if we remove the constraints (3.13 - 3.15) in (PHTD) formulation we are left with the pickup delivery, heterogeneous fleet VRP with the DOT regulations in an infinite time horizon. This variant is the same as the PHTD except that the pickup and delivery points can be visited any time without any restrictions. Two additional special cases of the PHTD that are more important for us in this research are the followings: i) the PHTD with one week time horizon (PHTD7) and ii) the PHTD with a single shift time horizon (PHTD1). The PHTD7 variant with weekly planning time horizon is appealing for both transportation companies and drivers. On one hand, companies can have a limited time horizon to plan and manage their resources easier. On the other hand, the drivers can have their weekly work schedule. The mathematical formulation of PHTD7 is derived by removing the Constraints (3.12) in (PHTD). The PHTD7 mathematical formulation is presented

in Appendix 3.B. The PHTD1 variant with the daily planning time horizon (without Constraints (3.10-3.12) in (PHTD)) is the least complicated model. The mathematical formulation of PHTD1 is presented in Appendix 3.C.

In the following section, we define and formulate another VRP variant with DOT regulation referred as the single shift pickup delivery, heterogeneous fleet VRP with backhauls and DOT regulations (PHBD1). The PHBD1 is a variation of PHTD1 in which the pickup points must be visited before the delivery points without any time windows restrictions on visited points. In Section 3.3.3, the ILP formulation of this variant is presented. This mathematical formulation is used later in Section 3.5 to solve a set of benchmark problems to compare the performance of ETP with the CPLEX solver. We choose the PHBD1 to compare the performance of ETP with the CPLEX solver for two reasons. First, the pickup delivery and backhauls VRP variants are the most popular models used to optimize the customers' network in J.B. Hunt. Second, CPLEX has shown the best performance on this variant. As a matter of fact, the single shift VRP variants with DOT regulations are the only variants that the CPLEX can find a feasible solution (even when we modified the benchmark instances by reducing the number of customer points to half, as explained in Appendix 3.A).

3.3.3 PHD with Backhauls in a Single Shift Time Horizon

The single shift pickup delivery, heterogeneous fleet VRP with backhauls and DOT regulations (PHBD1) is a variant of the pickup delivery heterogeneous fleet VRP with DOT regulations (PHD) in a single shift time horizon which follows the VRP with backhaul variants restrictions. The VRP with backhaul (VRPB) is a VRP variant where the vehicles must visit linehaul customers before considering backhauls. The customers are partitioned into two sets: i) linehaul customers, where to which shipments are received and ii) backhaul customers, from which shipments are picked up (Toth and Vigo, 2014). In the VRPB, each vehicle completes a route by starting from the depot and visiting a subset of linehaul customers before visiting any backhaul customers and returning to the depot while not exceeding the capacity of the vehicle (the vehicle can return to

the depot without visiting any backhaul customers). An example of a feasible solution for the VRPB is shown in Figure 3.2.

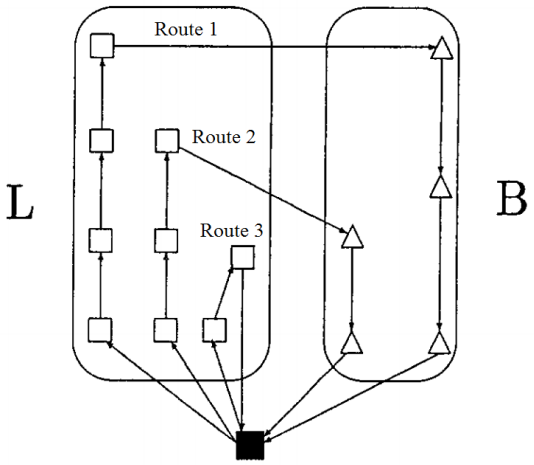


Figure 3.2: VRP with backhauls (a feasible solution example) (Toth and Vigo, 1997)

In Figure 3.2, the linehaul customer set is shown with "L" and backhauls (triangle shaped) with "B". Routes 1 and 2 include both linehaul customers (square shaped) and backhauls (triangle shaped) where linehauls are visited first. Route 3 consists of only linehaul customers without any backhauls.

Although VRPB is a realistic model for the movement of bulk and heavy shipments, there are fewer studies on this variant of VRP in the literature. The main reason in using backhaul variants is because the arrangement of loads in the trucks is mostly LIFO (last in, first out) and in some other cases, the linehaul customers have a higher priority compared to backhauls.

VRPB can be considered as a variant of the pickup delivery VRP (PDVRP). As shown in Figure 3.1, in PDVRP, each shipment has an origin and a destination where the origin must be visited first. If we restrict the PDVRP model by adding the constraint that all the pickup points must be visited before the delivery points, we will have the VRPB model. That means to convert the PDVRP to VRPB, we just need to set all the pickup points as the linehauls and the delivery points as the backhaul customers.

The notation of the PHBD1 is defined in Table 3.6. which are mostly the same as in 3.3 to 3.5 with some modified sets and decision variables. In this table, the customer locations are

partitioned into two sets of linehaul and backhaul customers, instead of pickup and delivery sets in Table 3.5.

Table 3.6: Modifications on the Sets and Decision Variables of the PHBD1

Notation	**Description**
Sets:	
• $V = \{0, 1, \ldots, n, n+1, \ldots, 2n, 2n+1\}$	The depot and customer locations (nodes).
◦ $\{0\}$	The start depot (source).
◦ $L = \{1, \ldots, n\}$	The linehaul customers.
◦ $B = \{n+1, \ldots, 2n\}$	The backhaul customers.
◦ $\{2n+1\}$	The end depot (sink).
Decision Variables:	
• $x_{ijk} \in \{0, 1\}$; $\forall i, j \in V, k \in K$	Equals 1 if vehicle k moves between location i and j; 0, otherwise.
• u_{ik}, $i \in V, k \in K$	The order of visiting customer i in the route k.

The mathematical formulation of the single shift PHBD1 is presented as follows. As mentioned before in this formulation, only the constraints related to the single shift of time horizon, DOT_ACTIVE, and DOT_DRIVE, are imposed.

$$\min \sum_{k\in K}\sum_{i\in V}\sum_{j\in V} c_{ij}x_{ijk} \qquad \text{(PHBD1)}$$

$$\sum_{j\in V}\sum_{k\in K} x_{ijk} = 1 \qquad \forall i \in L\cup B \tag{3.17}$$

$$\sum_{j\in L} x_{0jk} = 1 \qquad \forall k \in K \tag{3.18}$$

$$\sum_{i\in V} x_{ijk} - \sum_{i\in V} x_{jik} = 0 \qquad \forall j \in V, k \in K \tag{3.19}$$

$$Q_{ik} - Q_{jk} + q_j \le M(1 - x_{ijk}) \qquad \forall i, j \in V, k \in K \tag{3.20}$$

$$\max\{0, q_i\} \le Q_{ik} \le \min\{C_k,\, C_k + q_i\} \qquad \forall i \in V, k \in K \tag{3.21}$$

$$u_{ik} - u_{jk} + n x_{ijk} \le n - 1 \qquad \forall i, j \in L\cup B, k \in K \tag{3.22}$$

$$u_{0k} = 0 \qquad \forall k \in K \tag{3.23}$$

$$u_{ik} \ge 1 \qquad \forall i \in V, k \in K \tag{3.24}$$

$$u_{ik} - u_{jk} \le M(1 - x_{ijk}) \qquad \forall i \in L, j \in B, k \in K \tag{3.25}$$

$$\sum_{i\in V}\sum_{j\in V} (h_{ij}x_{ijk} + s_j x_{ijk}) \le \text{DA} \qquad \forall k \in K \tag{3.26}$$

$$\sum_{i\in V}\sum_{j\in V} h_{ij}x_{ijk} \le \text{DD} \qquad \forall k \in K \tag{3.27}$$

$$x_{ijk} \in \{0, 1\} \qquad \forall i, j \in V, k \in K \tag{3.28}$$

Similar to previous formulations, the objective function seeks to minimize the total cost of travel and M is a very large number. Constraints (3.17) are the degree constraints. Constraints (3.18) enforce the departure of each vehicle to one of the linehaul customers. Constraints (3.19) are the transition constraints for all the customers in both L and B. Constraints (3.20) and (3.21) limit the cumulative load of each vehicle to the capacity of the vehicle, while satisfying the demand/supply of each customer in L/B sets. Constraints (3.22-3.24) are the MTZ (Miller-Tucker-

Zemlin) inequalities (Miller et al., 1960) for each route which work as the sub-tour elimination constraints (SEC). Constraints (3.25) are the precedence constraints, which prevent visiting any backhaul customers, if there is any, before visiting all the linehaul customers in each route using the MTZ inequalities. The DOT regulations on the total active time (DOT_ACTIVE) and total drive time (DOT_DRIVE) in a single shift are imposed by Constraints (3.26) and (3.27), respectively.

3.4 ETP Optimization Methodology

In this section, the methodology for solving the different variants of the VRP is explained. We utilize customization of existing heuristic methods to generate solutions to large-scale real-world problems. These problems can include thousands of customer locations and hundreds of trucks. Only relatively small VRP instances, around 100 locations and 10 vehicles, can be solved optimally using CPLEX.

The evolution of VRP heuristics in the past 15 years emerged according to three major schemes: (i) combining methods that have been individually developed for VRPs, such as simulated annealing, genetic algorithms, and tabu search, (ii) exotic large neighborhoods, exact mathematical methods, and decomposition. (iii) hybridization, which are the methods that have been designed in a way that have the flexibility of applying to many variants of VRP without any major structural changes (Cordeau et al., 2001; Subramanian et al., 2013; Vidal et al., 2013). To optimize the VRP variants (Sections 3.3.2 and 3.3.3) proposed in the previous chapter, we pursue a hybridization approach in ETP. ETP performs two major steps: i) routing and ii) scheduling. Routing creates the sequence of shipments (locations) visited by a vehicle to find the candidate routes. The objective is to minimize the total cost of completing shipments. Scheduling determines the exact time of arrival and departure for each customer location in a given route with respect to feasibility constraints, such as DOT regulations, and pickup/delivery time windows. For each considered route, the scheduling step calculates the earliest feasible schedule. The routing process in ETP for any VRP model can be summarized in five steps: Step 0) grouping, Step 1)

merging, Step 2) improving, Step 3) filtering and Step 4) forcing as shown in Figure 3.3.

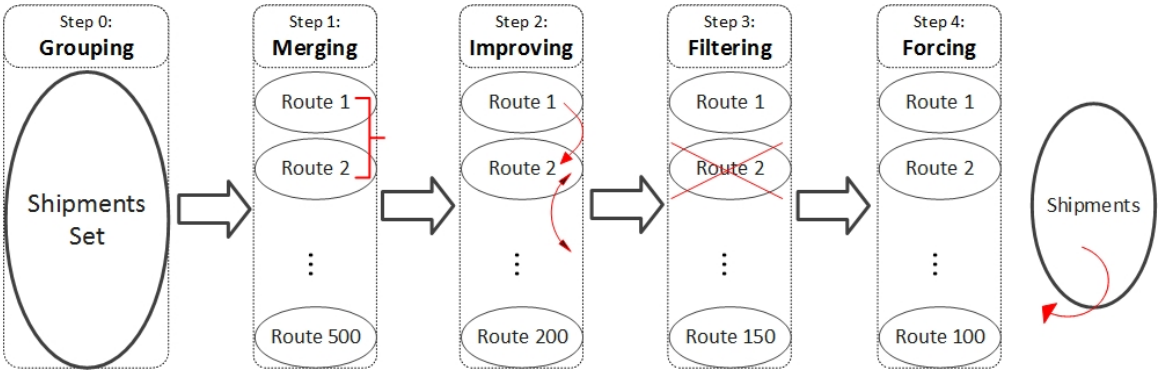


Figure 3.3: The ETP Routing Process

Grouping: The optimization process starts with a grouping algorithm. The main purpose of grouping is to identify sets of order lines that are most likely be in the same route or are unlikely to form a route together. Grouping helps to reduce the number of combinations of order lines in the next step by either putting similar orders in the same route or preventing other orders to create a route. This process is shown as step 0 in Figure 3.3. We develop and utilize a new algorithm referred to as Group Direct algorithm (Section 3.4.1) to complete this step. After step 0 is complete, all the order lines (shipments) in the dataset will be placed on to a route and passed to the merging methods of ETP (Section 3.4.2).

Merging: Step 1, merging, can be done by a variety of algorithms available to the user. The most frequently used algorithms are a modified Nearest Neighbor and a modified Clarke-Wright savings algorithm. The Nearest Neighbor search (Section 3.4.2.1), is a greedy heuristic in which the next move will be selected based on the largest savings from the last location. The Clarke-Wright method tries to merge the created routes based on the potential savings. Both Nearest Neighbor and Clarke-Wright algorithms are shown to perform well each on some benchmark instances considered in Section 3.5. Thus, in this chapter, we use both algorithms, separately, to merge the routes. As a result of this step, some routes will be combined. Therefore, the total number of routes is expected to decrease after Step 1.

Improving: Step 2 contains improvement procedures. We consider four improvement al-

gorithms: i) line move, ii) stop move, iii) line exchange, and iv) group stop move. These algorithms are based on the well-known λ-opt algorithms (Lin, 1965). The main purpose behind these algorithms is to alter the lines and stops in each route to obtain better solutions. Again, the route count is expected to decrease after improvement techniques are complete (see Section 3.4.3).

Filtering: Step 3 involves implementing a variety of user-defined filters on the created routes to satisfy special interests of customers. These filters do not coincide with optimization constraints, they are designed as post-optimization restrictions that each route must satisfy. For example, a customer wants to have an extra limitation on the duration of the routes in a certain area in the network due to some economic considerations. After the filters are implemented, some routes could be removed due to low utilization of the weight or volume in the associated vehicles (Section 3.4.4).

Forcing: In Step 4, all remaining single lines that do not belong to any route (or have been cut due to filters in Step 3) are either forced to be a part of a route or a single line route made into their route. Note that after step 0, if a line is not in a route, then it is not placed in any route in any iteration until Step 4 (Section 3.4.5).

Sections 3.4.1-3.4.5 offer more detailed explanations of the individual algorithms described above. Then, Section 3.5 discusses the computational performance of the ETP methodology.

3.4.1 Grouping Algorithms

In this study, we defined grouping algorithms as a fast scan of order lines in the network with the purpose of placing the order lines that share similar properties into the same sets to help accelerate the creating or merging routing process. Groups of orders are created by two sets of actions: i) cluster and ii) selection. Cluster narrows down the search space based on the location or time remoteness. It guides the optimization by not allowing some combinations of order lines (shipments) to be attempted in creating the routes. For example, it is unlikely that grouping a point on the east of the U.S. with a point on the west coast of the U.S. will lead to a feasible route, let alone an optimal route. It is even worse if the combination is feasible, because the algorithm will

explore the solution further. On the other hand, selection identifies the candidate location pairs for grouping consideration. Unlike the cluster algorithms, which is based on the differences of the location, the selection works with the similarities between orders. For example, if a pair of order lines have the same origins and destinations, they are most likely to be placed in the same route.

In this research, as we are working on the benchmark instances with small size comparing with the real-world problems, we do not need to perform extensive efforts to group the order lines. Our proposed Group Direrct algorithm is a fast and the only grouping algorithm that is used to optimize the benchmark instances in Section 3.5 and is explained as follows.

The Group Direct algorithm only groups order lines that have the same origin and destination locations. In other words, the lines that share the same pick and drop points are considered acceptable combinations. After such orders are identified, the method picks the orders with the highest volume, one by one, to the group. In the case of ties between the highest volume orders, it selects the order that has the earliest time of pickup to the grouped orders. Figure 3.4 represents the Group Direct algorithm, graphically.

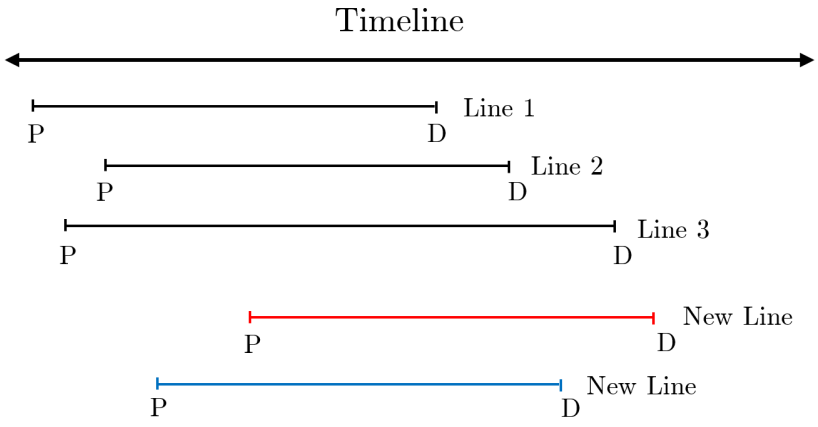


Figure 3.4: The Group Direct Algorithm

Figure 3.4, illustrates a group that already contains three lines (lines 1, 2 and 3). These lines have the same pickup and delivery locations. “P” stands for the beginning of the pickup time and “D” stands for the beginning of the delivery time. The algorithm wants to add two new

lines to this group. These lines have the same order volume and the same origin and destination as the other three lines in the group. At this point the Group Direct algorithm needs to decide which line to add to the group first based on the beginning of the pickup time. The algorithm is going to pick the one in blue because it has a pickup time closer to the earliest pickup time in the group (pickup time of line 1). At each iteration, the method only adds one line. So after adding the blue line, it is going to examine the red line to add to the group.

A parameter, which we refer to "Time_Gap", is associated with the Group Direct algorithm . It specifies how much difference we want to allow between the pickup time of orders in the same group. If the specified value for this parameter is smaller than the difference between the pickup time of the candidate order and the earliest pickup time of the underlying group, the line is not going to be added.

To clarify the implementation of Group Direct algorithm, the pseudo-code of this algorithm is stated as follows.

Group Direct (Shipments)

{

Step 1 Select an ungrouped shipment from Shipments. If no ungrouped shipment found, go to Step 6.

Step 2 Put the shipment in a new group.

Step 3 Find all ungrouped shipments with the same pickup and delivery locations as the shipment. If no shipment found, go to Step 1.

Step 4 Add another shipment with the largest size to the current group, if feasible. Otherwise, create a new group with the shipment and go to Step 3. In case of the tie between the shipments with the largest seize, pick the one with the earliest pickup time.

Step 5 Go to Step 1.

Step 6 Return all the created groups.

}

Where an ungrouped shipment is a shipment (order) that does not belong to any group yet. The feasibility check for grouping is the same as routing (capacity, time windows, etc.) plus the

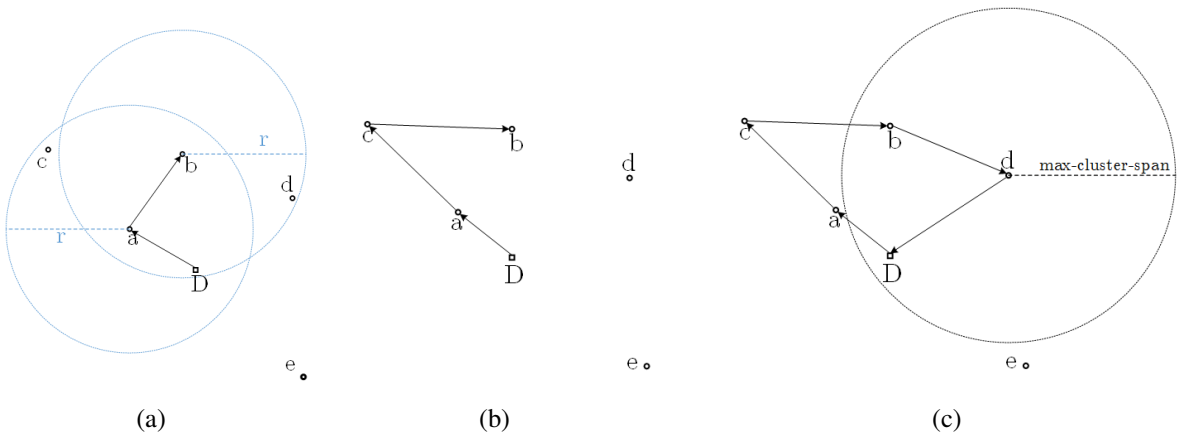
Time_Gap restriction.

3.4.2 Merging Algorithms

In Step 1, Merging algorithms combine the previously created groups (routes) to allow for better-utilized trucks. The order lines in the same group (route) in Step 0 are considered as a combined single order in this step. There are several types of proposed and modified heuristics included in ETP, however they are all savings based. In general, saving based algorithms dynamically evaluates combinations of orders and pick those which satisfy some criteria such as the biggest saving or the smallest cost to create routes. Then, they update the criteria and continue iterating until all the combinations are exhausted (Toth and Vigo, 2014). In this study, a modified Nearest Neighbor algorithm is proposed and is explained as follows.

