ABSTRACT

Title of dissertation:	PARALLEL AND SERIAL ALGORITHMS FOR VEHICLE ROUTING PROBLEMS
	Christopher Groër, Doctor of Philosophy, 2008
Dissertation directed by:	Professor Bruce Golden, Robert H. Smith School of Business

The vehicle routing problem (VRP) is a widely studied combinatorial optimization problem that has many applications. Due to its intrinsic difficulty and the size of problems encountered in practice, most solution methods for the VRP are heuristic in nature and lead to high quality, yet probably not optimal solutions. When one considers the additional constraints that can be encountered in practice, the need for high quality heuristic methods is clear.

We present two new variations of the VRP suggested to us by industry contacts, the Consistent VRP and the Balanced Billing Cycle VRP. We develop solution algorithms that incorporate heuristic methods as well as integer programming. Additionally, we develop a highly effective cooperative parallel algorithm for the classical VRP that generates new best solutions to a number of well-studied benchmark instances. We present extensive computational results and describe the C/C++ library that we developed to solve these vehicle routing problems. We describe the features

and design philosophy behind this library and discuss how the framework can be used to implement additional heuristic algorithms and incorporate additional constraints.

PARALLEL AND SERIAL ALGORITHMS FOR VEHICLE ROUTING PROBLEMS

by

Christopher Groër

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2008

Acknowledgments

I first thank my wife, Sharon, for her help during this time. Sharon allowed me to dedicate a great deal of time and effort to this project, and I would never have been able to complete this dissertation without her encouragement, support, and sacrifice. I also thank our young daughter, Pearce, who issued only mild protests when I repeatedly ignored her so that I could continue to work on this project.

This dissertation would not have been possible without the enthusiasm, support and direction of my advisor, Bruce Golden. He first introduced me to the world of Operations Research and the tantalizingly difficult VRP. His extensive experience and knowledge of this field was an invaluable asset. I also owe a debt of gratitude to Professor Edward Wasil who gave a great deal of valuable input into the various projects we worked on together. We had many valuable discussions, and his very careful editing of papers that we worked on together has allowed me to express my ideas and technical work more clearly.

I thank Zhi-Long Chen, Bill Dorland, and Paul Schonfeld, for taking time out of their busy schedules to serve on my dissertation committee. Alverda McCoy deserves a special thanks for taking care of many administrative matters over the years. I thank Larry Levy of RouteSmart Technologies, Inc. for providing data sets and for valuable conversations involving the complexities of real-world routing problems, my coworkers Bob Ebersole and Matt Fante for help in maintaining the Microsoft Compute Cluster, and Paul Kent of ILOG and fellow graduate student William Mennell for help with CPLEX-related issues. Fellow graduate students William Mennell and

Damon Gulczynski were early users of some of my VRP software and gave valuable feedback and bug reports. Finally, I thank my parents for encouraging me to pursue this degree and my employer for providing financial support.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

API	Application Programming Interface
BBCVRP	Balanced Billing Cycle Vehicle Routing Problem
ConVRP	Consistent Vehicle Routing Problem
GMPL	GNU MathProg Language
GLPK	GNU Linear Programming Kit
IP	Integer Program
IRP	Inventory Routing Problem
LKH	Lin-Kernighan-Helsgaun
LP	Linear Program
MIP	Mixed Integer Program
MPI	Message Passing Interface
PVRP	Period Vehicle Routing Problem
RTR	Record-to-Record
RRTR	Randomized Record-to-Record
TSP	Traveling Salesman Problem
UPS	United Parcel Service
VRP	Vehicle Routing Problem
VRPH	Vehicle Routing Problem Heuristic library
VRPTW	Vehicle Routing Problem with Time Windows

Chapter 1

Introduction

The efficient distribution of goods and services lies at the heart of many daily activities. These distribution problems often involve the routing of vehicles, and examples abound: a child being picked up by a school bus, a garbage truck traversing the streets of a residential neighborhood, the electric company reading the meters of its customers once a month, or the mailman visiting all the houses in a neighborhood. The cumulative costs of operating the vehicles participating in these activities can be very large, and so it is important to route these vehicles as efficiently as possible. However, developing efficient solutions to these types of problems requires us to solve a combinatorial optimization problem that has proven to be extremely difficult.

Problems of this type were first formally introduced in 1959 by Dantzig and Ramser [25], and hundreds of papers have been published on this topic in the last fifty years. In the classical VRP, we are given a fleet of homogeneous vehicles and a set of customer locations where each customer has a known demand. The task is to construct a minimum cost set of routes that meets all customer demand while also ensuring that the individual routes adhere to vehicle capacity and route length restrictions. Although the VRP is simple to state and easy to understand, experience has shown it to be very difficult to solve in practice. Furthermore, there are many additional complications that occur in practice: the firm may have more than one type of vehicle (the Heterogeneous VRP), certain customers may require service during a certain time period (the VRP with Time Windows), or the planning horizon may involve several days at a time (the Period VRP).

Because of the difficulty in solving the VRP and the number of additional considerations that arise in practice, problems of realistic size must typically be solved via heuristic methods. Exact solution methods for the VRP typically use integer programming or dynamic programming [9, 66], and even with modern computing resources, these methods are typically unable to solve to optimality problems with more than 100 customer locations. As real-world VRP instances can have several thousand nodes and can involve many additional constraints that complicate the problem even further, the need for high quality heuristic solvers is clear.

While the necessity of heuristic methods is undeniable, recent improvements in commercial mixed integer programming (MIP) solvers as well as increases in cheaply available computing power permit one to judiciously combine exact and heuristic methods. This combination of heuristic and exact methods is a powerful technique for solving the VRP that has become increasingly prevalent in recent years [17, 30, 79].

In this dissertation, we continue this trend of combining heuristic methods with exact integer programming methods. We create fast, powerful algorithms for the VRP and two real-world variations. Our procedures involve heuristic and integer programming methods, and we are able to solve problems of practical sizes using both real and simulated data.

The first contribution of this dissertation is the introduction of a new variant of the VRP that was suggested to us by UPS. We call this variation the Consistent VRP

(ConVRP). Here, we are presented with several days worth of service requirements for a large set of customers. Along with solving a VRP for several individual days, we must also satisfy *consistency* constraints: each customer must always be visited by the same driver at roughly the same time each day. In other words, if we are presented with a one week planning horizon and a particular customer requires service on three days, then we must ensure that the same driver visits this customer on each of these days at roughly the same time. We formulate the ConVRP as an integer program and solve small instances to optimality. We also develop a heuristic solution algorithm that we apply to larger problems. This procedure constructs a template of the most frequently visited customers and then modifies this template for each of the individual days in order to ensure that the consistency constraints are satisfied. We develop a set of benchmark problems for the ConVRP and also apply our solution algorithm to a real-world data set provided by UPS. We run a large number of computational experiments in order to measure the impact of these consistency constraints in terms of increased overall route length over the entire period. Finally, we use the UPS data to demonstrate that a solution to the ConVRP can be used to create consistent routes for future weeks of data with little loss in overall solution quality.

Next, we introduce a second VRP variant called the Balanced Billing Cycle VRP (BBCVRP). This variant is derived from a commercial application in the meter-reading industry where a utility must visit a large number of customer locations over the course of a one month billing period. The goal is to take an existing set of routes for each day of the billing cycle and then create a new, more efficient set of routes that is more balanced in terms of the distance traveled and the number of customers visited on each day of the billing cycle. Additionally, we must adhere to certain regulatory constraints that forbid a utility from changing a customer's billing day by more than a few days from one month to the next. We develop a solution algorithm for this complex VRP variant that combines heuristic and exact methods and apply our procedure to a real-world data set provided by RouteSmart Technologies, Inc.

Next, we present a cooperative parallel algorithm for solving the classical VRP. We develop and implement a completely asynchronous parallel algorithm. The majority of the processors are devoted to generating solutions to the VRP instance via a metaheuristic algorithm based on record-to-record travel [29]. We devote a second set of processors to solving a set covering integer program. These use routes extracted from the solutions generated by other processors and attempt to combine them in order to discover new solutions that are superior to any of the individual solutions. A single master processor stores the best solutions discovered during the search, manages the distribution of parameters to the other processors, and is responsible for the overall direction of the search. Our algorithm is able to quickly find solutions of high quality, and we find a number of new best solutions to benchmark problems that have been studied for decades. We run our procedure under a number of different scenarios in order to study the role of cooperation in the search as well as the influence of the number of processors on solution quality and computing time.

The three problems that we solved in this dissertation, the Consistent VRP, the BBCVRP, and the classical VRP, are different in several ways. However, they share enough similarities that we were able to develop a single C/C++ library to perform a great deal of the computational work. The final contribution of this dissertation is the description of our VRPH library (a library for VRP Heuristics). This library provides a large number of routines for creating and modifying solutions to VRP instances and offers a generic metaheuristic solver that can be used to solve nearly any instance of the classical VRP. The VRPH code will be released to the open source community and we document the library's design, describe several different ways to use the library, and discuss how VRPH can be further extended to handle additional VRP variants with relatively little modification.

The dissertation is organized as follows. We describe the Consistent VRP in Chapter 2 and the Balanced Billing Cycle VRP in Chapter 3. We present the parallel algorithm for the classical VRP in Chapter 4 before presenting the VRPH library in Chapter 5 and then giving our conclusions. Appendix A contains additional information and solutions related to the Consistent VRP. Appendix B contains additional information, data, and solutions related to the parallel algorithm. We include the best solutions that we found for the standard benchmark problems taken from the literature. Finally, Appendix C contains technical information about the VRPH library.

Chapter 2

The Consistent Vehicle Routing Problem

In the small package shipping industry (as in other industries), companies try to differentiate themselves by providing high levels of customer service. This can be accomplished in several ways including on-line tracking of packages, ensuring on-time delivery, and offering residential pick ups. Some companies want their drivers to develop relationships with customers on a route and would like the same drivers to visit the same customers at roughly the same time on each day that the customers need service. These service requirements together with traditional constraints on vehicle capacity and route length define a variant of the classical capacitated vehicle routing problem (VRP) that we call the Consistent VRP (ConVRP). In this chapter, we formulate the problem as a mixed integer program (MIP) and develop an algorithm to solve the ConVRP that is based on the record-to-record travel algorithm. We compare the performance of our algorithm to the optimal MIP solutions for a set of small problems and then apply our algorithm to five simulated data sets with 1,000 customers and a real-world data set with more than 3,700 customers. We provide a technique for generating ConVRP benchmark problems from VRP instances given in the literature and provide our solutions to these instances. The solutions produced by our algorithm on all problems do a very good job of meeting customer service objectives with routes that have a low total travel time.

2.1. Introduction

The traditional vehicle routing problem has been studied by researchers and practitioners for nearly 50 years, dating back to the early work of Dantzig and Ramser [25]. For the most part, the focus has been on developing a set of routes for a homogeneous fleet of vehicles that minimizes the total cost or the total distance traveled by the fleet, subject to a set of constraints.

Over the last five years or so, there has been a shift in practice from fleet-focused considerations to those that are customer-focused. Drivers for United Parcel Service (UPS) "...form a real bond with customers [and] take tremendous ownership of their customers and routes" [81]. This is a significant competitive advantage for UPS as drivers have gathered sales leads that have generated an additional "...volume of more than 60 million packages a year." Each day 103,500 UPS drivers visit 7.9 million customers and handle an average of 15.6 million packages [81]. In the last year, UPS acknowledged the following customer-service issue [59].

> "One of the weaknesses of existing research on [the] classical VRP is that not much attention is paid to the benefits of customers getting service from the same service provider at about the same time over multiple days. This service provider consistency is very important for UPS to provide good and efficient service."

We combine the service provider consistency requirements (the same driver visits the same customers at roughly the same time on each day that these customers need service) with traditional constraints on vehicle capacity and route length and define a variant of the traditional vehicle routing problem that we call the Consistent VRP (ConVRP). The ConVRP is the same problem faced by UPS over a multiple-day delivery period.

In Table 2.1, we show the modeling focus for traditional VRP variants including the ConVRP. In the past, researchers and practitioners considered the fleet, the driver, and the demand in their models. Customer service considerations are now being added to variants of the VRP. Campbell and Thomas [15] present a number of industry statistics and discuss the importance of both on-time delivery and consistent service within the highly competitive package delivery industry. The ConVRP is the first VRP variant that we have come across whose primary focus is customer satisfaction. For companies like UPS, providing consistent service (with respect to time) can be more important in practice than saving an incremental one, two, or three percent in travel costs.

The rest of this chapter is organized as follows. In Section 2.2, we model the ConVRP as a mixed integer program. In Section 2.3, we develop an algorithm for solving the ConVRP. In Section 2.4, we report computational results on test problems. In Section 2.5, we give our conclusions.

2.2. Modeling the ConVRP

In the standard version of the VRP, a homogeneous fleet of vehicles is based at a single depot. Each vehicle has a fixed capacity and must leave from and return to the depot. There are N customers and each customer has a known demand and is

Problem	Objective	Focus	Comments
Traditional VRP	Minimize cost or distance	Fleet	Toth and Vigo [80] provide an overview of the VRP and its variants. Cordeau et al. [22] review new heuristics that solve the VRP.
VRP with balance	Balance routes on each day	Driver	On a given day, each driver should have about the same amount of work. Levy and Bodin [49] present several criteria for generating good solutions including balanced daily work schedules. Levy and Bodin [50], Sniezek and Bodin [74], and Jozefowiez, Semet, and Talbi [41, 42, 43] extend this work.
Period vehicle routing problem (PVRP)	Account for time-sensitive demands	Demand	In the PVRP, routes are produced over several days to visit customers with known frequencies of service (e.g., a customer requires service on Tuesday and Thursday). Francis et al. [31] define operational complexity as the difficulty in implementing a solution to the PVRP. They present measures of operational complexity including arrival span (the variability of the time of day when customers are serviced) and crewsize (the number of different drivers to visit a customer) which are similar to our consistency requirements.
Inventory routing problem (IRP)	Ensure that customers do not run out of product	Demand	In the IRP, demands are stochastic and there are no customer orders. Instead, the delivery company decides when to visit each customer, based on forecasts, communications, and monitoring. The planning horizon is multiple days in length. Campbell et al. [14] discuss a wide variety of practical aspects. Moin and Salhi [55] provide a detailed overview of the IRP.
Consistent VRP (ConVRP)	Be consistent from one day to the next with customers	Customer	According to UPS (2007), many of its drivers have worked the same route for 20 years or longer. Customer schedule consistency is an important service criterion for UPS according to Wong [82]. Wong [83] discusses a number of additional practical issues.

Table 2.1: Modeling focus for the vehicle routing problem

serviced by exactly one visit of a single vehicle. A route must be developed for each vehicle so that all customers are serviced and the total distance traveled by the fleet is minimized.

In the ConVRP, there are D days of service requirements. Each customer must be serviced on a specific day (the days are known in advance) and each customer can receive service at most once on any day from any one of at most K identical vehicles. When a customer receives service, the same driver visits the customer at roughly the same time over the D-day planning horizon, so that the maximum arrival time variation (between latest and earliest arrival times) is no more than L time units. The service time at customer i on day d is denoted by s_{id} and the demand at customer i on day d is denoted by q_{id}. We define t_{ij} to be the deterministic, symmetric travel time between any two locations i and j. On each day, a vehicle has a capacity of Q units and can operate for no more than T units of time. The objective is to develop a set of routes for the fleet of vehicles that minimizes the total vehicle operating time over the D days.

We point out that the fleet is homogeneous: all K vehicles have the same capacity and are capable of servicing any customer. This matches the practice of many small package shipping companies which use a standard size vehicle to provide deliveries (e.g., the UPS Big Brown Truck ©). In addition, we do not account for hard time windows. In practice, most residential customers do not have hard time windows for delivery. In the ConVRP, by adhering to consistency requirements, we are able to guarantee that a customer will be serviced at roughly the same time over the next D days.

We formulate the ConVRP as a mixed integer program and we are able to use this formulation to solve several small problems. Set $w_{id} = 1$ if customer i requires service on day d and $w_{id} = 0$ otherwise ($i = 0$ is the depot). Let a_{id} equal the arrival time at customer i on day d and set $a_{id} = 0$ if customer i does not require service on day d. The decision variable x_{ijkd} equals 1 if vehicle k visits customer j immediately after customer i on day d and equals 0 otherwise. The decision variable y_{ikd} equals 1 if customer i is visited by vehicle k on day d and equals 0 otherwise. Using this notation, the objective function and constraints of the MIP are as follows:

$$\text{Minimize} \quad \sum_{d=1}^{D}\sum_{k=1}^{K}\sum_{i=0}^{N}\sum_{j=0}^{N} t_{ij}x_{ijkd} \tag{2.1}$$

$$\text{s.t.} \quad y_{0kd} = 1 \text{ for all } k, d \tag{2.2}$$

$$a_{0d} = 0 \text{ for all } d \tag{2.3}$$

$$\sum_{k=1}^{K} y_{ikd} = w_{id} \text{ for } i \geq 1 \text{ and all } d \tag{2.4}$$

$$\sum_{i=1}^{N} q_{id}y_{ikd} \leq Q \text{ for all } k, d \tag{2.5}$$

$$\sum_{i=0}^{N} x_{ijkd} = \sum_{i=0}^{N} x_{jikd} = y_{jkd} \text{ for all } j, k, d \tag{2.6}$$

$$w_{id_\alpha} + w_{id_\beta} - 2 \leq y_{ikd_\alpha} - y_{ikd_\beta} \leq -(w_{id_\alpha} + w_{id_\beta} - 2) \text{ for all days } d_\alpha \text{ and } d_\beta,\ \alpha \neq \beta, \text{ and all } i, k \tag{2.7}$$

$$a_{id} + x_{ijkd}(s_{id} + t_{ij}) - (1 - x_{ijkd})T \leq a_{jd} \text{ for all } d, k;\ i \geq 0,\ j \geq 1 \tag{2.8}$$

$$a_{id} + x_{ijkd}(s_{id} + t_{ij}) + (1 - x_{ijkd})T \geq a_{jd} \text{ for all } d, k;\ i \geq 0,\ j \geq 1 \tag{2.9}$$

$$\sum_{i \in S, j \in S} x_{ijkd} \leq |S| - 1 \text{ for } S \subseteq \{1, 2, \ldots, N\}, \text{ with } 2 \leq |S| \leq N \text{ for all } k, d \tag{2.10}$$

$$0 \leq a_{id} + w_{id}(s_{id} + t_{i0}) \leq Tw_{id} \text{ for all } i \geq 1 \tag{2.11}$$

$$-L + T(w_{id_\alpha} + w_{id_\beta} - 2) \leq a_{id_\alpha} - a_{id_\beta} \leq L - T(w_{id_\alpha} + w_{id_\beta} - 2) \text{ for all } i \text{ and days } d_\alpha \text{ and } d_\beta,\ \alpha \neq \beta \tag{2.12}$$

$$x_{ijkd} \in \{0, 1\}; y_{ikd} \in \{0, 1\}; a_{id} \geq 0 \text{ for all } i, j, k, d \tag{2.13}$$

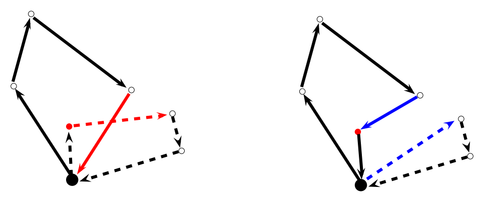
The objective function (2.1) minimizes the total travel time of the vehicles over all days. In (2.2) and (2.3), we require that the depot is visited at time 0 by all vehicles on all days. Constraint (2.4) ensures that customers are visited exactly once when they require service and (2.5) guarantees that each vehicle carries no more than Q units on any given day. In (2.6), each customer has only one predecessor and one successor and (2.7) ensures that each customer is served by the same driver whenever a customer requires service. Note that the quantity $(w_{id_\alpha} + w_{id_\beta} - 2)$ is negative unless customer i requires service on both days, in which case this quantity is zero. Constraints (2.8) and (2.9) determine the arrival times at the individual customers and (2.8) also serves to eliminate sub-tours in the individual daily routes (we prove this in Appendix A). Although they are redundant in this case, we also include the usual sub-tour elimination constraints (2.10). In our computational experiments with small problems, we found that including these constraints allowed us to halve our computation times. We note that constraint (2.9) could be removed if we wanted to allow the driver to wait at a location before traveling to the next customer in order to provide consistent arrival times. The vehicle travel time limit is defined in (2.11). In (2.12), the difference between arrival times at customer i on any two days α and β is no more than L units. As in (2.7), constraint (2.12) is redundant when customer i does not require service on both days.

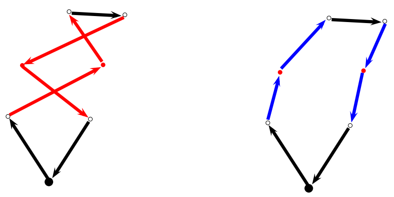
2.3. Solving Large ConVRPs

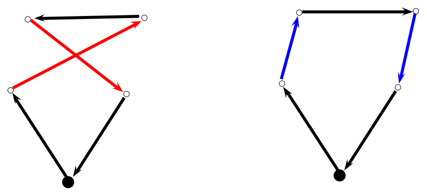
We now describe an algorithm for solving large-scale ConVRPs containing up to several thousand customer locations (solving our MIP is only practical for very small problems with a dozen or so customer locations). Many of the most successful metaheuristic algorithms for solving the standard VRP are quite complicated and involve many parameters. As the ConVRP adds several complicating constraints to the VRP, we wanted to develop an algorithm with a simple structure and relatively few parameters. The main feature of our procedure is a precedence principle, originally proposed by UPS: If customers a and b are both served by the same vehicle on a specific day and a is serviced before b, then customer a must receive service before customer b from the same vehicle on all days that they both require service. UPS believed that routes created by following this precedence principle would tend to adhere to the consistency constraints. In other words, if we consider only those customers that require service on more than one day and ensure that these customers are visited in the same order, then the resulting routes should lead to consistent service, even after we introduce additional customers that require service on only one of the D days. A key goal of this chapter is to study the performance of the precedence principle as a relatively simple heuristic approach to the problem of creating consistent routes.

Our algorithm has two stages. In the first stage, we generate a set of template routes by considering only those customers that require service on multiple days. In the second stage, using the template routes, we create routes for all days by removing customers from the template who do not require service on day d and then inserting customers who require service on only day d. This procedure guarantees that customers are always visited by the same vehicle when they require service and that the order of the customer visits will adhere to the precedence principle. Our hope is that the customers will also receive service at roughly the same time each day that they are serviced. We examine additional properties of the template such as the expected number of insertions and removals in Appendix A.

The template routes provide the primary structure of the routes for each of the D days and our procedure applies local search in an attempt to improve these routes. However, because the template routes themselves are never actually traversed by the vehicles, it is unclear how to treat the restrictions on a vehicle's total travel time and capacity. Thus, the actual travel time required to traverse a template route may be substantially larger than the time limit T, and the load allowed on a template route may be larger than the actual allowed capacity Q. To address these issues, when we are constructing and improving the template routes, we periodically derive each daily route from the template and check its feasibility with respect to travel time and capacity. If the total travel time of any daily route exceeds T or if a particular vehicle's load exceeds Q, then we decrease the bound on the length or capacity of a template route and regenerate the template routes until the daily route becomes feasible. On the other hand, if the template routes lead to daily routes that have travel times that are substantially smaller than T or capacities that are smaller than Q, then we increase the relevant limit and regenerate the template routes, thereby


(a) One-Point Move


(b) Two-Point Move


(c) Two-Opt Move

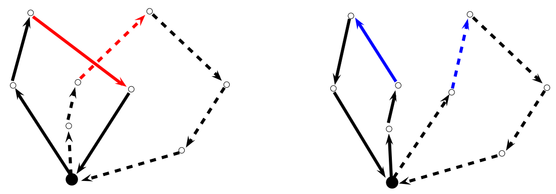

(d) Two-Opt Move

Figure 2.1: Improvement operators used in ConRTR

allowing the daily routes to either have larger total travel times or greater vehicle loads.

Our algorithm is based on the record-to-record (RTR) travel algorithm used by Li et al. [51] to solve very large-scale VRPs and is denoted by ConRTR (ConVRP Record-to-Record travel). We improve solutions using the following local search operators: one-point move, two-point move, and two-opt move. In a one-point move, we try to move each customer in the existing solution to a new position on the same route or on a different route. In a two-point move, we try to exchange the positions of two customers. We try the usual two-opt moves within a route and between routes by replacing two existing edges with two new edges. These improvement moves are shown in Figure 2.1.

The ConRTR algorithm attempts to create a high-quality solution to the ConVRP by creating a set of template routes that can be used to construct feasible routes for each of the days through simple removal and insertion procedures. The template is improved by repeatedly applying these three local search operators in a diversification phase followed by an improvement phase. In the diversification phase, we try to explore new areas of the solution space by accepting both improving and deteriorating moves. In the improvement phase, we attempt to improve the current solution as much as possible by accepting only improving moves until we reach a local minimum.

Step 1. Initialization

1.a We are given N total customers, a fleet of vehicles each with maximum total travel time T and capacity Q, and D days of service requirements. C denotes the current set of template routes being considered, F denotes the most recently generated set of template routes that is known to lead to feasible routes for each day, and F^* represents the set of template routes that leads to the lowest total travel time for the D days. The value of I is the number of iterations in the diversification phase, J is the maximum number of non-improving iterations allowed before returning a solution, α represents a tolerance for the amount of deterioration allowed in the local search, and λ is a parameter [84] used in the Clarke-and-Wright algorithm [21] to quickly generate multiple initial solutions. Given a set of template routes S, let $f(S)$ represent the total travel time of all D routes if they are feasible.

1.b Set $I = 30, J = 5, \alpha = .01, l = 1$, and $\lambda = \{0.6, 1.0, 1.4\}$.

1.c Set $C = F = F^* = \emptyset$.

1.d Partition the set of N customers into two groups, G_1 containing all customers requiring service on two or more days and G_2 containing all customers requiring service on only one day.

1.e Compute an expansion factor $E = |G_1|/\mu_{daily}$ where μ_{daily} is the mean number of stops required on each day and $|G_1|$ is the number of customers in the template. Make an initial estimate for the maximum capacity of the template routes by setting $Q_{template} = Q \times E = Q_0$ and estimate the maximum travel time for the template routes by setting $T_{template} = T/\sqrt{E} = T_0$.

1.f For all customers in G_1, set the demand amount and service time to be the mean values of these quantities taken across all days that the customer requires service.

Step 2. Create an initial set of template routes.

2.a Generate an initial set of template routes C for the customers in G_1 using the modified Clarke-and-Wright algorithm with parameter $\lambda[l]$, vehicle capacity $Q_{template} = Q_0$, and maximum travel time $T_{template} = T_0$.

2.b For each day d, create routes by removing customers from C not requiring service on day d and then inserting customers from G_2 requiring service only on day d.

2.c If the routes for all D days are feasible, set $F = C$, $Q_{old} = Q_{template}$, $T_{old} = T_{template}$. Go to Step 3.

2.d If at least one route on the D days is not feasible, then calculate the mean capacity violation (V_Q) and mean travel time violation (V_T) across all routes, and tighten the template constraints by setting $Q_{template} = Q_{template} - V_Q/2$ and $T_{template} = T_{template} - V_T/2$. Return to Step 2.a and try to generate a set of feasible template routes.

Step 3 Diversification Phase: Modify the current feasible template routes $C = F$. If $f(C) < f(F^*)$, set $F^* = C$.

3.a Set *Record* equal to the total travel time of all routes in the current template C. Set $Deviation = \alpha \times Record$.

3.b For $i = 1$ to I

Apply the one-point move, two-point move, and two-opt move with record-to-record travel to the current template routes C. Accept any improving move and only those deteriorating moves where the total travel time of all template routes is less than $Record + Deviation$ and all routes satisfy the template constraints.

Step 4. Improvement Phase: Improve the current solution C. Set $k = 0$.

4.a Apply the one-point move, two-point move, and two-opt move, accepting only improving moves until no further improvements can be found.

4.b Construct routes for each of the D days by applying the customer removal and insertion procedures.

4.c If the routes for all D days are feasible, set $F = C$. Compute the minimum slack amount across all daily routes in terms of capacity (S_Q) and travel time (S_T). Relax the template constraints by setting $Q_{template} = Q_{template} + S_Q/2$ and $T_{template} = T_{template} + S_T/2$. Go to Step 5.

4.d If at least one route on the D days is not feasible, then compute the mean violations (V_Q and V_T) as in Step 2.f and tighten the template constraints by setting $Q_{template} = Q_{template} - V_Q/2$ and $T_{template} = T_{template} - V_T/2$. Set $k = k + 1$. If $k < 5$, continue to find feasible improvements by returning to Step 4.a. Otherwise, return to the last known feasible template, setting $C = F$ and go to Step 5.

Step 5. If $f(C) < f(F^*)$, set $F^* = C$. If the objective function value of the current template routes C is less than *Record*, set $j = 0$. Set $j = j + 1$. If $j < J$, return to Step 3 and continue modifying the template. Otherwise, we have been unable to improve the current solution for J iterations, so stop modifying the current template and go to Step 6.

Step 6. Set $l = l + 1$ and return to Step 2 to generate a new initial solution if $l \leq 3$. Otherwise, use the best set of template routes (F^*) found during the search to generate the routes for each of the D days and return.

Before discussing the computational performance of our algorithm, we highlight a few of its more important features. The initial template routes that lead to feasible daily routes are generated by the modified Clarke-and-Wright algorithm using the λ parameter proposed by Yellow [84]. We make initial estimates of the template travel time and capacity limits in Step 1.e by comparing the number of template customers with the average number of stops per day and then computing an expansion factor E. After a template is found that leads to feasible routes for all D days, we run an improvement procedure while periodically altering the constraints on the template routes when we encounter daily routes that either violate the actual constraints or have significant slack. If a daily route violates a capacity limit or a travel time limit, then the offending limit (either Q or T) is decreased by one-half the mean violation amount (see Steps 2.d and 4.d). When the template is regenerated using this constraint, the idea is that since the template satisfies a tighter constraint, the daily routes will then be more likely to obey the actual constraints. On the other hand, if we have slack in all of the daily routes, then we increase each limit by one-half the mean amount of slack (see Step 4.c). This allows for more flexibility in the improvement operations and will hopefully lead to better solutions as the template is modified. At the end of ConRTR, we return to the template that led to the best solution across the D days and return these routes. Because the routes for each day are constructed from the template routes, the precedence principle holds on these final routes – any two customers requiring service on the same day more than once during the D days will be served in the same order and they will be serviced by the same driver.

2.4. Computational Experiments

In this section, we conduct several computational experiments that are designed to test the performance of ConRTR. First, we solve a set of small problems with 10 or 12 customer locations and compare the ConRTR solutions to the optimal solutions found by solving the mixed integer program given in Section 2. Second, we solve 40 randomly generated problems with 1,000 customer locations and examine the performance of ConRTR as we change the customers' service probabilities. Third, we construct a set of test problems from existing problems in the literature and solve them using ConRTR. Finally, we solve a problem with 3,715 customer locations that is based on five weeks of actual customer data provided by UPS. We coded ConRTR in C/C++ and all experiments were conducted on a machine with a 1.4 GHz Intel processor and 512 MB of RAM.

2.4.1 Small Problems

We generated five problems with 10 customer locations and five problems with 12 customer locations using the parameters below. The problem instances and the optimal solutions are given in Appendix A.

- Customer locations are generated uniformly at random within the square with vertices (0, 0), (10, 0), (10, 10), (0, 10).
- Travel times are defined to be the Euclidean distances.
- Depot is located at (0, 0).
- There are three days of known service requirements.
- Customers require service on each day with probability 0.7.
- For customers requiring service on a given day, demand is uniformly distributed on $[1, 3]$ and all service times are set to one unit.
- The maximum travel time is $T = 30$ and vehicle capacity is $Q = 15$ for all problems.
- The maximum arrival time differential is $L = 5$.

We compared the solutions produced by ConRTR on these 10 problems to the optimal solutions found by solving our MIP (we provide a modeling language formulation of the ConVRP in Appendix A). ConRTR does not explicitly account for the maximum arrival time differential L. Instead, it relies on the precedence principle to address this aspect of consistent service. To account for this, we assessed the performance of ConRTR in two ways. First, we solved each problem using ConRTR and calculated the largest arrival time differential (L_{max}) across all customers and then solved the MIP by setting $L = L_{max}$. Next, we ran ConRTR again, but discarded

Number of Nodes	ConRTR Solution $L = 5$	Optimal Solution $L = 5$	Gap	ConRTR Solution	L_{max}	Optimal Solution $L = L_{max}$	Gap
10	142.03	142.03	0	142.03	3.70	142.03	0
10	121.07	121.07	0	121.07	4.40	121.07	0
10	149.41	149.41	0	149.41	3.48	149.41	0
10	150.89	150.89	0	150.89	2.87	150.89	0
10	140.23	132.31	5.99%	130.77	12.37	130.77	0
12	171.05	171.02	0.02%	171.05	3.96	171.05	0
12	111.54	111.54	0	111.54	4.25	111.54	0
12	150.09	145.69	3.02%	145.67	5.26	145.67	0
12	168.45	166.37	1.25%	165.23	5.47	165.23	0
12	146.52	140.42	4.34%	140.29	7.90	139.75	0.38%

Table 2.2: Results for ConRTR and MIP on 10 small problems

all solutions that had $L > 5$ and compared these solutions to the optimal solutions produced by solving the MIP with $L = 5$.

In Table 2.2, we give the results produced by ConRTR and the MIP to these 10 problems. Columns 2 and 3 give the optimal and ConRTR solutions for the case of a fixed maximum arrival time differential of $L = 5$. The fourth column gives the best solution found by ConRTR when we ignore the $L = 5$ constraint along with the observed maximum arrival time differential L_{max}, and the final column lists the optimal solutions to the MIP with $L = L_{max}$. Note that, in several cases $L_{max} < 5$, so that the ConRTR solution satisfies the $L < 5$ constraint. The computing times for the MIP were very long, requiring up to several days using a state-of-the-art MIP solver (CPLEX 11.0), while ConRTR took less than one second to solve each problem. Nevertheless, the gap between the heuristic and optimal solutions is generally quite small and ConRTR finds the optimal solution in 14 of the 20 instances.

One interesting feature of the optimal solutions is that for all problems except one, the optimal solution adheres to the precedence principle and can be generated from a set of template routes. In other words, there is only one case where customers who require more than one visit over the three days are visited in a different order.

2.4.2 Large Simulated Problems

Next, we generated a set of random problems to assess ConRTR under a number of different scenarios. In particular, we were interested in the effect of the frequency of service requirements on its performance. For example, if all customers required service on every single day, then the template itself could be traversed by the vehicles, leading to a situation where the customers would be serviced at identical times across the different days. However, if customers required service with lower probabilities on each day, then the performance of ConRTR would suffer due to a larger number of modifications to the template as customers are inserted and removed when constructing the routes for each individual day.

In generating these problems, we set the travel time (in minutes) between two customers equal to the Euclidean distance. The service time was one minute and the maximum daily travel time for a vehicle was 500 minutes (this was based on discussions with Wong [82]). The demand at each customer was uniformly distributed on [0, 10] and the vehicle capacity was 500 units. In order to produce problems that would lead to routes similar to those found by a package delivery company on a typical day over $D = 5$ days of known service requirements, we generated customer locations randomly, according to a uniform distribution, in a quarter circle with a radius of 80 centered at the origin. With this distribution of customers and travel time and capacity restrictions specified above, there were 100 to 150 customers on a route, typical for a package delivery company.

First, we developed 35 homogeneous test problems. Here, we have only one type of customer and this customer is visited on each of the five days with a fixed probability p. We varied p from 0.6 to 0.9 in steps of 0.05 and generated five homogeneous test problems with 700 customers for each value of p. Second, we developed five heterogeneous test problems. Here, we envision two types of customers – commercial and residential. The commercial customers have a high probability of being visited each day and residential customers have a much lower probability of being visited each day. Consistent service by the same driver at the same time is not as important to residential customers (many are not present to receive a delivery) as it is to commercial customers. (We also point out that there is a very low probability that residential customers require service on more than one day.) We generated five heterogeneous test problems with 1,000 customer locations for each problem. Based on our discussions with Wong [82], each heterogeneous test problem has 70% commercial customers (there is a 0.9 chance that a commercial customer is visited each day) and 30% residential customers (there is a 0.1 chance that a residential customer is visited each day). For the five heterogeneous problems, there were an average of 659 commercial and residential customers who required service on a given day.

For both sets of test problems, we ran ConRTR to generate consistent routes and we calculated the mean maximum arrival time differential, the overall maximum arrival time differential, and the total travel time of the routes. Also, in order to get a sense for the increase in travel time resulting from the consistency constraints, we ran a generic record-to-record-travel (RTR) algorithm on each problem. This algorithm generally produces solutions that are within 1% to 2% of the best-known solutions on standard benchmark problems (see [51]). By comparing the total travel time of the ConRTR routes to the total travel time of the RTR routes, we get a sense of the increased cost added by the consistency constraints. In particular, we compare the total number of vehicles and the total travel time required in the different solutions. We point out that ConRTR and RTR each take roughly one to two minutes to generate the routes for all five days.

In Table 2.3, we show the results produced by ConRTR and RTR on the 35 homogeneous test problems. The results suggest that our precedence principle does a very good job preserving consistent arrival times for those customers requiring service on multiple days. For all 35 problems, taking into account the consistency requirements (denoted by consistent routes in Table 2.3), the average maximum arrival time differential for all 700 customers was between 4 minutes ($p = 0.9$, problem sets 2-5) and 11 minutes ($p = 0.6$, problem set 1) with an overall average of 7 minutes. In the worst case for one customer, the maximum arrival time differential was between 9 minutes ($p = 0.9$, problem set 3) and 50 minutes ($p = 0.65$, problem set 1) with an overall average of 22 minutes. In other words, using the routes generated by ConRTR, when a customer requires service on multiple days, the same driver visited a customer within 7 minutes (on average) of the same time each day. In the worst case for one customer, this time increased to 22 minutes (on average).

		ConRTR Results				RTR Results	
p	Problem Set	Average Maximum Differential	Overall Maximum Differential	Total Travel Time	Number of Routes	Total Travel Time	Average Number of Routes
0.6	1	11	42	6717	5	6312	4.8
	2	9	28	6691	5	6472	5.0
	3	8	25	6777	5	6458	5.0
	4	10	31	6764	5	6364	4.8
	5	10	28	7026	5	6523	4.8
0.65	1	10	50	7463	5	6576	5.0
	2	9	28	7397	5	6624	5.0
	3	8	22	6982	5	6655	5.0
	4	9	25	7349	5	6800	5.0
	5	7	30	7498	5	6683	5.0
0.7	1	7	18	7441	6	7383	5.4
	2	7	21	7900	6	7049	5.2
	3	9	24	7529	6	7188	5.4
	4	9	29	7714	6	7110	5.4
	5	7	20	7990	6	7171	5.2
0.75	1	7	19	7638	6	7160	5.6
	2	8	24	7995	6	7271	6.0
	3	7	30	7767	6	7519	6.0
	4	7	17	7558	6	7400	6.0
	5	7	33	7967	6	7430	6.0
0.8	1	5	14	8226	6	7701	6.0
	2	6	20	8391	6	7751	6.0
	3	6	15	7924	6	7691	6.0
	4	6	16	8225	6	7736	6.0
	5	6	16	8347	6	7613	6.0
0.85	1	6	16	8501	7	8046	6.4
	2	5	13	8695	7	8093	6.4
	3	7	23	8642	7	8125	6.6
	4	5	13	8557	7	8324	6.8
	5	6	21	8280	7	8114	6.4
0.9	1	6	14	9252	7	8316	7.0
	2	4	12	8652	7	8269	7.0
	3	4	9	8601	7	8414	7.0
	4	4	12	8776	7	8265	7.0
	5	4	12	8526	7	8342	7.0

Table 2.3: Results for ConRTR and RTR on homogeneous simulated problems

	ConRTR Results			RTR Results
	Average	Overall	Total	Total
	Maximum	Maximum	Travel	Travel
Problem	Differential	Differential	Time	Time
1	8	43	9581	8630
2	7	24	9079	8522
3	5	16	9069	8507
4	7	20	9584	8614
5	8	28	9464	8518

Table 2.4: Results for ConRTR and RTR on heterogeneous simulated problems

Over all 35 problems, the total travel time of the ConRTR-generated routes was slightly longer than the total travel time of the routes generated by RTR. On average, the total travel time of the consistent routes was 6.6% longer (the increased times were between 0.8% and 13.5%). Additionally, on several occasions RTR was able to produce solutions requiring one less vehicle. However, unless all of a particular driver's customers do not require service on a particular day, the consistent driver constraint implies that we must always have the same number of vehicles operating each day. As expected, we observe that, for the routes produced by ConRTR, the average arrival time differential decreases as p increases since the template routes account for more and more customers and are therefore able to do a better job of approximating each daily route.

In Table 2.4, we show the results produced by ConRTR and RTR on the five heterogeneous test problems. For all five problems, taking into account the consistency requirements, the average maximum arrival time differential for all commercial customers who require service on more than one day (recall that consistency is more important to the commercial customers than the residential customers) was between

5 minutes and 8 minutes with an overall average of 7 minutes. In the worst case for one customer, the maximum arrival time differential was between 16 minutes and 43 minutes with an overall average of 26 minutes. In other words, using the routes generated by ConRTR, when a customer requires service on multiple days, the same driver visited a customer within 7 minutes (on average) of the same time each day. In the worst case for one customer, this time increased to 26 minutes (on average).

Over all five problems, the total travel time of the ConRTR routes was slightly longer than the total travel time of the routes generated by RTR. On average, the total travel time of the consistent routes was 9.3% longer than the total travel time of the inconsistent routes (the increased times were between 6.5% and 11.3%). Finally, we note that for these five instances, ConRTR and RTR always generated solutions that contained seven routes.

2.4.3 Modified Benchmark Problems

There are a number of well-known benchmark problems for the classical VRP (see Christofides et al. [19, 20], Golden et al. [36], Li et al. [51]). Given an n-node VRP benchmark problem, some number of days D, and a service probability p, we developed a simple procedure to randomly generate a ConVRP benchmark from this existing problem. We took the VRP benchmark problems of Christofides et al. [19, 20] and constructed a single five day ConVRP benchmark from problems 1 to 12 using a daily service probability of $p = 0.7$ (these problems are available at `http://www.rhsmith.umd.edu/faculty/bgolden/vrp_data.htm`).

In Table 2.5, we present the solutions found by our algorithm as well as the best total route length found when consistency is not taken into account (Appendix A contains figures and details for all of these solutions). We also include the mean maximum arrival time differential as well as the overall largest maximum arrival time differential for the routes created by ConRTR.

Problem	Number of Customers	ConRTR Total Travel Time	ConRTR Number of Vehicles	Arrival Time Differentials Average	Arrival Time Differentials Maximum	RTR Total Travel Time	RTR Mean Number of Vehicles
1	50	2282.14	5	8.36	24.38	1963.39	3.4
2	75	3872.86	11	6.85	34.26	3182.31	7.8
3	100	3628.22	7	8.21	22.87	3127.77	5.8
4	150	4952.91	12	4.93	27.53	4121.73	8.6
5	199	6416.77	16	3.32	26.93	5108.19	12.2
6	50	4084.24	5	19.19	63.47	3954.32	4.6
7	75	7126.07	12	14.19	83.96	6325.39	8.6
8	100	7456.19	9	22.70	73.04	6902.10	6.8
9	50	11033.54	14	22.19	106.43	9932.90	10.4
10	199	13916.80	18	18.47	60.17	12399.40	13.2
11	120	4753.89	7	4.78	16.10	4244.48	5.2
12	100	3861.35	10	3.00	17.58	3209.88	6.6

Table 2.5: Results for ConRTR and RTR on 12 new benchmark problems

The solutions presented in Table 2.5 are different from those in Table 2.3 in several ways. First, it is difficult to assess the consistency of the arrival times as we have a much larger disparity in the total travel times and arrival time differentials from one problem to the next. For example, we have average maximum arrival time differentials ranging from 3.3 in problem 5 to more than 22 in problems 8 and 9. However, if we view the arrival time differential in proportion to the average route duration, then the average maximum arrival time differential is less than 10% in all 12 problems.

A second way that these solutions differ from those in Table 2.3 is that the solutions generated by RTR without regard for consistency generally require fewer vehicles and require roughly 15% less total travel time. Because the routes created by RTR are generally less costly than the ConRTR routes, we ran an experiment to transform the RTR solution into a solution that tries to satisfy the consistency constraints. To do so, we must first assign a single driver to each route on each of the D days so that as many customers as possible receive consistent service throughout the D days. Given this assignment of drivers to routes, we must then traverse each route in one of the two possible ways (i.e., clockwise or counterclockwise) in order to minimize the mean maximum arrival time differential for the customers that always receive service from the same driver.

The RTR-generated solutions sometimes require a different number of routes (drivers) on each day and so it is not clear how to analyze the consistency of these solutions. One option is to assign multiple drivers to individual vehicles on those days that require fewer routes. Another option is to have some drivers only work certain days. We chose to analyze the second option, reasoning that a firm would be unlikely to pay for more than a single driver per vehicle. Assuming that we have v total drivers on each day (those who do not work on a particular day are assigned a trivial depot-to-depot route), we then can find the optimal assignment of drivers to routes in the D-day RTR solution by solving a set partitioning problem with an additional constraint (we provide the details of this set partitioning problem in Appendix A).

After assigning drivers to routes, we next try to provide consistent service in terms of the arrival time differential. Each of the v vehicles traverses D routes, and

Problem	Number of Customers	Proportion of Customers Receiving Consistent Service	Arrival Time Differentials Average	Arrival Time Differentials Maximum
1	50	0.48	14.60	132.21
2	75	0.41	8.56	95.33
3	100	0.48	11.22	138.93
4	150	0.46	8.65	107.58
5	199	≤ 0.46	-	-
6	50	0.55	23.42	167.75
7	75	0.39	11.83	141.18
8	100	0.47	22.34	165.97
9	150	≤ 0.51	-	-
10	199	≤ 0.49	-	-
11	120	0.68	15.17	179.97
12	100	0.54	4.82	82.07

Table 2.6: Assigning drivers to routes in the optimal way

so there are 2^D different orientations of these routes. By examining all orientations, we are able to find the optimal orientation that minimizes the mean maximum arrival time differential over those customers that receive service from only one driver. We present the results of these computations in Table 2.6.

By assigning drivers to routes in this way, we are able to provide consistent service to between 39% and 68% of the customers in Table 2.6, with an average of 50% (we are only able to provide an upper bound for problems 5, 9, and 10 due to the size of the set partitioning problem.) In terms of the arrival time differentials, the mean maximum arrival time differential is more than 50% larger and the overall maximum arrival time differential is larger by a factor of more than 4.5. Thus, given efficient, inconsistent routes, this experiment suggests that it may not possible to transform these routes into consistent ones.

2.4.4 Real-World Problem

Finally, we tested ConRTR on a data set provided by UPS. This data set contained service requirements, demand amounts, and travel times for 3,715 customer locations over five weeks (five days per week). Although some of the customer locations had hard time windows, these were ignored because ConRTR does not account for this type of constraint.

We have two key interests in solving the real-world problem with ConRTR: (1) test the effectiveness of the precedence principle in handling actual service requirements and (2) examine the consistency of the arrival times at individual customers.

Week	Mean Number of Stops Per Day	Number of Customers With k Stops $k=1$	$k=2$	$k=3$	$k=4$	$k=5$	Number of Template Customers
1	597	838	213	100	60	132	505
2	591	801	215	98	58	133	504
3	566	755	216	84	52	135	487
4	573	807	219	96	44	123	482
5	572	818	201	94	43	130	468

Table 2.7: Properties of the five-week data set provided by UPS

Important properties of the data set are summarized in Table 2.7. The second column gives the mean number of stops per day and the remaining columns describe the distribution of customer stops per week. While most customers are visited on only one day ($k = 1$), there are a significant number of customers requiring service on every day of the week ($k = 5$). The right-most column gives the number of customers in the template (these customers require service on two or more days per week). A typical customer had a daily demand of three packages (some commercial customers required up to 90 packages) with a mean service time of three minutes. In general, the template contained roughly 85% as many customers as serviced on a typical day, implying that we will make more customer insertions than deletions to the template.

After reviewing the existing UPS routes, we set the maximum total travel time to 9.5 hours and the total vehicle capacity to 350 packages. We applied ConRTR to each of the five weeks and generated routes for each day in the week. As in previous experiments, we used RTR to generate routes that did not consider consistent service. In Table 2.8, we report the results of this experiment. For the consistent routes, the average maximum arrival time differential was between 19 minutes and 35 minutes with an overall average of 26 minutes. In the worst case for one customer, the maximum arrival time differential was between 64 minutes and 176 minutes with an overall average of 101 minutes. With the exception of the routes for week 4, ConRTR performed quite well with customers being visited within 25 minutes (on average) of the same time each day. In the worst case, this time increased to 83 minutes (on average). The consistency of the routes for the fourth week suffered due to a larger number of insertions on one day (causing late arrival times at some template customers) and a large number of deletions on another day (causing early arrival times for these same template customers).

The routes generated by RTR were slightly more efficient in terms of total travel time. However, the difference was only about 1%; this was quite different from the simulated problems where the difference was generally about 9%. A visual inspection of these routes indicates that there is significant clustering in the UPS data, and our

	ConRTR Results			RTR Results
Week	Average Maximum Differential	Overall Maximum Differential	Total Travel Time	Total Travel Time
1	19	64	6206	6107
2	29	101	6064	5998
3	24	81	5794	5755
4	35	176	5959	5910
5	25	85	5828	5777

Table 2.8: Results for ConRTR and RTR on the UPS data set

template-based approach seems to work well with clustered data as the insertion of customers into the template does not substantially increase the travel time.

In the final computational experiment with the UPS data, we tested ConRTR's performance in generating a template for a future planning horizon using historical data. We produced a set of template routes from the first four weeks (20 days) of service requirements and then used this template to generate routes for each day of the fifth week. We modified the criterion for including a customer in the template, now requiring that a customer need service on four or more days during this 20-day period. This criterion would allow us to provide consistent service to the most frequently visited customers by including them in the template routes.

The key point of this experiment is that the set of template routes was generated without ever considering the service requirements for the fifth week. Thus, we are trying to generate consistent routes for those customers requiring service in the fifth week by using only the historical service requirements from the first four weeks of data. This allows us to more accurately model the situation where a delivery company

Day	Routes derived from week 5 template	Routes derived from weeks 1 to 4 template	RTR solution
1	1190	1197	1183
2	1132	1136	1119
3	1147	1164	1136
4	1133	1137	1124
5	1226	1265	1214

Table 2.9: Generating consistent routes from historical data

wishes to provide consistent service over a longer horizon than the fixed D-day period that we have previously considered.

In Table 2.9, we provide the results of the final experiment. We compare the routes derived from the historical template to the routes produced from the template that used the actual data from the fifth week. The second column gives the total travel time if we generate consistent routes by creating a set of template routes from the service requirements for the fifth week. The third column gives the total travel time of the routes that were created by constructing a template from the first four weeks of data and then removing and inserting customers as determined by the service requirements for the fifth week. The final column gives the total travel time if consistency is disregarded and routes are generated for each day

The results of this experiment are encouraging in terms of the efficiency of the routes that we were able to produce. Using the template generated from the first four weeks of data, the routes are 1% longer (on average) in total travel time than the routes produced from the week five template. In terms of consistent arrival times, the maximum differential for customers requiring service on more than one day in week five was about 55 minutes. However, the overall maximum arrival time differential is much larger at just over three hours. This large time is due to the fact that there were several customers in the week five data set who required service on two or more days, but were not included in the template routes as they required service on less than four total days during the first four weeks. We expect that such large times would decrease as the number of weeks of historical data increased. Our results for the UPS data set indicate that the service requirements from one week to the next are quite similar. Thus, after choosing an appropriate criterion for including customers in the template, an effective template that leads to high-quality routes can be generated solely from historical data.

2.5. Conclusion

In today's highly competitive, small package shipping industry, companies are looking for ways to improve customer service. By providing consistent service, small package shipping companies can improve their relationships with customers by establishing a personal connection in the form of the same driver delivering packages at nearly the same time each day.

In this chapter, we developed a method for generating consistent delivery routes. The routes produced by ConRTR were very successful in achieving customer service objectives with a low total travel time. ConRTR performed well on randomly generated problems by producing routes on which drivers visited a customer within seven minutes (on average) of the same time each day. These consistent routes were only slightly longer in total travel time (less than 10% on average) than the corresponding inconsistent routes.

Furthermore, ConRTR performed well on four of five weeks of actual service requirements provided by UPS with customers being visited within 25 minutes (on average) of the same time each day. The total travel time of the consistent routes was only about 1% greater than the total travel time of the inconsistent routes generated without regard for the additional customer service constraints. Finally, we demonstrated that ConRTR generated high-quality, consistent routes by using purely historical data.

We presented our results to UPS managers in late 2006 and they remarked [59]: “We found their results very interesting and their approach novel and easy to implement. Their research also provides independent confirmation that our service constraints make good business sense.”

Chapter 3

The Balanced Billing Cycle Vehicle Routing Problem

Utility companies typically send their meter readers out each day of the billing cycle in order to determine each customer's usage for the period. Customer churn requires the utility company to periodically remove some customer locations from its meter-reading routes. On the other hand, the addition of new customers and locations requires the utility company to add new stops to the existing routes. A utility that does not adjust its meter-reading routes over time can find itself with inefficient routes and, subsequently, higher meter-reading costs. Furthermore, the utility can end up with certain billing days that require substantially larger meter-reading resources than others. However, remedying this problem is not as simple as it may initially seem. Certain regulatory and customer service considerations can prevent the utility from shifting a customer's billing day by more than a few days in either direction. Thus, the problem of reducing the meter-reading costs and balancing the workload can become quite difficult. We describe this *Balanced Billing Cycle Vehicle Routing Problem* in more detail and develop an algorithm for providing solutions to a slightly simplified version of the problem. Our algorithm uses a combination of heuristics and integer programming via a three-stage algorithm. We discuss the performance of our procedure on a real-world data set.

3.1. Introduction

A typical utility company issues bills to its customers on a monthly basis. However, because each customer's usage varies from one month to the next, the utility company must accurately determine the amount of each bill by physically inspecting the customer's meter. While some utility companies are using technological advances such as RFID to simplify this task [40], many others still send out a fleet of workers each day to individually read each customer's meter. These meter readers begin their day at a central location, visit a set of specified locations, and then return to this central facility.

Because of the periodic and largely predictable nature of this activity, we expect that most utility companies are very efficient in accomplishing this monthly meter-reading task. However, because the utility's customer set changes as accounts are closed and new accounts are added, there is the potential for the quality of these meter-reading routes to deteriorate over time if the routes and billing day assignments are not closely monitored.

As an example, consider a fixed customer location and assume that the meter belonging to this customer is read on the 10th day of each month. Since the utility company clearly wishes to minimize the total cost of the meter-reading process, the company will attempt to minimize the total driving time required to read all the necessary meters. Thus, we expect that many meters in the immediate vicinity of this particular location will also be read on the 10th. After a period of time, suppose that the current resident moves away and cancels service with the utility company. A new resident moves in at a later date and begins service, say on the 20th day of the month. We expect that this new resident's meter will be read on the 10th day of each month instead of the 20th as it is logical to believe that the meter reader servicing this particular area will simply add this new customer to the existing route. However, in practice, this is not always the case as the new resident's meter is sometimes read on the 20th of the month, rather than on the 10th day.

Over time, as specific meters are transferred among different customers and accounts are canceled and renewed, the utility company's meter-reading routes become less efficient as new customers are added to old, inappropriate routes. Thus, utility companies that fail to adjust their routes and billing day assignments in response to this customer turnover end up with inefficient, fractured routes, as well as meter-reading workloads that are not balanced across the billing cycle.

We expect that once a utility realizes that its routes are inefficient and are increasing its operating costs, it would simply adjust certain customers' monthly meter-reading dates in order to quickly cut costs by shifting customers to more appropriate routes. However, certain regulatory constraints forbid a utility company from shifting a given customer's billing date by more than a few days [12]. In addition, some utilities wish to avoid significant changes in billing dates as many customers resent the potentially large increases in their bills. Thus, even if the utility company knew that it could immediately reduce routing costs by shifting a customer's meter-reading day from the 20th to the 10th day of the month, there are often external considerations that prevent it from making such a change in a single billing period. After months and even years of this route fracturing phenomenon, a utility can find itself with terribly inefficient and unbalanced routes and faced with the daunting task of creating efficient and balanced routes. Some utilities resort to manual methods and attempt to repair such configurations by hand. Other approaches include a technique called string indexing that attempts to balance routes in terms of the number of customers. A single, giant tour is constructed and then partitioned into routes containing roughly the same number of customers by simply splitting the giant tour in certain places [46]. Clearly, these methods are ad hoc at best, and this chapter attempts to provide an effective and automated procedure for repairing such fractured configurations.

We now consider the goals of the utility. First and foremost, the utility would like to minimize its meter-reading costs by creating more efficient routes. By improving the routes for each billing day, the utility can reduce its operating costs by decreasing the total mileage traveled by individual meter readers. Additionally, it is possible that the utility can reduce its labor costs since the more efficient routes may also require fewer overall meter readers.

The utility company has two secondary goals related to balancing the workload across the different days of the billing cycle. First, the utility company wants to ensure that the number of required meter readers is constant across the billing cycle. In other words, the utility would like to develop meter-reading routes and billing day assignments that do not require extra meter readers or increased (overtime) labor costs on certain days of the billing cycle. Second, the utility wants to ensure that the actual number of customers requiring meter-reading service on any given day of the billing cycle is roughly constant. In certain cases, the billing data and other customer information are stored on older mainframe computer systems. When the meter readers download each day's billing data onto their personal hand-held devices and then upload the new readings to the main database, a particularly large number of customers on any given day can slow down the system to the point where the next day's meter reading is delayed [46].

In this chapter, we consider the following problem. We are given a set of existing meter-reading routes for each day of the billing cycle. We create meter-reading routes for each billing day and assign customers to these routes with the goal of reducing the total length of the meter-reading routes while observing regulatory and customer service considerations when assigning customers to new billing days. We have the secondary goal of balancing the meter readers' routes in terms of both total route length and the number of meters read per day.

The remainder of this chapter is organized as follows. In Section 3.2, we provide a mathematical formulation of this problem. In Section 3.3, we develop a method for obtaining solutions to this problem. In Section 3.4, we present the results of our solution algorithm on a real-world data set. We summarize our progress and presents ideas for future research in Section 3.5.

3.2. Problem Statement

The Vehicle Routing Problem (VRP) has been studied by the operations research community for nearly 50 years. The addition of real-world constraints to the problem has led to many variants including the VRP with Time Windows, the Inventory Routing Problem, and the Period Routing Problem [10, 11, 13, 18]. While our current problem shares some characteristics with well-known VRP variants, the Balanced Billing Cycle VRP (BBCVRP) has several aspects that are quite different.

In many VRP variants, the problem is solved from scratch. However, in the BBCVRP, we start with an existing set of unbalanced and inefficient routes and then move towards a more efficient solution that satisfies the routing and billing day constraints.

A second aspect of the BBCVRP that is largely absent from problems in the VRP literature is the notion of balancing the routes. Any company that employs multiple drivers to read meters or deliver goods would like each driver to individually work near capacity on each day, implying some degree of balance across the work days. Despite the simplicity and necessity of this objective in a wide variety of scenarios, it appears that only a handful of papers have considered the issue of balance in any detail.

Renaud, Boctor, and Laporte [71] use the notion of balance when constructing 2-petals as they develop their routes. The notion of balance is also addressed by the heuristic proposed in [6], but the problem they consider involves fewer than 30 nodes. More recently, balancing route length has been addressed in a series of papers by Jozefowiez, Semet, and Talbi [41, 42, 43]. They treat the problem as a multi-objective optimization problem and formally propose the Vehicle Routing Problem with Route Balancing (VRPRB). In this problem, the goal is to minimize the total route length as well as the difference between the longest route and shortest route. They solve this problem with a genetic algorithm and a local search method they refer to as Target Aiming Pareto Search. They report the performance of their procedure on some of the classical benchmark problems given in [19] and [20].