3.4.2.1 The Modified Nearest Neighbor Algorithm

The classic Nearest Neighbor search starts from the depot (or a random node) and repeatedly visits the next nearest node until a feasibility constraint is violated. We modify the classic NNS in three ways to increase the quality of the solutions. First, we add a new shipment with the minimum distance not only from the last shipment, but from any other shipments that already have been added to the route. Second, to add the nearest line to the current route we find the best position in between all the shipments that are already in the route. Finally, third, we define a parameter, which refer to as max_cluster_span, that limits the search space to considered for the next shipment. This positive-valued parameter is used as follows. First, we calculate the average minimum distance of each shipment from all other shipments in the network. Then, we multiply that average by max_cluster_span to limit the search space radius. The modified Nearest Neighbor search (NNS) algorithm is illustrated in Figure 3.5.



(a) (b) (c)

Figure 3.5: The Modified Nearest Neighbor Algorithm

In Figure 3.5 (a), route starts from depot "D" and the points "a" and "b" are added to the route, respectively. As shown in Figure 3.5 (b), point "c" is the next point added to the route even though point "d" is the closest point to "b" (i.e. $dist(a,c) < dist(b,d)$, here $dist(i,j)$ is the distance between i and j). According to Figure 3.5 (c), point "d" is the last point that we add to the current route, because point "e" is outside max_cluster_span from all the points in the current route. The pseudo-code of this algorithm is presented as follows.

Nearest Neighbor (Shipments)

{

Step 1 Select an unrouted shipment from Shipments. If no shipment found, go to Step 7.

Step 2 Create a new route with the shipment.

Step 3 Find the nearest shipment to any shipments in the current route from the unrouted shipments that have not tried before. If no shipment found go to Step 1.

Step 4 Add the shipment to the current route, if feasible. Otherwise, go to Step 3.

Step 5 Go to Step 3.

Step 6 Go to Step 1.

Step 7 Return all the routes.

}

The feasibility check int Step 4, is the regular feasibility check of routing (such as capacity

and time windows) plus the max_cluster_span check. Adding a shipment to the current route in Step 4 is greedy. It happens by finding the best position with the largest savings.

3.4.2.2 The Modified Clarke-Wright Algorithm

The Clarke-Wright algorithm (Clarke and Wright, 1964) has been used over the last five decades by researchers to find solutions for different variation of the VRP. The main principle of Clarke-Wright algorithm is to merge two routes with the biggest saving with respect to the feasibility constraints. This algorithm needs an initial solution to start the merging process with.

In this study, the initial solution of the modified Clarke-Wright algorithm (CLK) is obtained from the Group Direct algorithm (Section 3.4.1). After an initial feasible solution is obtained, the merging begins. The Clarke-Wright has two phases. The first phase contains a sub algorithm called the "Grenade Merge", which is only executed if there are more than 500 routes in the initial solution. This method is designed to combine the cherry pick routes in a greedy logic steps (Figure 3.6).

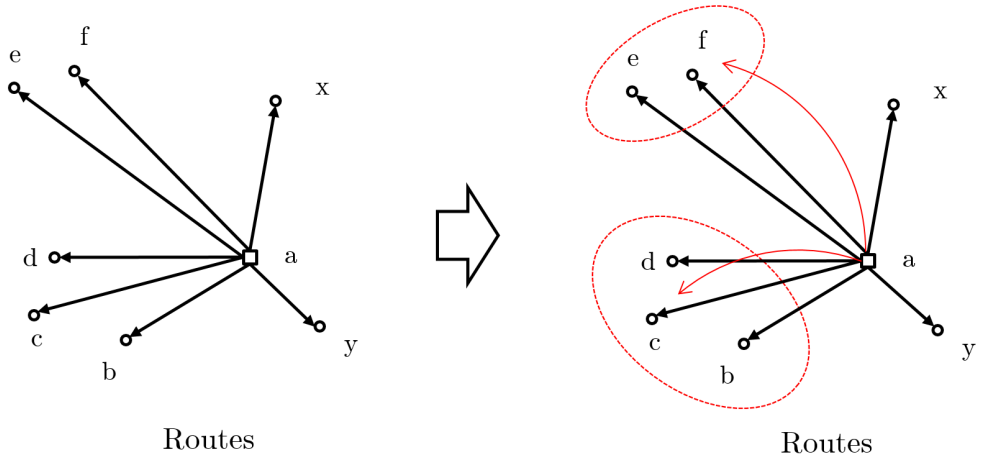


Figure 3.6: The Grenade Merge Method

The graphical representation of the Grenade method is shown in Figure 3.6. Depending on the size of the routing network, several routes are arbitrarily selected as the grenade routes. The origin and the destination points of these selected routes are used to create the blast radius pairs.

All the routes that are in these radius pairs are accepted for evaluation. The example in Figure 3.6 shows a single origin problem where all the routes originate from one origin to multiple destinations.

The second phase begins by arranging the routes based on their pick time begins and drop time ends. After that, for each route that have similar begin and end date, the code methodically checks all the possible merge pairs by trying each route with every other, and stores the savings generated. In the end, it executes the highest savings merge pair and updates all the merge pairs to reflect the new change. Iteratively this process keeps on going until all the possible combinations are exhausted. The graphical representation of the saving's methodology is shown in Figure 3.7. In Step 0, all the route pairs are combined to calculate the possible savings.

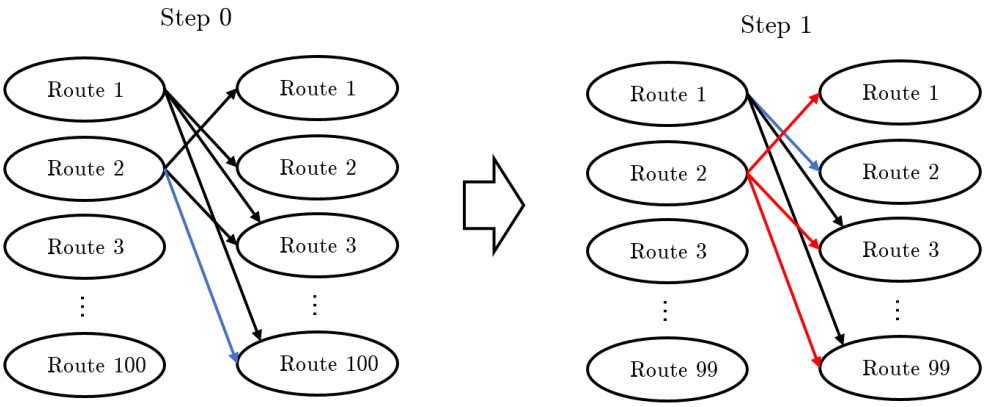


Figure 3.7: The Modified Clarke-Wright Algorithm

The example in Figure 3.7 contains 100 routes which means $99 \times 99 = 9801$ possible route pairs exist. After trying all, the code executes the best pair. The route 2 - 100 pair is the best in this example. The total route count drops to 99 since a two route merged into one, the new route 2. In the next step, first, it updates the pair combinations made by the executed pair, route 2 - 100, and then recalculates the new pairs than can be obtained with this new pair (shown with red lines). The savings list is updated again, and the merged pair that has the highest savings is selected. Step 1 is recursively repeated until there are no more route pairs left. The pseudo-code of this algorithm is provided as follows.

Clarke-Wright (Shipments)

{

Step 1 Execute the Group Direct algorithm on the Shipments to find the initial routes.

Step 2 If the number of routes in Step 1 is greater than 500, perform the Grenade Merge.

Step 3 Arrange the routes based on their pick time begins and drop time ends.

Step 4 Find the routes that have similar begin and end date. If no routes found, go to Step 7.

Step 5 Check all the possible merge pairs by trying each route with every other. Stores the savings generated. Executes the highest savings merge pair, if feasible. If not, execute the next highest savings merge pair.

Step 6 Updates all the merge pairs to reflect the new changes and go to Step 3.

Step 7 Repeat Step 5 until no savings is obtained by any of the merged pairs of routes.

Step 8 Return all the routes.

}

3.4.3 Improvement Algorithms

In Step 2, improvement procedures adjust the given network solution to improve their quality. The heuristic algorithms in this step are divided into two categories: i) inter-route and ii) intra-route categories. The first category includes the improvement algorithms that involve two routes. Improvements are made by moving order lines between the two routes of a solution. In this chapter, three such algorithms are used: i) 1-Move, ii) 1-Exchange and iii) 1-Reinsert. The algorithms in the second category work on the individual routes of a solution. This category consists three algorithms: i) Minimize Cost, ii) Break Cross and iii) 1-in-Move. Any time a solution enters to this step, all of these five algorithms will be implemented on the solution in the same order as stated above. The improvement procedure continues until no improvement is made by any of these algorithms (i.e., after making any improvement by any of these algorithms on the solution,

the improvement process restarts and all of them will be implemented on the solution again). Each algorithm is described in the following sections.

3.4.3.1 1-Move Algorithm

This algorithm considers the movement of a line from one route to another. The goal is to improve the overall savings at the iteration of the procedure. This algorithm is similar to the 1-opt algorithm described first by Lin (1965). An illustration of this algorithm is demonstrated in Figure 3.8.

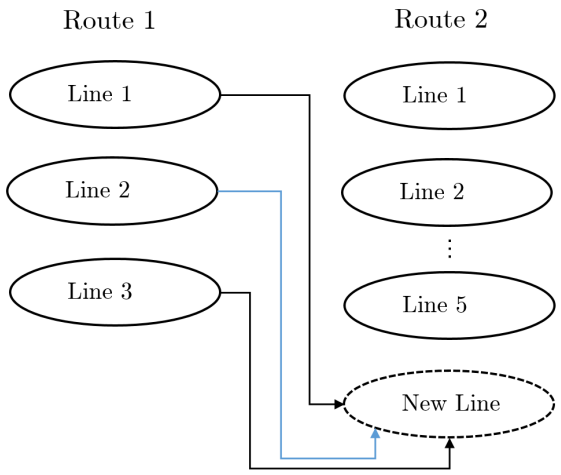


Figure 3.8: The 1-Move Algorithm

As shown in Figure 3.8, each line in route 1 is moved to route 2 one by one. After each line is moved, the algorithm checks whether a feasible combination of lines is found with respect to cost. If the cost is reduced, then the change will be kept. Otherwise, the next line is considered. Once all the lines are added or evaluated, another route is selected, and the same process is repeated until all the route pairs are tried. Selection of routes and the moving line is done exhaustively. That meant, all pair of routes are selected, and every line in the first route is moved to the second route.

One important aspect of this method is how a new line is placed into a route that already has some existing lines. It is not necessarily added to the end. Instead, it is placed in a way that minimizes the out of route miles.

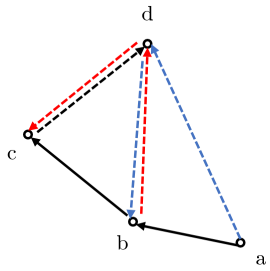


Figure 3.9: Adding a Line

An example of placing a line into a route is shown in Figure 3.9. The algorithm tries to add stop “d” to a route going through stops a−b−c. The algorithm tries to insert stop “d” in four different places: i) the first stop, ii) after stop “a”, iii) after stop “b” and iv) after stop “c”. It picks the one that minimizes the total distance, after stop “b”. If this best combination is not feasible, it does not try any other locations. The pseudo-code of this algorithm is presented as follows.

1-Move (Solution)

{

Step 1 Select a pair of routes from Solution that have not been selected before. If none, go to Step 5.

Step 2 Select a line in the first route that has not been selected before. If none, go to Step 1.

Step 3 Add the line to the second route, if possible. Record the saving and remove the line. Go to Step 2.

Step 4 Add the line with the largest saving to the second route, if feasible. Go to Step 1.

Step 5 Return Solution.

}

Note that we add the line to a route in Step 3 in the best feasible position with the largest saving by trying every possible position to add the line.

3.4.3.2 1-Exchange Algorithm

This algorithm tries to change every line in a route with every other line in all the other routes iteratively. The selection of the pair of routes is exhaustive and every combination of two lines from the two routes are tried. In the other words, the algorithm selects the first and second route of the solution and exchange the first line of the first route with the first line of the second route. If this exchange reduces the total cost of the solution, it will be performed. Each time an exchange performs, the pair of routes and total cost of the solution is updated. The next exchange happens between the first line of the first route and the second line of the second route. This continues until all the combination of lines in these two routes are exhausted. This algorithm terminates when every pair of routes in the solution are tried.

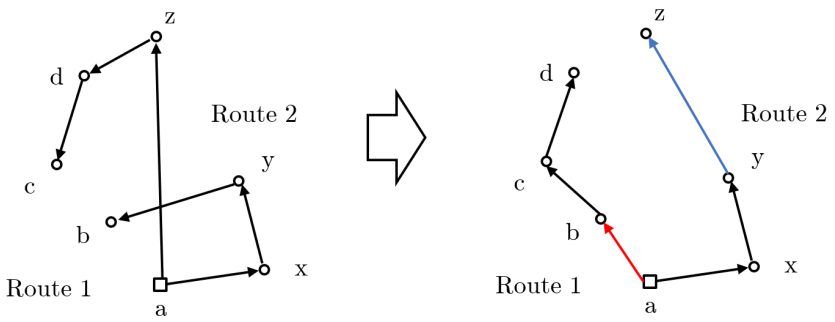


Figure 3.10: The 1-Exchange Algorithm

Figure 3.10 presents a graphical representation of this method. Line a−z on Route 1 and line y−b on Route 2 are swapped. The pseudo-code of this algorithm is stated as follows.

1-Exchange (Solution)

{

Step 1 Select a pair of routes from Solution that has not been selected before. If none, go to Step 5.

Step 2 Select a pair of lines from the routes that has not been selected before. If none, go to Step 1.

Step 3 Exchange the line between the routes, if possible. Record the saving and remove and move the lines to their original positions. Go to Step 2.

Step 4 Exchange the lines with the largest saving, if feasible. Go to Step 1.

Step 5 Return Solution.

}

Note that we exchange the line between routes by trying every possible position to add the line and select the one with the largest savings.

3.4.3.3 1-Reinsert Algorithm

The proposed 1-Reinsert algorithm is designed to move shipments between more than two routes. This newly developed algorithm removes the "worst" stop in each route of the solution and tries to reinsert them to a route that gives the most saving. The term worst can be defined as the furthest or the most costly. In this study, we remove the furthest stop in each route, say stop i, only if the sum of the distance to and from stop i in the route ($dist(i)$) is more than the average distance between stops multiplied by a user defined factor (INS_FACTOR). The average distance between stops of a route is calculated by the total travel distance of the route divided by the number of stops in the route. The algorithm is illustrated in Figure 3.11.

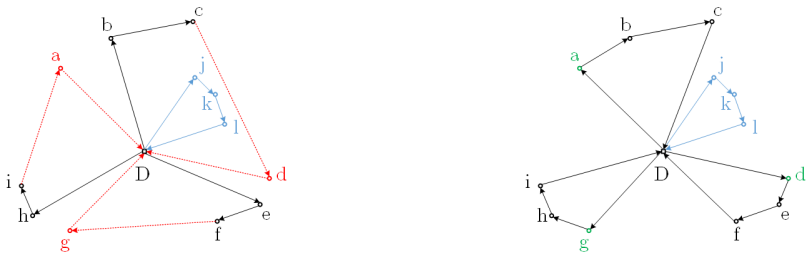


Figure 3.11: The 1-Reinsert Algorithm

In Figure 3.11, consider a solution with four routes in the network with the INS_FACTOR equal to 2. In the route "D-b-c-d-D", stop "d" is the furthest stop in the route. It means $dist(d) = dist(c,d) + dist(d,D)$ is more than the travel distances to and from stops "b" and "c" ($dist(d) > dist(b)$ and $dist(c)$). Also, $dist(d)$ is greater than the twice of the average distance between stops. Thus, stop "d" will be removed from the current route and become unrouted.

The same calculations remove stops "a" and "g" from the routes "D-h-i-a-D" and "D-e-f-g-D", respectively. Note that none of the stops of the route "D-j-k-l-D" will be removed from it because all the distances to and from each of these stops are less than the twice of the average distance between stops.

This improvement algorithm gives us the opportunity to switch stops between more than two routes. The one and two stop move algorithms switch lines between a pair of selected routes. This algorithm is implemented on the routes with more than two routes. The pseudo-code of this algorithm is provided as follows.

1-Reinsert (Solution)

{

Step 1 Find the worst line in each route of Solution. If none, go to Step 6.

Step 2 Remove the worst lines from their corresponding routes.

Step 3 Select a line from the worst lines, which has not been selected yet. If none, go to Step 6.

Step 4 Add the line to every route, if possible. Record the saving and remove the line.

Step 5 Add the line to the best route with the largest saving. Go to Step 3.

Step 6 Return Solution.

}

The worst line in Step 1 is the line that is furthest from its previous and next line. The minimum distance is provided by a user-defined factor multiple by the average of the distance of the line from each other. Similar to other algorithms, we add the line in Steps 3 and 4 to the best feasible position of a route with the largest saving.

3.4.3.4 Minimize Cost

This algorithm can find the optimal sequence of the lines in a route by trying every possible sequence of lines if the user wants to. We usually perform optimal search on a route with 5 lines or less. This number can be changed by the user. The number of sequence in this optimal search in a route with 5 lines is $5! = 120$. In the case of more than 5 lines, the Minimize Cost algorithm clusters the lines into subset of lines each with 5 lines using the a proposed K-means Cluster method and then performs the optimal search in each cluster. The K-means Cluster method is based on Lloyds algorithm (Lloyd, 1982), which is a very commonly encountered algorithm in the academia. The main goal of K-means is putting points into clusters such that the squared distance to the centroid of the cluster is minimized.

These type of methods usually start with random assignments between each point and the given *k* clusters. Based on these assignments, the cluster centroids are calculated. Then, it reas-

signs each point to the closest cluster, interactively and updates the centroids, respectively. This process is repeated until no more changes are observed. This iterative method is depicted in Figure 3.12 where each point is assigned to a cluster based on the distance to centroids. Each point is compared against the existing cluster centroids. And the closest one is picked (The red one in this case).

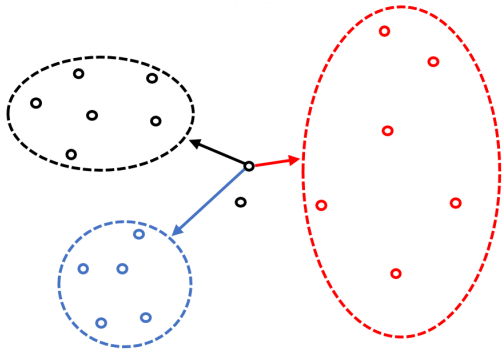


Figure 3.12: The K-means Cluster Algorithm

In ETP, K-means Cluster has used in two main areas: i) stop sequencing and ii) line pairing. The stop sequencing is the problem of reordering the stops in the route. A version of K-means Cluster is used to identify group pairs that are geographically close to each other. Line pairing is the general routing process where the optimization engine tries to pair lines to create routes. The clusters that are created using the K-means Cluster are then translated into 2-way and 3-way adjacency pairs. These pairs identify the line pair combinations that are allowed.

There is a parameter called “Cluster Ratio” which controls the allowed adjacencies. This parameter is defined as a percentage which determines the maximum normalized distance between two adjacent cluster pairs. If this value is 0 then there are no adjacencies allowed, if it is 1 then all the clusters are considered adjacent to each other. For any value in between, the code finds the distances from each cluster to the other. Then, it finds the maximum of these distances and uses it to normalize all the others. As a result, if the cluster ratio is set to 0.3, then only the cluster pairs that have less than 0.3 are allowed to create adjacencies. The pseudo-code of this algorithm is presented as follows.

Minimize Cost (Solution)

{

Step 1 Select a rout from Solution that has not been selected before. If none, go to Step 5.

Step 2 Cluster the lines of the route using K-means Cluster method.

Step 3 For each cluster, find the optimal feasible sequence of line by enumerating all sequences.

Step 4 Merge the clusters, if feasible. Go to Step 1.

Step 5 Return Solution.

}

In Step 4, all the possible combinations of clusters are tried, and the one with the largest saving is performed.

3.4.3.5 The Break Cross Algorithm

This new algorithm is designed to break the cross between two lines in a route. To do so, first, it selects two cut points in the route, and then reverses the order of lines between the two points while keeps the order of lines from the depot to the first cut point and from the second point to the depot unchanged. This new proposed algorithm is illustrated in the Figure 3.13.

Figure 3.13: The Break Cross Algorithm

As shown above, in Figure 3.13 (a) a cross between arcs “b-e” and “d-c” exists in the route. If we choose the two cut points as “b” and “f” and reverse the order of shipments between these

two points and keep the remaining unchanged, the cross will be broken and the total distance of the route will be reduced.

This algorithm is repeated for every possible pair of cut points in every route of a solution. The two cut points can be the same which means a single cut point in the route. In this case, the route is divided into two parts, first one from the depot to the cut point and the second one from the cut point to the depot. The first part of the route will be kept unchanged and the second part's order of shipments reversed. The pseudo-code of this algorithm is presented as follows.