We now discuss two different approaches to solving the BBCVRP. An iterative approach takes the existing configuration and tries to create the routes for the following billing period without considering future periods. After successively modifying several billing periods in this fashion, trying to balance the routes in the length and customer dimensions, this procedure ends when no further improvements can be made, thereby reaching a local minimum. A targeted approach has a longer term perspective and begins by constructing an efficient and balanced routing and billing day configuration while ignoring the initial routes altogether. This approach transitions to this final target configuration via a series of intermediate billing periods.

We adopted the targeted approach for two reasons. First, we can be certain that we will eventually reach a desirable configuration after a number of intermediate periods. Second, by ignoring the initial configuration altogether, we can use powerful VRP heuristics when creating the final target configuration from scratch, rather than relying on ad hoc methods to improve an existing set of routes. We note that the targeted approach does not explicitly account for the potential addition and removal of customer locations that may occur during the intermediate transition periods. However, because the number of such locations is likely to be small in any given month, it should be possible to accommodate these changes by slightly modifying the intermediate routes. We envision that our targeted procedure would be applied periodically to correct a severely fractured routing and billing day configuration.

Notation	Definition
N	The number of meters that must be read during each billing cycle
$(\underline{C}, \overline{C})$	The minimum and maximum number of customers that an individual meter reader can visit in a given day
$(\underline{L}, \overline{L})$	The minimum and maximum allowed route length for an individual meter reader
D	The number of days in the billing cycle
S	The maximum number of days that we are allowed to shift a customer's billing day in either direction
M	The number of meter readers used daily by the utility in its existing routes
$L_t(r, d)$	The length of route r on day d in billing cycle t
$C_t(r, d)$	The number of customers on route r on day d in billing cycle t
$d_t(i)$	The billing day of customer i during billing cycle t
$f(i)$	The final billing day of customer i

Table 3.1: Notation used to describe the BBCVRP

Using the notation presented in Table 3.1, we now state our objectives more precisely. In the targeted approach, we require a number of intermediate periods to reach the final configuration. The primary objective is to minimize the cost of the routes in the final configuration, and a secondary objective is to minimize the number of intermediate periods so that the transition to the new routes and billing day assignments is as short as possible. We let $K - 1$ denote the number of required intermediate periods and $L_K(r, d)$ denote the length of route r on billing day d in the final set of target routes. We wish to minimize the value of K as well as the quantity

$$\sum_{r=1}^{M} \sum_{j=1}^{D} L_K(r, j). \tag{3.1}$$

We now turn to the constraints. For all billing periods $1 \leq t \leq K$, all routes r, and days $d \in \{1, 2, \ldots, D\}$, we must satisfy the individual meter-reading length and capacity constraints

$$L_t(r,d) \le \overline{L} \tag{3.2}$$

$$C_t(r,d) \le \overline{C}. \tag{3.3}$$

We also attempt to satisfy the balance constraints imposed by the lower bounds

$$L_t(r,d) \ge \underline{L} \tag{3.4}$$

$$C_t(r,d) \ge \underline{C}. \tag{3.5}$$

In order to address the constraint that we cannot shift any customer's billing day by more than $\pm S$ days from one month to the next, we introduce the notion of billing distance. Given two billing days $1 \le u, v \le D$, we define $||u,v||_D$ as

$$||u,v||_D = ||v,u||_D = \min(u - v \bmod D, v - u \bmod D). \tag{3.6}$$

In other words, $||u,v||_D$ is simply the minimum number of billing days separating u and v in a D-day cycle if we allow for wraparounds. Thus, when we assign some customer i to a new billing day in period $t+1$, we must ensure that

$$||d_t(i), d_{t+1}(i)||_D \le S. \tag{3.7}$$

3.3. Solution Algorithm

As mentioned earlier, our approach to solving the BBCVRP is targeted. We construct an idealized set of target routes and then transition to this configuration through a series of intermediate routes. Before describing our three-phase procedure, we discuss our assumptions.

3.3.1 Assumptions

First, we assume that only a single meter reader operates on each day of the billing cycle. Although this clearly makes the problem simpler as we are now required to create only a single meter-reading route for each day of the billing cycle, we argue that this assumption is not very restrictive. Given existing routes for M meter readers, we can simply run our procedure M separate times, thereby producing a set of balanced target routes for each meter reader. Since the original customer sets for each meter reader are assumed to be disjoint, we produce a set of M routes per day that are balanced and efficient. If the initial routes are highly fractured and imbalanced so that this approach is not easy to execute, we could still proceed under the assumption of having a single (very fast) meter reader and then partition each route in the resulting solution into M separate routes.

Second, we treat the meter-reading problem as a node routing problem rather than an arc routing problem where a street network is traversed. While this assumption allows us to use many of the standard solution techniques developed for the VRP, many parts of our procedure would apply to an arc routing setting as well.

Third, we assume that the utility is willing to employ additional meter readers during the transition periods, if needed. Because we are able to present a final, balanced set of routes that will be reached after a bounded number of intermediate periods, this assumption is reasonable in that these additional resources are required for only a few months.

3.3.2 Phase 1

In Phase 1 of the algorithm, we completely ignore the existing routing configuration and create a set of target routes. Since there is only a single meter reader, we create a single meter-reading route for each day. We generate the routes by applying a modified version of the VRP record-to-record travel algorithm (see [16] and [51]). We chose this metaheuristic for its simplicity, its ability to handle different types of constraints, and its ability to quickly generate high-quality solutions for large VRPs as demonstrated in [51]. We modified the general VRP algorithm given in [51] by forbidding certain moves and including penalties and rewards for moves that hurt or improve the balance of the routes, and develop our own variant of their record-to-record travel algorithm, given in Algorithm 3.1.

We begin by generating a set of exactly D routes that obey the limits imposed by (3.2) and (3.3). Algorithm 3.1 does not explicitly attempt to reduce the number of routes. However, the algorithm's performance on benchmark problems indicates that it typically finds solutions with a minimum number of routes [51]. We rely on this property of the record-to-record travel procedure in Algorithm 3.2 where we

Algorithm 3.1 Record-to-Record Travel for the VRP

```
1: Input: An existing feasible solution to an $N$-node VRP
2: Output: An improved feasible solution to the VRP
3: Set $S = 30$, $T = 5$, $t = 0$, set $R$ equal to the current objective function value, and
   $D = (1.01) \times Record$
4: $Operators = \{One\ Point\ Move,\ Two\ Point\ Move,\ Two\text{-}Opt\ Move\}$
5: for $i = 1$ to $S$ do
6:     for $j = 1$ to 3 do
7:         for $k = 1$ to $N$ do
8:             Apply Operators[j] and search node number $k$'s neighbor list for moves
   that obey the length and capacity constraints, accepting the first improving move
   if one is found. If no improving moves are found, then accept the best deteriorating
   move if the resulting objective function is less than $D$
9:         end for
10:    end for
11:    if The current solution is a new record then
12:        Update Record and Deviation
13:    end if
14: end for
15: while Improving moves can be found do
16:    for $j = 1$ to 3 do
17:        for $k = 1$ to $N$ do
18:            Apply Operators[j] and search node number $k$'s neighbor list for im-
   proving moves that obey the length and capacity constraints, accepting the first
   improving move that is found.
19:        end for
20:    end for
21: end while
22: if The current solution is a new record then
23:    Update Record and Deviation
24:    Set $t = 0$
25: end if
26: $t = t + 1$
27: if $t = T$ then
28:    Return the best solution found
29: else
30:    Return to Start
31: end if
```

attempt to find a solution with exactly D routes. Additionally, we note that since we are typically provided with an initial (unbalanced and potentially inefficient) solution with D routes, we expect to encounter little difficulty in finding a solution with the desired number of routes.

Algorithm 3.2 Generate Initial Routes

```
1: Input: A set of $N$ customer locations
2: Output: A set of $D$ routes that visit all $N$ locations and obey the length and
   capacity upper bounds $\overline{L}$ and $\overline{C}$.
3: Construct an initial solution $\mathcal{S}$ using the Clarke-Wright algorithm [21], stopping
   if we reach a solution with $D$ routes.
4: while The number of routes in $\mathcal{S}$ is greater than $D$ do
5:     Apply Algorithm 3.1
6: end while
```

At the conclusion of Algorithm 3.2, we have a set of exactly D routes, one for each day of the billing cycle. Each of these routes is feasible in terms of the maximum route length and the maximum number of customers, but typically is not feasible in terms of the minimums imposed by constraints (3.4) and (3.5). The next step is to try to modify the existing routes in order to meet these balancing constraints.

We again utilize the general record-to-record travel framework, except that we now guide the search by forbidding certain moves and perturbing the evaluation of other types of moves. We focus solely on those moves that involve the exchange of nodes between two separate routes.

First, when we consider the acceptance of any inter-route move, we check to see if constraints (3.4) and (3.5) are satisfied by the current solution. If not, then we reject any move that decreases either the current minimum route length or the current minimum number of customers on a route. Second, as long as constraints (3.4) and (3.5) are violated by some route, we perturb the evaluation of certain moves as follows. Suppose we have some inter-route move m that involves route r. Let $\delta(m)$ represent the change in the objective function value if the move were made. If route r is currently the shortest route in the solution and the length of route r increases after making the move m, then we would like to reward m. Similarly, if route r contains the fewest number of customers and the number of customers in route r increases if we make the move m, then we would also like to reward this move. In either case, we set $\delta(m) = \delta(m) - \beta \times |\delta(m)|$ where $\beta \in (0, 1)$ is a fixed balance parameter. This modification has the effect of making m appear more attractive when compared to other potential moves.

On the other hand, if our guided record-to-record travel search discovers a solution that does not violate constraint (3.4) or (3.5), then we refrain from rewarding any more moves but we forbid any moves that lead to violations of these constraints. Computational experiments (discussed in Section 3.4) have shown that this procedure is typically able to find solutions that satisfy constraints (3.4) and (3.5) with little sacrifice in quality.

3.3.3 Phase 2

At the end of Phase 1, we have D different meter-reading routes, one for each day of the billing cycle. Each of the customers on a route had an original billing day. The goal of Phase 2 is to assign a single billing day to each of our target routes so that the transition from our initial routes and billing day configuration will be as quick and efficient as possible.

A simple way of assigning billing days to our target routes would be to use the modal value. For each route, we find the most common original billing day among all customers in this route, and then assign this billing day to each customer in this route. However, there are two problems with this approach. First, we could encounter two different routes with the same modal billing day. Second, if we choose this modal billing day for all the customers in this target route, there could be some customers that can only reach this billing day following many billing day shifts due to the shifting constraint (3.7). In other words, this strategy does not account for those customers which are far away from the modal value in terms of the billing day distance.

In order to address these problems, we formulate an assignment problem with an appropriate cost function to determine how to allocate our D possible billing days among the D different routes. We define b_{ij} to be the cost of assigning billing day j to customer i. This cost is calculated in terms of the billing distance:

$$b_{ij} = \begin{cases} 0, & \text{if } ||d_0(i), j||_D \leq S \\ ||d_0(i), j||_D, & \text{otherwise.} \end{cases}$$

If customer i can be shifted from the original billing day to day j in a single shift of size less than or equal to S, then we incur no cost. If the shift is greater than S, then we simply use the billing distance. We let C_{Rj} denote the cost of assigning billing day j to all the customers in target route R and set this quantity to be the sum of these billing shift costs for each customer in the route, that is, $C_{Rj} = \sum_{i \in R} b_{ij}$. Letting $x_{Rj} = 1$ if day j is assigned to route R and $x_{Rj} = 0$ otherwise, we have the following assignment problem.

$$
\begin{aligned}
\text{Minimize} \quad & \sum_{R=1}^{D} \sum_{j=1}^{D} C_{Rj} x_{Rj} \\
\text{s.t.} \quad & \sum_{R=1}^{D} x_{Rj} = 1 \text{ for } j = 1, 2, \ldots, D \\
& \sum_{j=1}^{D} x_{Rj} = 1 \text{ for } R = 1, 2, \ldots, D \\
& x_{Rj} \in \{0, 1\} \text{ for } R = 1, 2, \ldots, D \text{ and } j = 1, 2, \ldots, D.
\end{aligned}
\tag{3.8}
$$

In Table 3.2, we show how the solutions to the assignment problem change as we increase S in an illustrative example with a 10-day billing cycle. We give the original billing day mixtures, modal value, and assigned final billing days for a subset of the 10 days.

	(Original billing day, Number of customers)									
	(1,43)	(1,13)	(3,31)	(1,23)	(1,1)	(1,10)	(1,6)	(1,2)	(1,1)	(1,1)
Original	(2,62)	(3,19)	(9,13)	(2,38)	(6,62)	(4,27)	(5,36)	(5,64)	(3,70)	(4,36)
Billing	(9,6)	(4,37)	(10,51)	(6,1)	(8,29)	(7,39)	(6,16)	(6,1)	(9,12)	(7,41)
Day		(9,29)		(8,54)	(9,2)	(9,20)	(8,37)	(9,9)		(9,24)
Mixture				(9,1)			(9,4)	(10,20)		
							(10,9)			
	Final billing day assignment									
Mode	2	4	10	8	6	7	8	5	3	7
$S = 1$	1	3	10	9	7	8	6	4	2	5
$S = 2$	3	2	1	10	8	9	6	7	4	5
$S = 3$	4	2	10	9	5	7	8	3	1	6
$S = 4$	1	10	6	4	7	5	9	3	2	8

Table 3.2: Change in final billing day assignments as the value of S is varied

In Table 3.2, the modal value is not unique as day 7 is assigned to the two target routes corresponding to columns 7 and 11, and day 8 is assigned to the two target routes corresponding to columns 5 and 8. As an example of the effect of the shift size on the solution to the assignment problem, consider the fourth column that represents a route where 31 customers have original billing day 3, 13 customers have original billing day 9, and the remaining 51 customers have original billing day 10. For $S = 2, 3$, and 4, all customers are able to transition to the assigned final billing day in a single move. However, the actual final billing day itself varies as larger shifts are allowed. For $S = 3$, we see that these customers are assigned the final day of 10 and receive the billing day 6 for $S = 4$. Billing day 10 is given to the customers in the third column when $S = 4$, as this assignment allows all of these customers to transfer to their final day in a single shift.

After solving the assignment problem for a specific value of S, we have a set of D different target routes that have now been assigned a unique billing day. In other words, for each customer i, we now have a final billing day $f(i)$. Next, we construct a sequence of intermediate transition routes that allow us to move customers from their original routes and billing days to their final routes and billing days.

3.3.4 Phase 3

In Phase 3, we start with the initial routes at time $t = 0$ and iteratively construct the routes for billing period $t + 1$ in terms of the period t routes until all customers are transitioned to their final billing days. We first find all customers that can be transitioned from their billing day at time t to their final billing day in a single shift that does not violate (3.7). In terms of billing distance, we find all customers i such that $||d_t(i), f(i)||_D \leq S$. Letting $\mathcal{A}$ denote this set of assigned customers and $\mathcal{U}$ represent the complementary set of unassigned customers, we create routes for the next period by first constructing a set of skeleton routes. We do this by taking each target route and restricting it to only those customers that are in $\mathcal{A}$. Having constructed these skeleton routes, we take the unassigned customers in $\mathcal{U}$ and place them in an appropriate intermediate route while trying to ensure that these routes are as efficient and balanced as possible.

For the remaining unassigned customers in $\mathcal{U}$, we assign them to the existing skeleton routes by solving a sequence of mixed integer programs (MIPs). Given customer $i \in \mathcal{U}$ and skeleton route j, we let $x_{ij} = 1$ when customer i is inserted into route j and $x_{ij} = 0$ otherwise. We let c_{ij} represent the cheapest cost of feasibly inserting customer i into route j. One approach is to try to minimize the sum of all these insertion costs while attempting to maintain feasible and balanced routes. Letting L_j and C_j represent the length and number of customers on skeleton route j, we have the following MIP denoted by *INT-ROUTES*:

$$\text{Minimize} \quad \sum_{i \in \mathcal{U}} \sum_{j=1}^{R} c_{ij} x_{ij} \tag{3.9}$$

$$\text{s.t.} \quad \sum_{j=1}^{R} x_{ij} = 1 \text{ for } i \in \mathcal{U} \tag{3.10}$$

$$C_j + \sum_{i \in \mathcal{U}} x_{ij} \leq \overline{C} \text{ for all routes } j \tag{3.11}$$

$$C_j + \sum_{i \in \mathcal{U}} x_{ij} \geq \underline{C} \text{ for all routes } j \tag{3.12}$$

$$L_j + \sum_{i \in \mathcal{U}} c_{ij} x_{ij} \leq \overline{L} \text{ for all routes } j \tag{3.13}$$

$$L_j + \sum_{i \in \mathcal{U}} c_{ij} x_{ij} \geq \underline{L} \text{ for all routes } j \tag{3.14}$$

$$x_{ij} = 0, \text{ if } ||d_t(i), j||_D > S \tag{3.15}$$

$$x_{ij} = 0, \text{ if } ||d_t(i), f(i)||_D < ||j, f(i)||_D \tag{3.16}$$

$$x_{ij} \in \{0, 1\}. \tag{3.17}$$

In (3.9), we minimize the sum of the individual insertion costs. Constraint (3.10) ensures that each customer in $\mathcal{U}$ is assigned to a route, (3.11) and (3.12) require that we maintain balance across the routes in terms of the number of customers, and (3.13) and (3.14) attempt to ensure that each route remains feasible and balanced in terms of total length. Finally, constraint (3.15) restricts the choices of customer i's new billing day to only those days that can be reached by a feasible shift. Constraint (3.16) ensures that whenever we shift an unassigned customer's billing day, we are always moving this customer's billing day closer to the final billing day. We incorporate this constraint into the formulation in an attempt to minimize the number of required transition periods.

However, given skeleton route j with length L_j, the value $L_j + \sum_i c_{ij}x_{ij}$ in (3.13) and (3.14) can sometimes overestimate this route's length following the insertions that we made. For example, if two customers that are very near one another are inserted into a particular route, then the sum of these minimum insertion costs can be nearly double the actual increase in the route's length. We address this issue in two ways. First, rather than solving *INT-ROUTES* for all $|\mathcal{U}|$ customers at once, we solve a related formulation several times, each time using only a portion or batch of the customers in $\mathcal{U}$. Second, we initially solve a relaxation of the problem and then gradually tighten the constraint until no feasible solution can be found.

In order to determine which subset of customers from $\mathcal{U}$ to insert, we define a batch size B. We then select B customers at a time from $\mathcal{U}$ by considering their insertion costs. In particular, for each customer in $\mathcal{U}$, there is a particular subset of skeleton routes into which we can insert this customer without violating constraints (3.15) and (3.16). We compute the cheapest insertion costs for all these possibilities, find the largest insertion cost for each customer, and then sort the list of customers in $\mathcal{U}$ in terms of this largest insertion cost. We then solve a related integer program for the first B customers in this list. Our reasoning is that the customers with the largest insertion costs are potentially the most troublesome for our procedure and we would like to insert them early on while there is still some slack in the skeleton routes. After removing B customers from the set $\mathcal{U}$, we apply intra-route improvements to the individual routes that were affected by the insertion of these particular customers.

Then we recalculate the insertion costs for the remaining customers in $\mathcal{U}$ and sort the customers as before in order to select the next set of B customers.

The second part of our strategy in solving the MIP involves the right-hand side of constraint (3.13). Because of the potential error involved in the left-hand side of this constraint, we replace the route length maximum $\overline{L}$ with a larger value $\tilde{L}$. This change leads to the following alternative formulation of *INT-ROUTES*, denoted by *BATCH-ROUTES*:

$$\text{Minimize} \quad \sum_{i=1}^{B}\sum_{j=1}^{R} c_{ij}x_{ij} \tag{3.18}$$

$$\text{s.t.} \quad \sum_{j=1}^{R} x_{ij} = 1 \text{ for } i = 1, 2, \ldots, B \tag{3.19}$$

$$C_j + \sum_{i=1}^{B} x_{ij} \leq \overline{C} \tag{3.20}$$

$$L_j + \sum_{i=1}^{B} c_{ij}x_{ij} \leq \tilde{L} \text{ for all routes } j \tag{3.21}$$

$$x_{ij} = 0, \text{ if } ||d_t(i), j||_D > S \tag{3.22}$$

$$x_{ij} = 0, \text{ if } ||d_t(i), f(i)||_D < ||j, f(i)||_D \tag{3.23}$$

$$x_{ij} \in \{0, 1\}. \tag{3.24}$$

In *BATCH-ROUTES*, we reduce the number of customers under consideration, remove constraints (3.12) and (3.14), and relax (3.13). We embed this alternative formulation into Algorithm 3.3. Notice that we begin with some $\tilde{L} > \overline{L}$ for which a feasible solution can be found, and then gradually decrease this value until either no solution can be found, or $\tilde{L} \leq \overline{L}$.

Algorithm 3.3 Generate Intermediate Routes

```
1: Input: A set of skeleton routes and a set $\mathcal{U}$ of unassigned customers at time $t$
2: Output: A set of intermediate routes for time $t+1$
3: Set $B = 20$, $\alpha = .05$
4: while $|\mathcal{U}| > 0$ do
5:     for all $i \in \mathcal{U}$ do
6:         for $j = 1..D$ do
7:             Calculate $c_{ij}$, the minimum cost of inserting customer $i$ into route $j$, for all routes $j$ for which (3.22) and (3.23) are feasible
8:         end for
9:     end for
10:    Sort the list of customers in $\mathcal{U}$ in terms of the largest $c_{ij}$
11:    Select the first $\min(B, |\mathcal{U}|)$ customers from the list
12:    Set $\tilde{L} = 2 \times \overline{L}$
13:    while BATCH-ROUTES is infeasible do
14:        Set $\tilde{L} = \tilde{L} + \alpha \times \tilde{L}$
15:    end while
16:    while BATCH-ROUTES is feasible and $\tilde{L} > L$ do
17:        Set $\tilde{L} = \tilde{L} - \alpha \times \tilde{L}$
18:    end while
19:    Make the insertions determined by the $x_{ij}$ variables from the last feasible solution to BATCH-ROUTES
20:    Apply intra-route improvement operations to the resulting routes
21:    Update $\mathcal{U}$ by removing the just inserted $\min(B, |\mathcal{U}|)$ customers from the list
22: end while
```

Given the routes and billing period assignments at time t, we are able to use Algorithm 3.3 to generate new routes and billing period assignments for time $t + 1$. Since constraint (3.23) requires that a customer is always moving closer to the final billing day, this procedure will terminate after a finite number of iterations. We begin with the original routing and billing day configuration of time $t = 0$ and repeatedly apply Phase 3 of our algorithm in order to generate a sequence of intermediate billing periods and routes before finally arriving at the target configuration.

3.3.5 Additional Considerations

For a problem that is tightly constrained in terms of maximum length $\overline{L}$ and maximum number of customers $\overline{C}$, the intermediate routes of Phase 3 may be infeasible. This is simply due to the fact that these intermediate routes must be stretched in order to include customers which are temporarily visited on the corresponding billing day. Since our procedure guarantees that each customer's billing day moves closer by at least one day during successive billing periods, the maximum number of intermediate periods is bounded by $\lfloor D/2 \rfloor - 1$. Furthermore, because of the temporary nature of these intermediate routes as well as the promise of efficient balanced routes in the near future, it is reasonable for a utility to use additional meter-reading resources during the intermediate periods, especially since the exact number of required intermediate periods is completely determined upon the completion of Phase 3.

Finally, we note that after the intermediate routes are generated via Phase 3, the utility company can transition to the target routes at its own pace. If the utility is concerned about shifting a customer's billing day for several successive months, the utility can choose to slow down this process by using the same set of intermediate routes for two or more periods in order to prevent these successive shifts. In practice, utility companies are typically hesitant to shift customers' billing days several times in succession. Stretching out the transition period is a response to this hesitancy [46].

3.4. Computational Results

In this section, we conduct computational experiments with our procedure. We coded our algorithm in C/C++ and used the GMPL modeling language in conjunction with CPLEX 10.2 to handle the MIPs of Phase 3. When run on an existing set of routes containing between $3,000$ and $4,000$ customers, the entire procedure required roughly one hour of computing time on a 2GHz Intel processor. The MIPs themselves presented little difficulty and the construction of the balanced target routes generally required about half of the total computing time.

We had the opportunity to test the performance of our procedure on a real-world data set provided by Routesmart Technologies, Inc. Confidentiality agreements prevent us from describing all of the specifics of the data set, but we are able to offer the following details. The $17,775$ customer locations are shown in Figure 3.1 where the depot is represented by the large circle in the bottom center. These meter locations are serviced by a total of 13 meter readers over the standard 20 day (monthly) billing period. Four of these meter readers service the region on all 20 days of the billing cycle and nine readers visit customers anywhere from only one day per cycle to 10 days per cycle.

We focused on the four full-time meter readers, and ran our procedure on each of the four sets of 20 routes. The four data sets contain $2,894$, $3,011$, $3,942$, and $3,972$ customers, accounting for $13,819$ of the $17,775$ total customer locations. For each meter reader, we were provided with the customer latitude and longitude coordinates along with the existing billing day assignments. Because we treat the problem from a

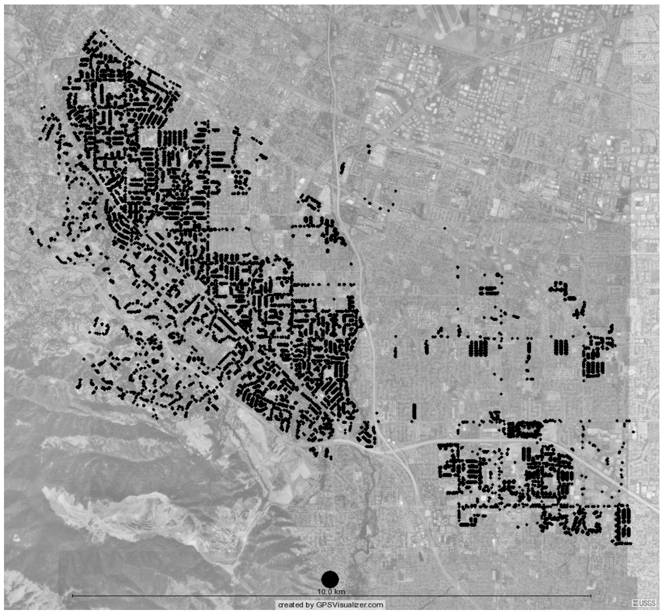


Figure 3.1: Original customer set

node-routing perspective rather than an arc-routing perspective, we did not use the existing routes that traversed a street network. Instead, we used the TSPLIB GEO [70] norm to calculate a symmetric distance matrix (in km) by using the customer coordinates. We then used these distances to create daily routes for each individual meter reader by computing a TSP tour traversing all customers on each day of the billing cycle. We used our record-to-record travel algorithm to generate these tours. In Table 3.3, we summarize important properties of the existing routes. It is clear that the existing routes are very imbalanced in terms of both the number of customers serviced each day as well as the total route length. For example, on one billing day, meter reader 1 visits 22 customers and travels 35 km. On another day of the cycle, this same meter reader visits 359 customers and travels 382 km (in this case, many of these locations are very near one another, possibly in an apartment complex where the meters are in a single location).

Meter Reader	Number of Total Stops	(Min, Max) Number of Customers	Total Length of all Routes	(Min,Max) Route Lengths
1	2894	(22,359)	3214	(35,382)
2	3011	(78,306)	3496	(94,324)
3	3942	(69,355)	4522	(89,381)
4	3792	(12,342)	4157	(31,366)

Table 3.3: Details of the existing routes

Next, we determined appropriate upper and lower bounds on the route length and number of customers per route. The maximum number of customers per route $\overline{C}$ is based on the existing routes from the data set. Determining the route length maximum $\overline{L}$ was more difficult. In order to generate solutions containing exactly 20 routes, we set $(\overline{L}, \overline{C}) = (170, 160)$ for meter readers 1 and 2, and $(\overline{L}, \overline{C}) = (225, 205)$ for readers 3 and 4. It is clear from Table 3.3 that these limits are substantially lower than the current maximum values. The balancing lower bounds $\underline{C}$ and $\underline{L}$ were set to 0.8 times the upper bounds.

We ran our procedure and generated a set of 20 target routes for each of the four meter readers using a balance parameter $\beta = 0.99$. Setting $\beta = 1$ causes unusual behavior in the record-to-record travel algorithm due to the distinction between deteriorating and improving moves, so we set β very close to 1 in order to reward balancing moves as much as possible. Our Phase 1 algorithm was able to construct routes that were well-balanced and efficient. We also ran a general record-to-record VRP algorithm on each of the four data sets, disregarding the lower bounds $\underline{L}$ and $\underline{C}$, in order to gain some measure of how the balancing constraints affect the final solution. The results of these computational experiments are presented in Table 3.4. Using the guided record-to-record procedure of Phase 1, we were able to find solutions

Meter Reader	Total Length (Balanced)	(Min, Max) Route Length	(Min, Max) Number of Customers	Total Length (Unbalanced)
1	3200	(137,170)	(126,150)	3205
2	3342	(146,170)	(135,157)	3342
3	4325	(187,224)	(177,200)	4322
4	4176	(183,225)	(166,205)	4174

Table 3.4: The quality of the target routes

that satisfy the upper and lower bound constraints with only one exception where a single route for meter reader 1 visited 126 customers compared to the lower bound of 128. Interestingly, we found that the balanced routes were very competitive with routes generated without using the lower bounds, and were even more efficient in certain cases (due to the nature of heuristics).

Phase 2 of the procedure provided billing day assignments for each of the target routes by solving the assignment problem (3.8). When running Phase 3 of the algorithm, we found that we were not always able to meet the balancing constraint (3.20), especially for small shift sizes when there are relatively few possibilities for each billing day shift. When we encountered such a situation where *BATCH-ROUTES* was infeasible, we increased the value of $\overline{C}$ by the minimum amount in order to produce a feasible solution and continued the procedure. The required increase was relatively modest. Due to the temporary nature of these intermediate routes, a small increase in $\overline{C}$ would seem tolerable, even if the utility company had to pay for additional overtime costs during some of the intermediate periods.

We ran our algorithm four times on each of the four data sets, using shift sizes $S = 2, 3, 4$, and 5. Each data set contains the customer locations, billing day assignments, and existing routes for one of the four full-time meter readers. The results are summarized in Table 3.5. The first column gives the instance and the maximum allowed shift size $(2, 3, 4$, or $5)$, the second column lists the total length of the original routes and the total length of the final target routes, and the third column gives the number of required intermediate periods. In column four, we count the number of customers who are not yet in their final route during each intermediate period. Column five lists the total length of all 20 routes during each required intermediate period, and the rightmost two columns list the minimum and maximum route lengths and the minimum and maximum number of customers across all intermediate periods.

As expected, we found that larger shift sizes led to fewer intermediate periods as well as more efficient and balanced intermediate routes. For meter readers 1 and 3, a shift size of $S = 2$ requires seven intermediate periods while meter readers 2 and 4 require five intermediate periods for this shift size. The routes for $S = 2$ are generally more costly and less balanced than the routes for larger shift sizes, and we were unable to meet the constraints regarding the route length and the number of customers, particularly for meter readers 3 and 4. If we allow shifts of size $S = 5$, we require at most three intermediate periods with meter readers 2 and 3 requiring no intermediate periods at all as we are able to transfer all customers from their original billing days to their final billing days in a single shift of fewer than five days.

In terms of total route length, the general trend is an initial increase over the target route length in the first intermediate period with the total route length of the subsequent intermediate periods then gradually decreasing before reaching the final configuration. In general, we were somewhat surprised at the high quality of the intermediate routes in terms of total length as we were typically able to avoid

(Instance, Shift Size)	(Original Length, Target Length)	# Inter-rmediate Periods	# Customers not in Final Route	Total Route Length	(Min, Max) Route Length	(Min, Max) Number of Customers
(1,2)		7	(384,213,105, (55,37,13,2)	(3231,3222,3214, 3214,3206,3203, 3201)	(108,208)	(95,194)
(1,3)	(3214,3200)	5	(175,114,57, 21,13)	(3219,3219,3215, 3204,3201)	(129,179)	(109,165)
(1,4)		3	(99,43,10)	(3208,3205,3200)	(128,174)	(114,157)
(1,5)		3	(25,11,3)	(3201,3201,3201)	(137,174)	(129,156)
(2,2)		5	(196,59,48, 31,6)	(3358,3343,3343, 3344,3343)	(124,198)	(107,176)
(2,3)	(3496,3342)	3	(59,34,5)	(3347,3343,3343)	(124,185)	(107,168)
(2,4)		1	(48)	(3343)	(124,187)	(107,168)
(2,5)		0	-	-	-	-
(3,2)		7	(468,326,194 53,50,18,2)	(4408,4372,4397, 4369,4363,4347, 4325)	(114,299)	(92,263)
(3,3)	(4522,4325)	5	(280,95,43, 23,6)	(4367,4363,4382, 4337,4326)	(140,242)	(122,215)
(3,4)		2	(63,19)	(4340,4342)	(187,231)	(171,205)
(3,5)		0	-	-	-	-
(4,2)		5	(641,412,317, 150,37)	(4264,4218,4258, 4233,4198)	(137,294)	(118,261)
(4,3)	(4157,4176)	5	(403,160, 86,28,4)	(4239,4210,4204, 4196,4181)	(138,254)	(118,226)
(4,4)		4	(192,65,34,3)	(4208,4198, 4217,4189)	(183,241)	(168,215)
(4,5)		2	(101,48)	(4185,4195)	(183,245)	(166,215)

Table 3.5: The performance of our procedure on four actual data sets

expensive insertions. In terms of percentage increase over the length of the target routes, the worst set of routes is the first set of intermediate routes for meter reader 4 and $S = 2$ with a total length of 4264 km compared to the target routes with a length of 4176 km. While this increase is very modest at just over 2%, it is important to note that these efficient intermediate routes do come at the expense of occasionally violating the maximum route length $\overline{L}$ and maximum number of customers per route $\overline{C}$, as seen in the rightmost two columns of Table 3.5. However, when compared to the existing extremes found in the original routes given in Table 3.3, these temporary violations would seem to be quite tolerable.

In Figures 3.2–3.7, we present a sequence of plots (produced by `http://www.gpsvisualizer.com`) derived from the data set for meter reader 4 with shift size $S = 4$. In these plots, the route for each billing day is represented by a different color and we have removed the edges radiating from the depot for clarity. In this example, the TSP tours that we computed for each of the existing billing days provide an initial solution with a total length of 4157 km. The intermediate routes are quite efficient, increasing the total route length by less than 0.3% over both the initial and target configurations. A visual inspection of the routes indicates that our procedure creates these efficient intermediate routes by moving clusters of customers from one route to the next, incurring minimal insertion costs. In other words, groups of neighboring customers are generally assigned the same final billing day and our procedure creates intermediate routes that preserve these clusters of customers.

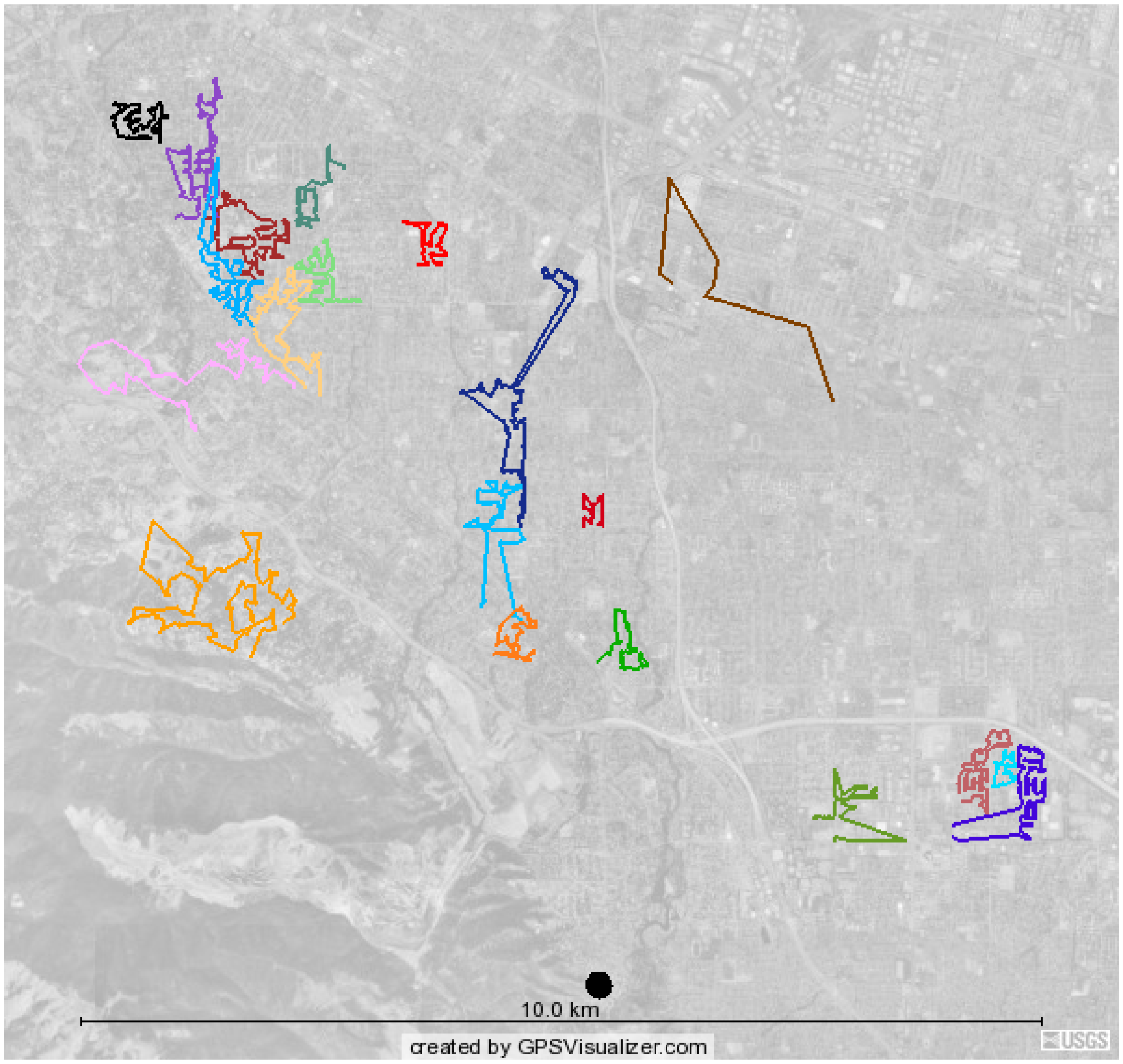


Figure 3.2: The original routes provided by the utility

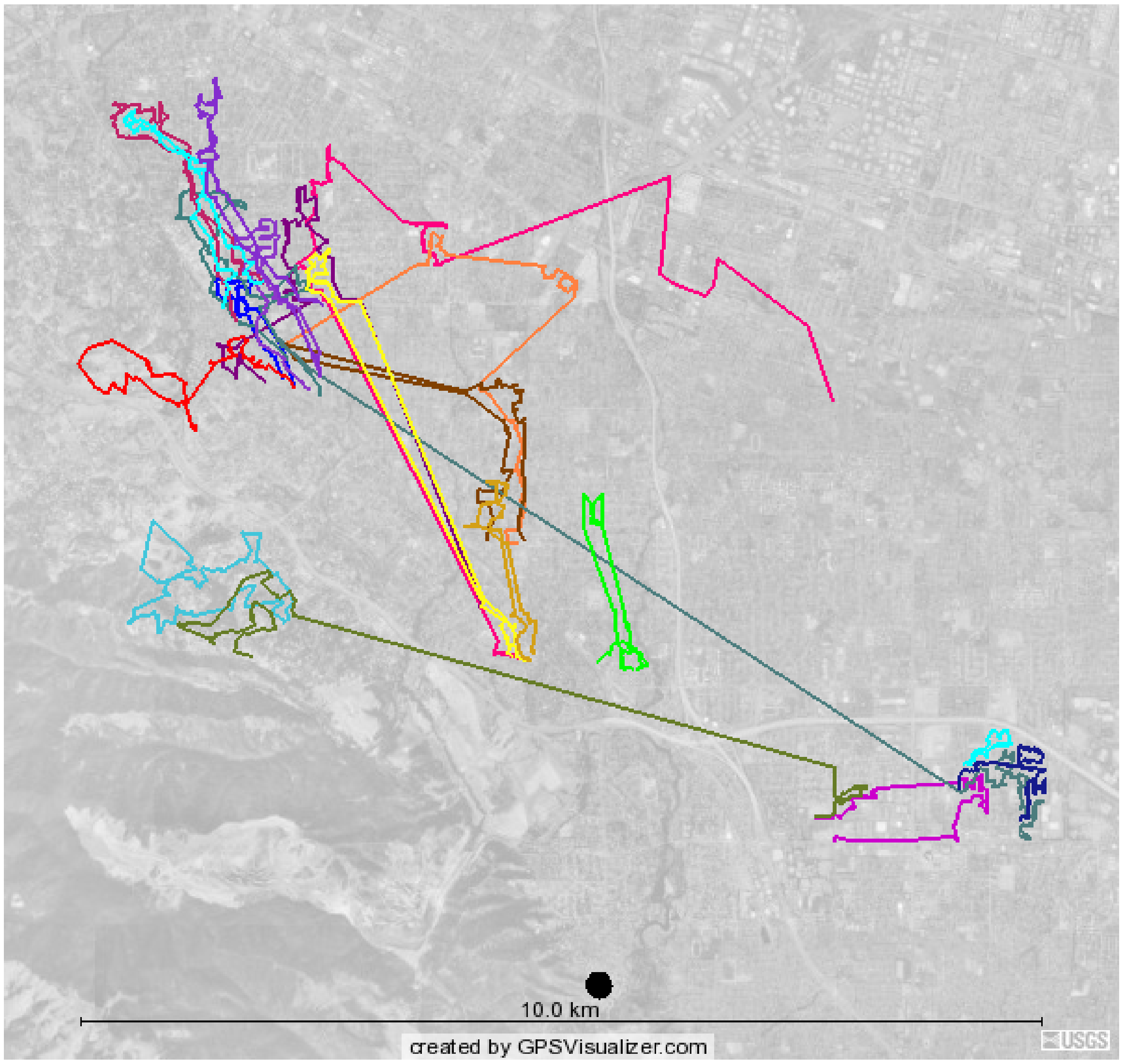


Figure 3.3: The routes for the first intermediate period

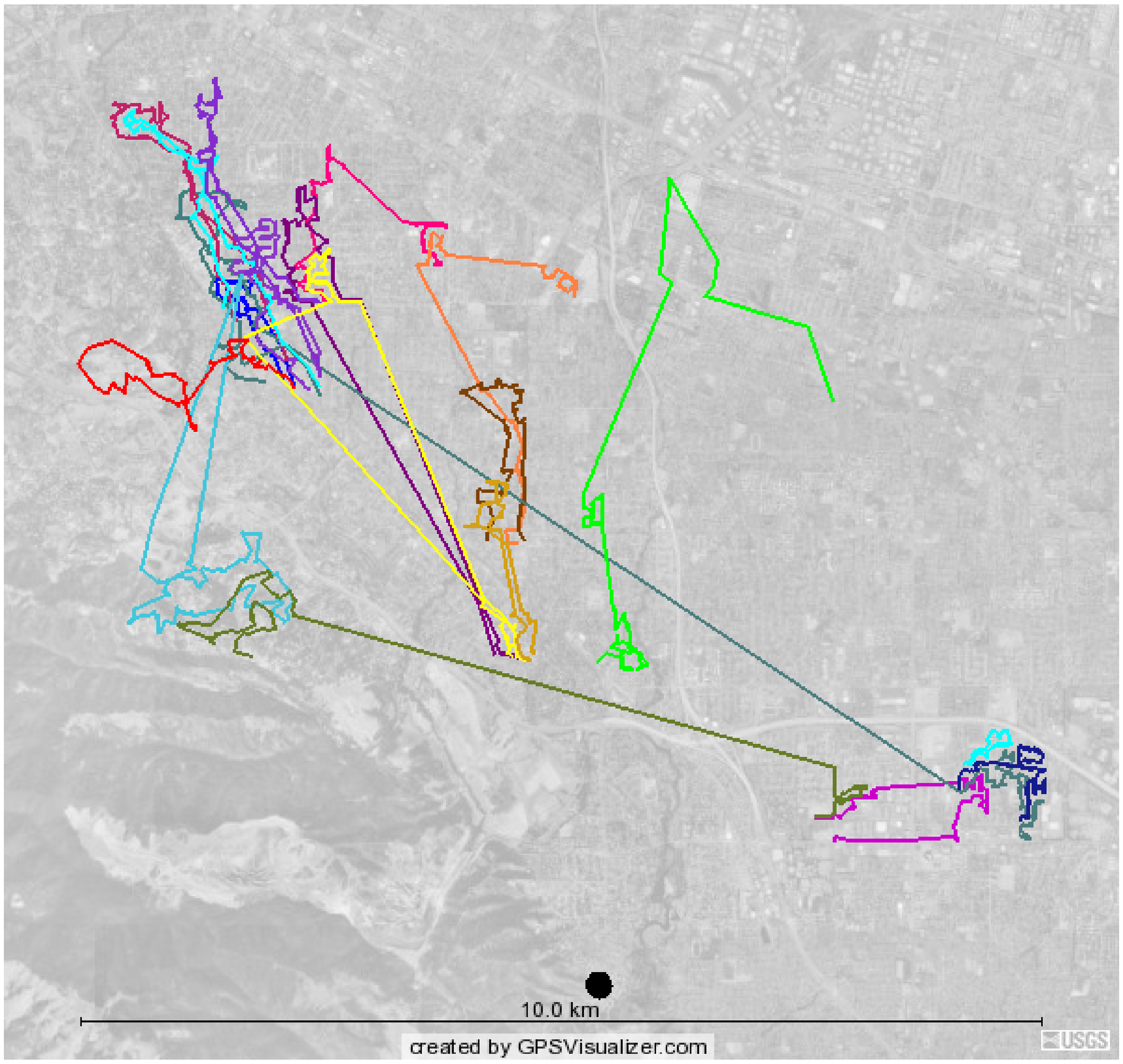


Figure 3.4: The routes for the second intermediate period

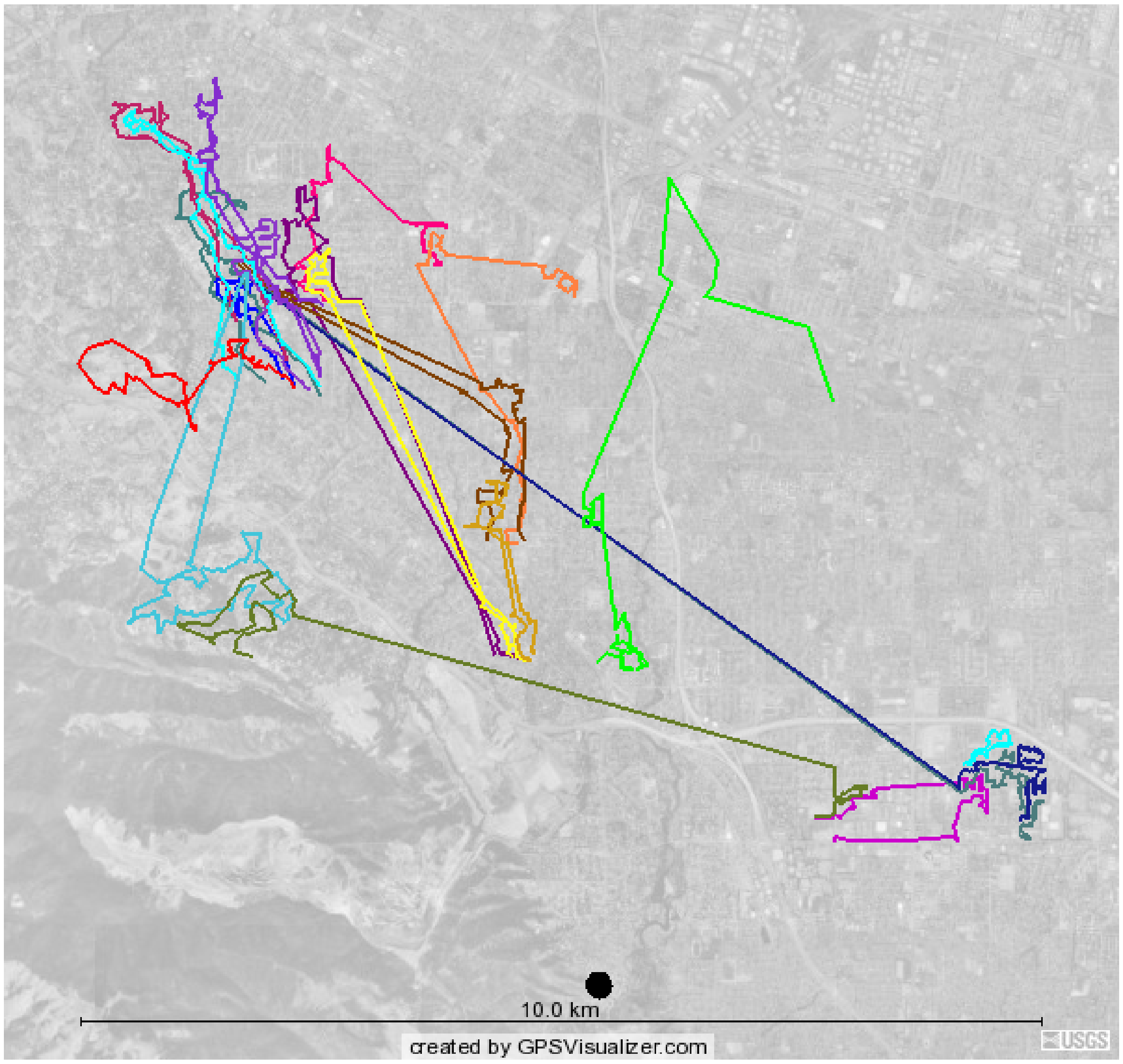


Figure 3.5: The routes for the third intermediate period

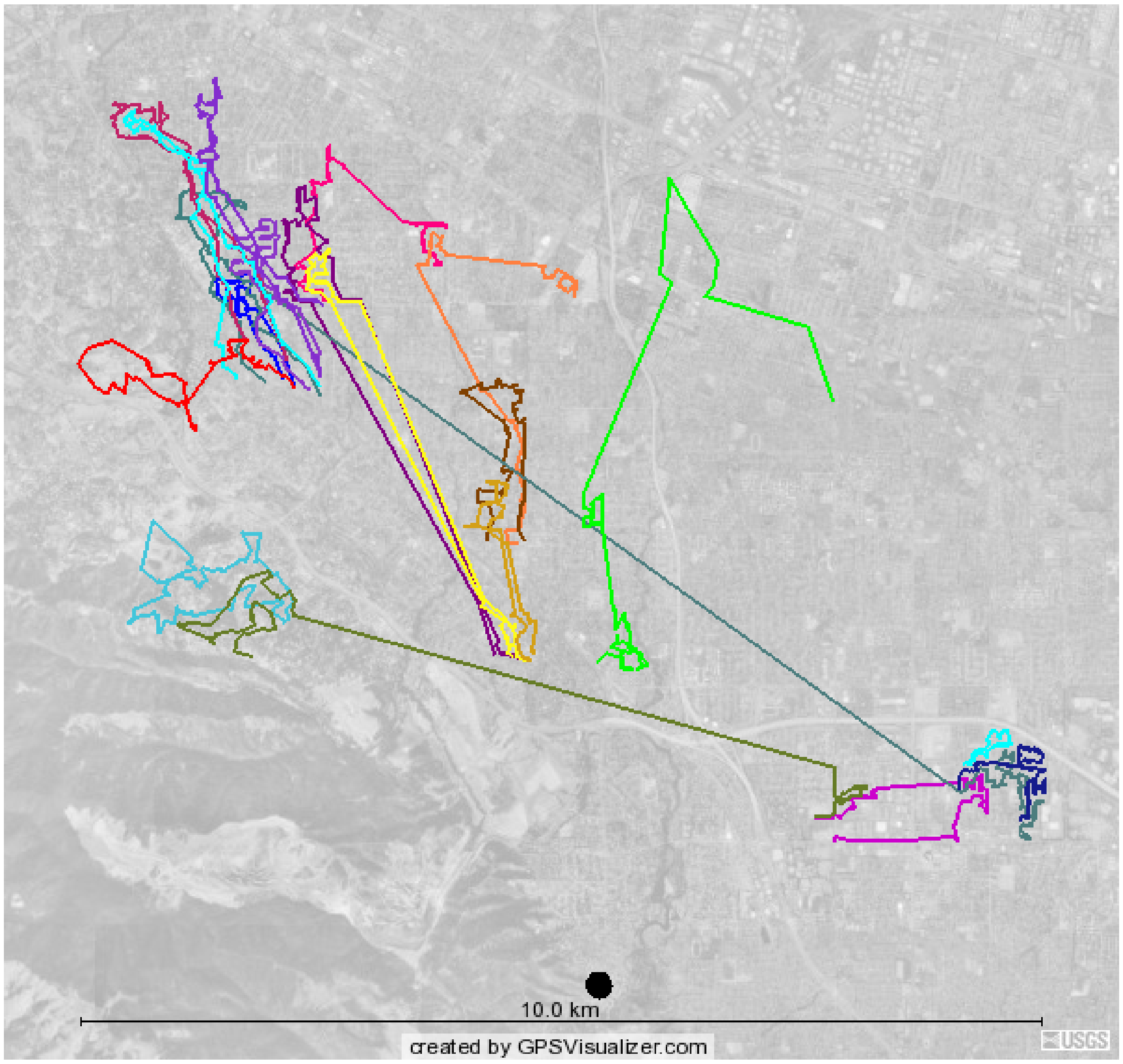


Figure 3.6: The routes for the fourth intermediate period

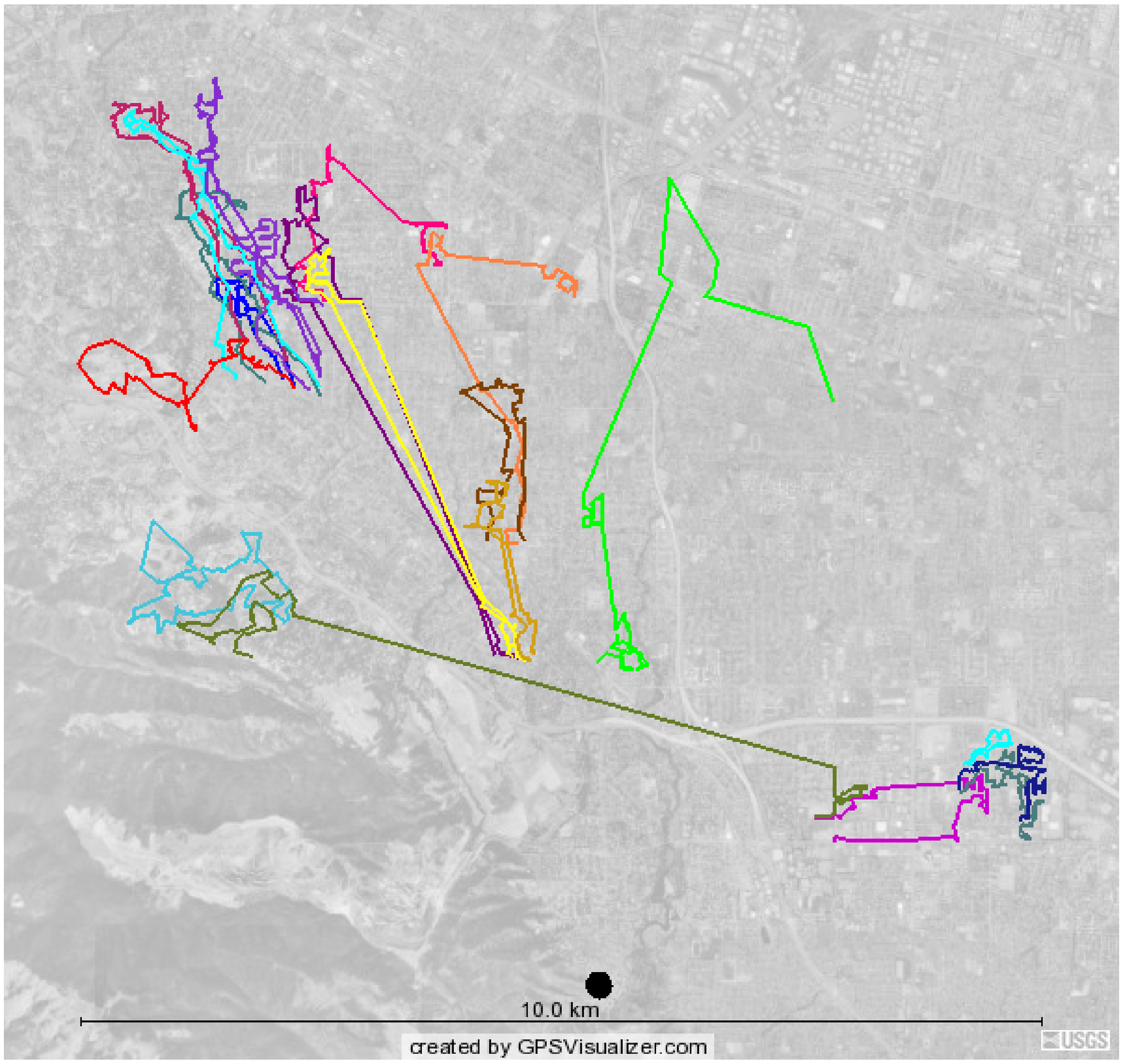


Figure 3.7: The final target routes

3.5. Conclusion

The Balanced Billing Cycle Vehicle Routing Problem is a new problem in the VRP literature. In the BBCVRP, we start with an initial billing day configuration that can be extremely poor. We are not allowed to start from scratch as is assumed in the typical vehicle routing problem. We developed a relatively simple heuristic method that produces solutions to the standard VRP with routes that are balanced in both the number of customers and length. Our procedure combines heuristic and exact methods to generate solutions to a new variant of the standard VRP that has unusual complicating constraints.

We tested the performance of our algorithm on a utility company's data set. Our algorithm produced efficient and balanced target routes along with a set of intermediate routes. In general, our procedure performed very well, allowing the utility to move from its existing, imbalanced initial configuration to a more efficient *and* more balanced configuration in a small number of steps. Furthermore, we found that the required intermediate routes remained quite balanced with a relatively small increase in cost.

In future work, we hope to investigate a modification of our approach that relaxes our assumption of having only one meter reader per day as this would possibly allow for a more efficient configuration when all the meter readers are taken into account. This modified approach may allow us to achieve a better balance of the workload across the different meter readers as well as across the different days of the billing cycle.

Finally, we note that some of our techniques may be useful for solving variants of the standard VRP. Our procedure for constructing balanced routes could be applied to any VRP where balancing is an important issue. In practice, balancing is rarely unimportant. Furthermore, there are real-world problems such as commercial sanitation collection which exhibit the features of the well-known Period Vehicle Routing Problem (PVRP) [16]. In such PVRP instances, some customers are visited once a week while others require more frequent service. In the literature, the PVRP is always solved from scratch. In practice, however, a current, inefficient set of daily routes may exist and the challenge is to improve these routes while minimally disrupting the current solution. The techniques presented in this chapter can be applied to such a scenario.

Chapter 4

A Cooperative Parallel Algorithm for the Vehicle Routing Problem

In this chapter, we develop a cooperative parallel algorithm for solving the classical VRP that combines a heuristic local search improvement procedure with a set covering formulation. We run our parallel algorithm with as many as 129 processors and are able to quickly find high-quality solutions to standard benchmark problems. We assess the impact of parallelism by analyzing our procedure's performance using different numbers of processors, and we assess the importance of the cooperative nature of the algorithm by running our procedure with different levels of information sharing among the various processors.

4.1. Introduction

Since its proposal by Dantzig and Ramser in 1959 [25], the classical vehicle routing problem has been the subject of a great deal of research. Given customers with known locations and demands, the goal is to find the minimum cost set of routes that satisfies all customer demands. Additionally, the vehicles assigned to these routes must carry no more than a fixed quantity of goods and can travel no more than a maximum distance. Although many variants of the VRP have been proposed over the last 50 years, even the simplest version of the classical VRP remains computationally difficult.

Because of its wide applicability and practical importance, many algorithms have been developed for solving the VRP. Over time, researchers have developed more powerful algorithms that find better and better solutions to both real-world VRP instances and well-studied benchmark instances by building on previous work and by taking advantage of faster computational resources. Despite the computational difficulty of the VRP, the literature contains relatively few algorithms that use the power of modern parallel computing resources.