Break Cross (Solution)

{

Step 1 Select a route in Solution that has not been selected yet. If none, Go to Step 6.

Step 2 Select two lines that have not been tried yet. If none, go to Step 1.

Step 3 Reverse the sequence of the lines between these two lines and keep the sequence of the other lines.

Step 4 Create a route with the new sequence, if possible. Record the saving. Go to Step 2.

Step 5 Create a route with the best sequence, if feasible. Go to Step 1.

Step 6 Return Solution.

}

3.4.3.6 1-in-Move

The one in-line move algorithm is similar to the one line move algorithm in Section 3.4.3.1 but in one route. In this algorithm, one line is removed from its position and inserted to another position in the route and then checked to be feasible and have a positive saving. In this research every line of each route of a solution is tried to every possible position in that route. A feasible move with the biggest saving will be performed.

3.4.4 Filtering

The filtering methods are the post-optimization method to satisfy the specific customer-defined criteria. The criteria restrict a subset of orders or routes in the network. Having these methods enables the user to have more flexibility to meet special needs of customers. These methods consider all the routes in a solution and try to remove or add lines that fit the criteria.

There are the four filters that each created route has to go through before they can be included in the final solution; weight limit, cube limit, count limit, and hours limit. They all individually check each created route and make sure that all the routes remain within the given limits.

3.4.5 Forcing

During the optimization and also due to filtering, it is natural to have lines without routes (unrouted). We use to functions to force unrouted shipments to be a part of a route: i) Unrouted Add and ii) Unrouted Switch.

The Unrouted Add function takes the unrouted lines and tries to add them back into previously created routes. The Unrouted Switch method is very similar to Unrouted Add, but instead of trying to add the unrouted lines it tries to switch them with others if there are more savings. The switched lines become unrouted, so the total unrouted count is not going to decrease, but the overall solution value is going to improve.

At the end of this step, if there is any unrouted shipments remain in the network, they will be considered as the LTL (less than truckload) shipments and will be shipped directly to the customer. On the other words, each unrouted shipment will be considered as a single shipment route.

3.5 Solutions and Results

In this section, the performance of ETP is compared with CPLEX Solver 12.6. We evaluate sixty problem instances of PHBD1 (see Section 3.3.3). We choose this model because the PHBD1 is one of the most common variants of VRP used to model customer transportation networks at

J.B. Hunt. We used a modified version of the famous VRPPD benchmark instances generated by Van Breedam (1994) to compare the results. The modifications to the benchmark instances are explained in Appendix 3.A.

The MIP formulation of PHBD1 was coded in C++ using the library of CPLEX. The library uses the IBM ILOG CPLEX solver 12.6.0 (*ibm.com*, 2013) to solve the MIP model. All test problems are run on a Core(TM) i7 CPU @ 2.93 GHz, 16 GB RAM computer.

The results of solving each benchmark instance with CPLEX and ETP, along with the running times, are reported in Table 3.7. All benchmark instances solved with CPLEX were given an 8-hour time limit, and the best feasible solution is reported.

In optimizing the benchmark instances with ETP, both NNS and CLK algorithms are executed, separately, and followed by all the improvement algorithms in Section 3.4.3. After creating a solution by either NNS or CLK, the improvement algorithms are implemented one after another in the same order as they appeared in Section 3.4.3. This process continues until no improvement on the solution obtained by any of the algorithms. No filtering method was applied to the solution. The reason is that in filtering step we put extra restrictions on the model usually demanded by customers which none of the benchmark instances have any.

The percentage of gap in objective function of PHBD1 between NNS and CPLEX (Gap_{NC}), CLK and CPLEX (Gap_{CC}), and ETP and CPLEX (Gap_{EC}) are reported in the last column. The calculation of the gap is as follows.

$$\text{Gap}_{NC} = \frac{\text{Obj}_{\text{NNS}} - \text{Obj}_{\text{CPLEX}}}{\text{Obj}_{\text{CPLEX}}} \times 100\% \tag{3.29}$$

$$\text{Gap}_{CC} = \frac{\text{Obj}_{\text{CLK}} - \text{Obj}_{\text{CPLEX}}}{\text{Obj}_{\text{CPLEX}}} \times 100\% \tag{3.30}$$

$$\text{Gap}_{EC} = \frac{\text{Obj}_{\text{ETP}} - \text{Obj}_{\text{CPLEX}}}{\text{Obj}_{\text{CPLEX}}} \times 100\%; \quad \text{where} \quad \text{Obj}_{\text{ETP}} = \min(\text{Obj}_{\text{NNS}}, \text{Obj}_{\text{CLK}}) \tag{3.31}$$

Where Obj_{NNS}, Obj_{CLK}, $\text{Obj}_{\text{CPLEX}}$ and Obj_{ETP} are the total cost of the best solution ob-

tained by NNS, CLK, CPLEX, and ETP, respectively. Note that the objective function of ETP is the minimum of the objective function of NNS and CLK, as these algorithms are a part of ETP optimization engine.

Table 3.7: Performance Comparison, ETP vs. CPLEX (Instances from Appendix 3.A)

Instance ID	NNS			CPLEX			CLK		ETP		
	Obj.	**Time**	**Gap$_{NC}$**	**Obj.**	**Time**	**Gap$_{CC}$**	**Obj.**	**Time**	**Obj.**	**Time**	**Gap$_{EC}$**
1P1	1071.2	2 s	1.3%	**1057.1**	8 h	4.3%	1102.6	2.8 s	1071.2	2 s	1.3%
2P1	1090.3	3.4 s	2.8%	**1060.1**	8 h	4.6%	1109.3	2.5 s	1090.3	3.4 s	2.8%
3P1	1124.6	2.9 s	0.0%	1124.7	8 h	0.0%	1124.6	5.7 s	1124.6	2.9 s	0.0%
4P1	1110.1	2.1 s	9.6%	1013.0	8 h	-1.2%	1000.7	2.1 s	**1000.7**	2.1 s	-1.2%
5P1	1115.9	2.2 s	20.4%	927.1	8 h	-5.7%	874	2.2 s	**874**	2.2 s	-5.7%
6P1	1130.7	3.1 s	-3.7%	1174.0	8 h	-10.8%	1046.8	2.1 s	**1046.8**	2.1 s	-10.8%
7P1	1301.7	2.8 s	-9.5%	1438.0	8 h	-12.7%	1255.5	2.8 s	**1255.5**	2.8 s	-12.7%
8P1	1376.9	3.7 s	-1.3%	1394.7	8 h	-3.9%	1340.2	1.9 s	**1340.2**	1.9 s	-3.9%
9P1	1521.6	5.4 s	-0.1%	1523.8	8 h	-2.6%	1483.6	2.1 s	**1483.6**	2.1 s	-2.6%
10P1	1182.3	5.3 s	1.4%	1165.7	8 h	-2.3%	1138.5	4.5 s	**1138.5**	4.5 s	-2.3%
11P1	1204.5	4.1 s	-12.2%	1371.3	8 h	-7.0%	1275.2	4.3 s	**1204.5**	4.1 s	-12.2%
12P1	1400.3	8.1 s	–	–	8 h	–	1336.8	3.4 s	1336.8	3.4 s	–
13P1	1187.0	3.2 s	16.4%	1020.1	8 h	8.3%	1104.3	2.5 s	1104.3	2.5 s	8.3%
14P1	1133.4	2.9 s	-5.1%	1194.6	8 h	-4.9%	1136	1.7 s	**1133.4**	2.9 s	-5.1%
15P1	1290.6	6.6 s	-15.1%	1520.4	8 h	-13.2%	1319.8	2.7 s	**1290.6**	6.6 s	-15.1%
16P1	774.7	3.6 s	–	–	8 h	–	856.6	3.2 s	774.7	3.6 s	–
17P1	806.5	2.9 s	-1.7%	820.6	8 h	-1.7%	806.5	2.4 s	**806.5**	2.9 s	-1.7%
18P1	963.6	2.8 s	-10.4%	1074.9	8 h	-10.4%	963.6	2.3 s	**963.6**	2.8 s	-10.4%
19P1	702.4	2.7 s	5.1%	**668.6**	8 h	11.4%	745.1	2.8 s	702.4	2.7 s	5.1%
20P1	670.6	4 s	-8.8%	735.0	8 h	-4.4%	702.6	2 s	**670.6**	4 s	-8.8%
21P1	811.7	2 s	-3.2%	838.4	8 h	-1.1%	829.3	2.6 s	**811.7**	2 s	-3.2%
22P1	777.0	1.8 s	-18.2%	949.7	8 h	-12.1%	834.4	2.1 s	**777**	1.8 s	-18.2%
23P1	833.4	3.1 s	-14.7%	976.9	8 h	-5.1%	926.6	2.1 s	**833.4**	3.1 s	-14.7%
24P1	1039.4	6.6 s	-16.9%	1250.8	8 h	-19.0%	1013.2	3.3 s	**1013.2**	3.3 s	-19.0%
25P1	669.7	2.3 s	-1.7%	681.4	8 h	-1.7%	669.7	2.3 s	**669.7**	2.3 s	-1.7%
26P1	721.3	2 s	-24.8%	959.5	8 h	-24.8%	721.3	2.3 s	**721.3**	2 s	-24.8%
27P1	893.5	3.1 s	–	–	8 h	–	893.5	2.3 s	893.5	3.1 s	–
28P1	828.0	4.4 s	-1.6%	841.7	8 h	-1.6%	828	2.3 s	**828**	4.4 s	-1.6%
29P1	879.0	4.7 s	-20.4%	1103.9	8 h	-20.4%	879	2.4 s	**879**	4.7 s	-20.4%
30P1	1046.3	4.7 s	-21.5%	1333.4	8 h	-12.5%	1167.3	3.3 s	**1046.3**	4.7 s	-21.5%
31P1	583.5	4.3 s	7.5%	542.6	8 h	0.0%	542.6	2.1 s	542.6	2.1 s	0.0%
32P1	757.2	2.9 s	10.7%	683.9	8 h	-0.8%	678.2	2.2 s	**678.2**	2.2 s	-0.8%
33P1	949.7	3.3 s	-10.2%	1057.8	8 h	-6.1%	992.8	3.0 s	**949.7**	3.3 s	-10.2%
34P1	679.1	2.1 s	29.8%	**523.2**	8 h	9.3%	571.9	1.7 s	571.9	1.7 s	9.3%

Continued on the next page

Table 3.7 – Continued from the previous page

Instance ID	NNS			CPLEX			CLK		ETP		
	Obj.	**Time**	**Gap$_{NC}$**	**Obj.**	**Time**	**Gap$_{CC}$**	**Obj.**	**Time**	**Obj.**	**Time**	**Gap$_{NCC}$**
35P1	683.2	4.3 s	-4.5%	715.2	8 h	-9.6%	646.4	2.9 s	**646.4**	2.9 s	-9.6%
36P1	932.5	4.7 s	-1.7%	948.9	8 h	-8.3%	869.8	3.0 s	**869.8**	3 s	-8.3%
37P1	798.2	8.1 s	9.6%	728.5	8 h	-0.9%	722.3	2.2 s	**722.3**	2.2 s	-0.9%
38P1	879.0	5.3 s	4.7%	839.6	8 h	-7.5%	777	4.7 s	**777**	4.7 s	-7.5%
39P1	937.1	3.9 s	-13.1%	1078.8	8 h	-13.1%	937.1	5.0 s	**937.1**	3.9 s	-13.1%
40P1	851.3	7.0 s	-2.7%	875.2	8 h	-10.5%	783.5	2.3 s	**783.5**	2.3 s	-10.5%
41P1	968.7	3.2 s	14.4%	**846.8**	8 h	2.9%	871.1	2.3 s	871.1	2.3 s	2.9%
42P1	1035.7	3.1 s	–	–	8 h	–	1110.6	2.3 s	1035.7	3.1 s	–
43P1	827.8	4.9 s	-11.4%	933.8	8 h	-14.9%	794.4	2.1 s	**794.4**	2.1 s	-14.9%
44P1	897.6	5.4 s	–	–	8 h	–	935.1	2 s	897.6	5.4 s	–
45P1	1109.9	4.6 s	-10.4%	1238.6	8 h	-6.6%	1156.5	2.3 s	**1109.9**	4.6 s	-10.4%
46P1	777.2	2.8 s	-0.6%	781.7	8 h	0.3%	784.2	2.1 s	**777.2**	2.8 s	-0.6%
47P1	710.0	1.6 s	0.0%	710.1	8 h	0.0%	710	2.3 s	710	1.6 s	0.0%
48P1	870.9	2.1 s	0.0%	871.0	8 h	0.0%	870.9	2.4 s	870.9	2.1 s	0.0%
49P1	674.8	2.7 s	8.6%	621.1	8 h	-1.5%	611.8	1.5 s	**611.8**	1.5 s	-1.5%
50P1	749.8	2.6 s	–	–	8 h	–	686	3.2 s	686	3.2 s	–
51P1	924.7	3.3 s	-2.9%	952.1	8 h	6.7%	1015.7	2.8 s	**924.7**	3.3 s	-2.9%
52P1	996.6	2.4 s	–	–	8 h	–	984.4	1.5 s	984.4	1.5 s	–
53P1	1102.5	2.5 s	-17.1%	1329.7	8 h	-26.8%	973	1.9 s	**973**	1.9 s	-26.8%
54P1	1443.6	2.8 s	-6.2%	1538.5	8 h	-17.9%	1262.7	1.8 s	**1262.7**	1.8 s	-17.9%
55P1	1045.5	2.9 s	-4.4%	1094.1	8 h	-2.5%	1067	3.2 s	**1045.5**	2.9 s	-4.4%
56P1	1151.9	2.7 s	-4.4%	1205.3	8 h	-11.8%	1062.9	1.5 s	**1062.9**	1.5 s	-11.8%
57P1	1340.5	5.7 s	-12.4%	1530.4	8 h	-14.2%	1312.5	2.3 s	**1312.5**	2.3 s	-14.2%
58P1	1057.7	1.5 s	3.6%	1021.1	8 h	-14.0%	878.6	2.8 s	**878.6**	2.8 s	-14.0%
59P1	1031.9	2.3 s	-10.3%	1149.9	8 h	-12.6%	1005	1.7 s	**1005**	1.7 s	-12.6%
60P1	1240.6	2.2 s	-5.3%	1310.2	8 h	-3.7%	1261.6	4.8 s	**1240.6**	2.2 s	-5.3%

In Table 3.7, the objective function and running time of the 60 benchmark instances solved by NNS, CLK and CPLEX methods are compared to each other. The NNS and CLK are parts of ETP optimization engine and the best performance of them are considered as the performance of ETP (last three columns). These two algorithms are used separately to create the feasible solutions (routes). After the solution obtained by each of these algorithms, the same improving algorithms (all improving algorithms in Section 3.4.3) are performed on the solution to reduce the objective function (total cost). Gap$_{NC}$ column (located between NNS and CPLEX columns) com-

pares the final solution of NNS and CPLEX regarding objective function. Negative sign shows that NNS performs better that CPLEX. Similarly, Gap_{CC} column compares the performance of CLK and CPLEX. The minimum of Gap_{NC} and Gap_{CC} are highlighted in gray in the instances that either of NNS or CLK has a better smaller objective function that CPLEX. The best objective function of ETP or CPLEX for each instance is shown in bold.

According to the results of experiments reported in Table 3.7, ETP outperforms CPLEX regarding solution obtained in 45 out of 60 instances (75%). To better illustrate the strength of ETP over CLEX the instances with negative Gap_{NNS} are shown in bold and the gap of the associated method is underlined. In 2 instances (3%), they produced the same results and finally, in 6 instances (10%) CPLEX was better than ETP. All the instances solved with CPLEX were terminated if an optimal solution had not been found within 8 hours. In 7 instances (12%), CPLEX could not find a feasible solution in the specified time. Regarding solution time, all the instances solved by ETP required less than 5 seconds. With CPLEX, no instance was solved to optimality in 8 hours. Both NNS with CLK have positive impacts on the performance of ETP, each in some instances. The running time of them are almost the same. However, the overall performance of CLK is better than CLK, specifically in 29 instances (48%). They both obtained the same results in 11 instances (19%) and in 20 instances (33%) CLK performed better.

In Figure 3.14, the objective function of CPLEX versus NNS and CLK solver are shown. In this figure, the objective function (total cost) of the problem instances that have been solved by these three algorithms are reported.

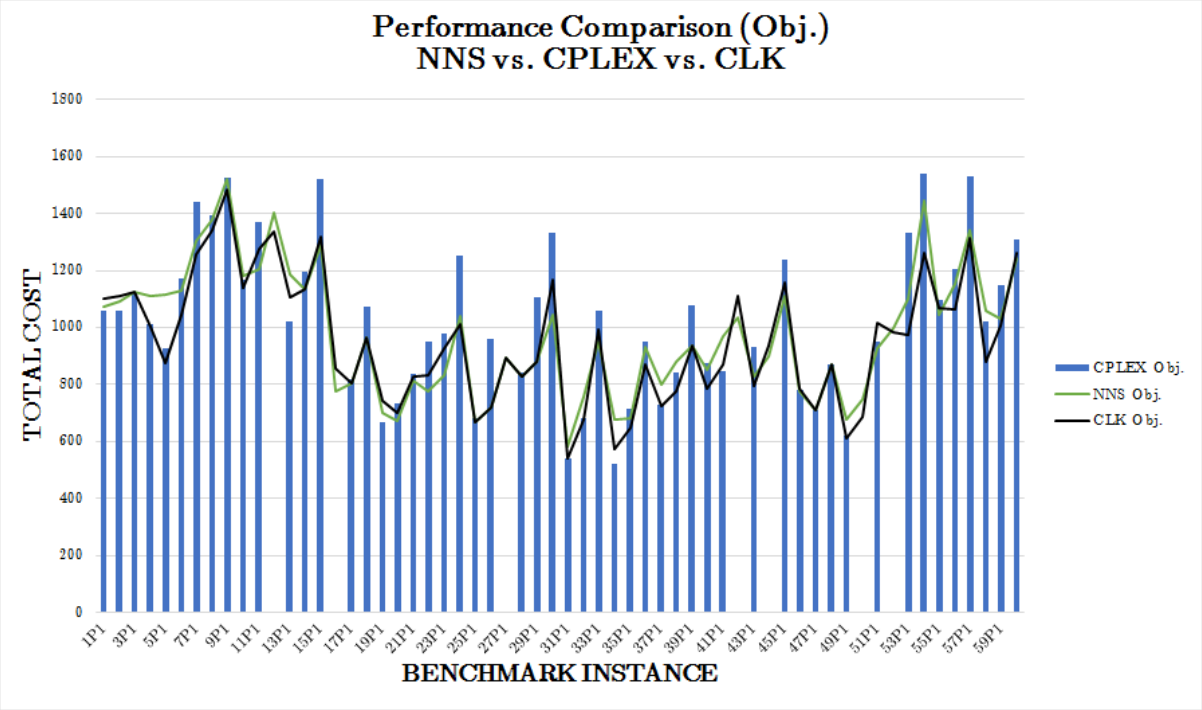


Figure 3.14: Performance on PHBD1, NNS vs. CPLEX vs. CLK

As shown in the above figure, in 15 instances NNS performed better, in 35 instances performed worse and in 3 instances performed the same as CPLEX. On the other hand, CLK outperformed CPLEX in 43 instances, got the same solution in 8 instances and in only 2 instances did worse than CPLEX.

Figure 3.15, illustrates the performance of ETP which is the base solution between NNS and CLK versus CPLEX regarding the objective function.