Parallel computing allows the user to simultaneously run multiple processes or threads on several processors with the common goal of solving a particular problem instance. "Parallelism thus follows from a decomposition of the total workload and the distribution of the resulting tasks to the available processors" [24]. The power of parallel processing has become increasingly available with the advent of computing clusters. These clusters are networked computers that contain commodity hardware components that run standard operating systems. We take advantage of a high-performance computing environment to develop, implement, and test a parallel algorithm for solving the VRP.

Our parallel solver combines local search methods with a set covering formulation for the VRP. Some processors improve existing solutions by running a metaheuristic algorithm, and other processors attempt to combine existing solutions into improved solutions by solving a set covering problem. The resulting parallel algorithm quickly generates solutions to benchmark problems that are highly competitive with the best-known solutions reported in the literature. We analyze the parallel algorithm in several ways by measuring its performance as we vary the number of processors and the degree of cooperation among them.

This chapter is organized as follows. In Section 4.2, we summarize several serial algorithms for solving the VRP, review parallel algorithms, and discuss several algorithms that combine both heuristic and exact methods. We describe the details of our parallel algorithm in Section 4.3, present computational results in Section 4.4, and give our conclusions in Section 4.5.

4.2. Literature Review

There is an extensive body of literature describing algorithms for solving the VRP and its variants. A broad summary of recent work is given in the book by Golden, Raghavan, and Wasil [35]. We review some of the most powerful serial algorithms that have been developed for the VRP over the last 15 years. This summary is not intended to be exhaustive, and we refer to [22] and [33] for more detailed, recent reviews. We finish by discussing parallel algorithms for the VRP as well as several algorithms that combine heuristic and exact methods.

4.2.1 Powerful Serial Algorithms

The Taburoute algorithm of Gendreau, Hertz, and Laporte [32] uses the tabu search heuristic to solve the VRP. The algorithm allows infeasible intermediate solutions that are later discouraged by adding dynamic penalties to the objective function. Certain solution features are kept in short-term memory and the placement of certain nodes into certain routes is sometimes forbidden (tabu) for a number of iterations. Solutions are modified by applying two advanced local search operators (GENI and US). A total of nine parameters must be set to run the algorithm. When published in 1994, Taburoute was responsible for many of the best-known solutions to the benchmark problems of Christofides et al. [19, 20].

The algorithm of Rochat and Taillard [72] is one of the most powerful in the VRP literature. This procedure begins by generating a large number of initial solutions and then stores the individual routes contained in these solutions. Routes are extracted from this set in order to create new solutions that are improved by a local search procedure that uses tabu search. At each iteration, the routes are chosen probabilistically according to the quality of the solution containing the route. This allows routes from better quality solutions to be selected with a higher probability. This procedure concludes by solving a set covering problem where a single set of minimum cost routes is chosen from a large subset of all the routes found during the search.

The D-Ants algorithm of Reimann et al. [69] uses an ant colony optimization algorithm to create high-quality solutions to the VRP. After generating a set of initial solutions to a problem containing several hundred nodes, the problem is decomposed into smaller instances containing between 50 and 75 nodes. The information gained from solving these smaller problems is used to update the global memory which is used to aid in the search for solutions to the original problem instance.

Prins [64] presents a hybrid genetic algorithm (GA) for solving the VRP. This procedure was the first to demonstrate that a genetic algorithm could be competitive with tabu search and other sophisticated metaheuristics. The algorithm uses a simple local search procedure as a mutation operator and incorporates ideas from other successful GAs developed for solving different combinatorial optimization problems. At the time of its publication, the algorithm was able to produce several new best solutions to the benchmark problems presented in Golden et al. [36].

The SEPAS algorithm introduced by Tarantilis [78] shares many characteristics with the procedure of Rochat and Taillard [72]. Rather than constructing new solutions to be improved from routes, SEPAS uses smaller elite parts of previous solutions. These elite parts are connected strings of nodes that appear to be essential parts of high-quality solutions. A sophisticated tabu search algorithm improves full solutions by using short- and long-term memories. The algorithm requires a total of eight parameters.

The algorithm of Li et al. [51] improves the procedure given in [36] and is based on the record-to-record travel (RTR) procedure first suggested by [29] in the context of computational physics. Their algorithm alternates between a diversification phase and an improvement phase. In the diversification phase, three local search operators are applied and worsening moves are accepted as long as the objective function does not deteriorate by more than a threshold amount. The improvement phase applies the same local search techniques but allows only improving moves, i.e., those that decrease the total route length. The algorithm has a simple structure and requires only four parameters.

Pisinger and Røpke [62] develop a flexible heuristic that is capable of solving several different variants of the classical VRP. For each type of VRP variant, the authors transform the problem into what they call a "rich pickup and delivery problem with time windows." Then they apply adaptive large neighborhood search to iteratively improve solutions. Their procedure generates very good solutions to benchmark problems for several VRP variants, but requires fairly long computing times.

Two recent algorithms have established themselves as the clear leaders in terms of solution quality and are responsible for most of the best solutions to the standard benchmark problems. The active-guided evolution strategy (AGES) algorithm developed by Mester and Bräysy [53, 54] uses a set of well-known local search operators as part of a two-phase algorithm. In the first phase, edges are penalized according to certain strategies in an attempt to diversify the search and escape local minima. In the second phase, customers are ejected from the solution and then reinserted into new positions. Several different strategies are used to select the customers to eject, and the algorithm quickly produces high-quality solutions. The procedure is rather intricate and involves up to 10 parameters.

The memetic edge assembly crossover algorithm of Nagata [57] and Nagata and Bräysy [58] uses a genetic crossover operation that was first introduced as an effective approach for solving the traveling salesman problem in [56]. After creating an initial population, solutions are selected and merged to produce an offspring solution. This solution is improved by a local search algorithm that accommodates infeasible intermediate solutions using penalties as in [32]. A drawback of this procedure is that the genetic crossover operation is only capable of handling vehicle capacity constraints and not route-length restrictions. Nevertheless, this procedure is quite different from the other successful VRP algorithms and demonstrates that a cleverly designed genetic algorithm can provide very high-quality solutions.

4.2.2 Parallel Algorithms

In reviewing the literature, we were surprised to find that relatively few parallel VRP algorithms have been proposed. This view is shared by Crainic [24] who provides a survey of exact and heuristic parallel algorithms for the VRP and its variants. We describe some of the more notable parallel algorithms developed for solving the VRP and its variants.

Rego [68] presents a parallel tabu search algorithm that uses the ejection chain neighborhood as a component of the local search. His implementation uses four slave processors and a single master. Initial diversity is created by providing each slave with the same initial solution and then running the serial tabu search procedure with different parameters. The evaluation of some of the more complex ejection chain moves is done in parallel, and a post-optimization procedure is accelerated by determining high-quality TSP tours for each of the individual routes by sending individual routes to different slave processors.

Alba and Dorronsoro [2] propose a parallel cellular genetic algorithm where genetic crossover and mutation operations are combined with a local search procedure. The algorithm allows for infeasible intermediate solutions using a penalized objective function. The population is arranged in a mesh that limits the choice of mating solutions to four neighbors. The population is regenerated at each generation in parallel, leading to decreased computation time and better solution quality. This work is improved and extended in Dorronsoro et al. [28] where the authors run a modified procedure on a grid platform containing 125 machines and conduct computational experiments requiring up to 75 hours of computing time. The algorithm produces very competitive solutions for the large-scale problems proposed in Li et al. [51]. This work represents the only research that we are aware of where the VRP is solved in a modern, high-performance computing environment. Additional details on cellular genetic algorithms can be found in the book [3].

A parallel implementation of the D-Ants algorithm is presented by Doerner et al. [27]. The authors explore several different parallelization strategies, but their primary goal is the speedup obtained by parallelizing the search and they do not attempt to improve solution quality in their parallel implementation.

Finally, we mention two parallel algorithms that solve the VRP with Time Windows (VRPTW). Schulze and Fahle [73] develop an effective algorithm for solving the VRPTW that combines a tabu search metaheuristic with a set covering formulation. In this algorithm, each processor begins with a set of solutions to the problem and then improves the solution by running tabu search. After completing the tabu search procedure, each processor broadcasts out a subset of routes discovered during tabu search and receives a set of routes sent out by other processors. Each processor then runs a heuristic algorithm to solve a set covering problem that attempts to recover a new solution that can be better than any of the individual solutions. The individual processors then update their set of solutions and repeat this process until a stopping condition is met. The authors report very competitive results when running their algorithm with eight processors.

Le Bouthillier and Crainic [47] develop a powerful parallel solver for the VRPTW that uses two different tabu search metaheuristics and four different methods for constructing initial solutions. After creating an initial solution, each processor improves a solution using one of the two metaheuristics and sends the best solution that it found to a central solution pool or warehouse. Post-optimization procedures are run on the solutions in this central pool, and the individual processors proceed by running one of the two tabu search metaheuristics on a new solution obtained from this central pool. The algorithm produces very competitive solutions to a large number of VRPTW benchmark problems. This algorithm was subsequently improved in Le Bouthillier et al. [48] where they extended the algorithm by considering sets of directed edges shared by the best solutions.

4.2.3 Combining Exact and Heuristic Methods

We now present several algorithms for solving the VRP that combine exact and heuristic methods. Many of these algorithms involve the set covering formulation of the VRP. Let $\mathcal{R} = \{1, 2, \ldots, R\}$ denote the set of all feasible routes, and let c_r represent the cost of route r. The decision variable $x_r = 1$ if route r is chosen and $x_r = 0$ otherwise. Then, with $a_{ir} = 1$ if node i is contained in route r and $a_{ir} = 0$ otherwise, the set covering formulation (SC) for the VRP is given by

$$\text{Minimize} \quad \sum_{r=1}^{R} x_r c_r \tag{4.1}$$

$$\text{s.t.} \quad \sum_{r=1}^{R} x_r a_{ir} \geq 1 \text{ for all nodes } i, \tag{4.2}$$

$$x_r \in \{0, 1\}.$$

The objective function (4.1) minimizes the sum of the costs of the selected routes. Constraint (4.2) guarantees that every node is contained in at least one route. If the triangle inequality is not satisfied, then (4.2) must be replaced by an equality constraint. When the triangle inequality is satisfied, if we encounter a set of routes where some node is contained in more than one route, we modify this set of routes so that the node is contained in only one route and choose the set of routes with lowest cost.

Because the set $\mathcal{R}$ contains only feasible routes, this formulation can handle additional constraints with no modification and is the foundation for many exact solution methods for the VRP and several variants. The typical exact algorithm solves the linear programming (LP) relaxation of SC via column generation and then attempts to find an optimal or near-optimal integer solution by solving SC as an integer program with the final set of columns from the solution to the column generation problem. For examples of this approach, see [1, 7, 26, 38]. Recent work is summarized by Bramel and Simchi-Levi [9].

Heuristic algorithms have also used this set covering formulation. The algorithm of Rochat and Taillard [72] solves the set covering formulation as a final optimization step. Taillard and Boffey [77] combine the set covering formulation with a tabu search heuristic to produce very high-quality solutions to the heterogeneous fleet VRP. Alvarenga et al. [4] develop a serial algorithm that combines local search and genetic crossovers to generate solutions to the VRPTW. The individual routes from these solutions are added to a large set of routes that is used as input to a set partitioning solver as in [72]. The authors use an integer programming solver as an integral part of their algorithm, whereas Rochat and Taillard use the set covering formulation only as a post-optimization procedure.

Recent research has attempted to incorporate more complicated integer programming formulations into heuristic methods. DeFranceschi et al. [30] develop a technique for improving very good VRP solutions by extracting certain nodes and then inserting them into new positions by solving an integer program. Toth and Tromantani [79] extend this technique by considering the column generation problem in more detail. In both cases, very good solutions are obtained with fairly long computing times.

4.3. Description of the Parallel Algorithm

In this section, we provide the details of our parallel algorithm. We incorporate a heuristic algorithm and the set covering formulation described in Section 4.2.3 into a cooperative parallel search procedure. We begin this section by classifying our parallel algorithm according to the taxonomy suggested by Crainic and Nourredine [23]. We then describe our serial heuristic algorithm and the details of our parallel algorithm.

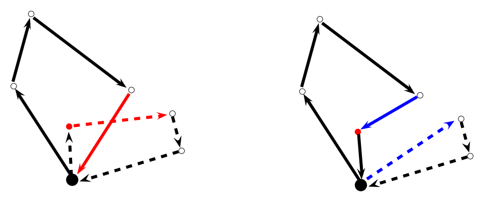
Crainic and Nourredine use three dimensions to classify parallel metaheuristic strategies. The first dimension is search control cardinality and measures how the search is controlled. The second dimension deals with the type and amount of information exchanged among the processors. The third dimension addresses the similarities and differences of the search strategies employed by the different processors.

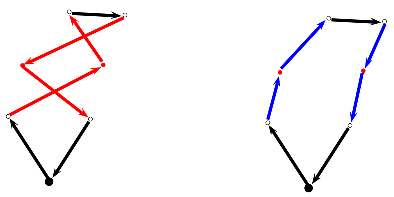
According to this system, our algorithm fits into the *pC/KC/MPDS* classification. The *pC* classification indicates that the global search is controlled by multiple, collaborating processors. *KC* stands for Knowledge Collegial information exchange and indicates that multiple processors exchange information asynchronously and that new solutions are created from the exchanged information (this is achieved by solving the set covering problem). Finally, our procedure fits the *MPDS* classification (Multiple Points, Different Strategies) since the heuristic solvers run a different strategy from different points in the solution space.

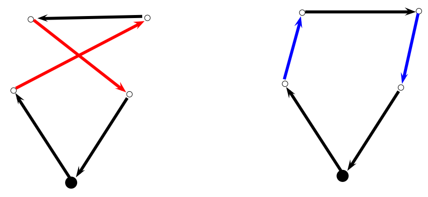
4.3.1 A Serial VRP Algorithm

In our parallel algorithm, we implement the record-to-record travel algorithm (RTR) used by Chao et al. [16] to solve the period vehicle routing problem and later used by Golden et al. [36] and Li et al. [51] to solve the classical VRP. These RTR-based algorithms all have a very simple structure, require a small number of parameters, and perform consistently well on a variety of benchmark problems. Our implementation maintains the basic structure of these previous algorithms, and enhances them by adding more local search operators and by using randomly generated parameters.

The RTR algorithm given in Li et al. [51] uses the three local search operators shown in Figure 4.1: one-point move, two-point move, and two-opt move. The one-point move shifts a single node to a new location, the two-point move swaps the


(a) One-Point Move


(b) Two-Point Move


(c) Intra-route Two-Opt Move

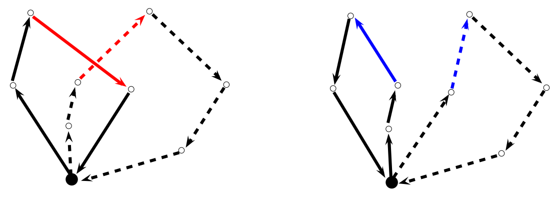
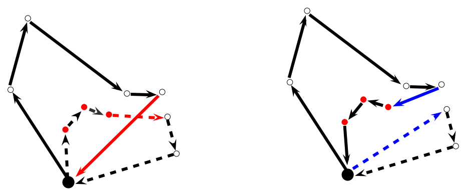

(d) Inter-route Two-Opt Move

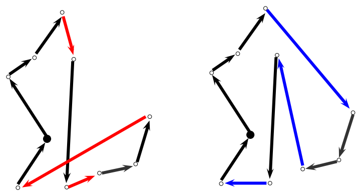
Figure 4.1: The improvement operators used in Li et al. [51]

locations of two nodes, and the two-opt move removes two edges from the existing solution and replaces them with two new edges.

In addition to the three local search operators shown in Figure 4.1, we use three more in our implementation: Or-opt move, three-opt move, and three-point move. These operators are shown in Figure 4.2. In the Or-opt move [60], we remove a string of two, three, or four customers from the solution and insert this string into a new position. The three-opt move [52] removes three edges from a single route and adds three new edges so that a feasible route is maintained. The three-point move is a special case of the λ-interchange operator of Osman [61] where we exchange the positions of two consecutive nodes with the position of a third node.

For all six local search operators, we perform the local search in a straightforward way. For each node i, we create a list L_i that contains i's nearest N neighbors (N is a parameter). In the diversification phase, for any local search operator and a given node i, we search the list L_i for a feasible move. If we find an improving


(a) Or-Opt Move


(b) Three-Opt Move

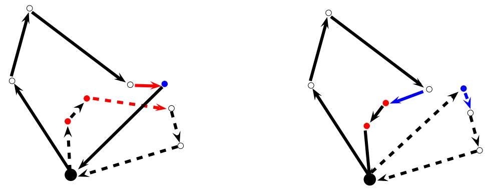

(c) Three-Point Move

Figure 4.2: Additional improvement operators used in our serial algorithm

move, then we make this move. If no improving move is found, then we search L_i and store the feasible move with the least amount of deterioration. If this move does not increase the total route length by more than a fixed amount based on the best solution (known as the record), then we make the move. This fixed amount of allowed deterioration changes as we find better and better records.

In the improvement phase, we use all six operators, and look for an improving move for each node i, again by searching the neighbor list L_i. If no improving move is found, then we perform no modifications and move to the next node. We generate the parameters for each run of the RTR algorithm at random within a reasonable range. This eliminates the need for parameter tuning and adds diversity to the search. We refer to our implementation of the algorithm as Randomized RTR (RRTR).

In the RRTR algorithm, after generating a set of random parameters in steps 1 to 3, steps 5 to 11 attempt to diversify the solution by accepting deteriorating moves according to the record-to-record travel strategy. Steps 12 to 18 represent

Algorithm 4.1 The Randomized RTR algorithm for the VRP

```
1: Generate random parameters $I \in \{25, 75\}$, $P \in \{5, 6, \dots, 10\}$, $N \in$
   $\{25, 26, \dots, 75\}$, $\delta \in (0.005, 0.015)$, and $K \in \{5, 6, \dots, 10\}$
2: Let $\mathcal{V}$ denote the set of six local search operators, and randomly select a subset
   of operators, $\mathcal{U} \subseteq \mathcal{V}$
3: Set the record $R$ equal to the current total route length, set the threshold $T =$
   $(1 + \delta)R$, and set $k = p = 0$
4: **while** $p < P$ **do**
5:     **for** $i = 1$ to $I$ **do**
6:         **for all** Operators $u \in \mathcal{U}$ **do**
7:             **for** $j = 1$ to $n$ **do**
8:                 Apply operator $u$ to node $j$ using record-to-record travel
9:             **end for**
10:        **end for**
11:    **end for**
12:    **while** Improving moves can be found **do**
13:        **for all** Operators $v \in \mathcal{V}$ **do**
14:            **for** $j = 1$ to $n$ **do**
15:                Apply operator $v$ to node $j$ accepting only improving moves
16:            **end for**
17:        **end for**
18:    **end while**
19:    **if** The current solution is a new record **then**
20:        Update $R$ and $T$ and set $k = 0$
21:    **end if**
22:    $k++$
23:    **if** $k == K$ **then**
24:        Perturb the solution
25:        $p++$
26:    **end if**
27: **end while**
28: Return a list of the 50 best solutions found
```

the improvement phase where only improving moves are accepted and we reach a local minimum in the search space. We apply all six local search operators in the improvement phase and only a randomly selected subset of these operators in the diversification phase. If the local minimum at the end of the improvement phase is a new record, then we update this value and the threshold in step 20 and reset the counter (k) that keeps track of the number of times we have been unable to escape a particular local minimum. In steps 23 to 26, we check to see if we have been unable to improve the current record after K times through the main loop of the algorithm. If there is no improvement, then we perturb the solution by removing a set of customers from the solution and reinserting them into new locations (this technique is given in Li et al. [51]). Once we have perturbed the solution P times, we return the 50 best solutions found during the search.

In preliminary experiments on several benchmark problems, we ran the RRTR algorithm using all six local search operators and a single processor. We found that the solution quality was equivalent to the results given in [51] where only three local search operators were used. Our motivation in using these additional operators is simple: we want to have each processor running a slightly different algorithm using different parameters in order to diversify the search.

4.3.2 Architecture of the Parallel Algorithm

We begin with a general outline of the parallel algorithm and then provide a detailed description of the roles played by the individual processors. Each processor is devoted to one of three different tasks or processes. We have a single master processor to guide the search and distribute solutions (the master), a set of processors dedicated to generating solutions with the RRTR algorithm (heuristic solvers), and a group of processors that solve the set covering formulation (set covering solvers). We exchange information among the different processors in an attempt to improve the overall solution quality and reduce the time required to reach a high-quality solution.

Our overall strategy uses the heuristic solvers to generate many solutions to a problem. We then send the routes produced by the heuristic solvers to the set covering solvers by a completely asynchronous mechanism. This allows the different processors to complete their tasks at different times without having to wait for the others to finish. We transfer this information from the heuristic solvers to the set covering solvers by using a file-based approach. In particular, we create a single file directory for each set covering solver and assign a number of heuristic solvers to each set covering solver.

After a heuristic solver completes the RRTR algorithm, it writes a large set of routes discovered in the most recent run to a file in the appropriate directory. The heuristic solver then sends a set of solutions to the master, receives a new solution to be improved from the master, generates a new set of parameters, and repeats the RRTR algorithm. Meanwhile, each set covering solver searches for new routes (columns) by reading the files written by the heuristic solvers in its directory and then attempts to solve the current set covering problem to optimality. After solving the problem to optimality or stopping after reaching a pre-determined time limit, the set covering solver sends the best solution that it has found back to the master. This series of steps continues until a time limit is reached, at which point the master writes the overall best solution discovered during the search to a file. The overall flow of information amongst the processors is illustrated in Figure 4.3. The dashed lines represent the flow of information between the heuristic solvers and the set covering solvers, and the solid lines represent information sent to and from the master (we omit some lines that involve the master to keep the diagram simple).

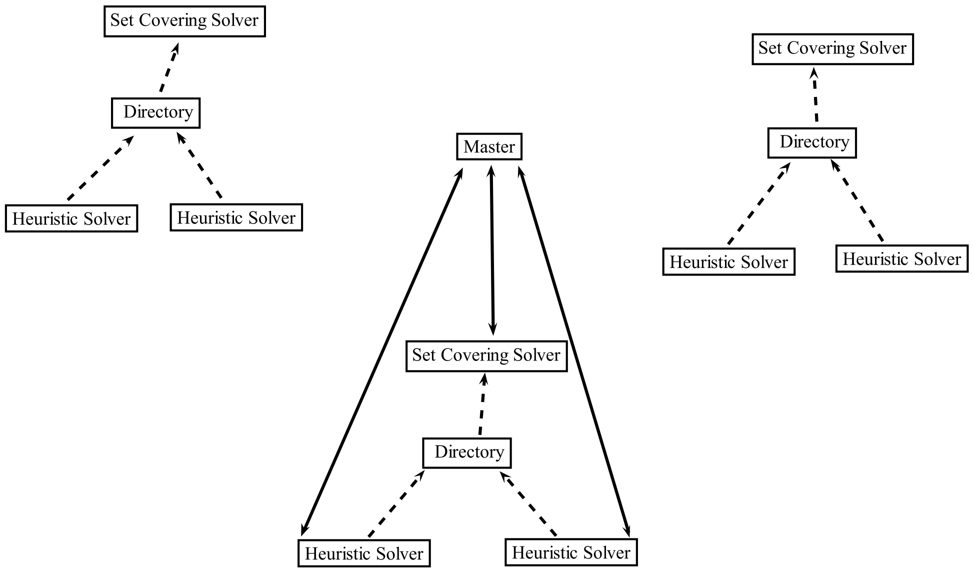


Figure 4.3: The architecture of the parallel algorithm

With the general outline and structure of the parallel algorithm as background, we now provide a more detailed description of the roles played by the heuristic solvers, the set covering solvers, and the master processor.

4.3.3 The Role of the Heuristic Solvers

After generating an initial solution with the Clarke-Wright algorithm using a randomly generated shape parameter λ as described by Yellow [84], the heuristic solvers repeatedly perform the following steps: 1) receive a solution from the master, 2) attempt to improve the procedure by running the RRTR algorithm, 3) write a file containing a set of routes discovered during the search, and 4) send the 50 best unique solutions produced during the most recent run to the master.

When running the RRTR algorithm, we keep track of the 50 best unique solutions found during the search. We also store the solutions that are found at the end of the *while* loop in step 18 of the RRTR algorithm since each of these solutions represents a local minimum in the search space and can contain promising routes. Next, we take this set of solutions, perform three-opt improvement on each route until no improvements can be found, and then create a list of all unique routes found in this set of solutions. We sort these routes by the total route length of the solution containing the route, and then write these routes to a file. By sorting the list of routes in this way, when the set covering solver reads this file and adds the routes to the integer program, routes derived from higher quality solutions are selected first. After creating this sorted list of unique routes, the heuristic solver writes a file named *u_v.rte*, where u is the total route length of the best solution found during the search and v is the average total route length of the solutions. By naming the files in this way, when the set covering solvers search for files, they can select files that contain routes derived from better quality solutions simply by inspecting the name of the file.

4.3.4 The Role of the Set Covering Solvers

Our heuristic solvers are quite fast. They complete the RRTR algorithm in 5 to 20 seconds, depending on the problem size. In contrast, an integer program (IP) can be very difficult to solve, even with the best commercial solvers. In order to combine heuristic procedures and IP solvers into a single, cooperative, parallel VRP algorithm, we must ensure that the computation times for both methods are roughly equivalent. Otherwise, if the set covering problems grow too large and become difficult to solve, the IP solver may spend all of its time solving a single problem that contains routes derived from heuristic solutions generated very early on in the search. In order to increase the cooperation among the heuristic solvers and the set covering solvers, we must carefully manage the number of columns or routes we allow into the set covering problem.

We initially set two parameters: s, the maximum number of seconds allowed to solve a single set covering problem, and m, the minimum number of new columns that must be found before attempting to solve a new set covering problem. We control the total number of columns in the set covering formulation by monitoring the time required to solve the integer program. Each set covering solver keeps track of a quantity M_j, the maximum number of columns allowed when solving integer program j. We set an initial value $M_0 = 500$ and then, after solving this initial problem, we set M_{j+1} in terms of M_j via a simple rule:

$$M_{j+1} = \begin{cases} M_j, & \text{if problem } j \text{ required between } s/2 \text{ and } s \text{ seconds,} \\ \lfloor 1.1 * M_j \rfloor, & \text{if problem } j \text{ required less than } s/2 \text{ seconds,} \\ \lfloor .9 * M_j \rfloor, & \text{if problem } j \text{ was not solved to optimality in } s \text{ seconds.} \end{cases} \tag{4.3}$$

After initially finding M_0 columns and then solving this first problem, M_1 is computed using this rule and the set covering solver starts the search for additional columns by reading the files written by the heuristic solvers. After finding m new columns, the problem is solved again. Note that recursion (4.3) implies that, in some cases, we remove existing columns from the integer program. In these cases, we select columns to remove at random, except that we never remove a column that was once part of an optimal solution to a set covering problem.

4.3.5 The Role of the Master

The master processor is responsible for storing the solutions and for coordinating the overall search. The master keeps track of the 1000 best unique solutions and is responsible for sending new solutions to the other processors when they complete their tasks.

When the master receives a new solution from a set covering solver, it sends the 10 best solutions to the set covering solver. This ensures that each set covering solver always has the routes from the current 10 best solutions.

When the master receives new solutions from the heuristic solvers, it adds them to the list of the 1000 best unique solutions (we use a hash table to increase efficiency) and sorts this list in increasing order by the total route length. After adding these solutions to its list, the master then determines which solution(s) to send back to the heuristic solver.

In the simplest case, the master selects one solution and sends it back to the heuristic solver. We experimented with two different strategies to select this solution. In the first case, the master selects the best solution found so far. In the second strategy, the master selects a random solution whose quality is biased in favor of better solutions as the search progresses. In particular, if, after s seconds, the master has k total unique solutions and a total allowed time of t seconds, we select solution j, where j is generated by the following procedure:

- Generate $r \in (0, 1)$ uniformly at random;
- Compute $j = \lfloor k(1 - s/t)^{r+2} \rfloor$;
- Return solution j from the sorted list.

This procedure tends to send out worse solutions early on in the search, and better solutions near the end. In Figure 4.4, we show the average value of j for $k = 1000$ and $t = 100$ as time progresses from 0 to 100.

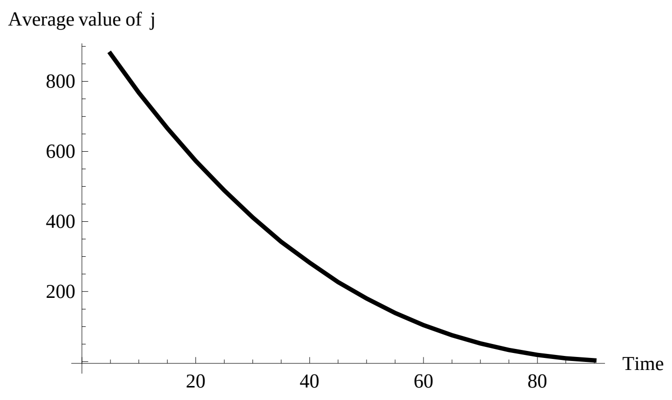
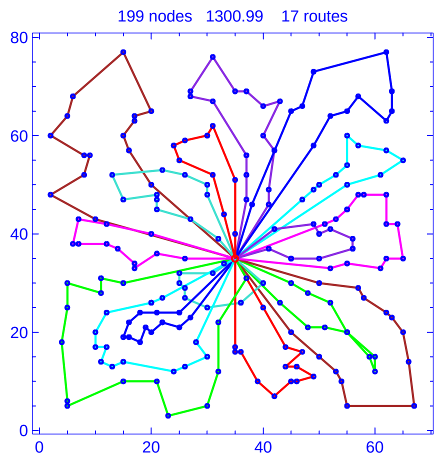


Figure 4.4: The average rank of the randomly selected solution

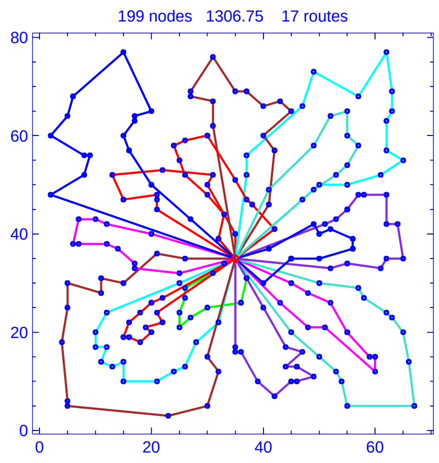
We also employed two strategies to divide the problem into smaller subproblems that contain fewer nodes than the original problem. By running the RRTR algorithm on these smaller problems, we hope to find additional routes that might not be discovered when running the algorithm on the entire original problem. We use these strategies in the second half of the search process after we have (hopefully) created a very high-quality set of solutions. In the first strategy, the master selects one of the

10 best solutions, and a second, inferior solution that is chosen at random from the remaining pool of solutions. The master then attempts to create a smaller problem for the heuristic solver by taking the superior solution and removing any routes that are shared with the inferior solution. The master sends the resulting sub-problem to the heuristic solver. The logic behind this idea is straightforward. At some point in the search, certain routes become obvious and are shared by many solutions. By removing routes shared by the two solutions, we can focus more intensely on the "non-obvious" parts of the problem. In Figure 4.5, we show an example of this route removal procedure where two solutions (17 routes, 199 nodes) have four routes in common. After removing these four routes, we have a smaller subproblem with 13 routes and 158 nodes.