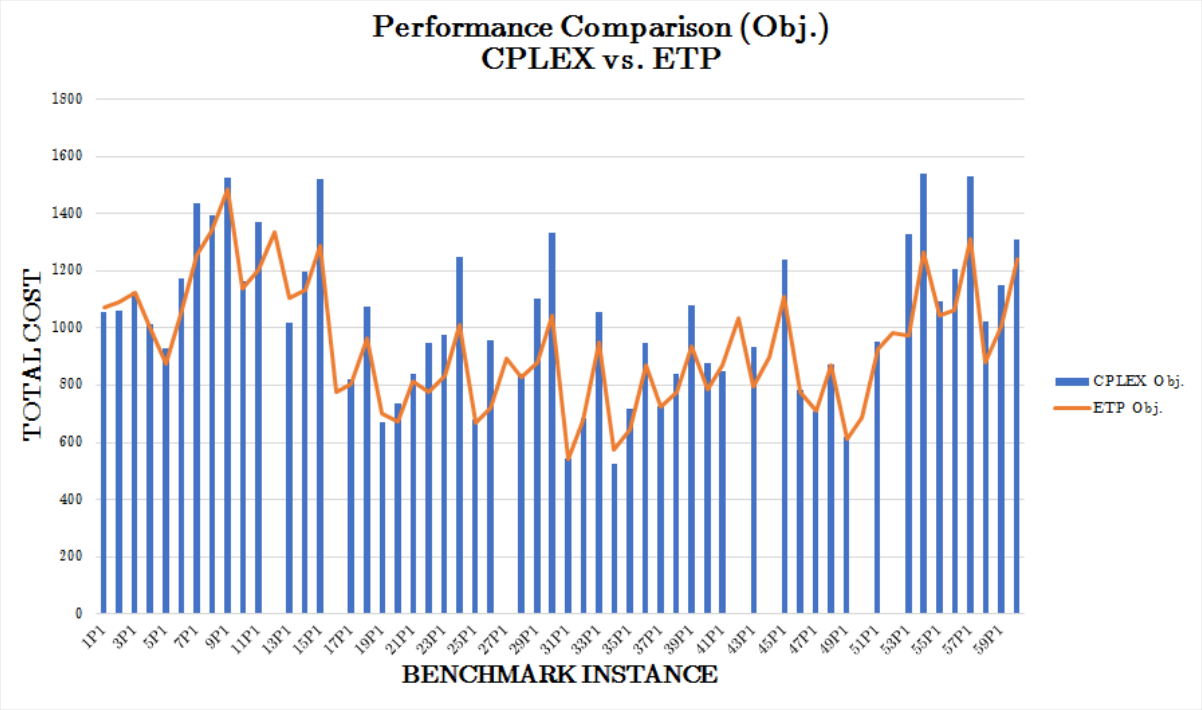


Figure 3.15: Performance on PHBD1, CPLEX vs. ETP

As shown in Figure 3.15, the overall performance of ETP is much better than CPLEX in terms of both objective function and running time. In the worst case, the performance of ETP is 9.3% worse than the CPLEX solver (in 1P34 instance). In the best case, the solution found by ETP is 26.8% better than the CPLEX solver (in instance 1P53). More extensive computational study on individual heuristic algorithms used in optimizing these instances is provided in the next chapter (Chapter 4).

3.5.1 A Case Study

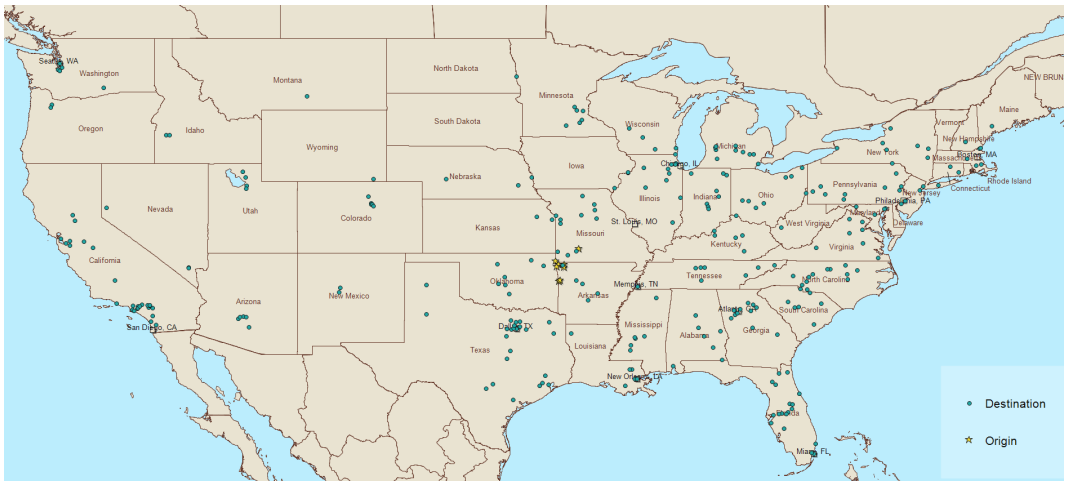
In this section, to show the strength of ETP in solving instances with customer networks consisting of thousands of locations and hundreds of vehicles, we present and solve a real dataset provided by J.B. Hunt. The dataset is associated with the food industry with the characteristics described in Table 3.8.

Table 3.8: The Case Study Characteristics

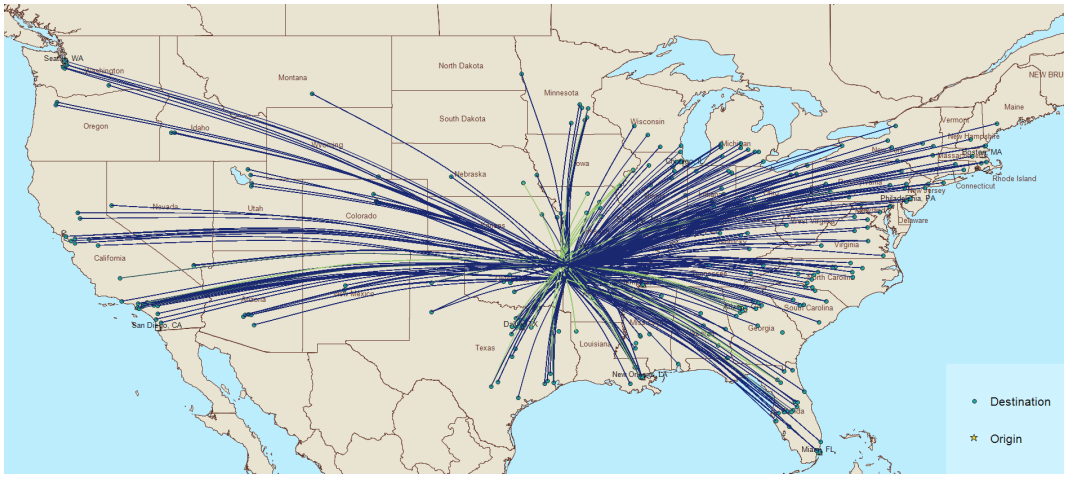
Item	Value/Description
Locations:	
• Total	292
◦ Origins	8
◦ Destinations	284
Shipments:	
• Total	14,356
◦ Fresh	4,009
◦ Frozen	10,347
• Average weight	18,718 lbs
• Average pallet count	17 pallets
• Average length of haul	773 miles
Model:	
• Truck capacity	
◦ Weight	44,000 lbs
◦ Number of pallets	57 pallets
• Fleet of trucks	Heterogeneous (fresh & frozen)

This network is modeled as the multiple shifts, pickup delivery, heterogeneous fleet, open VRP with backhauls, time windows, and DOT regulation (PHOBTD). Before optimizing this the network, the graphical representation of the nodes and shipments is presented using ETP's graphical tools.

Figure 3.16 shows the shipping network before optimizing by ETP. The origin and destinations locations and the shipments between these nodes are depicted in Figure 3.16 (a) and (b), respectively. In Figure 3.16 (b) the fresh shipments are shown in green and the frozen ones in blue.



(a) The network points: origins (stars) and destinations (dots)



(b) The network shipments: fresh (green) and frozen (blue).

Figure 3.16: The graphical representation of the shipment network of the case study

The result summary of optimizing the network using ETP is reported in Table 3.9. 572 multi-stop routes are created by ETP to transport the shipments.

Table 3.9: The Optimization summary of the case study network by ETP

Solution Method	Mode	Miles	Routes	Orders	Weight (lb)
Direct					
	LTL	11,097,679	14,356	14,356	323,284,553
ETP Optimization					
	CTL	27,372	62	202	1,424,916
Run Time:	LTL	10,147,321	12,860	12,860	312,628,972
$2:30':36''$	MSTL	245,347	298	1,294	9,230,665
	Total	10,420,039	13,220	14,356	323,284,553
	Improv.	6.11%	7.91%	–	–

In Table 3.9, results of this case study are shown by comparing with and without ETP. Without ETP, the direct method utilized in industry, which only makes use of LTL (less than truckload) transportation is used. However, in the ETP modes of CTL (customized truck load), LTL, and MSTL (multi-stop truckload) are used to transport the shipments. According to the table, ETP saved more than 6% in total miles of shipments and nearly 8% in the number of the shipments in only 2 h, 30 m, 36 sec of solution time.

3.6 Conclusion

In this research, variants of the Vehicle Routing Problem (VRP) serving the transportation logistics industry are studied. New VRP variants that adhere to working hour regulations for truck drivers are proposed. These variants restrict the hours of service by the truck drivers according to the regulations imposed by the U.S. Department of Transportation (DOT). The DOT regulations limit the work and drive hours of drivers in a daily and weekly shifts and impose minimum hours of rest between them.

A comprehensive decision tool that integrates multiple solution techniques referred to as Enterprise Transportation Planning (ETP) is developed to address the interests of both academic and industry parties. ETP yields quality solutions in a short time that are flexible to a wide range of problems and are easy to make a use of that are desired by industry and can compete with the

more sophisticated optimization techniques in academia. Several solution techniques, primarily based on customized heuristic methods, are incorporated into the optimization engine of ETP.

To test the quality of solutions of ETP we compared the performance of ETP versus the exact method of finding optimal solution using CPLEX. The results have shown that ETP can produce a quality solutions in a very short amount of time comparing with CPLEX. Another advantage of using ETP is the size of the problems that can be optimized. CPLEX can only solve small problem instances around 50 locations with 5 vehicles while ETP optimizes a transportation network including thousands of customer locations with hundreds of vehicles.

ETP is a very flexible integrated optimization tool that can solve a variety of VRP instances all together. ETP can solve the multiple shifts, pickup delivery, heterogeneous fleet, open allowed, multi-depot, split delivery, capacitated VRP with backhauls, time windows, simultaneous pick-drops and DOT regulations in a resonable time and obtain quality solutions.

In future, we try to add new exact and heuristic methods to ETP while improving the current heuristic algorithms. Also we want to automate the selection of best input parameters and optimization methods in ETP.

Bibliography

Ascheuer, N., Jünger, M., and Reinelt, G. (2000). A branch & cut algorithm for the asymmetric traveling salesman problem with precedence constraints. Computational Optimization and Applications, 17(1):61–84.

Baldacci, R., Battarra, M., and Vigo, D. (2008). Routing a heterogeneous fleet of vehicles. pages 3–27.

Balinski, M. L. and Quandt, R. E. (1964). On an integer program for a delivery problem. Operations Research, 12(2):300–304.

Bent, R. and Van Hentenryck, P. (2006). A two-stage hybrid algorithm for pickup and delivery vehicle routing problems with time windows. Computers & Operations Research, 33(4):875–893.

Berbeglia, G., Cordeau, J.-F., Gribkovskaia, I., and Laporte, G. (2007). Static pickup and delivery problems: a classification scheme and survey. Top, 15(1):1–31.

Braekers, K., Ramaekers, K., and Van Nieuwenhuyse, I. (2016). The vehicle routing problem: State of the art classification and review. Computers & Industrial Engineering, 99:300–313.

Brandão, J. (2004). A tabu search algorithm for the open vehicle routing problem. European Journal of Operational Research, 157(3):552–564.

Bräysy, O. and Gendreau, M. (2005a). Vehicle routing problem with time windows, part i: Route construction and local search algorithms. Transportation science, 39(1):104–118.

Bräysy, O. and Gendreau, M. (2005b). Vehicle routing problem with time windows, part ii: Metaheuristics. Transportation science, 39(1):119–139.

Christofides, N., Mingozzi, A., and Toth, P. (1981). Exact algorithms for the vehicle routing problem, based on spanning tree and shortest path relaxations. Mathematical programming, 20(1):255–282.

Clarke, G. and Wright, J. W. (1964). Scheduling of vehicles from a central depot to a number of delivery points. Operations research, 12(4):568–581.

Cordeau, J.-F., Gendreau, M., Laporte, G., Potvin, J.-Y., and Semet, F. (2002). A guide to vehicle routing heuristics. Journal of the Operational Research society, pages 512–522.

Cordeau, J.-F., Laporte, G., and Mercier, A. (2001). A unified tabu search heuristic for vehicle routing problems with time windows. Journal of the Operational research society, 52(8):928–936.

Cordeau, J.-F., Laporte, G., and Ropke, S. (2008). Recent models and algorithms for one-to-one pickup and delivery problems. pages 327–357.

Dantzig, G. B. and Ramser, J. H. (1959). The truck dispatching problem. Management science, 6(1):80–91.

Desrochers, M. and Verhoog, T. (1989). A matching based savings algorithm for the vehicle routing problem. Cahiers du GERAD.

Duhamel, C., Potvin, J.-Y., and Rousseau, J.-M. (1997). A tabu search heuristic for the vehicle routing problem with backhauls and time windows. Transportation science, 31(1):49–59.

Fisher, M. L. (1994). Optimal solution of vehicle routing problems using minimum k-trees. Operations research, 42(4):626–642.

Fleszar, K., Osman, I. H., and Hindi, K. S. (2009). A variable neighbourhood search algorithm for the open vehicle routing problem. European Journal of Operational Research, 195(3):803–809.

Goetschalckx, M. and Jacobs-Blecha, C. (1989). The vehicle routing problem with backhauls. European Journal of Operational Research, 42(1):39–51.

Golden, B. L., Raghavan, S., and Wasil, E. A. (2008). The vehicle routing problem: latest advances and new challenges, volume 43. Springer Science & Business Media.

Hernández-Pérez, H. and Salazar-González, J.-J. (2004a). A branch-and-cut algorithm for a traveling salesman problem with pickup and delivery. Discrete Applied Mathematics, 145(1):126–139.

Hernández-Pérez, H. and Salazar-González, J.-J. (2004b). Heuristics for the one-commodity pickup-and-delivery traveling salesman problem. Transportation Science, 38(2):245–255.

Hernández-Pérez, H. and Salazar-González, J.-J. (2007). The one-commodity pickup-and-delivery traveling salesman problem: Inequalities and algorithms. Networks, 50(4):258–272.

Hernández-Pérez, H. and Salazar-González, J.-J. (2009). The multi-commodity one-to-one pickup-and-delivery traveling salesman problem. European Journal of Operational Research, 196(3):987–995.

Johnson, D. S. and McGeoch, L. A. (1997). The traveling salesman problem: A case study in local optimization. Local search in combinatorial optimization, 1:215–310.

Kumar, S. N. and Panneerselvam, R. (2012). A survey on the vehicle routing problem and its variants. Intelligent Information Management, 4(03):66.

Lau, H. C. and Liang, Z. (2002). Pickup and delivery with time windows: Algorithms and test case generation. International Journal on Artificial Intelligence Tools, 11(03):455–472.

Li, F., Golden, B., and Wasil, E. (2007). The open vehicle routing problem: Algorithms, large-scale test problems, and computational results. Computers & Operations Research, 34(10):2918–2930.

Lin, S. (1965). Computer solutions of the traveling salesman problem. The Bell System Technical Journal, 44(10):2245–2269.

Lloyd, S. (1982). Least squares quantization in pcm. IEEE transactions on information theory, 28(2):129–137.

Miller, C. E., Tucker, A. W., and Zemlin, R. A. (1960). Integer programming formulation of traveling salesman problems. Journal of the ACM (JACM), 7(4):326–329.

Miller, D. L. (1995). A matching based exact algorithm for capacitated vehicle routing problems. ORSA Journal on Computing, 7(1):1–9.

Min, H. (1989). The multiple vehicle routing problem with simultaneous delivery and pick-up points. Transportation Research Part A: General, 23(5):377–386.

Montoya-Torres, J. R., Franco, J. L., Isaza, S. N., Jiménez, H. F., and Herazo-Padilla, N. (2015). A literature review on the vehicle routing problem with multiple depots. Computers & Industrial Engineering, 79:115–129.

Murray, D. C. and Short, J. (2015). Quantifying impacts from 34 hr restart provision. http://atri-online.org/wp-content/uploads/2015/04/Quantifying-Impacts-from-34-hr-Restart-Provision-FINAL-04-2015.pdf. April 2015.

Nagata, Y. and Bräysy, O. (2009). Edge assembly-based memetic algorithm for the capacitated vehicle routing problem. Networks, 54(4):205.

Nagata, Y., Bräysy, O., and Dullaert, W. (2010). A penalty-based edge assembly memetic algorithm for the vehicle routing problem with time windows. Computers & operations research, 37(4):724–737.

Nanry, W. P. and Barnes, J. W. (2000). Solving the pickup and delivery problem with time windows using reactive tabu search. Transportation Research Part B: Methodological, 34(2):107–121.

Analysis Division, Federal Motor Carrier Safety Administration (2011). 2010-2011 hours of service rule regulatory impact analysis. https://www.fmcsa.dot.gov/sites/fmcsa.dot.gov/files/docs/2011_HOS_Final_Rule_RIA.pdf. RIN 2126-AB26.

Osman, I. H. (1993). Metastrategy simulated annealing and tabu search algorithms for the vehicle routing problem. Annals of operations research, 41(4):421–451.

Pisinger, D. and Ropke, S. (2007). A general heuristic for vehicle routing problems. Computers & operations research, 34(8):2403–2435.

Potvin, J.-Y. and Bengio, S. (1996). The vehicle routing problem with time windows part ii: genetic search. INFORMS journal on Computing, 8(2):165–172.

Prins, C. (2004). A simple and effective evolutionary algorithm for the vehicle routing problem. Computers & Operations Research, 31(12):1985–2002.

Psaraftis, H. N. (1983). k-interchange procedures for local search in a precedence-constrained routing problem. European Journal of Operational Research, 13(4):391–402.

Reimann, M., Doerner, K., and Hartl, R. F. (2003). Analyzing a unified ant system for the vrp and some of its variants. pages 300–310.

Renaud, J., Boctor, F. F., and Laporte, G. (1996). An improved petal heuristic for the vehicle routeing problem. Journal of the operational Research Society, 47(2):329–336.

Rochat, Y. and Taillard, É. D. (1995). Probabilistic diversification and intensification in local search for vehicle routing. Journal of heuristics, 1(1):147–167.

Ropke, S., Cordeau, J.-F., and Laporte, G. (2007). Models and branch-and-cut algorithms for pickup and delivery problems with time windows. Networks, 49(4):258–272.

Ruland, K. and Rodin, E. (1997). The pickup and delivery problem: Faces and branch-and-cut algorithm. Computers & mathematics with applications, 33(12):1–13.

Ryan, D. M., Hjorring, C., and Glover, F. (1993). Extensions of the petal method for vehicle routeing. Journal of the Operational Research Society, 44(3):289–296.

Sariklis, D. and Powell, S. (2000). A heuristic method for the open vehicle routing problem. Journal of the Operational Research Society, 51(5):564–573.

Short, J. (2013a). Assessing the impacts of the 34-hour restart provisions. http://atri-online.org/wp-content/uploads/2015/09/ATRI-HOS-Restart-Impacts-06-13-FINAL.pdf. June 2013.

Short, J. (2013b). Operational and economic impacts of the new hours-of-service. https://www.protectmycdl.com/wp-content/uploads/2013/11/ATRI-Operational-and-Economic-Impacts-of-New-HOS.pdf. November 2013.

Spoorendonk, S. and Desaulniers, G. (2010). Clique inequalities applied to the vehicle routing problem with time windows. Infor, 48(1):53.

Subramanian, A., Uchoa, E., and Ochi, L. S. (2013). A hybrid algorithm for a class of vehicle routing problems. Computers & Operations Research, 40(10):2519–2531.

Tarantilis, C. D. (2005). Solving the vehicle routing problem with adaptive memory programming methodology. Computers & Operations Research, 32(9):2309–2327.

Federal Highway Administration (2012). Freight management and operations, federal highway administration, u.s. department of transportation. https://ops.fhwa.dot.gov/freight/freight_analysis/nat_freight_stats/docs/12factsfigures/table3_10.htm. Accessed: Oct 2017.

fmcsa.dot.gov. Summary of hours of service regulations. Accessed: 2014-12-16.

ibm.com (2013). ibm ilog cplex optimization studio v12.6.0. https://www.ibm.com/support/knowledgecenter/SSSA5P_12.6.0/ilog.odms.studio.help/Optimization_Studio/topics/COS_home.html.

Toth, P. and Vigo, D. (1997). An exact algorithm for the vehicle routing problem with backhauls. Transportation science, 31(4):372–385.

Toth, P. and Vigo, D. (2003). The granular tabu search and its application to the vehicle-routing problem. Informs Journal on computing, 15(4):333–346.

Toth, P. and Vigo, D. (2014). Vehicle routing: problems, methods, and applications, volume 18. Siam.

Van Breedam, A. (1994). An Analysis of the Behavior of Heuristics for the Vehicle Routing Problem for a Selectrion of Problems with Vehicle-related, Customer-related, and Time-related Constraints. RUCA.

Vidal, T., Crainic, T. G., Gendreau, M., Lahrichi, N., and Rei, W. (2012). A hybrid genetic algorithm for multidepot and periodic vehicle routing problems. Operations Research, 60(3):611–624.

Vidal, T., Crainic, T. G., Gendreau, M., and Prins, C. (2013). A hybrid genetic algorithm with adaptive diversity management for a large class of vehicle routing problems with time-windows. Computers & Operations Research, 40(1):475–489.