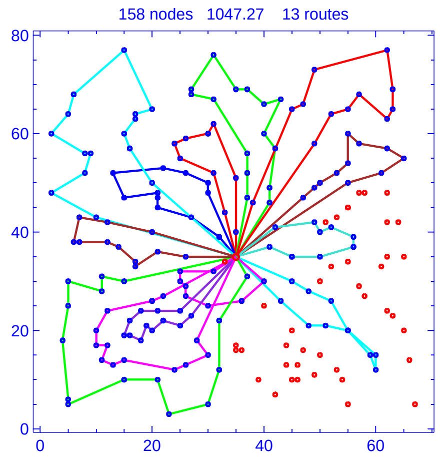
In the second strategy, the master selects a single solution at random and then draws a single randomly generated line through the depot in order to split the problem in two disjoint parts. We accept this splitting if the two parts are roughly equal in size, so that each part has 40–60% of the total number of nodes. Then, in order to preserve some of the routes found in the initial solution, we create two problems by first selecting all routes that have at least one node above the line and then selecting all routes that have at least one node below this line. The master then sends the larger of the two resulting subproblems to the heuristic solver. This procedure is illustrated in Figure 4.6 where two routes are removed from the solution, creating a problem instance with 28 fewer nodes.



(a) Solution 1



(b) Solution 2

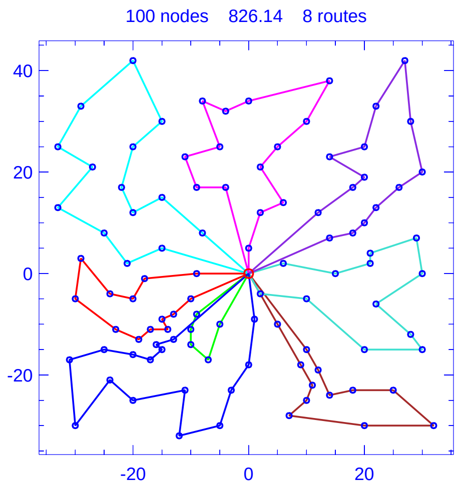


(c) Solution after removal of routes in common

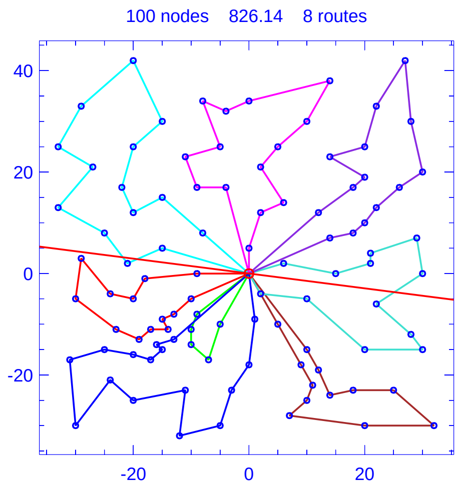
Figure 4.5: The route removal operation

4.4. Computational Results

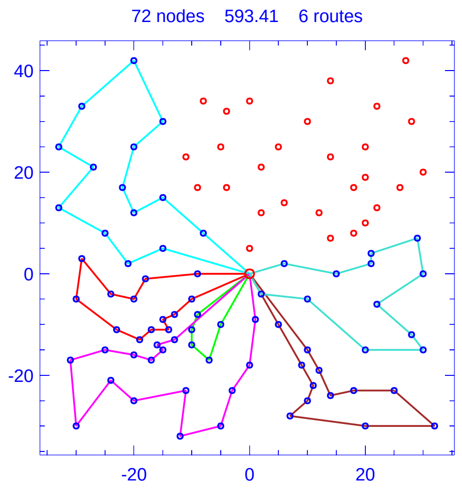
We implemented our algorithm in C/C++, using the well-known Message Passing Interface (MPI) to handle the inter-process communication and CPLEX 11.1 as the IP solver for the set covering problems. We had access to two different computing clusters: the heterogeneous 1200 compute core Linux-based Deepthought cluster at the University of Maryland, and a separate 50-node Windows-based Microsoft Com-



(a) Solution before splitting



(b) Splitting the solution



(c) Solution after splitting

Figure 4.6: The solution splitting operation

pute Cluster with 200 compute cores (2.3 GHz Intel Xeon processors). We conducted a large part of the development and testing of our algorithm on the Deepthought cluster. Our final computational results are generated from runs on the Microsoft Compute Cluster.

We provide computational results for our parallel algorithm's performance on 55 problems taken from four well-known sets of benchmark problems: 14 problems from Christofides et al. [19, 20], 9 problems from Taillard [75], 20 problems from Golden et al. [36], and 12 problems from Li et al. [51].

In the first subsection, we report the solutions found by our parallel algorithm and compare them to the results reported in the literature. In the second subsection, we solve selected problems with our parallel algorithm under different scenarios in order to analyze the impact of the parallelism and cooperation on the algorithm's speed and performance. In the third subsection, we vary the number of processors but use a fixed amount of total computing time to measure the parallel speedup of our algorithm.

4.4.1 Solutions to Benchmark Problems

We ran our cooperative parallel algorithm on the 55 benchmark problems with the following parameters: 129 total processors, 16 set covering solvers, 5 second time limit for each set covering problem, and 600 seconds of total wallclock time. We ran our procedure five times on each problem and report the best solution found during these runs.

In Tables 4.1–4.4, we present the results of our computational experiments. The cooperative parallel algorithm generated very high-quality solutions to all 55 benchmark problems, discovering 13 new best solutions. We found 31 solutions that equaled the existing best-known solutions. On the remaining 11 problems, our algorithm produced solutions that were slightly worse than the best-known solutions (an average deviation of 0.15%). Appendix B contains the details of these solutions.

A key aspect of our algorithm is its cooperative nature. In order to assess the importance of this cooperation, we ran a non-cooperative parallel version of our algorithm on the 55 benchmark problems. We used 129 processors for 600 seconds and ran the procedure five times on each problem. In this version, we remove all cooperation among the processors and use 128 heuristic solvers and no set covering solvers. In this variant of our algorithm, each heuristic solver repeatedly generates a new random initial solution using the parameterized Clarke-Wright algorithm and then improves this solution by running the RRTR algorithm. When the time limit is reached, each heuristic solver sends back the best solution it found and the master records the best overall solution discovered during the search. This strategy is equivalent to running the serial algorithm 128 times for 600 seconds each time on a single processor and then returning the best solution. In this case, parallel processing allows us to complete this computational task 128 times faster.

In Tables 4.1–4.4, the first two columns give some details about the problem, and the third column provides the previous best-known solution along with the earliest source that we could find in the literature. The fourth column contains the solutions found by our cooperative parallel algorithm and the fifth column contains the solutions found by the parallel algorithm using no cooperation among the nodes (an entry in bold indicates a new best-known solution and an entry in italics indicates a solution equal to the current best-known solution). The sixth column gives the lowest value of the LP relaxation solution found by the set covering solvers. The LP relaxation cannot be used to provide a lower bound for the solution to the VRP as we solve it using only a subset of columns without considering the reduced cost. However, we found it interesting that the gap between the best VRP solution and the lowest LP relaxation was smaller for the more structured geometric problems, particularly the large benchmark problems of [51]. For 10 of these 12 problems, the lowest LP relaxation was equal to the best solution that our algorithm found. We do not have an explanation for this observation.

The cooperative version of our algorithm finds a better solution than the non-cooperative version for 44 of the 55 problems. On average, the solutions are 0.10% better with the largest gap being 0.84% (problem 12 from [36]). The non-cooperative version produces a better solution than the cooperative algorithm for problems 2 and 7 from [36]. For both problems, the non-cooperative version finds a new best-known solution, while our cooperative parallel algorithm finds the same solutions as several serial algorithms. These are probably very good solutions from which it is very difficult to escape. Because the non-cooperative version of our parallel algorithm solves the problem from scratch many times using different initial solutions, sometimes it is able to find parts of the solution space that have not been discovered by other VRP algorithms.

4.4.2 Analyzing the Effect of Parameters on Performance

In order to study the importance of certain parameters in our parallel algorithm, we ran our procedure with different settings on three benchmark problems: problem 15 from [36], problem 385 from [76], and problem 25 from [51] (we refer to these problems as G15, T385, and L25). These three problems are dissimilar in problem

Problem	Number of Nodes, Number of Routes	Previous Best-known Solution	Best Solution (With Cooperation)	Best Solution (Without Cooperation)	Minimum LP Relaxation
1	50, 5	524.61[a]	*524.61*	*524.61*	523.67
2	75,10	835.26[a]	*835.26*	*835.26*	819.82
3	100, 8	826.41[a]	*826.41*	*826.41*	807.32
4	150,12	1028.42[a]	*1028.42*	*1028.42*	1016.04
5	199,16	1291.29[b]	1291.45	1293.24	1286.52
6	50, 6	555.43[a]	*555.43*	*555.43*	539.80
7	75,11	909.68[a]	*909.68*	*909.68*	897.35
8	100, 9	865.94[a]	*865.94*	*865.94*	846.68
9	150,14	1162.55[a]	*1162.55*	*1162.55*	1145.97
10	199,18	1395.85[a]	*1395.85*	1399.93	1389.88
11	120, 7	1042.11[a]	*1042.11*	*1042.11*	1041.18
12	100,10	819.56[a]	*819.56*	*819.56*	816.57
13	120,11	1541.14[a]	*1541.14*	1541.25	1491.06
14	100,11	866.37[a]	*866.37*	*866.37*	846.52

[a]Rochat and Taillard [72]; [b]Mester and Bräysy [54]

Table 4.1: Solutions to the problems of Christofides et al. [19, 20]

Problem	Number of Nodes, Number of Routes	Previous Best-known Solution	Best Solution (With Cooperation)	Best Solution (Without Cooperation)	Minimum LP Relaxation
100A	100,11	2041.34[a]	*2041.34*	*2041.34*	1993.48
100B	100,11	1939.90[a]	*1939.90*	*1939.90*	1896.04
100C	100,11	1406.20[a]	*1406.20*	*1406.20*	1340.42
100D	100,11	1580.46[b]	*1580.46*	*1580.46*	1572.62
150A	150,15	3055.23[a]	*3055.23*	*3055.23*	3027.72
150B	150,14	2727.20[b]	*2727.20*	2728.32	2716.34
150C	150,15	2341.84[a]	2358.66	2358.92	2298.81
150D	150,14	2645.40[a]	*2645.40*	2646.10	2594.36
385	385,47	24369.13[b]	**24366.69**	24462.71	24296.64

[a]Mester and Bräysy [54]; [b]Nagata and Bräysy [58]

Table 4.2: Solutions to the problems of Taillard [76]

size, spatial construction, and constraints. They also appear to be *difficult* in the sense that there is considerable variation in the best solutions reported in the literature.

We ran our algorithm 10 times on each problem, varying the following parameters:

Problem	Number of Nodes, Number of Routes	Previous Best-known Solution	Best Solution (With Cooperation)	Best Solution (Without Cooperation)	Minimum LP Relaxation
1	240, 9	5627.54[a]	**5623.47**	5628.95	5608.52
2	320,10	8447.92[b]	*8447.92*	**8435.00**	8372.73
3	400,10	11036.22[b]	*11036.22*	11037.42	11036.22
4	480,10	13624.52[b]	*13624.52*	13625.72	13624.52
5	200, 5	6460.98[b]	*6460.98*	*6460.98*	6460.98
6	280, 7	8412.80[b]	8412.90	8412.90	8412.90
7	360, 9	10181.75[c]	10195.59	10195.59	10183.30
8	440,10	11663.55[a]	*11663.55*	**11649.89**	11662.31
9	255,14	580.02[d]	**579.71**	582.30	577.68
10	323,16	738.44[d]	**737.28**	740.24	736.29
11	399,18	914.03[d]	**913.35**	918.63	912.47
12	483,19	1104.84[d]	**1102.76**	1112.02	1102.17
13	252,26	857.19[d]	*857.19*	859.44	851.00
14	320,30	1080.55[d]	*1080.55*	1083.43	1076.91
15	396,33	1340.24[d]	**1338.19**	1342.89	1335.83
16	480,37	1616.33[d]	**1613.66**	1622.74	1611.38
17	240,22	707.76[d]	*707.76*	707.96	702.79
18	300,27	995.13[d]	*995.13*	1000.27	993.65
19	360,33	1365.99[d]	**1365.60**	1369.39	1359.57
20	420,38	1819.99[d]	**1818.25**	1826.47	1815.55

[b]Mester and Bräysy [54]; [b]Prins [64]; [c]Pisinger and Røpke [62]; [d]Nagata and Bräysy [58]

Table 4.3: Solutions to the problems of Golden et al. [36]

Problem	Number of Nodes, Number of Routes	Previous Best-known Solution	Best Solution (With Cooperation)	Best Solution (Without Cooperation)	Minimum LP Relaxation
21	560,10	16212.74[a]	16212.83	16214.03	16207.43
22	600,15	14597.18[a]	**14584.42**	14611.10	14553.28
23	640,10	18801.12[a]	18801.13	18802.33	18801.13
24	720,10	21389.33[a]	21389.43	21391.83	21389.43
25	760,19	16902.16[b]	**16763.72**	16835.12	16763.72
26	800,10	23971.74[a]	23977.73	23980.13	23977.73
27	840,20	17488.74[a]	**17433.69**	17528.58	17433.69
28	880,10	26565.92[a]	26566.03	26568.43	26566.03
29	960,10	29154.34[c]	*29154.34*	29155.54	29154.34
30	1040,10	31742.51[a]	31742.64	31745.04	31742.64
31	1120,10	34330.84[a]	34330.94	34333.34	34330.94
32	1200,11	36919.24[c]	37185.55	37256.25	37185.55

[a]Mester and Bräysy [54]; [b]Pisinger and Røpke [62]; [c]Estimated solution from Li et al. [51]

Table 4.4: Solutions to the problems of Li et al. [51]

- Number of processors (8, 16, 32, or 64);
- Number of set covering solvers (up to 8);
- Total time allowed (200 or 400 seconds);
- Strategy used by the master to distribute solutions (randomly selected solutions, best solution).

We summarize the results of this experiment in Table 4.5 (the complete results are given in Appendix B). Each row in Table 4.5 corresponds to a particular parameter setting, and each entry in the table contains two values. The first value is the overall average solution. The second value is the average best solution using the results of all runs with a particular parameter setting. For example, the first row shows the overall average and average best solution values found when using 8 processors and varying the remaining parameters over their possible settings.

In the first four rows of Table 4.5, we see that, for all three problems, using more processors usually leads to better solutions, both in terms of the overall average solution and the average best solution. The next two rows indicate that while doubling the computing time from 200 to 400 seconds leads to better solutions, the difference is very slight as the average improvement is less than 0.1%.

The strategy used by the master in distributing solutions appears to have little effect on the average solution value and the average best solution value. For all three problems, the average solutions generated by the two strategies are within 0.04% of each other. The average best solutions of the two strategies differ by at most 0.05%. It appears that there is little to be gained by trying to inject some diversification into the procedure by having the master distribute randomly selected solutions, since the strategy of sending the best solution appears to work equally well.

In the last five rows of Table 4.5, we present the results when the algorithm is run with 64 processors and the number of nodes assigned to solving the set covering problem is varied. The results suggest that using more set covering solvers leads to better solutions. In particular, the configuration with 8 set covering solvers generates the best results for all three problems based on average solution value and average best solution value.

4.4.3 Measuring the Parallel Speedup

In our final experiment, we attempted to measure the parallel speedup of the algorithm. The parallel speedup is defined to be the ratio t_s/t_p, where t_s is the

	G15		T385		L25	
Parameter Setting	Overall Average	Average Best	Overall Average	Average Best	Overall Average	Average Best
8 processors	1344.8	1342.7	24443.3	24406.6	16966.2	16864.8
16 processors	1344.0	1341.8	24440.2	24400.2	16950.0	16821.9
32 processors	1342.8	1340.7	24425.4	24390.3	16889.6	16800.3
64 processors	1342.6	1340.4	24415.0	24383.1	16839.8	16790.1
200 seconds	1343.9	1341.8	24434.3	24396.3	16920.1	16822.4
400 seconds	1343.0	1340.8	24424.6	24391.4	16879.9	16801.8
Random solution	1343.6	1341.4	24431.6	24394.8	16905.8	16808.7
Best solution	1343.3	1341.2	24427.3	24392.9	16894.2	16815.4
64 processors, 0 IP solvers	1343.4	1341.4	24424.4	24386.8	16847.8	16794.1
64 processors, 1 IP solver	1343.0	1341.3	24415.8	24381.6	16836.0	16797.8
64 processors, 2 IP solvers	1342.2	1339.9	24411.7	24385.7	16828.3	16788.5
64 processors, 4 IP solvers	1341.7	1339.7	24408.1	24378.5	16847.1	16780.0
64 processors 8 IP solvers	1341.6	1339.7	24401.5	24372.4	16825.4	16773.7

Table 4.5: Summary of the effect of various parameters

amount of time required by a single sequential computer and t_p represents the amount of time required for the parallel computation when using p processors. For many computational tasks, this quantity is easy to measure and a linear parallel speedup of p is typically the goal. However, in our experiments, this measure cannot be applied as easily since we typically obtain different final solutions from different runs of the algorithm. Therefore, in our study of the parallel speedup, we fix a total amount of computing time (i.e., the total wallclock time multiplied by the number of processors), and then compare the average solution value and best solution value when using a different number of processors and a fixed amount of computing time.

In this experiment, we again ran our cooperative parallel algorithm on three problems (G15, T385, and L25). For each problem, we ran the algorithm 20 times with four different configurations: 8 processors, 1 set covering solver, 800 seconds; 16 processors, 2 set covering solvers, 400 seconds; 32 processors, 4 set covering solvers, 200 seconds; 64 processors, 8 set covering solvers, 100 seconds. In each case, we have a total computing time of 6400 seconds, and we use one set covering solver for every eight heuristic solvers.

Problem	Number of Processors	Total Time (seconds)	Average Solution	Best Solution
	8	800	1342.9	1340.0
G15	16	400	1342.2	1339.4
	32	200	1342.0	1339.3
	64	100	1342.4	1338.8
	8	800	24428.4	24384.5
T385	16	400	24408.6	24371.2
	32	200	24409.1	24371.2
	64	100	24405.8	24370.6
	8	800	16940.0	16809.3
L25	16	400	16906.5	16782.0
	32	200	16905.2	16797.2
	64	100	16920.4	16796.9

Table 4.6: Analyzing the parallel speedup for three problems

The results of this experiment are summarized in Table 4.6. For all three problems, we generate solutions of roughly the same quality when using the configurations with 16, 32, or 64 processors, while the solutions generated with 8 processors for 800 seconds are generally the worst. The configuration with 64 processors finds the best overall solution to two problems, while the configuration with 16 processors finds the best solution to the largest problem, L25. These results suggest that the parallel speedup of our algorithm is quite satisfactory as we are able to obtain solutions of equivalent (or better) quality by cutting the computation time in half and doubling the number of processors. We analyze the parallel speedup in more detail by studying the solution trajectories under the different configurations. For all 20 runs on the three problems, we had the master record the best solution found every five seconds. We then calculated the average solution value at each time interval for each of the configurations. The average solution trajectories are given in Figures 4.7, 4.8, and 4.9. In each figure, there are two key observations. A steep slope indicates more rapid convergence towards a minimum. The final ending point of the trajectory represents the average final solution that was generated in the allotted time limit. These plots clearly show that, by using more processors, our algorithm finds better solutions more quickly. In addition, it is interesting to note that when running the algorithm with 8 processors, we typically reach a point of diminishing returns with respect to time where the search trajectory flattens out. In contrast, for the 64 processor runs of 100 seconds, we observe that the solution value decreases almost linearly with time.

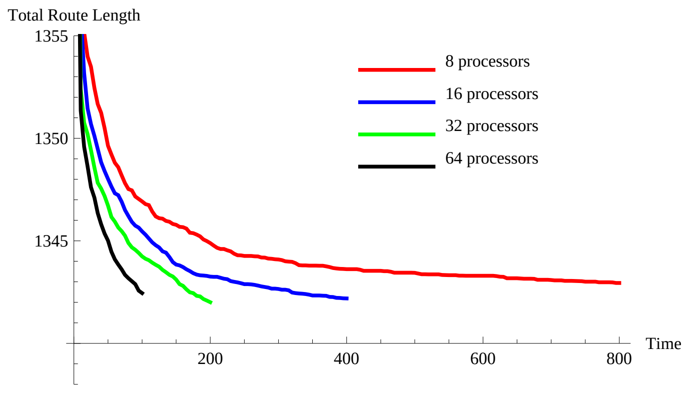


Figure 4.7: Average solution trajectories for problem G15

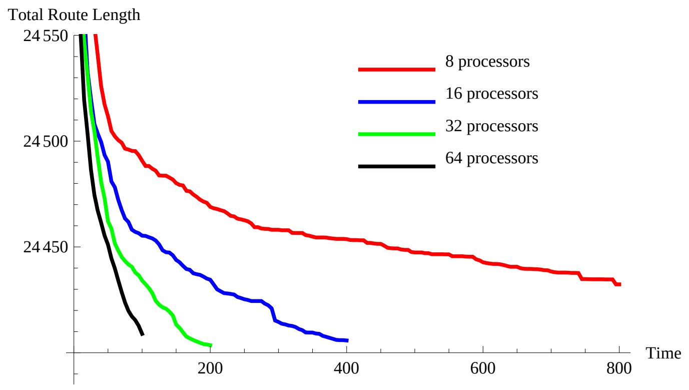


Figure 4.8: Average solution trajectories for problem T385

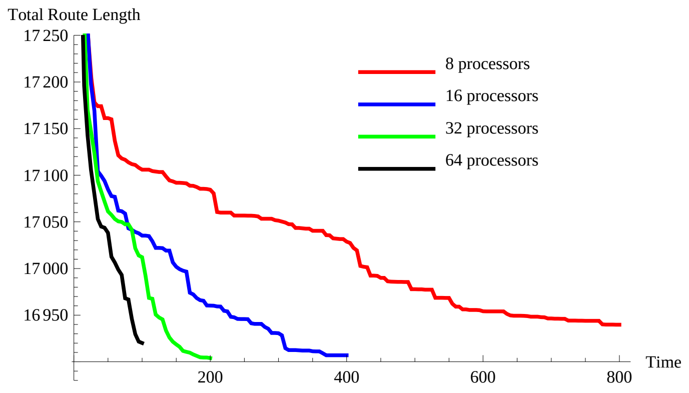


Figure 4.9: Average solution trajectories for problem L25

Finally, we note that if we continue to double the number of processors and halve the amount of wallclock time allowed, we will eventually reach a point where the solutions will begin to worsen. This is due to the fact that the heuristic solvers generally require at least five seconds per run. For example, if we were to distribute our 6400 seconds of computing time among 1024 processors and use only 6.25 seconds of wallclock time, then each heuristic solver would solve the problem at most once, and the set covering solvers would probably not have enough time to solve a single problem to optimality. Thus, we have very little (if any) cooperation among the processors in this case and the solution quality would almost certainly be worse.

4.5. Conclusion

We developed a parallel algorithm that combines heuristic and exact methods for solving the vehicle routing problem, and implemented it in a modern, high-performance computing environment. Our algorithm found high-quality solutions and produced new, best-known solutions to 15 benchmark problems. We analyzed the effects of the algorithm's cooperative nature by comparing its performance with a non-cooperative version. We assessed the impact of various parameters by running the algorithm many times on several different types of problems. We found that our algorithm scales quite well in terms of its parallel speedup.

It is worthwhile to mention some important lessons we learned when developing, implementing, and testing our procedure. One key to developing a successful implementation of a parallel algorithm is to minimize the time spent sending and receiving data amongst the processors. In an early implementation of our parallel algorithm, we did not have the heuristic solvers write their solutions to files. Instead, each heuristic solver sent a large number of solutions to the master at the end of each run of the RRTR algorithm, and the set covering solvers would then obtain routes by sending a request to the master and then receiving a large set of routes. This implementation worked reasonably well with a single set covering solver and 10 to 20 heuristic solvers. However, when we ran this implementation with more processors, the master quickly became overwhelmed and the other processors experienced substantial delays of several seconds. We developed the file-based approach in order to prevent these delays as this allows the set covering solvers to acquire new routes without direct involvement from the master. However, implementing this file-based approach required substantial effort as we had to handle the subtleties and speeds of two different types of file systems when moving our algorithm from the Linux cluster to the Windows-based cluster.

A second issue we encountered when developing the algorithm was the heterogeneity of the computing environment. We conducted a large portion of the development and testing on the Linux-based Deepthought cluster at the University of Maryland. This cluster has nodes that use different computing hardware, and different sets of nodes are connected by different types of networking hardware. We were not aware of this when testing and timing various parts of our procedure using a small number of processors. When performance and solution quality decreased, we did not consider that it was due to a hardware change and instead tried to find algorithmic and parameter changes that could be responsible for the decreases. Although we eventually discovered that the majority of these fluctuations in performance were due to hardware changes, we spent a considerable amount of time and effort looking elsewhere for the sources of the variations.

The parallel structure of the algorithm and the use of the set covering solvers could be extended to handle richer variants of the VRP, simply by replacing our RRTR algorithm with one capable of handling additional constraints. In future work, we would like to study the set covering problem in more detail. In particular, we would like to explore more sophisticated strategies for removing and adding columns, such as considering the reduced cost obtained by solving the LP relaxation. This may provide insight into the sizes of the gaps that we observed between the best integer solution and the solution to the LP relaxation for different problems.

Chapter 5

The VRPH Library

The vehicle routing problem (VRP) is a difficult and well-studied combinatorial optimization problem. Real-world instances of the VRP can contain hundreds and even thousands of nodes, requiring the use of heuristic methods. We present a software library of heuristics for solving the VRP. Our code uses efficient data structures to store and modify solutions, and contains implementations of several techniques for generating initial feasible solutions. Our library includes seven local search operators that can be used to develop very good solutions to large instances of the VRP. The code is well-documented, has been compiled on several different platforms, and has been designed so that it can be easily extended to handle additional constraints.

5.1. Introduction

In the standard form of the VRP, a minimum cost set of routes is constructed for a fleet of identical vehicles. These routes must satisfy the demands of all customers, and the vehicles traversing these routes have a fixed capacity. Computational experience indicates that the VRP is difficult to solve to optimality. Typically, exact methods are unable to handle problems with more than about 100 nodes. Thus, most real-world vehicle routing problems are solved using heuristic methods.

We are aware of only one freely available, non-commercial software package for solving the VRP. This is the VRP code of Ralphs et al. [66, 67] that is included with the open source mixed integer programming package, *Symphony*. In contrast, there are several open source software packages for generating solutions to the traveling salesman problem (TSP). The freely available and widely used *Concorde* package is an exact solver for the TSP that has been used to produce optimal solutions for problems with nearly 100,000 nodes [5, 8]. The C source code for *Concorde* was developed by a team of seven authors and can be downloaded for academic research. *Concorde* also provides an interactive graphical interface for Windows. The code contains heuristic tour construction techniques and implements fast, specialized linear and integer programming algorithms in order to find provably optimal solutions.

A second code for solving the TSP is the Lin-Kernighan-Helsgaun (LKH) package developed by Helsgaun [39]. The author provides an optimized implementation of the original Lin-Kernighan heuristic [52] and enhances it in several ways. Although the LKH code does not implement exact methods, it has generated optimal tours to problems with up to 85,000 nodes. LKH is responsible for the current best-known solution to the World TSP with nearly two million nodes. The LKH code is freely available for academic purposes. Users can obtain the source code as well as a precompiled Windows executable from Helsgaun's website [39].

In this paper, we present an open source software library of heuristics for solving the VRP. Our library of Vehicle Routing Problem Heuristics (VRPH) is written in C/C++ and uses efficient data structures to implement methods for constructing an initial feasible solution and several local search heuristics that can be used to improve solutions. Our library provides a solver based on a metaheuristic algorithm that can be used to quickly generate high-quality solutions. The library's design is flexible enough to incorporate new constraints and implement different solution algorithms with minimal effort.

We describe the main ideas behind the library's design, summarize the most important functions, and describe ways that the library can be extended to handle variants of the standard VRP. This paper is organized as follows. We begin by describing a standard file format for VRP instances. Next, we show how VRPH represents solutions to a VRP, describe the most important data structures, and discuss various functions that are implemented in the library. We conclude by describing our metaheuristic solver. In Appendix C, we provide instructions for compiling VRPH and describe the interfaces between VRPH and the open source PLPlot graphics library and the commercial mathematical programming software CPLEX.

5.2. Input File Format

The TSPLIB format [70] is a widely-used file format for representing TSP instances that offers several options for representing the VRP. The file format used by VRPH adheres closely to the TSPLIB format, and we add a few additional options specifically designed for the VRP.

In Table 5.1, we give the supported keywords and their meanings. For each keyword, we indicate whether or not the keyword is mandatory (the problem will not load successfully without these mandatory fields). The case-sensitive keywords can be included in any order.

Field Name	Mandatory	Description
NAME	Yes	A string describing the problem
TYPE	Yes	Must be *CVRP* to denote capacitated VRP
BEST_KNOWN	No	Floating point number
DIMENSION	Yes	The total number nodes including the depot
CAPACITY	Yes	Vehicle capacity
DISTANCE	No	Maximum route length - defaults to *VRP_INFINITY* if not given
SERVICE_TIME	No	The identical service time at each node (defaults to 0 if not given)
EDGE_WEIGHT_TYPE	Yes	Either *EXPLICIT* or *FUNCTION*. This describes how to determine the inter-node distances
EDGE_WEIGHT_FORMAT	No	Describes the distance function used if *EDGE_WEIGHT_TYPE* is *FUNCTION*
NODE_COORD_SECTION	Yes	Header that begins the section containing the node coordinates
DEMAND_SECTION	Yes	Header that begins the section containing the demands
DEPOT_SECTION	Yes	Header that begins the section containing the coordinates of the depot. This section must end with a “-1”
SVC_TIME_SECTION	No	Header that begins an optional section containing the service time at each non-depot location
TIME_WINDOW_SECTION	No	Header that begins an optional section containing the time window at each non-depot location
EOF	Yes	Indicates the end of the file

Table 5.1: Keywords for the VRPH input file

Most of the fields in Table 5.1 are self-explanatory. The 10-node example file in Figure 5.1 illustrates the most commonly used options.

```
NAME: Example-10
TYPE: CVRP
DIMENSION: 11
CAPACITY: 30
EDGE_WEIGHT_TYPE: FUNCTION
EDGE_WEIGHT_FORMAT: EUC_2D
NODE_COORD_TYPE: TWOD_COORDS
NODE_COORD_SECTION
1 37.00000 52.00000
2 49.00000 49.00000
...
10 51.00000 21.00000
DEMAND_SECTION
1 7
2 30
...
10 5
DEPOT_SECTION
30.00000 40.00000
-1
EOF
```

Figure 5.1: An example of a properly formatted VRPH input file

There are several options not used in this example that are worth mentioning. First, VRPH supports five of the different distance functions described in the TSPLIB documentation [70]. Given two points with coordinates, (x_1, y_1) and (x_2, y_2), these distance functions are specified as follows:

- Euclidean distance (EUC_2D): $\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$;
- Geographical distances (GEO) - node locations are latitude and longitude coordinates, and distances are given in kilometers ;
- Manhattan distance (MAN_2D): $|x_1 - x_2| + |y_1 - y_2|$;
- Maximum distance (MAX_2D): $\max(|x_1 - x_2|, |y_1 - y_2|)$;
- Rounded Euclidean distance (CEIL_2D): $\lceil \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} \rceil$.

These distance functions produce a symmetric distance matrix. The user can specify an explicit distance matrix by adding the line EDGE_WEIGHT_TYPE: EXPLICIT and by then including either a full matrix or half of a symmetric matrix following the line EDGE_WEIGHT_SECTION. The files distributed with VRPH include two examples of this type of problem, *VRPH/data/misc/explicit_full.vrp* and *VRPH/data/misc/explicit_upper_row.vrp* (these directory paths are explained fully in Appendix C). Finally, VRPH can handle problems where the planning horizon has multiple days. This is done by adding an additional line NUM_DAYS: D for a D-day problem, and by then providing D days worth of demand and service times in the appropriate sections. The file *VRPH/data/misc/multi-day.vrp* gives an example of this type of problem.

5.3. Solution Representation

We now describe the way that our code stores solutions. Given a feasible solution to a VRP instance, VRPH attempts to improve it by performing local search, moving from one feasible solution to another. These operations can occur millions of times when running a metaheuristic algorithm, so it is necessary to have an efficient and compact method of storing solutions.

VRPH keeps track of the current solution information as it is modified throughout the improvement process via the *VRP* class. This class contains the information about a VRP instance and contains a number of data structures and methods that can be used to access problem information, modify the solution, and output the solution to buffers and files.

For an n-node problem, the current solution at any stage in the solution procedure is stored in a doubly linked list contained in two arrays of length $(n+1)$: *next_array* and *pred_array*. We use the method suggested in Kytöjoki et al. [45] where negative indices in these arrays indicate the beginning of a new route. For example, consider a 10-node VRP with the depot labeled as 0, the 10 customer nodes numbered $1, 2, \ldots, 10$, and the three routes given in Table 5.2. This solution is stored in the *next_array* and *pred_array* as shown in Table 5.3. Along with these two arrays, the *VRP* class keeps track of several properties of the current solution such as the total route length and the number of routes.

Route	Ordering
1	0-2-5-8-0
2	0-1-3-4-0
3	0-6-7-9-10-0

Table 5.2: A 10-node VRP example

i	0	1	2	3	4	5	6	7	8	9	10
next_array$[i]$	-2	3	5	4	-6	8	7	9	-1	10	0
pred_array$[i]$	-10	-8	0	1	3	2	-4	6	5	7	9

Table 5.3: The *next_array* and *pred_array* corresponding to the solution in Table 5.2

In any VRP solution, each customer is assigned to a single route (we do not allow for split deliveries). VRPH stores this information in the *route_num* array where *route_num*$[i] = j$ indicates that node i is assigned to route j. In our 10-node example, we have *route_num*$[2] = 1$, *route_num*$[10] = 3$, and so on.

The *VRP* class maintains an array of *VRPRoute* objects. Each *VRPRoute* represents a single route and contains the following information:

- *start*: First non-depot node in the route;
- *end*: Last non-depot node in the route;
- *length*: Total length of the route (including any service time);
- *load*: Total load carried by the vehicle on this route (this is equal to the total demand of the customers on the route);
- *num_customers*: Number of customers or nodes on the route;
- *total_service_time*: Total service time required by all customers visited on this route.

As an example of how to access this information, suppose we would like to know the start node and the length of route number 2 in the 10-node example given in Table 5.2. Assuming that we are using a *VRP* object called V, then we access this information by referencing $V.route[2].start$ and $V.route[2].length$.

An important feature of VRPH is that all information about the solution and the individual routes is automatically updated whenever the solution is modified by any of the other routines contained in VRPH. For example, VRPH is capable of performing a two-opt move that removes two edges from a solution and replaces them with two new edges (more on this later). When a two-opt move is performed, the ordering is updated in the *next_array* and *pred_array*, and all information about the solution is automatically updated to reflect the changes resulting from this move. This updating is completely transparent to the caller of the function, allowing the user to ignore the rather tedious bookkeeping.

5.4. Creating an Initial Solution

After loading a properly formatted file, VRPH can create an initial feasible solution with two well-known procedures: the Clarke-Wright algorithm and the sweep algorithm.

Given an n-node problem, the Clarke-Wright algorithm starts with n routes where each vehicle visits a single node and returns to the depot. These routes are then merged together in an order determined by the *savings* criterion. Representing the depot with index 0, we let $d_{i,j}$ represent the distance between nodes i and j. The initial solution has a total route length of $\sum_{i=1}^{n} 2d_{0,i}$. For each pair of locations (i, j), we define $s_{i,j}$ to be

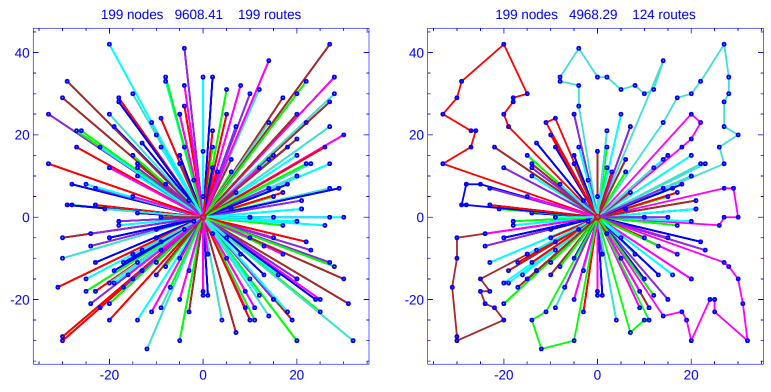
$$s_{i,j} = (2d_{0,i} + 2d_{0,j}) - (d_{0,i} + d_{i,j} + d_{0,j}) = d_{0,i} - d_{i,j} + d_{0,j},$$

which gives the savings (or reduction) created by merging the two initial routes servicing nodes i and j. The Clarke-Wright algorithm uses these savings values to determine the order in which the original routes are merged together to form larger routes.

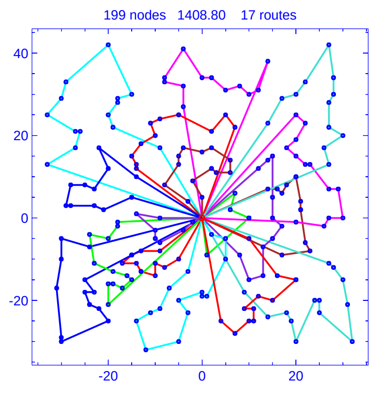
In VRPH, we implement a variant of the Clarke-Wright algorithm [84] that incorporates a shape parameter λ into the savings, $\overline{s}_{i,j} = d_{0,i} - \lambda d_{i,j} + d_{0,j}$ (see Algorithm 5.1). In Figure 5.2, we show the solution at various stages of the algorithm.

Algorithm 5.1 The Clarke-Wright algorithm

```
1:  for all $(i, j)$ with $0 < i < j \leq n$ do calculate $\overline{s}_{i,j} = d_{0,i} + \lambda d_{i,j} + d_{0,j}$
2:  end for
3:  Create a list $L$ of the triples $(i, j, \overline{s}_{i,j})$ and sort $L$ in descending order by the value
    of $\overline{s}_{i,j}$.
4:  while $L \neq \emptyset$ do
5:      Select the top entry from the list $L$, $(i, j, \overline{s}_{i,j})$
6:      if Nodes $i$ and $j$ are currently adjacent to the depot and are in different routes
    then
7:          if The route obtained by merging these two routes is feasible then
8:              Merge these two routes together by adding the edge linking $i$ and $j$
9:          end if
10:     end if
11:     Remove the triple $(i, j, \overline{s}_{i,j})$ from $L$.
12: end while
```



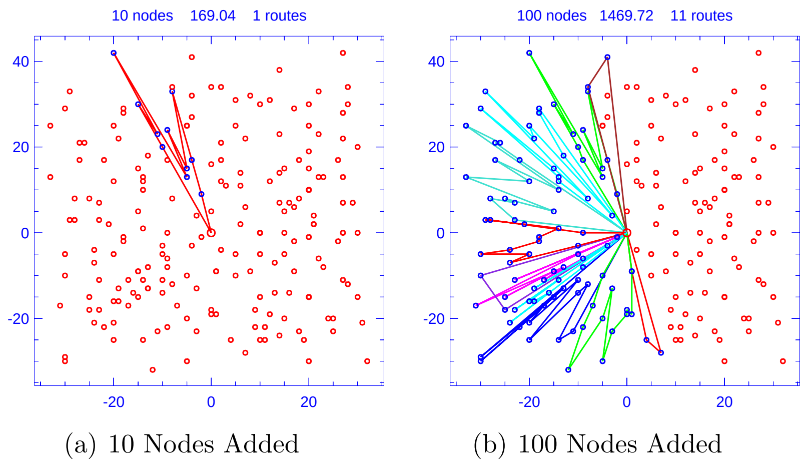
(a) Initial Solution (b) After 100 Merges



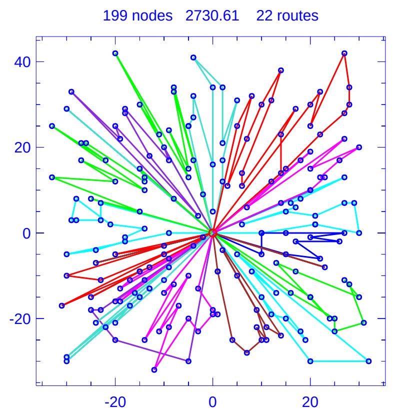
(c) Final Clarke-Wright Solution

Figure 5.2: Progress of the Clarke-Wright algorithm

To construct an initial solution using the sweep algorithm, we select a random *seed* customer and draw a ray connecting this node to the depot. This seed customer is the first node on the first route. We then sweep this ray around the depot until the ray touches another node which we then add to the current route. When the route is about to become infeasible due to a violation of a capacity constraint or a route length constraint, we start a new route and repeat the process until all customers are routed. In Figure 5.3, we show the routes generated by the sweep algorithm.



(a) 10 Nodes Added (b) 100 Nodes Added



(c) Final Solution

Figure 5.3: Progress of the sweep algorithm

We enhance both the Clarke-Wright and sweep algorithms by allowing the user to specify different heuristic operations that attempt to improve the solution during its construction. To do so, the user selects a set of local search operators (discussed in detail in the next section) and an improvement interval, k. For the Clarke-Wright algorithm, we apply the selected local search operators after every k-th merging of two routes. When running the sweep algorithm, we apply the operators after adding every k-th node to the solution.

5.5. Solution Modification

VRPH contains different routines that allow the user to modify an existing VRP solution. When any of these routines is called, the relevant fields in the solution storage arrays are updated to reflect the change in the solution. Thus, the user should seldom (if ever) have any need to directly modify a field such as a route's *length* or *load*. The core of VRPH is its flexible implementation of seven different heuristic local search operators described in the next section.

5.5.1 Local Search Operators

VRPH generates high-quality solutions to vehicle routing problems by applying seven different local search operators. We describe the general idea behind local search, discuss each of the operators, and describe how the local search is implemented and how it can be extended.

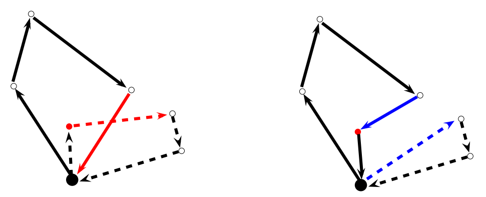
Local search is a commonly used technique in combinatorial optimization. We are given a general minimization problem,

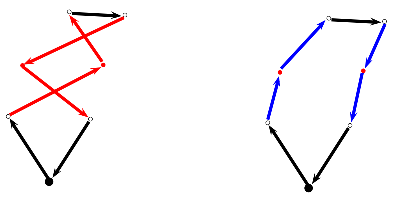
$$\min f(x)$$
$$\text{subject to } x \in \mathcal{S},$$

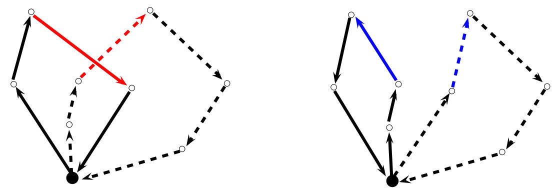
where $\mathcal{S}$ is a discrete solution space and $f : \mathcal{S} \mapsto \mathbb{R}$ is the objective function. Given a feasible solution $x \in \mathcal{S}$, we construct a neighborhood of feasible solutions, $N(x) \subset \mathcal{S}$, and then search this neighborhood for a new solution $x' \in N(x)$. When selecting x', we may require $f(x') < f(x)$ if we wish to consider only improving moves, or we may use a probabilistic acceptance strategy such as simulated annealing [44] where we may accept the new solution when $f(x') > f(x)$.

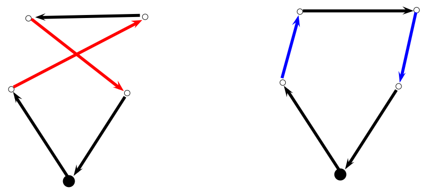
For the VRP, these neighborhoods are typically defined in terms of a heuristic operation where a node is moved to a new position in the solution, or several edges are removed and replaced with new edges. In other words, given a solution and a heuristic operation, the neighborhood consists of all solutions that can be created by applying the heuristic operation to the current solution. Local search has proven to be very effective for the VRP and this technique is used in many VRP metaheuristics.

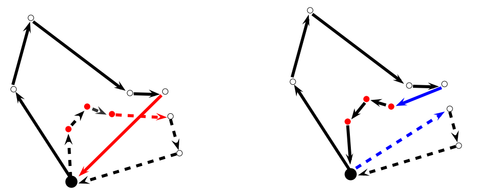
The seven different local search operators are shown in Figure 5.4. For each operator, we provide a diagram illustrating the change to the solution and a brief description. Each heuristic operator is implemented as a separate class within VRPH and is designed to search for and perform solution modifications that meet a set of user-defined *criteria* that can incorporate different considerations. For example, these

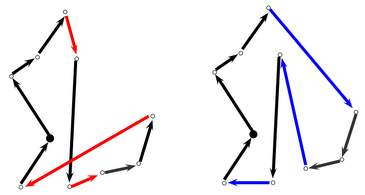

(a) **One-Point move**: Relocate an existing node into a new position.

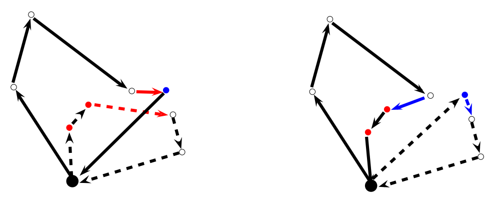

(b) **Two-Point Move**: Swap the position of two nodes.


(c) **Inter-route Two-Opt Move**: Remove one edge from two different routes and replace them with two new edges.


(d) **Intra-route Two-Opt Move**: Remove two edges from a single route and replace them with two new edges.


(e) **Or-Opt Move**: Remove a string of two, three, or four nodes and insert the string into a new position.


(f) **Three-Opt Move**: Remove three edges from a route and replace them with three new edges.


(g) **Three-Point Move**: Swap the position of a pair of adjacent nodes with the position of a third node.

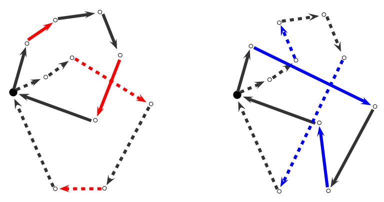

(h) **Cross-Exchange Move**: Remove four edges from two different routes and replace them with four new edges.

Figure 5.4: Local search operators in VRPH

criteria can specify that a particular heuristic improvement strategy such as simulated annealing or record-to-record travel will be used in the local search, they can refine the neighborhood that is searched by insisting on only inter- or intra-route moves, and they can specify that certain edges remain fixed or unchanged when considering solution modifications.

We represent these *criteria* as a 32-bit integer, and we reserve a single bit for each option. We list the most commonly used options in Table 5.4.

Option	Meaning
DOWNHILL	Accept only those moves that decrease the total route length.
RECORD_TO_RECORD	Accept moves according to the record-to-record acceptance strategy [29].
SIMULATED_ANNEALING	Accept moves according to the simulated annealing metaheuristic [44].
FIRST_ACCEPT	Make the first solution modification that is found that meets the other specifications given in the current *criteria*.
BEST_ACCEPT	Evaluate all moves in the neighborhood and make the move that leads to a feasible solution with minimum total route length (this is the *best* move).
LI_ACCEPT	If any improving move is found, make this move. If no improving move is found, then make the move that increases the total route length by the smallest amount (a similar acceptance strategy is used in [51]).
INTER_ROUTE_ONLY	Search for moves that involve the modification of more than one route.
INTRA_ROUTE_ONLY	Search for moves that involve only a single route.
USE_NEIGHBOR_LIST	Limit the search to those moves that only involve a particular node's neighbor list.
FORWARD	Create an ordered solution buffer by concatenating all routes and search for moves that involve nodes that are found when moving forward in the solution buffer (see [54]).
BACKWARD	Create an ordered solution buffer by concatenating all the routes and search for moves that involve nodes that are found when moving backward in the solution buffer (see [54]).
RANDOMIZED	When examining the search neighborhood for moves, evaluate these moves in a random order.

(continued on next page)

Option	Meaning
SAVINGS_ONLY	When comparing two moves, evaluate them by considering only the savings or improvement offered by the moves.
MINIMIZE_NUM_ROUTES	When comparing two moves that involve more than a single route, attempt to minimize the number of routes by trying to maximize the sum of the squares of the number of nodes in the routes involved in the move.
FIXED_EDGES	Forbid those moves that disrupt any edges that are currently set as *fixed*.
TABU	Forbid those moves that result in routes that are *tabu* according to the solution's current memory.

Table 5.4: Different options allowed in the *criteria*

We now provide additional details about these heuristic operators and then give a few examples. Each heuristic operator is a single class in VRPH that implements at least three of the four different methods described below.

- *search*: Given a specific node and a valid *criteria*, construct the search neighborhood and find solution modifications that satisfy the given *criteria.* Modify the solution if one is found and return *true.* Return *false* if no new solution can be found.

- *route_search*: Given one or more route numbers and a *criteria*, construct the search neighborhood and search for acceptable solution modifications involving the routes. Modify the solution if one is found and return *false* otherwise.

- *evaluate*: Given a particular solution modification and a valid *criteria*, return *true* if the move is feasible and satisfies the criteria and *false* otherwise.

- *move*: Given an evaluated, feasible solution modification, modify all data structures and return *true*. Return *false* if an error or infeasibility is encountered.

In order to use a local search operator, we proceed in the following way. The user selects a particular local search operator, constructs a well-defined *criteria* by combining several options from Table 5.4, and chooses a particular node j or one or more routes (r_1 and r_2) that are to be involved in the eventual solution modification. The user then calls *search(criteria, j)* or *route_search(criteria, r_1, r_2)*. A neighborhood of solutions is constructed by considering the provided *criteria*, and each new solution in this neighborhood is evaluated. If an acceptable solution modification is found, then the *move* method is called and the solution is modified. Typically, the user only calls the *search* method which then internally calls the *evaluate* and *move* functions.

We illustrate the flexibility of this approach with a few examples. We assume that we have a properly instantiated *VRP* object, V, and that we have loaded a 100-node problem with a current solution containing 8 routes.

```
criteria = FIRST_ACCEPT+RECORD_TO_RECORD+RANDOMIZED+USE_NEIGHBOR_LIST;
two_opt.search(V,37,criteria);
```

In this example, we search the nearest neighbors of node 37 for a valid *two_opt* move. We construct the search neighborhood consisting of the nearest nodes (the actual number is an internal parameter that can be modified) and then randomly permute it before searching for moves. Since our *criteria* includes *FIRST_ACCEPT*, we make the first feasible move that results in a new total route length that is allowed when using the *RECORD_TO_RECORD* travel acceptance strategy.

```
criteria = FIXED+BEST_ACCEPT+DOWNHILL+INTRA_ROUTE_ONLY;
one_point_move.route_search(V,5,7,criteria);
```

In this example, we search for the best *one_point_move* involving routes five and seven. VRPH allows the user to fix certain edges to ensure that they remain in the routes. In this example, since *FIXED* is part of the *criteria*, any moves that remove fixed edges from the solution are forbidden. Since the *criteria* includes *BEST_ACCEPT*, we search all moves in the neighborhood before selecting the best one, and accept only those moves that reduce the current total route length since the *DOWNHILL* bit is set.

Finally, we discuss some of the details of how we implemented these local search operators to give the reader an idea as to how these implementations can be extended easily. Suppose we are applying two-opt to the current solution, and are evaluating a particular two-opt move in terms of a particular *criteria*. If the move results in a feasible solution, then we store all relevant changes to the current solution in a *VRPMove* data structure. This structure contains information such as the change in the total route length, the change in route loads, and the change in the number of customers per route. After populating this data structure, we call the function *check_move* which determines whether or not the *VRPMove* meets the *criteria* or not. As this function is called in the same manner by all seven local search operators, if the user wanted to try, say a variant of simulated annealing using a different probability function, then the user would need to incorporate this into the *criteria* and then add code to handle this evaluation in the *check_move* function. The fact that all seven operators are implemented in a very similar manner allows extensions to be accomplished easily by modifying a small amount of code.

5.5.2 Miscellaneous Operations

We now discuss some of the additional solution modification functions provided by VRPH. When solving a VRP with n customers, it may be useful to consider a smaller problem with fewer nodes. For example, in a multi-day VRP where we visit customers over a period of several days, we typically would not visit every customer on every day, so we would be faced with smaller problems on each day. Alternatively, it may be useful to restrict attention to a certain geographic cluster of nodes in order to focus more intently on a partial subproblem of a particular VRP instance.

The *VRP* class contains an array *routed*[] where *routed*[i] is true if customer i is in the current solution, and false otherwise. All heuristic operators discussed in the previous section handle partial problems and unrouted nodes by checking this array when evaluating proposed moves, returning false whenever a proposed move involves an unrouted node. VRPH offers several routines that allow the user to take a solution and eject and insert nodes and sets of nodes. The routines are described below and the file *VRPH/src/demos/ejection.cpp* contains examples of how these routines are used.

- *eject_node(j)*: Remove node j from the solution, replacing the old edges $i - j$ and $j - k$ with the new edge $i - k$.

- *eject_route(r)*: Remove all nodes contained in route r from the solution.

- *inject_node(j)*: Insert the unrouted node j into the solution so that the increase in total route length is minimal and all routes remain feasible.
- *inject_set*$(a[], k)$: Insert all k nodes from the array $a[]$ into the solution so that the total route length increase is minimal and so that the resulting routes are feasible. VRPH uses several different heuristic approaches to search for an efficient insertion of the set.
- *eject_neighborhood(j)*: Select a random set of nodes near j and remove them from the solution.

5.6. Solution Input and Output

VRPH allows the user to store solutions to a problem in buffers and to write solutions to files in a simple format using the commands *export_solution_buff* and *write_solution_file*. The formats of the solution are identical for these two commands. The first entry in the solution buffer or file is the number of non-depot nodes in the solution. Following this entry, we list the order in which customers are visited, using a negative index to indicate the first node in each route. Referring to the 10-node example in Table 5.2, the solution buffer and file have the form

```
10 -2 5 8 -1 3 4 -6 7 9 10 0.
```

The buffer or file produced by these functions can then be imported into a properly initialized VRP object by calling either *import_solution_buff* or *read_solution_file*. When these functions are called, the current solution stored by VRPH is discarded and all internal data structures are updated to reflect this new solution.

5.6.1 Solution Hashing

Along with providing several ways to export and import solutions, VRPH stores a large pool of the best s solutions (s is an internal parameter that can be varied). When a potentially new solution is encountered, we need to quickly determine whether or not the solution is already in the pool. We do this by using an h-bit hash table and associate an h-bit integer with each solution. We are unaware of any other uses of hashing in the VRP literature. Next, we describe the details of our procedure.

The idea behind our hash function is to transform the solution into a standard form, export the standardized solution into a buffer, and then associate an h-bit integer with this buffer. We are given an R-route solution to an n-node VRP. For each j from 1 to R, we represent the j-th route as an ordered list, r_j. In particular, we write $r_j = \{a_j, \ldots, z_j\}$ where a_j is the index of the first customer visited in route j and z_j is the index associated with the last customer visited in route j. Additionally, we store a list of n random 32-bit integers, $Y_0, Y_1, \ldots, Y_{n-1}$. Given this notation and this list Y, we associate a positive integer with each solution by performing the steps given in Algorithm 5.2.

Algorithm 5.2 Hashing a VRP Solution

```
1: for $j = 1$ to $R$ do
2:     if $a_j > z_j$ then Reverse the ordering of route $j$
3:     end if
4: end for
5: Create a list $L$ containing the resulting, possibly reversed routes, and sort this
   list in terms of the index of the first node visited in the route
6: With $L = \{r_1, r_2, \ldots, r_R\}$, concatenate these routes into a single list
   $\{m_1, m_2, \ldots, m_n\}$
7: Return $H = \bigoplus_{i=1}^{n-1} Y_{(m_i + m_{i+1}) \bmod n} \bmod 2^h$
```

In Step 2, we ensure that each route is oriented so that the index of the first node in each route is smaller than the index of the last node visited in each route and then concatenate these routes to form a single list in Steps 5 and 6. In Step 7, we associate a 32-bit integer $Y_{(m_i+m_{i+1}) \bmod n}$ with each pair of nodes in the ordered list and then sum them together (when computing this sum, we use the bitwise XOR operation rather than normal addition). We produce the hash value H by taking the low h bits.

We illustrate this procedure using the example given in Table 5.2. Since the index of the start node is less than the index of the end node in each of the three routes, no reversals are required, and so we sort these routes by the start node in Step 5 and concatenate them together to from the list $\{1, 3, 4, 2, 5, 8, 6, 7, 9, 10\}$. We then associate one of our 10 random numbers $Y[0], Y[1], \ldots, Y[9]$ with each pair of consecutive numbers in this list and XOR them together to compute an h-bit hash value:

$$H = (Y[4] \oplus Y[7] \oplus Y[6] \oplus Y[7] \oplus Y[3] \oplus Y[4] \oplus Y[3] \oplus Y[6] \oplus Y[9]) \bmod 2^h.$$

This provides a very fast method of associating an h-bit integer with a given solution. When a potentially new solution is encountered, VRPH hashes the solution, producing an integer H. Then we check the entry in position H in the hash table. If the location is empty, then we instantly know that the solution is not currently one of the s best solutions. If entry H in the hash table is not empty, then we perform additional checks to determine whether or not this is a new solution.

5.7. Using the Default Solver

Up to this point, we have described some of the most important aspects of the VRPH library. In this section, we briefly describe a metaheuristic algorithm that we have implemented using VRPH. This solver is produced when compiling VRPH (see Appendix A for instructions on compiling VRPH), and we explain the details of how to use this solver and set the values of various parameters.

We implement a variant of the fast and flexible record-to-record travel (RTR) algorithm presented in [51]. Typically, our implementation produce solutions to problems containing several hundred nodes in less than 20 seconds of computing time on a laptop or desktop machine. When this solver is run on benchmark problems, the solutions are generally within 3% of the best-known solutions.

This algorithm alternates between two phases, a diversification phase and an improvement phase. In the diversification phase, the goal is to accept some worsening moves and explore new parts of the solution space. This is followed by an improvement phase where only improving moves are allowed as we seek a local minimum in the search space. The algorithm terminates after it has been unable to escape a particular local minimum after several attempts. The algorithm requires the following parameters:

- D: Controls the size of the main loop in the diversification phase. A value between 25 and 50 is reasonable.
- δ: Controls how much deterioration is allowed in the diversification phase. Defaults to $\delta = .01$.
- K: Number of local minima that must be reached before perturbing the solution or terminating the algorithm.

- N: Size of the neighbor list that is searched when running the local search operators. A larger value will slow the algorithm down as there are more possibilities to consider.
- P: Number of times the solution is perturbed once the search is stuck in a local minimum.