Wark, P. and Holt, J. (1994). A repeated matching heuristic for the vehicle routeing problem. Journal of the Operational Research Society, 45(10):1156–1167.

Young, J. (2013). Simulation-based truck fleet analysis to study the impact of federal motor carrier safety administration's 2013 hours of service regulation changes. In Simulation Conference (WSC), 2013 Winter, pages 3395–3405. IEEE.

Zachariadis, E. E. and Kiranoudis, C. T. (2010). A strategy for reducing the computational complexity of local search-based methods for the vehicle routing problem. Computers & operations research, 37(12):2089–2105.

Zhong, Y. and Cole, M. H. (2005). A vehicle routing problem with backhauls and time windows: a guided local search solution. Transportation Research Part E: Logistics and Transportation Review, 41(2):131–144.

Appendix

Appendix 3.A Modified Breedam's Benchmark Instances

The modifications on the PDVRP benchmark instances are as follows.

- In all of the instances, the number of locations is reduced from 100 to 50 by keeping the first two current locations and removing the next two consecutive locations. That means in each instance, the stop numbers are reduced from $\{0,1,2,3,4,5,6,\ldots,97,98,99,100\}$ to $\{0,1,2,5,6,\ldots,99,100\}$.
- The capacity of each vehicle is reduced from 100 to 50.

The size of instances are reduced, so that they could be solved by CPLEX.

Appendix 3.B The PHTD Variant with One Week Time Horizon

Table 3.10: The Modified Set of Table 3.3 for the PHTD7 Formulation

Notation	Description
• $S=\{1,\ldots,d\},\ d=7$ or 8	The shifts (days) of a week.

$$\min \sum_{t\in S}\sum_{k\in K}\sum_{i\in V}\sum_{j\in V} c_{ij}x^t_{ijk} \qquad \text{(The PHTD7)}$$

$$\sum_{t\in S}\sum_{k\in K}\sum_{j\in V} x^t_{ijk} = 1 \qquad \forall i\in P \tag{3.32}$$

$$\sum_{t\in S}\sum_{j\in V} x^t_{ijk} - \sum_{t\in S}\sum_{j\in V} x^t_{n+i,j,k} = 0 \qquad \forall i\in P, k\in K \tag{3.33}$$

$$\sum_{t\in S}\sum_{j\in V} x^t_{jik} - \sum_{t\in S}\sum_{j\in V} x^t_{ijk} = 0 \qquad \forall i\in P\cup D, k\in K \tag{3.34}$$

$$\sum_{t\in S}\sum_{j\in P} x^t_{0jk} = 1 \qquad \forall k\in K \tag{3.35}$$

$$\sum_{t\in S}\sum_{i\in V} x^t_{i,2n+1,k} = 1 \qquad \forall k\in K \tag{3.36}$$

$$Q_{ik} - Q_{jk} + q_j \le M(1-x^t_{ijk}) \qquad \forall i,j\in V, k\in K, t\in S \tag{3.37}$$

$$\max\{0,q_i\} \le Q_{ik} \le \min\{C_k,\ C_k+q_i\} \qquad \forall i\in V, k\in K \tag{3.38}$$

$$\sum_{i\in V}\sum_{j\in V}(h_{ij}x^t_{ijk} + s_j x^t_{ijk}) \le \text{D\_A} \qquad \forall k\in K, t\in S \tag{3.39}$$

$$\sum_{i\in V}\sum_{j\in V} h_{ij}x^t_{ijk} \le \text{D\_D} \qquad \forall k\in K, t\in S \tag{3.40}$$

$$\sum_{t\in S}\sum_{i\in V}\sum_{j\in V} h_{ij}x^t_{ijk} \le \text{D\_WD} \qquad \forall k\in K \tag{3.41}$$

$$T_{ik} - T_{jk} + s_i + h_{ij} \le \text{D\_R}\left(1 - \left| x^t_{ijk} - \sum_{w\in V} x^{t+1}_{jwk} \right|\right) \qquad \forall i\in V;\ j\in P\cup D\cup\{2n+1\};\ k\in K;\ t\in S \tag{3.42}$$

$$T_{ik} - T_{jk} + s_i + h_{ij} \le M(1-x^t_{ijk}) \qquad \forall i\in V;\ j\in P\cup D\cup\{2n+1\};\ k\in K;\ t\in S \tag{3.43}$$

$$T_{n+i,k} - T_{ik} - s_i - h_{i,n+i} \ge 0 \qquad \forall i\in P;\ k\in K \tag{3.44}$$

$$b_i \le T_{ik} \le e_i \qquad \forall i\in V, k\in K \tag{3.45}$$

$$x^t_{ijk} \in \{0,1\} \qquad \forall i,j\in V, k\in K, t\in S \tag{3.46}$$

Appendix 3.C The PHTD Variant with a Single Shift Time Horizon

Table 3.11: The Modified Decision Variable of Table 3.3 for the PHTD1 Formulation

Notation	Description
• $x_{ijk} \in \{0,1\};\ \forall i,j \in V, k \in K$	Equals 1 if vehicle k moves between location i and j; 0, otherwise.

$$\min \sum_{k\in K}\sum_{i\in V}\sum_{j\in V} c_{ij}x_{ijk} \quad \text{(The PHTD1)}$$

$$\sum_{k\in K}\sum_{j\in V} x_{ijk} = 1 \qquad \forall i \in P \tag{3.47}$$

$$\sum_{j\in V} x_{ijk} - \sum_{j\in V} x_{n+i,j,k} = 0 \qquad \forall i \in P, k \in K \tag{3.48}$$

$$\sum_{j\in V} x_{jik} - \sum_{j\in V} x_{ijk} = 0 \qquad \forall i \in P\cup D, k \in K \tag{3.49}$$

$$\sum_{j\in P} x_{0jk} = 1 \qquad \forall k \in K \tag{3.50}$$

$$\sum_{i\in V} x_{i,2n+1,k} = 1 \qquad \forall k \in K \tag{3.51}$$

$$Q_{ik} - Q_{jk} + q_j \le M(1 - x_{ijk}) \qquad \forall i,j \in V, k \in K \tag{3.52}$$

$$\max\{0, q_i\} \le Q_{ik} \le \min\{C_k,\ C_k + q_i\} \qquad \forall i \in V, k \in K \tag{3.53}$$

$$T_{2n+1,k} - T_{0k} \le \text{D\_A} \qquad \forall k \in K \tag{3.54}$$

$$\sum_{i\in V}\sum_{j\in V} h_{ij}x_{ijk} \le \text{D\_D} \qquad \forall k \in K \tag{3.55}$$

$$T_{ik} - T_{jk} + s_i + h_{ij} \le M(1 - x_{ijk}) \qquad \forall i \in V;\ j \in P\cup D\cup\{2n+1\};\ k \in K \tag{3.56}$$

$$T_{n+i,k} - T_{ik} - s_i - h_{i,n+i} \ge 0 \qquad \forall i \in P \tag{3.57}$$

$$b_i \le T_{ik} \le e_i \qquad \forall i \in V, k \in K \tag{3.58}$$

$$x_{ijk} \in \{0,1\} \qquad \forall i,j \in V, k \in K \tag{3.59}$$

Appendix 3.D Certification of Student Work

UNIVERSITY OF ARKANSAS

College of Engineering
Department of Industrial Engineering

Date: December 5, 2017

Graduate School
University of Arkansas

Dear Dr. Needy:

I am writing to verify that Fereydoun Adbesh completed more than 51% of the work for the chapter titled “Vehicle Routing Problems with Hours of Service Regulations for Trucking Industry” in his dissertation.

Sincerely,

Chase Rainwater
cer@uark.edu
479-575-2687
Associate Professor
Department of Industrial Engineering
University of Arkansas

4. A Genetic Algorithm for Unified Vehicle Routing Problems

Abstract: A comprehensive tool to analyze real-world transportation networks, consisting of thousands of shipments, and solve different variations of the Vehicle Routing Problem (VRP) on the networks was developed in Chapter 3. The tool consists of several heuristic algorithms that can be categorized in constructive and improvement algorithms. In this study, we focus on improving the implementation of this tool via extensive exploration on the impact of these algorithms, individually, on the quality and running time of the solutions. This analysis becomes very useful in optimizing large real-world networks. To further improve the quality of the solutions obtained by our optimization tool, a Genetic Algorithm (GA) is added to our solution suite. The GA is designed in a way that utilizes the both constructive and improvement algorithm to create and improve its population. Experimental results on the collection of instances (the same benchmark instances as in the previous study) shows a trade-off in using GA in the optimization tool. Specifically, the GA produces better solutions regarding objective function, the total travel cost, but requires more computational time to run. A sensitivity analysis on the GA parameters is conducted to determine the best parameter setting for the GA.

4.1 Introduction

In Chapter 3, a comprehensive optimization tool (referred to as ETP) to analyze and solve a variety of the VRP models for real-world problem size transportation networks is developed. A typical real-world problem consists of 1,000s of shipments which need 100s of vehicles to deliver. For example, the case study of a real customer data in Chapter 3 includes more than 14,000 of shipments. In general, optimizing large-scale real-world problems requires the consideration of instances magnitudes larger than the benchmark problems in the VRP literature. The largest benchmark problem that has been solved by researchers consist of 1,000 shipments (Gehring and Homberger, 2001). In reality, to provide the best service for customers, each with different net-

work specifications, a flexible and fast optimization tool is required that can produce reasonably good quality solutions such as ETP. To further improve our proposed optimization tool in Chapter 3 regarding both quality and running time, we analyze the existing heuristic methods in this optimization tool. This analysis helps us to identify which algorithms have the most impact in obtaining feasible quality solutions and running time. Then, we used the results of this analysis to design a new Genetic Algorithm (GA) (Holland, 1975) which uses these heuristic algorithms throughout its optimization process.

In a nutshell, in this chapter, the performance of the implementation of the solution approach introduced in Chapter 3 is more comprehensively considered. A study as to which individual heuristic methods have the largest impact on the quality and running time of solutions is provided. This analysis plays a very important role when we deal with real-world problems consisting of very large size datasets. This optimization tool consists of several heuristic algorithms that can be categorized into i) constructive and ii) improvement algorithms. Also, we designed a more sophisticated meta-heuristic algorithm, a GA, to further improve the quality (regarding objective function) of solutions.

GA is an evolutionary algorithm designed by Holland (1975) to mimic biological evolution in solving optimization problems. The algorithm starts with an initial population of chromosomes and repeatedly modifies them in successive generations. Each chromosome is a feasible solution to the problem that is being optimized. In each generation, some chromosomes are selected (randomly or based on some merit-based rules) to breed and some others are extinct. These parents use one of the two operations, crossover and mutation, to produce new chromosomes. The number of chromosome in each generation which is called the population size remain constant through the evolution process. Each chromosome (solution) is evaluated based on its fitness (objective function) and the ones with higher fitness have more chance of survival. The evolution process terminates after a certain number of generations has passed or other criteria such as the running time. At the end, the best chromosome in the last generation is reported as the best solution of the optimization process.

The existing constructive algorithms in ETP along with a random sequence generation method are used to produce the initial population of the GA. To produce the new offspring in each generation, a crossover and a mutation operators are designed and used. In the mutation phase, the improvement algorithms that are already a part of ETP optimization engine, are called to produce a better offspring out of the selected parent. The other components of GA are independent from the existing algorithms in our optimization tool (ETP) which are explained in Section 4.6. Build upon the results of analysis on the constructive (which are used in generating the initial population) and improvement algorithms (which are used as the mutation operator), the GA is tuned and could find quality solutions in a reasonable amount of time.

Note that similar to other heuristic algorithms in ETP, this GA is also part of the optimization engine and of course compatible with the structure of ETP. Thus, we can easily use the heuristic algorithms of ETP that are already a part of this software in searching for solutions in the GA. The sensitivity analysis in the first part of this chapter enables us to effectively choose the best heuristics to use in each generation.

After designing the mechanic of the GA, a sensitivity analysis on its components is performed to choose the best setting. Then, the performance of the GA is compared with CPLEX and with the ETP heuristics on the suite of benchmark instances from Chapter 3. Results show that the GA outperforms CPLEX in almost all the instances regarding both the quality of the solutions and the running times. In comparing the GA with the ETP heuristics, the GA obtains more quality solutions but requires an increased amount of computational time.

The rest of this study is organized as follows. Section 4.2 reviews the relevant works in the literature. In Section 4.3, the structure of ETP is described. The optimization phase of ETP is explained in Section 4.4 by representing the main constructive and improving heuristic algorithms used in ETP. Section 4.5 is dedicated to the sensitivity analysis of the heuristic algorithms used in the optimization engine of ETP. The performance of each of these algorithms is reported after solving benchmark problems from the literature. In Section 4.6, a GA is proposed to solve variety of VRP variants. In Section 4.7 the performance of the GA is evaluated by solving the same VRP

variant (the single shift (PHBD1) on the same benchmark problems (Breedam's benchmark) of Chapter 3. Finally, in Section 4.8 concluding remarks are provided.

4.2 Literature Review

In this section, we review some key prior works on the literature of VRP that utilize Genetic Algorithm (GA) (Holland, 1975) to find quality solutions for the associated variant of VRP which are the foundation of our proposed GA in this study. The reason that we choose GA to improve the quality of our solution in this study is that GA is one of the common meta-heuristic approaches used in the literature to solve variations of the VRP (Baker and Ayechew, 2003; Berger and Barkaoui, 2004; Ho et al., 2008; Tan et al., 2006).

Prins (2004) implemented a simple and an effective GA on the capacitated VRP (CVRP). In his study, the chromosome structure is a sequence of locations to be visited including all the nodes (a big tour). Then, another procedure based on the optimal shortest path problem is used to optimally split the tour into individual routes.

Nagata and Bräysy (2009) proposed another effective GA for CVRP by adapting an Edge Assembly Crossover (AEC) from their previous study (Nagata, 2007). The AEC merges the edges from parents and creates several cycles from the merged graphs resulting in new children. Any infeasible solutions are repaired with a method to reconnect the sub tours.

A hybrid GA is implemented by Vidal et al. (2012), which combines the population-based evolutionary search of GA to explore solutions and local search improvement with a population diversity control scheme. The algorithms are based on the chromosome representation in Prins (2004) and evaluating each solution by both the objective function and the contribution to the diversity of the population.

4.3 ETP Structure

Studying the system structure is necessary to integrate with existing systems. In our optimization tool, without knowing the structure of data and how they are organized, we cannot develop,

add, modify, or change any algorithm in its optimization engine. Algorithms in ETP work with shipment data, points, routes, geographical locations and other customer information. The optimization process in ETP occurs via a chain of algorithms, so the output of one algorithm is the input of another one (more details on Section 4.4).

In this section, the structure of optimization tool in ETP is described. We show how the input data needs to be prepared to optimize the network in each problem. First, the characteristics of the network such as the shipment locations and amounts, the vehicle types and their capacity, the time windows, costs, etc. are defined and interpreted for ETP. Second, the parameters of the heuristic algorithms, either constructive or improving, are set to start these algorithms.

To improve the performance of ETP, new optimization algorithms are developed, and existing ones from the literature are modified and added to the engine. Unlike many existing heuristic algorithms in the literature of VRP that sequence, move or exchange the points (Lin, 1965; Toth and Vigo, 2014), several ETP heuristic algorithms work with shipments. A shipment is a line in the network that connects two points, and includes the information about the origin, the destination and the amount of shipment. More details about the optimization algorithms are provided in Section 4.4.

4.3.1 The Input Datasets

In the ETP framework, a database is a network consisting of nodes and arcs. The nodes, in our context, refer to the points on the map where a pick or drop is supposed to occur. The arcs, on the other hand, show the direction of the movement by connecting pick and drop points. ETP can obtain this node/arc information through various sources; Access, Excel, text and SQL tables are the typical data sources used in ETP.

ETP works with in-memory databases which are replicas of the input databases. The in-memory database contains all the necessary information for the network optimization. The data is categorized into four main tables: i) “Points”, ii) “Lines”, iii) “Parameters” and iv) “Routes” which are explained as follows. The correlation between the ETP datasets and tables are shown

in Figure 4.1.

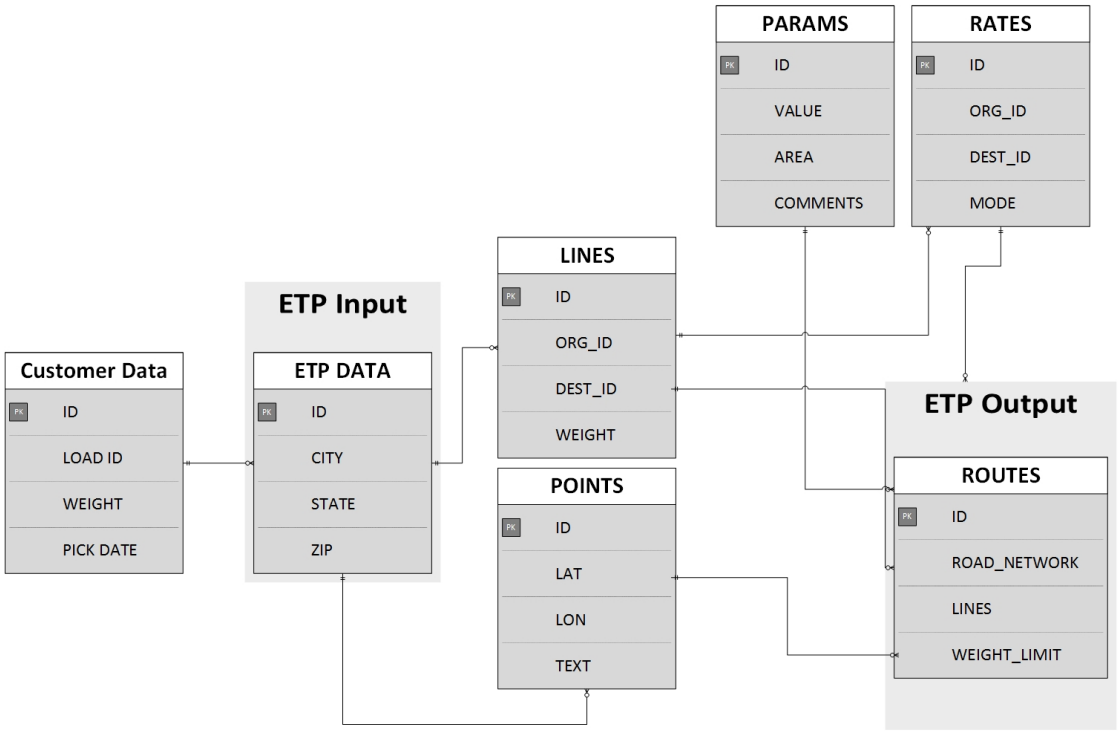


Figure 4.1: The Correlation Between Datasets in ETP

Figure 4.1 shows how the data is organized in ETP. The raw data is obtained from the customer databases. It is then modified and sorted in a more meaningful format for ETP. After that, LINES and POINTS table are created which holds the basic building blocks of the customer data. Finally, PARAMS, RATES and APPOINTMENT tables are created to define the optimization parameters, shipping costs, and customer specific constraints, respectively. At this point the problem is ready to be optimized. The results will be stored in ROUTES tables.

4.3.1.1 Points

The "POINTS" table stores the location information of the origin and the destination of all orders. These information include but not limited to each point's latitude, longitude, state, city, country, graphical data such as size, color, and shape on the map.

4.3.1.2 Lines

The “LINES” table categorizes all the orders and their details. Each order can be a parcel, box, pallet, etc. depending on the shipment type of each customer with corresponding characteristics and details. The orders need to be picked up from their origin and be delivered to the destination. Each line row corresponds to a unique order and includes the details such as origin, destination, weight, width, length, service time at the origin and the destination, volume, required vehicle type, cost rate, pickup time windows in the origin, delivery time window at the destination, distance (if the distance is not determined, ETP calculates it), etc.

4.3.1.3 Parameters

The “PARAMETER” table contains the parameters that guides creating the VRP model, imposing the constraints, and running the optimization process. The parameters are grouped into four categories regarding their purpose: i) function, ii) constraint, iii) schedule, and iv) algorithm. The function parameters are the basic functionality parameters that have a small influence on routing and scheduling logic. The constraint parameters put restrictions on each route and they have a significant influence over the routing and scheduling algorithms. The schedule parameters are the ones that effect the scheduling. Finally, the algorithm parameters are used to alter the optimization logic. They have a significant influence on both routing and scheduling.

4.3.1.4 Routes

The “ROUTES” table stores two sets of information, the default and created routes. Default routes are the blueprint of each route object created routes are the output of optimization process which includes the assignment of vehicles to a set of order lines and scheduling of the pickups and deliveries.

4.4 Optimization in ETP

In ETP, several heuristic algorithms are developed and included in the optimization engine of ETP. In Chapter 3, the network optimization of any VRP models in ETP is separated into five steps process: i) grouping, ii) merging, iii) improving, iv) filtering, v) and forcing. A chain of heuristic algorithms is implemented for each step to optimize the network. More generally, the heuristic algorithms in ETP can be categorized into two categories: i) constructive and ii) improving algorithms. The constructive algorithms create the feasible solutions (routes). Feasible solutions are improved using the improving heuristic algorithms in ETP. In the following two sections we describe the main constructive and improving algorithms, respectively.

4.4.1 The Constructive Algorithms in ETP

Two main constructive heuristic algorithms of the ETP optimization engine are the two following algorithms: i) a modified nearest neighbor search algorithm and ii) a modified Clarke-Wright algorithm.

4.4.1.1 The Modified Nearest Neighbor Algorithm

The well-known nearest neighbor search (NNS) is one of the first and fastest algorithms to find feasible solutions for the traveling salesman problem (TSP). In VRP, this algorithm creates a route by starting from a depot or a random shipment. It interactively adds the “nearest” shipment to the current route until the capacity is exhausted while maintaining the feasibility of the route. The “nearest” term can be defined differently. It could be defined as the closest point regarding distance, the point with the minimum cost to travel to, or any other criteria.

4.4.1.2 The Modified Clarke-Wright Algorithm

The Clarke-Wright algorithm (Clarke and Wright, 1964) is one of the most popular saving algorithms over the last five decades in the VRP literature. It merges the routes with a positive saving

while keeping the solution feasible. The merging process usually starts with the routes with single node. In this research, we develop a new heuristic method to produce the initial solution for the modified Clarke-Wright (CLK) algorithm referred to as Group Direct algorithm. The idea behind the Group Direct algorithm is as follows. It groups the shipments that have the same origins and destinations, and list them with the priority of the bigger shipment size and the earliest pickup time, respectively. It adds the shipments in the same group to a route one by one from the list until the capacity of the associated truck is exhausted while keeping the route feasible. A maximum gap between the start time of the route and the adding shipment is enforced to the algorithm.

4.4.2 Improvement

The improvement algorithms are the ones that try to improve the existing routes in a solution. The improvement algorithms are implemented on a single route or a pair of routes by altering the sequence of order lines in them. The algorithms that alter lines between a pair of routes in a solution are referred to as Inter-Route improvement algorithms. The ones that work within a route of a solution are called Intra-Route algorithms. The Inter-Route category include three algorithms: i) 1-Move, ii) 1-Exchange and iii) 1-Reinsert. The first two algorithms are described in details in Chapter 3. The third one, 1-Reinsert, is a new algorithm we added to the optimization engine of ETP in this chapter. The Intra-Route category also includes three algorithms: i) Minimize Cost, ii) Break Cross and iii) 1-in-Move algorithms. In the following sections, we briefly describe these algorithms.

4.4.2.1 Inter-Route Improvement Algorithms

Line move algorithms are the improvement algorithms that work with moving shipment lines between the routes in a solution. In this research, three different line move algorithms are presented and their impacts on improving solutions are studied. These algorithms are i) one line move, ii) one line exchange, and iii) one line reinsert. Note that in this research the word line, order line

and shipment mean the same and are used interchangeably.

4.4.2.1.1 1-Move

In this algorithm we remove one line from a route and insert to another route. We select a pair of routes and a line from the first route exhaustively. It means that this algorithm tries every possible pair of routes in the solution and move every line from the first route and insert it to every possible position in the second route as long as we improve in the solution.

4.4.2.1.2 1-Exchange

This algorithm works similar to one line move algorithm but instead of moving one line from a route to another one it switches a line from the first route with another line in the second route. Similar to one line move, this algorithm search trough the routes to find the best position to switch lines exhaustively. The combination of lines is exhaustive, too. That means exchanging every possible pair of lines of the routes are evaluated.

4.4.2.1.3 1-Reinsert

The One-Reinsert algorithm has the potential of moving shipments between more than two routes. In this algorithm, an "outliner" stop in each route is determined and removed. The outliner is the stop that has the furthest distance from its previous and next stop in the current route. If the summation of the distance from a stop to its previous stop and its distance to the next stop is greater than a certain value it will be considered as the outliner and be removed from the route. The value is calculated by multiplying a user-defined factor by the average distance of the stops in their current route from their previous and next stops. After the outliners are removed from their routes, the algorithm tries to place each of them into the best position of the best route with the largest savings.

4.4.2.2 Intra-Route Improvement Algorithms

In this research, the route improvement algorithms are the ones that are performed in only one route a solution by altering the sequence of orders wile maintaining the feasibility of the route. These algorithms are applied to each route of the solution until no improvement is made by any of them. Currently, we use three route improvement algorithms in the same order that are described as follows.

4.4.2.2.1 Minimize Cost

This algorithm, as described in Chapter 3, Section 3.4.3.4, uses a clustering method referred to as K-means Cluster to divide the order lines of a route to a user-defined number of subsets (clusters) of lines with few elements. Then, an extensive search is implemented for each cluster.

4.4.2.2.2 Break Cross

This algorithm is described in details in Chapter 3, Section 3.4.3.5. This algorithm is designed to break the intersection of two lines in a route. In general, a route will improve regarding total distance by breaking the intersection between lines. For example, in Figure 4.2 (a) the two lines a−d and b−c are crossed each other. Breaking this cross, as shown in Figure 4.2 (b), decreases the total distance of the route.

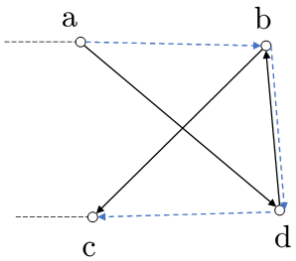


Figure 4.2: Breaking Line Cross

4.4.2.2.3 1-in-Move

As stated in Chapter 3, Section 3.4.3.6, the 1-in-Move algorithm moves a line in a single route to find the position with the minimum cost. A line is selected and placed in every possible position

in a route. This algorithm selects the lines one by one as their sequence in the route.

4.5 Heuristic Algorithms Analysis

In this section, we analyze the heuristic algorithms that are explained in Section 4.4, regarding their impact on the quality of the solutions and running time of the ETP optimization process. This analysis gives us valuable information on how to choose the constructive and improvement algorithms especially when we want to optimize large datasets of the real-world problems. We use the VRP variant, the PHBD1, and the benchmark instances, the modified Breedam's benchmark (Van Breedam, 1994) in Chapter 3 (for more detail see Appendix 3.A) to analyze each algorithm's impacts.

Table 4.1: The Impact of the Nearest Neighbor and Heuristics Algorithms on the Modified Breedam's Instances

			Improvement Heuristics					
	Nearest Neighbor		**Inter-Route**		**Intra-Route**			
Instance ID	*Cost*	*Time*	*Cost Improv.*	*Time*	*Cost Improv.*	*Time*	**Total Cost**	**Total Time**
1P1	1166.1	0.3 s	90.2	0.7 s	-4.7	0.1 s	1071.2	2.0 s
2P1	1147.7	0.9 s	57.4	1.8 s	0.0	0.7 s	1090.3	3.4 s
3P1	1144.4	0.9 s	19.8	1.7 s	0.0	0.3 s	1124.7	2.9 s
4P1	1110.1	0.9 s	0.0	1.1 s	0.0	0.1 s	1110.1	2.1 s
5P1	1115.9	0.9 s	0.0	1.2 s	0.0	0.1 s	1115.9	2.2 s
6P1	1167.0	0.9 s	36.3	1.8 s	0.0	0.4 s	1130.7	3.1 s
7P1	1368.2	0.7 s	49.8	1.8 s	16.7	0.3 s	1301.7	2.8 s
8P1	1538.0	0.7 s	133.2	2.6 s	27.9	0.4 s	1376.9	3.7 s
9P1	1692.7	0.8 s	160.0	3.9 s	11.1	0.7 s	1521.6	5.4 s
10P1	1305.7	0.9 s	116.2	3.7 s	7.2	0.7 s	1182.3	5.3 s
11P1	1562.2	0.8 s	357.7	2.9 s	0.0	0.4 s	1204.5	4.1 s
12P1	1694.4	1.0 s	290.3	5.8 s	3.8	1.3 s	1400.3	8.1 s
13P1	1335.8	0.3 s	81.2	2.6 s	67.6	0.3 s	1187.0	3.2 s
14P1	1251	0.3 s	68.1	2.3 s	49.5	0.3 s	1133.4	2.9 s
15P1	1423.3	0.3 s	124.5	5.6 s	8.2	0.7 s	1290.6	6.6 s
16P1	819.9	1.0 s	0.0	2.1 s	45.2	0.5 s	774.7	3.6 s
17P1	815.0	0.9 s	0.0	1.7 s	8.5	0.3 s	806.5	2.9 s
18P1	970.8	0.8 s	0.0	1.8 s	7.2	0.2 s	963.6	2.8 s

Continued on the next page

Table 4.1 – Continued from the previous page

			Improvement Heuristics					
	Nearest Neighbor		Inter-Route		Intra-Route			
Instance ID	*Cost*	*Time*	*Cost Improv.*	*Time*	*Cost Improv.*	*Time*	**Total Cost**	**Total Time**
19P1	702.4	1.0 s	0.0	1.5 s	0.0	0.2 s	702.4	2.7 s
20P1	675.3	1.1 s	0.0	2.5 s	4.7	0.4 s	670.6	4.0 s
21P1	811.7	0.9 s	0.0	0.9 s	0.0	0.2 s	811.7	2.0 s
22P1	777.0	0.9 s	0.0	0.8 s	0.0	0.1 s	777.0	1.8 s
23P1	849.9	0.9 s	3.5	1.9 s	13.0	0.3 s	833.4	3.1 s
24P1	1119.3	0.9 s	73.7	4.8 s	6.2	0.9 s	1039.4	6.6 s
25P1	669.7	1.1 s	0.0	1.0 s	0.0	0.2 s	669.7	2.3 s
26P1	721.3	0.9 s	0.0	0.9 s	0.0	0.2 s	721.3	2.0 s
27P1	907.1	0.9 s	0.0	1.9 s	13.6	0.3 s	893.5	3.1 s
28P1	910.4	0.9 s	36.0	2.9 s	46.4	0.6 s	828.0	4.4 s
29P1	922.7	0.9 s	34.5	3.2 s	9.2	0.6 s	879.0	4.7 s
30P1	1103.7	0.9 s	34.7	3.3 s	22.7	0.5 s	1046.3	4.7 s
31P1	673.8	0.7 s	89.3	3.1 s	1.0	0.5 s	583.5	4.3 s
32P1	781.5	0.8 s	0.0	1.8 s	24.3	0.3 s	757.2	2.9 s
33P1	957.8	0.9 s	0.0	2.0 s	8.1	0.4 s	949.7	3.3 s
34P1	679.1	0.5 s	0.0	1.5 s	0.0	0.1 s	679.1	2.1 s
35P1	734.9	0.6 s	41.9	3.3 s	9.8	0.4 s	683.2	4.3 s
36P1	976.1	0.9 s	43.6	3.2 s	0.0	0.6 s	932.5	4.7 s
37P1	985.4	0.8 s	187.2	6.2 s	0.0	1.1 s	798.2	8.1 s
38P1	913.2	0.8 s	24.3	3.8 s	9.9	0.7 s	879.0	5.3 s
39P1	1059.3	0.8 s	97.5	2.6 s	24.7	0.5 s	937.1	3.9 s
40P1	1032.2	1.0 s	152.1	5.1 s	28.8	0.9 s	851.3	7.0 s
41P1	996.5	0.9 s	22.1	1.9 s	5.7	0.4 s	968.7	3.2 s
42P1	1036.6	0.9 s	0.0	1.9 s	0.9	0.3 s	1035.7	3.1 s
43P1	922.1	0.3 s	91.9	4.0 s	2.4	0.6 s	827.8	4.9 s
44P1	1022.9	0.7 s	76.4	4.0 s	48.9	0.7 s	897.6	5.4 s
45P1	1214.8	0.7 s	84.8	3.4 s	20.1	0.5 s	1109.9	4.6 s
46P1	785.8	0.9 s	0.1	1.7 s	8.5	0.2 s	777.2	2.8 s
47P1	710.0	0.7 s	0.0	0.8 s	0.0	0.1 s	710.0	1.6 s
48P1	870.9	1.0 s	0.0	0.9 s	0.0	0.2 s	870.9	2.1 s
49P1	770.9	0.6 s	84.0	1.8 s	12.1	0.3 s	674.8	2.7 s
50P1	795.9	0.6 s	46.1	1.8 s	0.0	0.2 s	749.8	2.6 s
51P1	943.5	0.9 s	18.8	2.0 s	0.0	0.4 s	924.7	3.3 s
52P1	1100.7	0.5 s	101.8	1.7 s	2.3	0.2 s	996.6	2.4 s
53P1	1349.8	0.5 s	210.2	1.7 s	37.1	0.3 s	1102.5	2.5 s
54P1	1557.3	0.6 s	100.6	1.9 s	13.1	0.3 s	1443.6	2.8 s
55P1	1234.1	0.7 s	171.0	1.9 s	17.6	0.3 s	1045.5	2.9 s
56P1	1256.7	0.6 s	82.7	1.8 s	22.1	0.3 s	1151.9	2.7 s
57P1	1476.3	0.8 s	122.3	4.1 s	13.5	0.8 s	1340.5	5.7 s

Continued on the next page

Table 4.1 – Continued from the previous page

			Improvement Heuristics					
	Nearest Neighbor		Inter-Route		Intra-Route			
Instance ID	*Cost*	*Time*	*Cost Improv.*	*Time*	*Cost Improv.*	*Time*	**Total Cost**	**Total Time**
58P1	1057.7	0.3 s	0.0	1.1 s	0.0	0.1 s	1057.7	1.5 s
59P1	1085.0	0.3 s	48.5	1.7 s	4.6	0.3 s	1031.9	2.3 s
60P1	1336.2	0.3 s	72.7	1.7 s	22.9	0.2 s	1240.6	2.2 s
Average:	**1051.9**	**0.8 s**	**59.3**	**2.4 s**	**11.7**	**0.4 s**	**977.8**	**3.6 s**

As shown in the above table, the average of improvements made by the Inter-Route improvement (Live Move, Line Exchange and Line Reinsert algorithms) is 59.3 or 5.6% of the average total cost of the constructive NNS algorithm in only average time of 2.4 seconds. On the other hand, the Intra-Route algorithms (Minimize Cost, 1-in-Move and Break Cross algorithms), has made an 11.7 improvement (1.1%) in the objective function but in a very short amount of time. In total the improvement algorithms improved the total cost by 6.7% in average time of 2.8 seconds. As we can see the Inter-Route improvement algorithms take 2.4 seconds versus the con algorithm that just needs 0.8 seconds in average to create solutions. That means in the improvement step we need three times more time in average to improve the solution.

A similar statistics for the impact of improvement algorithms on the NNS, as presented in Table 4.1, is provided for the CLK algorithms in Table 4.2.

Table 4.2: The Impact of the Clarke-Wright and Improvement Heuristics on the Modified Breedam's Instances

			Improvement Heuristics					
	Clarke-Wright		Inter-Route		Intra-Route			
Instance ID	*Cost*	*Time*	*Cost Improv.*	*Time*	*Cost Improv.*	*Time*	**Total Cost**	**Total Time**
1P1	1166.8	1.2 s	64.2	1.3 s	0.0	0.2 s	1102.6	2.7 s
2P1	1158.5	0.7 s	49.2	1.6 s	0.0	0.2 s	1109.3	2.5 s

Continued on the next page

Table 4.2 – Continued from the previous page

			Improvement Heuristics					
	Clarke-Wright		Inter-Route		Intra-Route			
Instance ID	*Cost*	*Time*	*Cost Improv.*	*Time*	*Cost Improv.*	*Time*	**Total Cost**	**Total Time**
3P1	1186.1	1.3 s	61.5	3.7 s	0.0	0.8 s	1124.6	5.8 s
4P1	1000.7	1.1 s	0.0	0.9 s	0.0	0.1 s	1000.7	2.1 s
5P1	874.0	1.3 s	0.0	0.8 s	0.0	0.1 s	874.0	2.2 s
6P1	1046.8	1.2 s	0.0	0.8 s	0.0	0.1 s	1046.8	2.1 s
7P1	1316.3	0.8 s	53.5	1.8 s	7.3	0.2 s	1255.5	2.8 s
8P1	1340.2	1.1 s	0.0	0.8 s	0.0	0.0 s	1340.2	1.9 s
9P1	1483.6	1.2 s	0.0	0.8 s	0.0	0.1 s	1483.6	2.1 s
10P1	1179.3	1.0 s	40.8	3.1 s	0.0	0.4 s	1138.5	4.5 s
11P1	1314.8	0.6 s	33.9	3.3 s	5.7	0.4 s	1275.2	4.3 s
12P1	1353.9	1.4 s	17.1	1.8 s	0.0	0.2 s	1336.8	3.4 s
13P1	1146.9	0.7 s	42.6	1.6 s	0.0	0.3 s	1104.3	2.6 s
14P1	1136.0	0.7 s	0.0	0.8 s	0.0	0.1 s	1136.0	1.6 s
15P1	1332.8	0.8 s	13.0	1.7 s	0.0	0.2 s	1319.8	2.7 s
16P1	882.0	1.4 s	12.1	1.6 s	13.3	0.2 s	856.6	3.2 s
17P1	806.5	1.4 s	0.0	0.9 s	0.0	0.2 s	806.5	2.5 s
18P1	963.6	1.3 s	0.0	0.8 s	0.0	0.2 s	963.6	2.3 s
19P1	745.1	1.3 s	0.0	1.3 s	0.0	0.1 s	745.1	2.7 s
20P1	702.6	1.3 s	0.0	0.7 s	0.0	0.1 s	702.6	2.1 s
21P1	829.3	1.4 s	0.0	1.0 s	0.0	0.2 s	829.3	2.6 s
22P1	834.4	1.2 s	0.0	0.8 s	0.0	0.0 s	834.4	2.0 s
23P1	926.6	1.3 s	0.0	0.7 s	0.0	0.0 s	926.6	2.0 s
24P1	1023.2	1.3 s	10.0	1.7 s	0.0	0.2 s	1013.2	3.2 s
25P1	669.7	1.3 s	0.0	0.8 s	0.0	0.2 s	669.7	2.3 s
26P1	721.3	1.3 s	0.0	0.8 s	0.0	0.2 s	721.3	2.3 s
27P1	893.5	1.3 s	0.0	0.8 s	0.0	0.2 s	893.5	2.3 s
28P1	828.0	1.2 s	0.0	0.8 s	0.0	0.2 s	828.0	2.2 s
29P1	879.0	1.4 s	0.0	0.8 s	0.0	0.1 s	879.0	2.3 s
30P1	1171.0	1.0 s	3.7	1.9 s	0.0	0.2 s	1167.3	3.1 s
31P1	542.6	1.1 s	0.0	0.9 s	0.0	0.1 s	542.6	2.1 s
32P1	678.2	1.1 s	0.0	0.9 s	0.0	0.2 s	678.2	2.2 s
33P1	997.8	1.1 s	5.0	1.6 s	0.0	0.2 s	992.8	2.9 s
34P1	571.9	0.8 s	0.0	0.8 s	0.0	0.1 s	571.9	1.7 s
35P1	656.6	0.9 s	10.2	1.8 s	0.0	0.2 s	646.4	2.9 s
36P1	883.6	0.9 s	13.8	1.9 s	0.0	0.2 s	869.8	3 s
37P1	722.3	1.1 s	0.0	0.8 s	0.0	0.1 s	722.3	2.0 s
38P1	800.3	1.1 s	21.4	3.2 s	1.9	0.3 s	777.0	4.6 s
39P1	976.2	1.2 s	37.1	3.1 s	2.0	0.7 s	937.1	5 s
40P1	783.5	1.2 s	0.0	1.0 s	0.0	0.1 s	783.5	2.3 s
41P1	871.1	1.3 s	0.0	0.9 s	0.0	0.2 s	871.1	2.4 s

Continued on the next page

Table 4.2 – Continued from the previous page

Instance ID	Clarke-Wright		Improvement Heuristics				Total Cost	Total Time
			Inter-Route		Intra-Route			
	Cost	*Time*	*Cost Improv.*	*Time*	*Cost Improv.*	*Time*		
42P1	1110.6	1.2 s	0.0	0.8 s	0.0	0.1 s	1110.6	2.1 s
43P1	794.4	1.1 s	0.0	0.9 s	0.0	0.1 s	794.4	2.1 s
44P1	935.1	1.1 s	0.0	0.8 s	0.0	0.1 s	935.1	2.0 s
45P1	1156.5	1.1 s	0.0	1.0 s	0.0	0.1 s	1156.5	2.2 s
46P1	784.2	1.2 s	0.0	0.8 s	0.0	0.1 s	784.2	2.1 s
47P1	710.0	1.1 s	0.0	0.9 s	0.0	0.2 s	710.0	2.2 s
48P1	870.9	1.3 s	0.0	0.9 s	0.0	0.2 s	870.9	2.4 s
49P1	611.8	0.7 s	0.0	0.7 s	0.0	0.1 s	611.8	1.5 s
50P1	710.7	0.9 s	18.6	1.9 s	6.1	0.4 s	686.0	3.2 s
51P1	1019.9	0.9 s	4.2	1.8 s	0.0	0.2 s	1015.7	2.9 s
52P1	984.4	0.6 s	0.0	0.7 s	0.0	0.1 s	984.4	1.4 s
53P1	973.0	0.9 s	0.0	0.9 s	0.0	0.1 s	973.0	1.9 s
54P1	1262.7	0.8 s	0.0	0.9 s	0.0	0.1 s	1262.7	1.8 s
55P1	1082.5	1.1 s	15.5	1.8 s	0.0	0.2 s	1067.0	3.1 s
56P1	1062.9	0.7 s	0.0	0.7 s	0.0	0.1 s	1062.9	1.5 s
57P1	1312.5	1.3 s	0.0	0.8 s	0.0	0.1 s	1312.5	2.2 s
58P1	885.6	0.8 s	1.6	1.8 s	5.4	0.3 s	878.6	2.9 s
59P1	1005.0	0.7 s	0.0	0.9 s	0.0	0.1 s	1005.0	1.7 s
60P1	1306.8	0.7 s	14.9	3.6 s	30.3	0.4 s	1261.6	4.7 s
Average:	**966.3**	**1.1 s**	**9.1**	**1.3 s**	**1.2**	**0.2 s**	**956.0**	**2.6 s**

4.6 A Genetic Algorithm

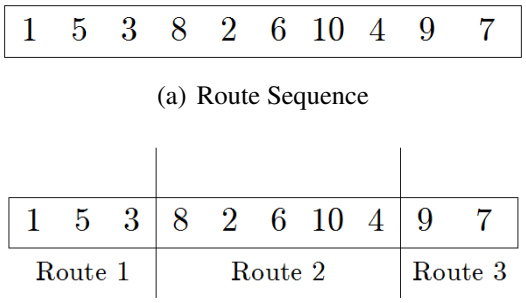
In this section, a Genetic Algorithm is proposed that follows the route first, cluster second procedure (Beasley, 1983) to create the chromosomes. Most of the initial population are generated randomly, and the remaining are created by the constructive heuristic algorithms in ETP (the modified Clarke-Wright and Nearest Neighbors) and another random greedy selection algorithm within specified time and distance. In each generation, new chromosome are produced using the crossover and mutation operations. The crossover operation creates two offspring out of two parents which have been used by a tournament selection from the current population. After choosing the parents, there will be more chance to move the best route from one parent to another by a ruin

and recreate procedure or selecting the worst routes in one parent and add its nodes to the other parent. The mutation is performed by one parent using the previously developed improvement algorithms in the ETP optimization engine.

This GA is designed in a way that can be applied to a variety of a VRP variants. The proposed GA is based on the work of Ombuki et al. (2006). Each component of the GA is explained in details in the following sections.

4.6.1 Chromosome Representation

We used a route first cluster second approach to create the solutions. Each solution in a generation is represented by a chromosome. A chromosome is a sequence of all the nodes in the network which will be divided to some routes later. We begin building up the routes by starting from the first node in the sequence and adding it to a vehicle. We keep adding the nodes in thew sequence one by one to the current route until the capacity of the associated vehicle is exhausted. Then we start a new route by adding the next node in the sequence to the new route. We continue this process until add the last node to a route. This process is illustrated in Figure 4.3.



(a) Route Sequence

(b) Route Clusters

Figure 4.3: The Chromosome Representation

In Figure 4.3 (a), the sequence of the potential routes of a solution is presented. We start from node 1 and add it to route 1 without violating any constraints in the model. Then, nodes 5 and 3 are added to route 1 while maintaining the feasibility of the route. Next, we try to add node 8 to route 1 but it violates a constraint (could be any constraint such as the capacity of the

vehicle). So, we stop adding any more nodes to route 1 and start route 2. Route 2 in created by adding nodes 8, 2, 6, 10, and 4 to it, respectively. Route 3 is formed the same way as the others by adding nodes 9 and 7, respectively (Figure 4.3 (b)).

4.6.2 The Initial Population

The most of the chromosomes in the initial population is generated randomly. Two chromosomes are created using the two constructive algorithms in our optimization tool, the Clarke-Wright and the Nearest Neighbor algorithms. The rest of the initial population is generated by producing a random sequence of the nodes in the network and clustering the routes as explained in Section 4.6.1, repeatedly, until we have the desired number of feasible chromosomes.

4.6.3 Selection Strategy

In each generation, the survived chromosomes become the next generation's population. The evolution stops when a stopping criteria has met. The population in current generation consists of the chromosomes from previous generation plus the ones we produced in this generation. In each generation, the new chromosomes are produced by the crossover and mutation operations. In this study, the crossover operates on two parents and produces two children and the mutation on one parent and create one child. Two types of selection strategies are used in this GA for: i) selecting parent(s) and ii) selecting surviving chromosomes which is explained as follows.

4.6.3.1 Selecting Parent(s)

In this GA, we use two selections method to select parent(s) for the crossover and mutation operations: i) random selection and ii) tournament selection. In random selection, we just randomly select parent(s) from the current population. In tournament selection, two parameters are involved: i) tournament_size and ii) elite_cutoff $\in [0, 1]$. The default values of the tournament_size and elite_cutoff parameters are 4 and 0.8, respectively. First, we select tournament_size number of chromosomes as the parent candidates. Second, we generate a number between 0 and 1 ran-

domly. If the generated number is less than the elite_cutoff we select the chromosome with the lowest cost from the candidates. Otherwise, we randomly select a chromosome from the candidates. We repeat these steps until we select as many parents as we need.

4.6.3.2 Selecting The Surviving Chromosomes

To Select the surviving chromosomes to go to the next generation the current population we need to set a value for two out of the following three parameters: i) elite_percent, ii) crossover_percent, and iii) mutation_percent because elite_percent + crossover_percent + mutation_percent = 100. The default values for these parameters are 10, 80 and 10, respectively. The elite_percent of the best chromosomes with the lowest cost from current population go straightly to the next generation. The crossover and mutation operations produce the crossover_percent and mutation_percent of the next generation, respectively.

4.6.4 Crossover

The crossover operation is responsible for the diversity of chromosomes in each generation. With this operation, the feasible solution area is widely explored not to get trapped in a local optima. The crossover needs two chromosomes as the parents to breed and produce two new chromosomes as the children. The crossover algorithm's pseudo-code is as follows.

Crossover (parent1, parent2)

{

Step 1 Randomly select a route (route2) from parent2.

Step 2 Create a sequence of shipments (sequence1) by removing the route clusters of parent1 (similar to Figure 4.3 (a)).

Step 3 Remove the shipments (lines) of route2 from sequence1.

Step 4 Insert the shipments of route2 into sequence1.

Step 5 Create a child (child1) by clustering the sequence1.

Step 6 Switch parent1 and parent2 and repeat steps 1 to 5 to create offspring2.

Step 7 Return offspring1 and offspring2.

}

In this algorithm, there are three different insert method to find the best position for the shipments of route2 to sequence1 in step 4. These methods are i) exhaustive, ii) semi-exhaustive, and iii) random search. In the exhaustive search, every position in sequence1 is tried to add each shipment of route2 and the best position with the largest saving is selected. In the semi-exhaustive search, we keep the sequence of shipments in route2 and try to find the best position in sequence1 for all them at once. It means we search for the best position for route2 in sequence1 instead of each shipment of route2. In random search which is the fastest method, first we set a value for the crossover_random_num parameter with the default value of the number of routes in parent1. Second, similar to the semi-exhaustive search, we try to find the best position for route2 only for the "crossover_random_num" number of randomly selected positions in sequence1. In this GA, the default insert method for step 4 of the crossover operation is the random search method.

4.6.5 Mutation

In this GA, the mutation operation is used to improve the quality of the chromosomes in each generation. We first select a parent as explained in Section 4.6.3.1 and then perform the improvement algorithms in our optimization tool as described in Section 4.4.2 on it. If the number of

shipments is less than 500 we execute all the line move and stop move improvement algorithms on the selected parent to reduce the total cost of the child as much as possible. Otherwise, we use the results of analysis of the improvement algorithms in Section 4.5 to choose the best algorithm(s) to apply on the parent.

4.6.6 The Termination Criteria

Four criteria are considered to terminate the GA. The criteria are considered in the order that follows. A violation of any criteria terminates the algorithm. These criteria are listed as follows.

1. Maximum run time: After each generation, the GA checks the running time and terminates the process if the elapsed time is greater than the maximum run time. The default value for this parameter is 60 minutes.

2. Maximum number of generations: The GA terminates after the maximum number of generation is exceeded. The default value for this parameter is 50.

3. Maximum number of attempts in producing different offspring: The crossover and mutation operators in the GA are designed to select different parent(s). These operators are executed until they produce the number of required new solutions different than the other solutions in the current generation. The default value for this parameter is 10 times the population size.

4. The number of consecutive generations without any improvement in the objective function of the best solution. The default value for this parameter is 2.

4.7 The GA Experimental Results

In this section, we report the performance of the GA versus CPLEX and the heuristic methods in ETP. The result of the comparison of the objective function (total cost) and running time on a subset of the benchmark instances in Chapter 3 (43 out of 60) are reported in Table 4.3. These

benchmark instances include those for which heuristic methods in ETP did not outperform CPLEX (20 instances). More specifically, these instances have a non-negative values in the Gap_{NC} or Gap_{CC} columns in Table 3.7. Additional benchmark instances are considered to increase the validity of the derived results. In Table 4.3, the gap between the objective function of ETP and CPLEX is shown by Gap_{EC} and the gap between the GA and CPLEX is shown by Gap_{GC}. Their expressions are as follows.

$$\text{Gap}_{\text{EC}} = \frac{\text{Obj}_{\text{ETP}} - \text{Obj}_{\text{CPLEX}}}{\text{Obj}_{\text{CPLEX}}} \times 100\% \tag{4.1}$$

$$\text{Gap}_{\text{GC}} = \frac{\text{Obj}_{\text{GA}} - \text{Obj}_{\text{CPLEX}}}{\text{Obj}_{\text{CPLEX}}} \times 100\% \tag{4.2}$$

Where Obj_{ETP}, $\text{Obj}_{\text{CPLEX}}$ and Obj_{GA} are the total cost of the best solution obtained by ETP, CPLEX and GA, respectively.

Table 4.3: Performance Comparison, ETP vs. CPLEX vs. GA (default settings as in Table 4.4)

	ETP			**CPLEX**			**GA**	
Instance ID	*Obj.*	*Time*	**Gap**$_{\text{EC}}$	*Obj.*	*Time*	**Gap**$_{\text{GC}}$	*Obj.*	*Time*
1P1	1071.2	2.0 s	1.3%	1057.1	8 h	0.0%	1057.1	1116.3 s
2P1	1090.3	3.4 s	2.8%	1060.1	8 h	0.0%	1060.0	1255.6 s
3P1	1124.6	2.9 s	0.0%	1124.7	8 h	0.0%	1124.6	2016.5 s
4P1	1110.1	2.1 s	9.6%	1013.0	8 h	**-10.5**%	906.2	1018.3 s
5P1	1115.9	2.2 s	20.4%	927.1	8 h	**-6.8**%	864.5	1003.2 s
8P1	1376.9	3.7 s	-1.3%	1394.7	8 h	**-9.7**%	1259.2	866.5 s
9P1	1521.6	5.4 s	-0.1%	1523.8	8 h	**-9.8**%	1374.6	949.9 s
10P1	1182.3	5.3 s	1.4%	1165.7	8 h	**-3.5**%	1125.0	1089.7 s
11P1	1204.5	4.1 s	-12.2%	1371.3	8 h	**-14.6**%	1170.9	1185.8 s
13P1	1187.0	3.2 s	16.4%	1020.1	8 h	1.9%	1039.2	1438.0 s
14P1	1133.4	2.9 s	**-5.1**%	1194.6	8 h	**-5.1**%	1133.4	1047.2 s
17P1	806.5	2.9 s	**-1.7**%	820.6	8 h	**-1.7**%	806.5	1338.1 s
19P1	702.4	2.7 s	5.1%	668.6	8 h	**-0.9**%	662.8	1539.2 s
21P1	811.7	2.0 s	-3.2%	838.4	8 h	**-4.1**%	803.9	1420.5 s
25P1	669.7	2.3 s	**-1.7**%	681.4	8 h	**-1.7**%	669.7	1487.4 s
28P1	828.0	4.4 s	**-1.6**%	841.7	8 h	**-1.6**%	828.0	1862.0 s
31P1	583.5	4.3 s	7.5%	542.6	8 h	0.0%	542.6	693.3 s
32P1	757.2	2.9 s	10.7%	683.9	8 h	**-0.8**%	678.2	834.6 s

Continued on the next page

Table 4.3 – Continued from previous page

	ETP			CPLEX			GA	
Instance ID	*Obj.*	*Time*	**Gap$_{EC}$**	*Obj.*	*Time*	**Gap$_{GC}$**	*Obj.*	*Time*
34P1	679.1	2.1 s	29.8%	523.2	8 h	0.0%	523.2	789.7 s
35P1	683.2	4.3 s	-4.5%	715.2	8 h	**-8.2%**	656.6	649.1 s
36P1	932.5	4.7 s	-1.7%	948.9	8 h	**-9.4%**	860.1	780.0 s
37P1	798.2	8.1 s	9.6%	728.5	8 h	**-0.9%**	722.3	1063.9 s
38P1	879.0	5.3 s	4.7%	839.6	8 h	**-4.7%**	800.3	849.5 s
40P1	851.3	7.0 s	-2.7%	875.2	8 h	**-10.8%**	780.9	777.2 s
41P1	968.7	3.2 s	14.4%	846.8	8 h	1.8%	862.0	943.1 s
46P1	777.2	2.8 s	-0.6%	781.7	8 h	**-6.6%**	730.4	1110.6 s
47P1	710.0	1.6 s	0.0%	710.1	8 h	0.0%	710.0	2270.9 s
48P1	870.9	2.1 s	0.0%	871.0	8 h	0.0%	870.9	3306.7 s
49P1	674.8	2.7 s	8.6%	621.1	8 h	**-1.5%**	611.8	953.9 s
51P1	924.7	3.3 s	-2.9%	952.1	8 h	**-6.3%**	892.2	1021.3 s
54P1	1443.6	2.8 s	-6.2%	1538.5	8 h	**-23.6%**	1175.9	706.0 s
55P1	1045.5	2.9 s	-4.4%	1094.1	8 h	**-13.3%**	948.3	1102.1 s
58P1	1057.7	1.5 s	3.6%	1021.1	8 h	**-14.0%**	878.6	1422.6 s
Average:	**956.8**	**3.4 s**	**2.9%**	**939.3**	**8 h**	**-5.0%**	**882.7**	**1209.4 s**

In Table 4.3, the performance of ETP and the GA is compared with CPLEX regarding both objective function and running time. Gap$_{EC}$ and Gap$_{GC}$ columns show the percentage of difference in the objective function of CPLEX compared to ETP and the GA, respectively. Negative signs indicate improvement. The maximum improvement (or minimum gap%) in the total cost of the instances are shown in bold.

According to the results in Table 4.3, the GA outperforms CPLEX in almost all the benchmark instances (except 13P1 and 41P1, shown in highlighted gray, in which the gap is less than 2%) regarding objective function. The GA is always expected to obtain better or at least not worse solutions than ETP. To optimize a network ETP either uses the NNS or CLK to construct a solution and then perform the improvement algorithms on the solution. In this GA, as mentioned in Section 4.6.2, two of the initial solutions of are the NNS's and CLK's solutions. Also, the GA uses all the improvement methods in the improvement step of ETP in its mutation process (Section 4.6.5). Therefore, the GA's solution is most likely to be at least as good as ETP unless

the solutions of NNS and CLK are not selected to mutate in any generation. The performance of the GA is also graphically represented in Figure 4.4.

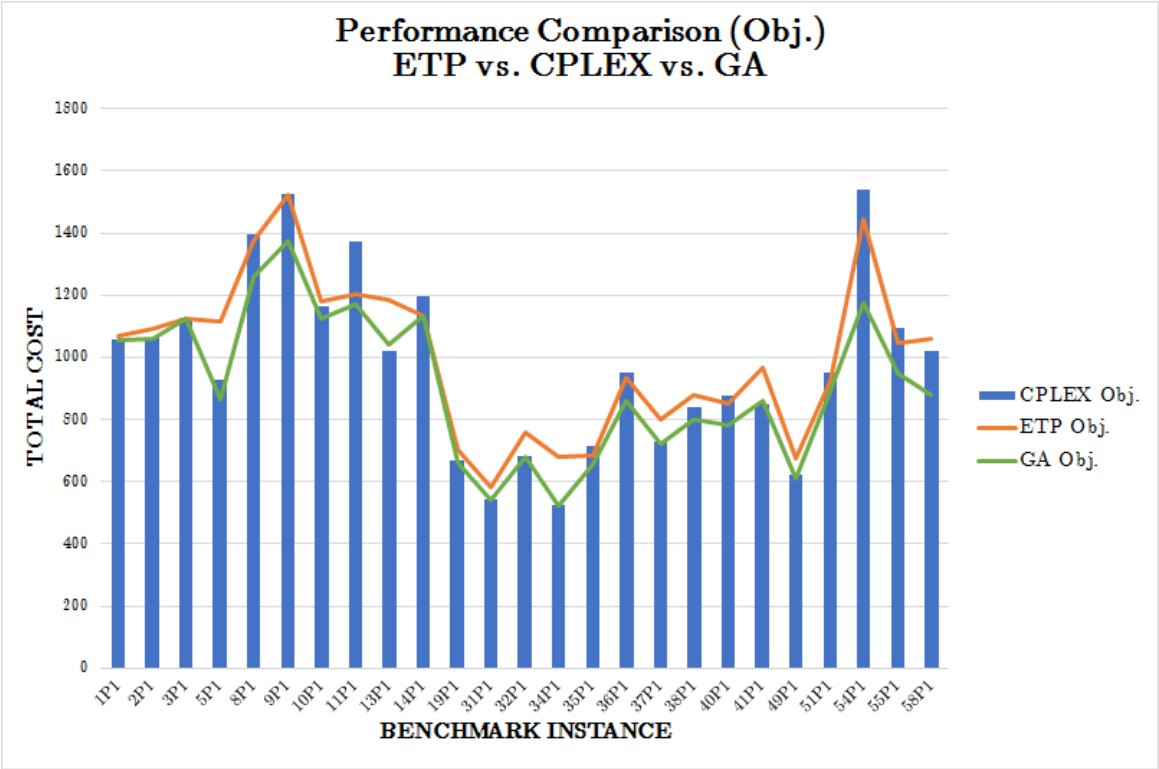


Figure 4.4: Performance (obj. function) of ETP vs. CPLEX vs. GA (default settings)

According to Figure 4.4, the GA performs well in finding high-quality solutions for the benchmark instances. The weak point of the GA is the running time which can make solving real-world problems impractical. In the following section we consider the components of the GA in each generation and determine the best parameter settings for reducing the running time.

4.7.1 GA Parameter Settings Analysis

The default parameter settings for all the benchmark instances are stated in Table 4.4. These parameters are derived from executing the GA on a subset of benchmark instances.

Table 4.4: Default Parameter Setting of the GA

Parameter	Value
# of generations	50
Population size	50
Crossover	80%
Mutation	10%
Max run time	60 min

According to Table 4.4, 80% of the population of the next generation are created by crossover and 10% by mutation. The remaining 10% are the best chromosomes of the current generation that move directly to the next. The percentage of the population in each generation that are produced by mutation is set to only 10% to reduce the total time of the GA. According to the analysis of the heuristic methods in Tables 4.1 and 4.2, the improvement algorithms take a large proportion of the total running time after both NNS and CLK constructive heuristics (78% and 58%, respectively). As mentioned in Section 4.6.5, we use the same improvement algorithms to mutate chromosomes. Therefore, reducing the number of mutant chromosomes will reduce the running time of the GA significantly.

In Table 4.5, the performance of each component of the GA (initial population, crossover, and mutation) are reported. In this table, the best objective function of the initial population and the run time required to create the chromosomes are shown. The same information about the crossover and mutation process is also provided.

Table 4.5: The Performance of the Components of GA (default settings as in Table 4.4)

Instance	**Ini. Population**		**Crossover**		**Mutation**			**GA**	
ID	*Best Obj.*	*Time*	*Best Obj.*	*Time*	*Best Obj.*	*Time*	**# Gen.**	*Best Obj.*	*Time*
1P1	1179.3	1.4 s	1681.5	0.3 s	**1057.1**	126.9 s	6	1057.1	127.2 s
2P1	1117.0	1.7 s	1534.8	0.3 s	**1060.0**	14.1 s	1	1060.0	14.5 s
3P1	1144.4	1.4 s	1745.0	0.3 s	**1124.6**	33.1 s	1	1124.6	33.4 s
4P1	1000.7	1.7 s	911.7	8.0 s	**906.2**	410.0 s	21	906.2	418.1 s
5P1	874.0	1.9 s	870.4	5.2 s	**864.5**	264.2 s	15	864.5	269.4 s
8P1	1340.2	1.4 s	**1259.2**	6.7 s	1274.4	266.9 s	18	1259.2	273.6 s

Continued on the next page

Table 4.5 – Continued from previous page

Instance	Ini. Population		Crossover		Mutation			GA	
ID	*Best Obj.*	*Time*	*Best Obj.*	*Time*	*Best Obj.*	*Time*	**# Gen.**	*Best Obj.*	*Time*
9P1	1483.6	1.6 s	**1374.6**	5.0 s	1382.4	190.1 s	13	1374.6	215.1 s
10P1	1179.3	1.4 s	1681.5	0.3 s	**1125.0**	20.4 s	1	1125.0	20.7 s
11P1	1314.8	1.5 s	1794.0	0.3 s	**1170.9**	19.5 s	1	1170.9	19.8 s
13P1	1146.9	1.3 s	1581.2	0.3 s	**1039.2**	18.9 s	1	1039.2	19.2 s
14P1	1136.0	1.3 s	1687.2	0.3 s	**1133.4**	16.2 s	1	1133.4	16.5 s
19P1	702.4	1.5 s	**662.8**	2.7 s	665.5	109.4 s	8	662.8	112.1 s
21P1	811.7	1.5 s	1365.9	0.3 s	**803.9**	20.7 s	1	803.9	21.1 s
34P1	571.9	1.2 s	**523.2**	7.5 s	548.7	283.3 s	21	523.2	290.8 s
36P1	883.6	1.4 s	867.7	3.1 s	**860.1**	127.6 s	9	860.1	130.7 s
40P1	783.5	1.3 s	**780.9**	2.6 s	781.1	97.2 s	8	780.9	99.9 s
41P1	871.1	1.4 s	**862.0**	1.6 s	880.3	75.4 s	5	862.0	77.0 s
46P1	784.2	1.5 s	763.7	1.0 s	**730.4**	56.3 s	3	730.4	57.3 s
51P1	943.2	1.9 s	**892.2**	3.4 s	895.5	243.1 s	8	892.2	246.5 s
54P1	1262.7	1.4 s	1176.8	5.2 s	**1175.9**	76.0 s	15	1175.9	194.6 s
55P1	1053.6	1.9 s	961.1	11.6 s	**948.3**	445.0 s	21	948.3	456.6 s
58P1	885.6	1.6 s	**878.6**	1.0 s	884.0	100.4 s	2	878.6	101.4 s
Average:	**1021.4**	**1.5 s**	**1175.3**	**3.0 s**	**968.7**	**137.0 s**	**8.2**	**965.1**	**146.2 s**

The "#Gen." column in Table 4.5 shows the last generation in which the GA solution improved. For example, in 1P1 instance, we found the best solution of the GA in the first generation after initialization. In this instance, the crossover time is 0.3 seconds and the mutation time is 20.4 seconds. In instance 15P1 the procedure executed for 15 generations before finding the best solution. Implementing the crossover and mutation in these generations took 5.2 and 264.2 seconds, respectively. In some benchmark instances, such as instance 31P1, the initial population contains the best solution of GA and the remaining 50 generations do not improve this solution. These instances are not reported in Table 4.5 to prevent misleading information on the average values provided in the last row of the table.

Note that, unlike in the mutation process, the solution after crossover can be worse than the parents regarding the objective function. In Table 4.5, these instances are shown in highlighted gray. In the GA, the chromosomes that are produced by crossover are needed for exploring the

feasible region and skipping from local optima. In Table 4.5, the best objective function found by the crossover or mutation process is shown in bold. In 8 out of 22 instances (36%) the best solution is found by the crossover operator and in the remaining by mutation.

On average, creating the initial population takes only 1.5 seconds. The crossover is much faster than the mutation process. However, the average total cost of the best solutions found by mutation is 17.6% less than crossover. The running time of crossover with 80% contribution in creation of the next generations is 0.4 seconds per generation, on average. The average time of mutation per generation is 16.7 seconds with the GA parameters stated in Table 4.4. On average, the best solutions are found after 8.2 generations. The maximum number of generations required to obtain the best found solution is 21 (in 4P1, 34P1 and 55P1 instances). In instance 34P1, the 21 generations require 290.8 seconds versus 789.7 required to execute all 50 generations (see Table 4.3). By reducing the number of generations from 50 to 21 (58%), we can save 63% in running time. This almost linear relationship between the number of generation and the total running time holds for all other instances. Therefore, to reduce the time required by the GA to solve large real-world problems, the most sensitive parameters are the number of generations and the mutation percentage.

So far in this section we have provided an analysis on the three parameters of the GA: i) the number of generations, ii) the crossover and iii) the mutation percentage of the GA. In Table 4.6, we analyze the impact of the last parameter of the GA, the population size, on the quality and running time of the problem instances in Table 4.5. The population size of each generation is reduced to 25 (from 50) while the other parameters are set to their default values as Table 4.4.

Table 4.6: The Impact of the Population Size on the Performance of GA (with default values for other parameters as in Table 4.4)

Instance	**Pop. Size = 50**			**Pop. Size = 25**			**Pop. Size = 10**		
ID	**# Gen.**	**Best Obj.**	**Time**	**# Gen.**	**Best Obj.**	**Time**	**# Gen.**	**Best Obj.**	**Time**
1P1	6	1057.1	1116.3 s	4	1057.1	496.6 s	25	1057.1	104.2 s
2P1	1	1060.0	1255.6 s	1	1060.0	478.5 s	2	1060.0	215.1 s

Continued on the next page

Table 4.6 – Continued from previous page

Instance	Pop. Size = 50			Pop. Size = 25			Pop. Size = 10		
ID	**# Gen.**	**Best Obj.**	**Time**	**# Gen.**	**Best Obj.**	**Time**	**# Gen.**	**Best Obj.**	**Time**
3P1	1	1124.6	2016.5 s	1	1124.6	464.0 s	2	1124.6	205.0 s
8P1	18	1259.2	866.5 s	5	1259.2	380.0 s	21	1259.2	189.2 s
9P1	13	1374.6	949.9 s	6	1381.9	447.1 s	48	1374.6	198.1 s
10P1	1	1125.0	1089.7 s	12	1125.0	420.9 s	13	1125.0	263.6 s
11P1	1	1170.9	1185.8 s	1	1170.9	489.1 s	6	1170.9	290.7 s
13P1	1	1039.2	1438.0 s	8	1039.2	684.0 s	2	1039.2	188.6 s
14P1	1	1133.4	1047.2 s	1	1133.4	411.1 s	4	1133.4	183.8 s
19P1	8	662.8	1539.2 s	4	662.8	622.7 s	16	662.8	253.3 s
21P1	1	803.9	1420.5 s	1	803.9	737.8 s	1	811.7	212.6 s
34P1	21	523.2	789.7 s	2	523.2	326.8 s	9	523.2	143.2 s
36P1	9	860.1	780.0 s	11	860.1	422.9 s	7	860.1	209.1 s
40P1	8	780.9	777.2 s	19	766.0	371.4 s	34	782.9	148.2 s
41P1	5	862.0	943.1 s	1	862.0	377.2 s	2	862.0	212.1 s
46P1	3	730.4	1110.6 s	11	730.4	442.2 s	20	730.4	195.7 s
51P1	8	892.2	1021.3 s	6	892.2	469.7 s	48	892.2	202.9 s
54P1	15	1175.9	706.0 s	9	1175.9	439.4 s	24	1176.8	199.4 s
55P1	21	948.3	1102.1 s	7	994.1	436.0 s	20	994.1	149.7 s
58P1	2	878.6	1422.6 s	9	878.6	367.8 s	22	878.6	241.4 s
Average:	**7.2**	**973.1**	**1128.9 s**	**5.95**	**975.0**	**464.3 s**	**16.3**	**975.9**	**200.3 s**

As shown in Table 4.6, the population size is reduced from 50 which is the default value to 25, 10 and 5. On average, we saved 58.9% of running time with the price of 0.2% increment in the total cost by reducing the population size from 50 to 25. The reduction of population size from 25 to 10 results in 56.9% decrease in running time and only 0.1% increase in total cost, on average. Finally, reducing the population size from 10 to 5 decreases the running time by 21.4% and increases the total cost by only 0.4%, on average.

Based on these results, reducing the population size from 50 down to 5 has a positive impact on the solution obtained by the GA. This reduction makes the running time shorter without significant increment in the total cost. Thus, we choose the population size equal to 3 and ran the GA for the benchmark instances, again. This number is the smallest number that the population size can get without redesigning the GA, one for each reproduction operators (crossover and mu-

tation) and one for storing the best solution of the current generation to move it directly to the next generation (elitism). The result of having the population size of 3 in the GA is compared with the default number of population of 50 in Table 4.7.

Table 4.7: The Impact of the Default (50) and the Smallest (3) Population Size on the Performance of GA (with default values for other parameters as in Table 4.4)

Instance	**Pop. Size = 50**			**Pop. Size = 3**		
ID	**# Gen.**	**Best Obj.**	**Time**	**# Gen.**	**Best Obj.**	**Time**
1P1	6	1057.1	1116.3	9	1057.1	105.6
2P1	1	1060.0	1255.6	3	1060.0	29.4
3P1	1	1124.6	2016.5	2	1124.6	115.0
8P1	18	1259.2	866.5	25	1296.3	132.7
9P1	13	1374.6	949.9	26	1374.6	105.1
10P1	1	1125.0	1089.7	15	1125.0	213.1
11P1	1	1170.9	1185.8	5	1170.9	30.0
13P1	1	1039.2	1438.0	4	1039.2	96.5
14P1	1	1133.4	1047.2	2	1133.4	43.5
19P1	8	662.8	1539.2	9	665.5	85.6
21P1	1	803.9	1420.5	1	811.7	21.2
34P1	21	523.2	789.7	29	528.0	140.7
36P1	9	860.1	780.0	48	861.0	142.2
40P1	8	780.9	777.2	1	783.5	18.9
41P1	5	862.0	943.1	15	862.0	81.7
46P1	3	730.4	1110.6	29	730.4	179.1
51P1	8	892.2	1021.3	17	902.4	85.9
54P1	15	1175.9	706.0	20	1175.9	61.3
55P1	21	948.3	1102.1	3	994.1	96.9
58P1	2	878.6	1422.6	20	878.6	116.5
Average:	**7.2**	**973.1**	**1128.9 s**	**14.15**	**978.7**	**95.0 s**

As shown in Table 4.7, by reducing the population size from 50 to 3, we can save up to 91.6% in running time while increasing the total cost by 0.6%, on average. Therefore, the best population size ofr this GA is the smallest number which is equal to 3.

Interestingly, the number of generations required to find the best solution in GA is decreased by 17.4% after reducing the population size from 50 to 25. However, the GA has a 2.7 times increase in the number of generations by reducing the population size from 25 to 10, as ex-

pected. Again, the number of generations decreased by 23.9% by reducing the population size from 10 to 5. Finally, on average, reducing the population size from 50 to 5 almost doubles the number of generations (96.5%).

4.8 Conclusion

In this chapter, we provided extensive study on the performance of the heuristic algorithms that are introduced in Chapter 3 to find quality solutions for VRP variants. These heuristic algorithms are divided into two categories: i) constructive and ii) improvement. The constructive algorithms are used to create feasible solutions for the VRP problems and then improved by the improvement heuristics. The impact of each algorithm on the quality and running time of the solutions are studied. This study on both constructive and improvement algorithms provides insightful information to solve large real-world problems consist of thousands of shipments and hundreds of vehicles. Two constructive algorithms are used to find feasible solutions for the benchmark instances: i) modified Nearest Neighbor Search (NNS) and ii) modified Clarke-Wright (CLK) algorithms. The experimental results show that NNS is faster than CLK but the quality of the solutions obtained by CLK is higher that NNS. In solving 60 benchmark instances with 50 locations, the average running time of NNS is 0.8 seconds versus 1.1 seconds for CLK. On the other hand, the total travel cost of CLK is 8.1% lower than NNS. The 0.3 seconds difference between the NNS and CLK in the benchmark instances with 50 locations can make a huge difference in real-world problem size with thousands of shipment locations. The improvement algorithms in ETP are also categorized in two categories: i) Inter-Route and ii) Intra-Route algorithms. Inter-route improvements work on two or more routes of a solution and try to move or swap shipments or locations between them. On the other hand, Intra-Route algorithms improve each route of the solution by altering the sequence of the visited locations at the route. Note that both types of algorithms keep the solution feasible. According to the results of the analysis, the Inter-Route algorithms have a higher impact on improving the solution cost found by both NNS and CLK. The improvement in the solution of NNS in average (on the 60 benchmark instances) is 5.6% in

2.4 seconds and CLK is 0.9% in 1.3 seconds. Note that the running time of the Inter-Route improvements on the solutions found by NNS or CLK are more than when creating the solutions by these constructive algorithms (2.4s vs. 0.8s for NNS and 1.3s vs. 1.1s for CLK). The Intra-Route improvement algorithms could improve the solutions of NNS and CLK by 1.1% and 0.1% in 0.4 seconds and 0.2 seconds, respectively. In summary, the results of analyzing the constructive and improvement algorithms in this study on the small benchmark instances in the literature can guide us to choose the best parameter settings in solving the much larger real-world problems.

The second part of this study improves the quality of the solutions found by our developed optimization tool. A Genetic Algorithm (GA) is proposed to increase the quality of the solutions in the previous chapter. The GA utilizes the constructive and improvement heuristic methods of our optimization tool to evolve the generations. We used the results of the analysis on the heuristic methods in the first part of this chapter to set the parameters for the GA. The constructive algorithms are used to create the initial population and the improvement algorithms mutate the chromosomes in each generation. Computational study of the same benchmark instances as the previous chapter shows that the GA outperforms other tools regarding the solution quality at the cost of increased runtime. On average, the quality of the solutions obtained by GA is 7.7% better than ETP and 5% better than CPLEX. To reduce the running time of the GA while maintaining the quality an analysis on the components of the GA is conducted. This experimental study showed a 91.6% reduction in running time while increasing the total cost only by 0.6%.

Bibliography

Baker, B. M. and Ayechew, M. (2003). A genetic algorithm for the vehicle routing problem. Computers & Operations Research, 30(5):787–800.

Beasley, J. E. (1983). Route firstcluster second methods for vehicle routing. Omega, 11(4):403–408.

Berger, J. and Barkaoui, M. (2004). A parallel hybrid genetic algorithm for the vehicle routing problem with time windows. Computers & operations research, 31(12):2037–2053.

Clarke, G. and Wright, J. W. (1964). Scheduling of vehicles from a central depot to a number of delivery points. Operations research, 12(4):568–581.

Gehring, H. and Homberger, J. (2001). A parallel two-phase metaheuristic for routing problems with time windows. Asia-Pacific Journal of Operational Research, 18(1):35.

Ho, W., Ho, G. T., Ji, P., and Lau, H. C. (2008). A hybrid genetic algorithm for the multi-depot vehicle routing problem. Engineering Applications of Artificial Intelligence, 21(4):548–557.

Holland, J. H. (1975). Adaptation in natural and artificial systems. an introductory analysis with application to biology, control, and artificial intelligence. Ann Arbor, MI: University of Michigan Press.

Lin, S. (1965). Computer solutions of the traveling salesman problem. The Bell System Technical Journal, 44(10):2245–2269.

Nagata, Y. (2007). Edge assembly crossover for the capacitated vehicle routing problem. In European Conference on Evolutionary Computation in Combinatorial Optimization, pages 142–153. Springer.

Nagata, Y. and Bräysy, O. (2009). Edge assembly-based memetic algorithm for the capacitated vehicle routing problem. Networks, 54(4):205–215.

Ombuki, B., Ross, B. J., and Hanshar, F. (2006). Multi-objective genetic algorithms for vehicle routing problem with time windows. Applied Intelligence, 24(1):17–30.

Prins, C. (2004). A simple and effective evolutionary algorithm for the vehicle routing problem. Computers & Operations Research, 31(12):1985–2002.

Tan, K. C., Chew, Y. H., and Lee, L. (2006). A hybrid multiobjective evolutionary algorithm for solving vehicle routing problem with time windows. Computational Optimization and Applications, 34(1):115.

Toth, P. and Vigo, D. (2014). Vehicle routing: problems, methods, and applications, volume 18. Siam.

Van Breedam, A. (1994). An Analysis of the Behavior of Heuristics for the Vehicle Routing Problem for a Selectrion of Problems with Vehicle-related, Customer-related, and Time-related Constraints. RUCA.

Vidal, T., Crainic, T. G., Gendreau, M., Lahrichi, N., and Rei, W. (2012). A hybrid genetic algorithm for multidepot and periodic vehicle routing problems. Operations Research, 60(3):611–624.

Appendix

Appendix 4.A Certification of Student Work

UNIVERSITY OF ARKANSAS

College of Engineering
Department of Industrial Engineering

Date: December 5, 2017

Graduate School
University of Arkansas

Dear Dr. Needy:

I am writing to verify that Fereydoun Adbesh completed more than 51% of the work for the chapter titled “A Genetic Algorithm for Unified Vehicle Routing Problems” in his dissertation.

Sincerely,

Chase Rainwater
cer@uark.edu
479-575-2687
Associate Professor
Department of Industrial Engineering
University of Arkansas

5. Conclusions and Future Work

In this dissertation, we study on and propose novel approaches for two applied operation research optimization problems, scheduling, and routing. We first study on the allocation and scheduling of a fleet of dredges in inland and intra-coastal waterways of the United States under environmental and resource constraints. Secondly, we focus on the several variants of the Vehicle Routing Problem (VRP). We investigate applying the different variations of well studied VRP on modeling and solving the large scaled real-world problems. We model new variants and develop heuristic and meta-heuristic algorithms to find good feasible solutions for our transportation problems. Finally, we identify the most efficient algorithms and parameter settings for them.

In Chapter 2, dredge fleet allocation and scheduling under environmental work windows and resource constraints are studied. Environmental work windows are the only time that dredging are allowed during the planning time horizon to prevent any harm to the local wild life. The horizon period is usually a year and the Restricted Periods (RP) are the time dredging is not allowed. We used Constraint Programming (CP) approach to solve this heavily constraints scheduling problem while ILP model fails to solve even very small problem instances. In general, CP works well in both scheduling problem and in heavily constraints system. In scheduling problems, CP finds high quality solutions by defining the the decision variables as the interval constraints. Complicated constraints in CP can be handled effectively by the aim of the global (logical) constraints. Using this privilege, we developed a CP model for the dredge scheduling problem with objective of maximizing the total cubic yards of dredging. In this chapter, we further investigated the different variations of the dredge scheduling by altering some fundamental assumptions of our base model. We successfully model and solve different variations of the problem such as the partial dredging during RP, the variable dredge job sizes, the multiple trip to the same job by dredges, the simultaneous work of different dredges on the same job, the sequence dependent mobilization/demobilization cost and the job dependent production rate of dredges. Our computational efforts demonstrated that the CP models perform very well on the problems by finding high-quality feasible solutions very quickly.

In Chapter 3, several variations of the well-known VRP is stated and combined to model the real-world transportation problems. We introduce a new variant of the VRP by considering the driver time regulations and develop some heuristic algorithms to solve it along with other variants of the VRP derived from this model. We provide a comprehensive tool to solve the VRP variants and their combinations in a single software. Computational results on several benchmark instances show that this software can find quality solutions in a very short amount of time comparing with the exact methods of solving the corresponding MIP formulations of the problems in CPLEX. At the end of this chapter, a case study with real data consist of thousands of customer locations and hundreds of vehicles is presented and solved.

Finally, in Chapter 4, we prioritize the efficiency of the heuristic methods used to solve and improve the VRP variants in Chapter 3. The remaining of this chapter belongs to developing a Genetic Algorithm (GA) to solve our comprehensive model. As some of the heuristic algorithms in the previous chapter are used in the GA, we used the results of the first part of this chapter to choose the best algorithm and parameter settings for the GA. Computational results on the same benchmark instances in Chapter 3 demonstrate the power of the GA to find high quality solutions with the price of increasing the running times.

5.1 Future Work

Postdoctoral work relating to chapter 2 includes, but is not limited to, development of dredge scheduling problem in extended periods of time. We seek to find the best plan for operation and maintenance (O&M) of the waterways concerning how frequent we need to dredge a plant to keep it navigable over an extended period.

Future work relating to chapters 3 and 4 involves the comprehensive analysis of the impact of the hours of service regulations on the drivers, trucking companies and transportation industry. In addition, investigating new variants of VRP with hours of service regulations to provide convenient work shifts for drivers in an extended planning time horizon is of our interest to get closer and closer to the real problems challenged by transportation companies. We also plan to contin-

uously develop new heuristic and meta-heuristics algorithms to make our optimization tool more effective and efficient.