Algorithm 5.3 An algorithm for the VRP based on record-to-record travel

```
1: Generate a random $\lambda \in (0.5, 2)$ and generate an initial feasible solution using
   Algorithm 5.1
2: Select a set of local search operators $\mathcal{U}$
3: Set the record $R$ equal to the current total route length, set the threshold $T =
   (1 + \delta)R$, and set $k = p = 0$
4: while $p < P$ do
5:     for $i = 1$ to $D$ do
6:         for all Operators $u \in \mathcal{U}$ do
7:             for all Nodes $j$ in the solution do
8:                 Apply operator $u$ to node $j$ using record-to-record travel
9:             end for
10:        end for
11:    end for
12:    while Improving moves can be found do
13:        for all Operators $u \in \mathcal{U}$ do
14:            for all Nodes $j$ in the solution do
15:                Apply operator $u$ to node $j$ accepting only improving moves
16:            end for
17:        end for
18:    end while
19:    if The current solution is a new record then
20:        Update $R$ and $T$ and set $k = 0$
21:    end if
22:    $k++$
23:    if $k == K$ then
24:        Perturb the solution
25:        $p++$
26:    end if
27: end while
28: Return the best solution found
```

After compiling VRPH, Algorithm 5.3 can be run by entering the command *RTR* from the *VRPH/bin* directory. Table 5.5 describes the options that can be used when running this solver.

Option	Possible Values	Description
-a	Integer in $\{0, 1\}$	Accept Type: Can be either *first accept* (0) or *best accept* (1). If first accept is chosen, then the first acceptable move is accepted when performing local search in the improvement phase. If best accept is selected, then the entire neighborhood is searched and the overall best move is selected.
-d	Real in $(0, 1)$	Deviation: Determines the amount of deterioration allowed when implementing the heuristic operations. Default is 0.01.
-D	Integer > 0	This determines the parameter D which is the number of times we run through the loop in the diversification phase of the algorithm. Default is $D = 30$.
-h	String	Heuristic: This option designates that a particular heuristic operation should be used in the search (can be repeated).
-K	Integer≥ 0	This determines the parameter K which is the limit on the number of times we fail to beat the current local minimum before stopping. Default is $K = 5$.
-L	Integer > 0	Lambdas: This is the number of different randomly generated λ values used when generating initial Clarke-Wright solutions. The default setting is to use three random λ's.
-N	Integer ≥ 0	This is the size of neighbor list, N, that is used when running the local search operators. Defaults to $N = 25$ and using the option -N 0 will not use neighbor lists at all and all nodes will be searched when running local search.
-out	String	Output File: The solution will be written to this file. Default is to write a file named *name_len* where *name* is the problem's name as determined from the input file, and *len* is the best total route length found.
-P	Integer ≥ 0	The parameter P which is the number of times we perturb the solution after reaching a local minimum that we cannot escape. Default is $P = 1$.
-sol	String	Starting solution: The algorithm will start with the solution contained in the provided file. The default mode does not import an initial solution and starts by generating an initial solution with the Clarke-Wright algorithm.

Table 5.5: Options for the RTR solver

There are many ways to run Algorithm 5.3. We provide an example to demonstrate some of the options.

```
RTR -f Christofides_10.vrp -D 50 -L 5 -N 30 -h ONE_POINT_MOVE
-h TWO_OPT -h THREE_OPT -out C10.sol
```

This command will run the RTR solver on the Christofides_10.vrp benchmark problem using 5 different initial solutions, three heuristic operations (One-Point Move, Two-Opt, and Three-Opt), a diversification loop of 50 iterations, neighbor lists of size 30, with the best solution written to the file C10.sol.

5.8. Conclusions

We developed a C/C++ library for solving the standard vehicle routing problem. Our library uses a compact, efficient method for storing solutions and provides a simple interface for importing problem data. It includes well-known techniques for constructing initial solutions as well as seven local search operators. These heuristic operators are implemented in a similar, logical way that allows for many extensions. Our software also provides a solver that is able to generate very good solutions in a short amount of computing time. We hope that this library will be useful to researchers studying the VRP and that it will provide a flexible tool for solving new problem variants.

Chapter 6

Conclusion

In this dissertation, we developed computational algorithms for solving two new real-world vehicle routing problems. The first problem is the Consistent VRP which was proposed to us by UPS. Here, we increase customer service quality by having the same driver visit each customer at roughly the same time each day. The second new variant is the Balanced Billing Cycle VRP, described to us by RouteSmart Technologies, Inc. Here, we are given an initial, potentially inefficient and unbalanced set of routes for each business day of the month, and we must construct a new set of routes for each day so that the individual routes are balanced, ensuring that we also respect various regulatory constraints.

For each of these problems, we developed an efficient computational algorithm to generate solutions. We tested our algorithms on both simulated benchmark problems as well as real-world data sets provided by contacts in industry. In both cases, our algorithms generated high quality solutions in a reasonable amount of computing time.

Next, we developed a very powerful parallel algorithm for solving the classical VRP. The parallel algorithm combines a set-covering formulation of the VRP with a metaheuristic algorithm and we implemented it in two different high-performance computing environments. Our algorithm produced new best-known solutions to fifteen benchmark problems from the literature that have been extensively studied for more than a decade. We studied the role of cooperation and information sharing among the different processors by conducting extensive computational experiments and varying the algorithm's parameters, and we demonstrated that the algorithm exhibits a very good parallel speedup.

The majority of the computational work presented in this dissertation was done with a single software library that we developed. The final contribution of this dissertation is a description of this C/C++ library that will be released to other academic researchers. The source code is well-documented and has been compiled and run on several different platforms. We provide both a fast executable solver as well as an application programming interface that allows others to use our code in a variety of ways. Our hope is that this library will provide an easy-to-use tool to continue research into using heuristic methods for solving this very difficult combinatorial optimization problem.

Appendix A

Additional Information Related to the Consistent VRP

In this Appendix, we provide additional information and data related to our work on the Consistent Vehicle Routing Problem (ConVRP) discussed in Chapter 2.

A.1. The redundancy of the sub-tour constraints

In this section, we prove that the sub-tour constraints are redundant in the integer programming formulation provided on page 11.

Lemma. *The subtour constraints* (2.10) *are redundant in our formulation of the ConVRP.*

Proof. Sub-tours are only relevant for a particular vehicle on a particular day, so fix k and d and consider a single vehicle on a single day. Then if $x_{ijkd} = 1$, then node j is visited after node i by the selected vehicle on the selected day. We want to show that the constraint

$$a_{ikd} + x_{ijkd}(s_{id} + t_{ij}) - (1 - x_{ijkd})T \leq a_{jkd} \text{ for all } i \geq 0,\ j \geq 1$$

guarantees that there are no sub-tours that do not contain the depot. Select some i and j and assume that the edge $i - j$ is in the solution so that $x_{ijkd} = 1$. Suppose that we actually do have a sub-tour where we return to node i after visiting node

j and then some arbitrary number of other locations. If our inter-node travel times and service times are strictly positive, then $a_j > a_i$ and so any node k that we visit after node j will have $a_k > a_j > a_i$, implying in particular that we cannot return to node i. We conclude that subtours are prevented by (2.11). □

A.2. Implementing the ConVRP Formulation in a Modeling Language

The GNU MathProg Language (GMPL) is an open source modeling language that is part of the GNU Linear Programming Kit (GLPK) [34]. GMPL is similar to the widely-used commercial modeling language, AMPL. The model file below describes the ConVRP formulation.

```
###############################################
###  GMPL: CVRP Formulation for Euclidean   ###
###        Instances                        ###
###############################################

##################
### PARAMETERS ###
##################
param N integer > 2;       # Number of non-depot nodes
set AllNodes := 0..N;
set Arcs := {i in AllNodes,
    j in AllNodes};
param V integer >=1;       # Number of vehicles
set Vehicles := 1..V;
param D integer >=1;       # Days
set Days := 1..D;
param L >=1;               # Maximum Service Time Differential
param T;                   # The maximum allowed per vehicle daily
                           #    total travel time
param Q;                   # The maximum load per vehicle
param s{AllNodes, Days};   # The service time at each node
param q{AllNodes, Days};   # The demand at each node
set Depot := {0};
set NonDepotNodes:= AllNodes diff Depot;
param w{AllNodes, Days};
set S := 1 .. (2**(N) - 1);
set Cutset {k in S} := {i in NonDepotNodes:
    (k div 2**(i-1)) mod 2 = 1 and i!=0};
param xcoord {AllNodes};
param ycoord {AllNodes};
```

```
# Define the travel time by Euclidean distance
param t {(i,j) in Arcs} := sqrt((xcoord[i] - xcoord[j])^2 +
(ycoord[i] - ycoord[j])^2);

##########################
### Decision variables ###
##########################
var x {i in AllNodes, j in AllNodes,k in Vehicles, d in Days} binary;
var y {i in AllNodes, k in Vehicles, d in Days} binary;
var a{AllNodes, Days};

##########################
### Objective Function ###
##########################
minimize Tourlength: sum {(i,j) in Arcs,k in Vehicles, d in Days}
    (t[i,j] + s[j,d])* x[i,j,k,d];

###################
### Constraints ###
###################

#########################
### Constraint (2.2)  ###
#########################
subject to DepotService{k in Vehicles, d in Days}: y[0,k,d]=1;
#########################
### Constraint (2.3)  ###
##########################
subject to DepotArrival{d in Days}:
a[0,d]=0;
#########################
### Constraint (2.4)  ###
#########################
subject to NodeService  {i in NonDepotNodes, d in Days}:
sum{k in Vehicles} y[i,k,d]=w[i,d];
#########################
### Constraint (2.5)  ###
#########################
subject to DemandConstraint{k in Vehicles, d in Days}:
sum{i in NonDepotNodes} q[i,d]*y[i,k,d] <= Q;
#########################
### Constraint (2.6)  ###
#########################
subject to InDegree  {j in AllNodes, k in Vehicles, d in Days}:
sum{i in AllNodes}  x[i,j,k,d]=y[j,k,d];
subject to OutDegree {j in AllNodes, k in Vehicles, d in Days}:
sum{i in AllNodes}  x[j,i,k,d]=y[j,k,d];
#########################
### Constraint (2.7)  ###
#########################
subject to SameVehicle1{i in AllNodes, k in Vehicles, d1 in Days,
d2 in Days: d1!=d2}:
w[i,d1]+w[i,d2]-2 <= y[i,k,d1]-y[i,k,d2];
subject to SameVehicle2{i in AllNodes, k in Vehicles, d1 in Days,
d2 in Days: d1!=d2}:
 y[i,k,d1]-y[i,k,d2]<=-w[i,d1]-w[i,d2]+2;
#########################
### Constraint (2.8)  ###
#########################
subject to ArrivalTime1{i in AllNodes, j in NonDepotNodes,
d in Days, k in Vehicles}:
a[j,d]>=a[i,d]+x[i,j,k,d]*t[i,j]+x[i,j,k,d]*s[i,d]-T+T*x[i,j,k,d];
#########################
### Constraint (2.9)  ###
#########################
subject to ArrivalTime2{i in AllNodes, j in NonDepotNodes,
    d in Days, k in Vehicles}:
    a[j,d]<=a[i,d]+x[i,j,k,d]*t[i,j]+x[i,j,k,d]*s[i,d]+
        T-T*x[i,j,k,d];
##########################
### Constraint (2.10)  ###
##########################
subject to Subtours {k in Vehicles, d in Days,
m in S: card(Cutset[m]) >= 2 and card(Cutset[m]) <= N-1}:
    sum {(i,j) in Arcs: (i in Cutset[m]) and (j in Cutset[m])}
    x[i,j,k,d] <= card(Cutset[m])-1;
##########################
### Constraint (2.11)  ###
##########################
subject to BoundArrivalTime{i in NonDepotNodes, d in Days}:
0<=a[i,d]+s[i,d]*w[i,d]+t[i,0]*w[i,d]<=T*w[i,d];
##########################
### Constraint (2.12)  ###
##########################
subject to ConsistentMin{i in AllNodes, d1 in Days,
    d2 in Days: d1!=d2}:
    a[i,d1]-a[i,d2]>= (-L+T*w[i,d1]+T*w[i,d2]-2*T);
subject to ConsistentMax{i in AllNodes, d1 in Days,
    d2 in Days: d1!=d2}:
    a[i,d1]-a[i,d2]<=L-T*w[i,d1]-T*w[i,d2]+2*T;

end;
```

A.3. Analysis of the Template Routes

In this section, we analyze several quantities related to our heuristic algorithm for solving the ConVRP. One of the key properties of this algorithm is the construction of a set of template routes that consists of all customers that require service on more than one day over the D-day period. After constructing this template, we determine the routes for an individual day by first removing all customers from the template that do not require service on this day. We then insert all non-template customers into the resulting routes in order to create a set of routes that visits all customers requiring service on this day.

Assuming that each customer requires service on a particular day with equal probability $p \in [0, 1]$, we derive the expected values of several quantities related to this set of template routes. Given n customer locations, we assume that the service probabilities are independent from one day to the next, and calculate the following quantities:

- The expected number of customers in the template
- The expected number of removals from the template on a single day
- The expected number of additions to the restricted template on a single day

We compute a closed form expression for these three quantities in terms of D, n, and p. We begin by giving expressions for the probabilities of various events, and then use these to compute the expected values of the three quantities of interest.

We first derive the probability that a given customer is in the template. Recall that a customer is in the template if the customer requires more than one visit during the D days. Let X_k denote the event that a customer requires service on exactly k days during the D-day period, and let Y denote the event that a customer is in the template. Then $P(Y) = 1 - (P(X_0) + P(X_1))$, where $P(X_0) = (1-p)^D$ and $P(X_1) = D(1-p)^{D-1}p$, so that

$$P(Y) = 1 - ((1-p)^D + D(1-p)^{D-1}p). \tag{A.1}$$

Next, we derive the probability that a customer in the template requires service on exactly k days where $k \in \{2, 3, \ldots, D\}$ (this probability is 0 by definition when $k = 0$ or 1 since customers requiring less than two services are not in the template). For each such k, we are interested in $P(X_k|Y)$, the probability that a customer requires service on exactly k days, given that the customer is in the template. We use the identity

$$P(X_k|Y) = \frac{P(X_k) - P(X_k|\overline{Y})P(\overline{Y})}{P(Y)}. \tag{A.2}$$

Note that $P(X_k|\overline{Y})$ is the probability that a customer requires service on $k \geq 2$ days, given that the customer is not in the template. However, since the template consists of only those customers that require service on 2 or more days, $P(X_k|\overline{Y}) = 0$, implying that $P(X_k|Y) = P(X_k)/P(Y)$ when $k \geq 2$. As we have already computed $P(Y)$ in (A.1), we are left with $P(X_k)$, the probability that a customer requires service on exactly k of the D days. This is simply

$$P(X_k) = \binom{D}{k} p^k (1-p)^{D-k}. \tag{A.3}$$

Combining (A.2) with (A.3), it follows that the probability that a template customer requires service on exactly k days is

$$P(X_k|Y) = \frac{\binom{D}{k} p^k (1-p)^{D-k}}{1 - ((1-p)^D + D(1-p)^{D-1}p)}.$$

Next, we compute the probability that a template customer does not require service on a particular day. Denote this event as Z, and note that $P(Z)$ can be computed by partitioning the customers into groups based on the number of days that they require service. Suppose that a particular template customer requires service on exactly $2 \leq k \leq D$ days. Then this customer does not require service on $D-k$ days, and as we assume that all probabilities are independent from one day to the next, the probability that this customer does not require service on a particular day is just $\frac{D-k}{D}$. We can then sum across all $D-1$ values of k to find the desired probability

$$\begin{aligned} P(Z) &= \sum_{k=2}^{D} \Big(\frac{D-k}{D}\Big) P(X_k|Y) \\ &= \sum_{k=2}^{D} \Big(\frac{D-k}{D}\Big) \frac{\binom{D}{k} p^k (1-p)^{D-k}}{1-((1-p)^D + D(1-p)^{D-1}p)}, \end{aligned}$$

which simplifies to

$$P(Z) = \frac{1-(\frac{1}{1-p})^{D-2} + p(D-2)}{1-(\frac{1}{1-p})^{D-1} + p(D-1)}. \tag{A.4}$$

Note that $\overline{Z}$ represents the event that a customer in the template does require service on a particular day.

The final quantity we require is the probability that we will have to insert a non-template customer on a given day. Denoting this event as W, we reason as follows. The probability that a customer receives service on a given day is p. A customer that receives service falls into one of two categories. The first category consists of non-template customers who receive service only on this day. The second category consists of customers in the template who are serviced on this day. Thus, $p = P(\overline{Y})P(W) + P(Y)P(\overline{Z})$, and solving for the desired $P(W)$, we have

$$P(W) = (p - P(Y)P(\overline{Z}))/P(\overline{Y}). \tag{A.5}$$

Combining (A.1), (A.4), and (A.5) and simplifying the resulting expression, we find that

$$P(W) = \frac{p}{(1-p) + pd}. \tag{A.6}$$

Having calculated these various probabilities, we can now compute the desired expected values. First, we compute the expected number of customers in the template,

$$nP(Y) = n - n((1-p)^D + D(1-p)^{D-1}p).$$

Next, we compute the expected number of removals from the template on each day:

$$nP(Y)P(Z) = n\big(1 - ((1-p)^D + D(1-p)^{D-1}p)\big)\left(\frac{1 - (\frac{1}{1-p})^{D-2} + p(D-2)}{1 - (\frac{1}{1-p})^{D-1} + p(D-1)}\right).$$

Finally, we compute the expected number of non-template customers requiring service on a given day. This is the expected number of additions we have to make to the template.

$$nP(\overline{Y})P(W) = n((1-p)^D + D(1-p)^{D-1}p)\left(\frac{p}{(1-p) + pd}\right).$$

A.4. Creating a Consistent Solution from an Inconsistent Solution

In this section, we present a set partitioning formulation that allows us to take a solution generated without regard for the consistency constraints and transform it into a consistent solution by assigning drivers to routes and by determining the orientation of each route.

We are given a set of routes that satisfies D days worth of customer demands and we wish to make them as consistent as possible in terms of both the same driver constraint and the same time constraint. We assume that there are v routes per day and add trivial depot-to-depot routes when the number of required routes is less than v. Next, we enumerate all possible v^D assignments of drivers to routes where each of the v drivers is assigned to one of the v possible routes on each of the D days. For each assignment i, we let c_i be the number of customers that receive consistent service in terms of the same driver constraint and define the decision variable x_i to be 1 if assignment i is chosen and 0 otherwise. We define a_{ir} to be 1 if route r is contained in assignment i and 0 otherwise, with r ranging over all $v \times D$ routes. As there are v^D possible assignments and $v \times D$ total routes, we have Dv^{D+1} total a_{ir} variables. We assign drivers to routes by solving the following integer program.

$$\text{Maximize} \quad \sum_i c_i x_i \tag{A.7}$$

$$\text{s.t.} \quad \sum_i x_i = v \tag{A.8}$$

$$\sum_i x_i a_{ir} = 1 \text{ for every route } r \tag{A.9}$$

$$x_i \in \{0,1\} \text{ for all } i. \tag{A.10}$$

The objective function (A.7) maximizes the number of customers receiving consistent service. Constraint (A.8) ensures that we assign a single set of D routes to each of the v drivers. Constraint (A.9) guarantees that each route r is assigned to exactly one driver.

After assigning drivers to routes in the optimal way, we minimize the maximum arrival time differential experienced by the customers that receive consistent service from the same driver. We minimize the mean maximum arrival time differential by simply taking the D routes and then computing the mean maximum differential if we traverse the route in either direction. In other words, for each vehicle, we examine all 2^D possible ways of traversing these routes and choose the option that minimizes the mean maximum arrival time differential. We note that choosing the optimal orientations for each vehicle across the D days may not be optimal for all customers since it is possible that some customers are serviced several times by more than one vehicle.

A.5. Consistent VRP Benchmark Problems

We first provide the problem instances and optimal solutions for the 10- and 12-node problems. In the second section, we provide solutions we have found to the benchmark instances that we developed and solved.

A.5.1 Small Benchmark Problems

We provide the details of the 10 small ConVRP problems that we solved to optimality using a mixed integer program (MIP). For each problem, we describe the specifics of each customer via a list of the form (k, x, y, q_1, q_2, q_3) where k is the customer number, x, y represents the location of the customer and q_i represents the demand at this customer on day i. If no service is required on day i, then we set $q_i = 0$. Whenever a customer requires service, there is a service time of one unit, and the travel times between nodes are set to the Euclidean distances. The depot is located at the origin, the total vehicle daily travel time limit is $T = 35$, the maximum daily capacity is $Q = 15$, and the maximum arrival time differential is $L = 5$.

10-node problems:

- Problem 1. (1, 8.180, 9.781, 3, 3, 1), (2, 0.864, 3.187, 0, 1, 0), (3, 3.660, 2.820, 1, 0, 2), (4, 0.866, 2.470, 1, 3, 0), (5, 3.045, 8.571, 3, 3, 0), (6, 8.371, 8.022, 0, 1, 1), (7, 2.040, 2.697, 0, 3, 0), (8, 1.409, 5.733, 3, 3, 1), (9, 9.236, 1.998, 0, 3, 1), (10, 1.755, 2.412,3, 0, 1)
- Problem 2. (1, 8.954, 5.615, 1, 3, 1), (2, 7.778, 6.524, 3, 2, 0), (3, 4.151, 3.729, 3, 0, 0), (4, 3.880, 1.690, 2, 3, 0), (5, 1.014, 5.966, 3, 3, 3), (6, 7.049, 2.998, 2, 0, 2), (7, 2.434, 4.323, 2, 3, 2), (8, 3.423, 2.062, 3, 2, 1), (9, 1.185, 1.765, 0, 1, 0), (10, 4.220, 4.711,1, 1, 0)
- Problem 3. (1, 0.020, 0.780, 3, 1, 1), (2, 7.609, 3.414, 1, 3, 1), (3, 9.463, 1.078, 2, 3, 3), (4, 6.845, 5.040, 2, 1, 0), (5, 0.435, 6.355, 0, 3, 1), (6, 3.758, 5.630, 0, 2, 3), (7, 2.594, 4.659, 1, 3, 2), (8, 1.410, 2.461, 1, 2, 0), (9, 8.794, 3.059, 1, 2, 1), (10, 0.745, 8.250,1, 2, 1)

- Problem 4. (1, 5.816, 2.818, 2, 2, 1), (2, 0.599, 6.659, 1, 1, 1), (3, 8.097, 5.370, 0, 3, 0), (4, 1.917, 3.790, 2, 3, 0), (5, 0.969, 4.120, 0, 2, 3), (6, 8.925, 3.601, 3, 3, 0), (7, 4.750, 7.027, 2, 1, 2), (8, 8.097, 6.313, 2, 0, 2), (9, 0.902, 9.245, 3, 2, 2), (10, 3.022, 4.613,3, 1, 2)
- Problem 5. (1, 9.574, 5.474, 1, 0, 0), (2, 0.637, 3.747, 1, 2, 3), (3, 7.541, 3.132, 2, 1, 2), (4, 4.448, 0.854, 2, 3, 2), (5, 2.152, 9.587, 2, 3, 0), (6, 4.914, 4.111, 2, 3, 0), (7, 0.561, 4.104, 3, 0, 3), (8, 2.152, 1.214, 3, 2, 3), (9, 5.801, 0.251, 3, 2, 1), (10, 2.532, 2.137,0, 1, 0)

12-node problems

- Problem 1. (1, 6.052, 3.088, 2, 3, 0), (2, 5.891, 3.533, 2, 2, 0), (3, 5.527, 1.645, 3, 0, 0), (4, 0.802, 9.435, 1, 1, 0), (5, 2.517, 3.295, 0, 1, 2), (6, 9.474, 4.196, 2, 0, 1), (7, 3.135, 6.523, 0, 0, 1), (8, 1.046, 2.026, 1, 1, 3), (9, 8.144, 0.351, 3, 1, 3), (10, 6.816, 9.033,3, 1, 3), (11, 8.537, 2.535, 2, 1, 2), (12, 2.266, 6.815, 0, 3, 2)
- Problem 2. (1, 1.458, 8.872, 0, 3, 0), (2, 4.936, 1.479, 0, 1, 0), (3, 2.792, 4.057, 1, 3, 0), (4, 0.229, 2.017, 2, 0, 0), (5, 2.274, 5.382, 0, 2, 3), (6, 5.940, 5.560, 0, 0, 2), (7, 1.948, 3.185, 2, 0, 3), (8, 8.701, 3.896, 2, 3, 0), (9, 1.720, 1.825, 1, 3, 1), (10, 4.994, 0.400,2, 1, 1), (11, 6.688, 4.925, 0, 2, 3), (12, 6.522, 3.583, 2, 1, 0)
- Problem 3. (1, 6.411, 7.437, 3, 1, 1), (2, 7.263, 5.660, 1, 2, 3), (3, 7.485, 3.395, 3, 1, 0), (4, 1.432, 3.817, 1, 2, 0), (5, 1.921, 4.169, 0, 2, 3), (6, 9.957, 5.770, 2, 0, 1), (7, 2.847, 5.453, 2, 2, 3), (8, 1.674, 0.576, 0, 3, 0), (9, 2.228, 4.792, 3, 1, 1), (10, 5.879, 2.075,3, 3, 0), (11, 4.724, 9.378, 0, 3, 2), (12, 6.587, 8.195, 2, 0, 0)
- Problem 4. (1, 7.444, 2.887, 2, 2, 1), (2, 8.946, 2.130, 2, 0, 0), (3, 7.182, 4.225, 3, 0, 0), (4, 5.843, 9.515, 1, 0, 3), (5, 9.963, 0.776, 3, 2, 3), (6, 1.668, 5.045, 2, 1, 2), (7, 6.246, 4.734, 1, 2, 2), (8, 2.759, 3.211, 0, 3, 0), (9, 9.485, 5.047, 2, 2, 0), (10, 4.494, 3.050,2, 2, 0), (11, 6.972, 7.585, 2, 0, 1), (12, 2.839, 6.113, 0, 2, 1)
- Problem 5. (1, 2.205, 8.162, 0, 0, 2), (2, 2.153, 9.199, 3, 1, 3), (3, 2.467, 3.594, 3, 3, 1), (4, 3.120, 1.909, 1, 1, 1), (5, 5.185, 5.413, 0, 0, 3), (6, 5.504, 5.719, 2, 2, 3), (7, 4.233, 8.845, 2, 2, 3), (8, 6.289, 2.841, 2, 0, 1), (9, 8.490, 9.219, 2, 0, 0), (10, 2.481, 4.513,1, 0, 0), (11, 6.634, 1.767, 0, 0, 1), (12, 7.083, 6.334, 0, 0, 2)

A.5.2 Solutions to ConVRP Benchmark Problems

For each problem, we provide a plot of the solution we found as well as a table describing the routes. We list the tours traveled by each vehicle in the solution, the total length of each route, the total load carried by each vehicle, and the sum of the lengths of all routes. All calculations are performed using double precision floating point numbers (64-bit) and we give our results rounded to three decimal places. We provide the optimal solutions for the 10-node and 12-node problems as well as the best solutions we found for the 12 larger benchmark problems based on the Christofides [19, 20] VRP instances.

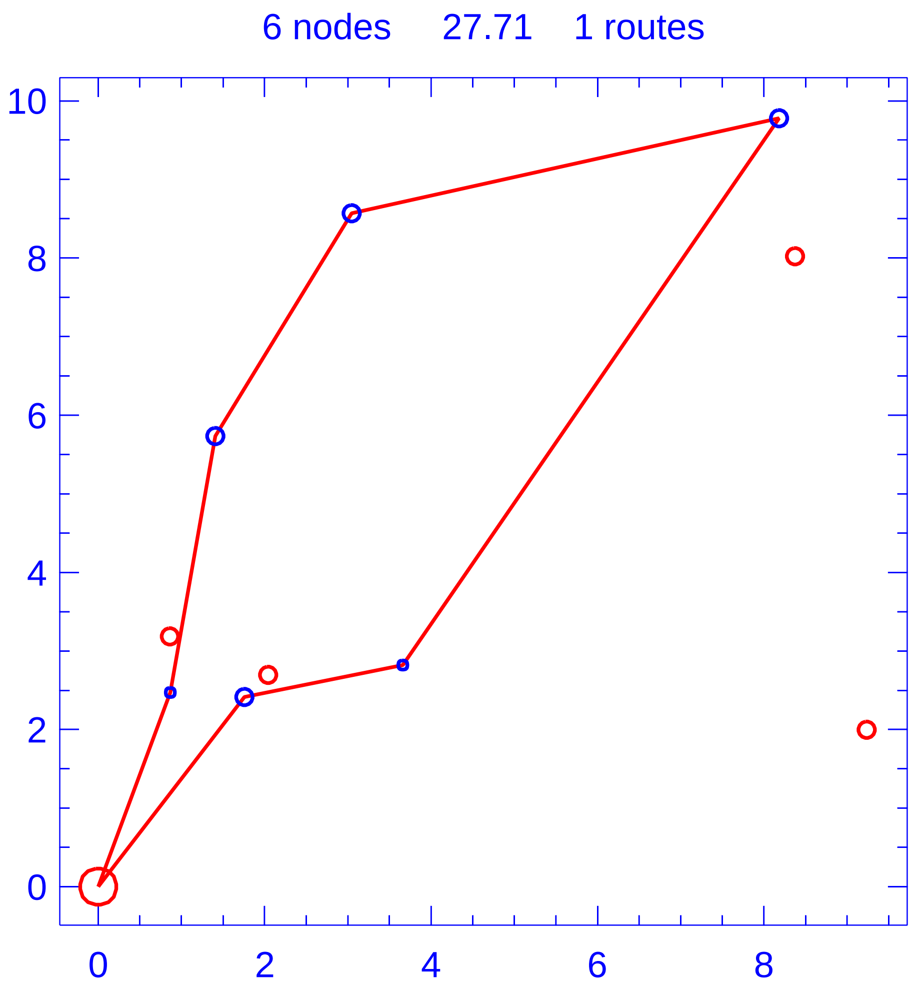


Figure A.1: Solution for day 1 of problem ConVRP-10-3-1

Problem			ConVRP-10-3-1
Vehicle capacity			15
Maximum route length			35.000
Number of nodes			6
Total route length			27.708
Total number of routes			1
Route	Length	Load	Ordering
1	27.708	14	(0, 4, 8, 5, 1, 3, 10, 0)

Table A.1: Details of solution to day 1 of problem ConVRP-10-3-1

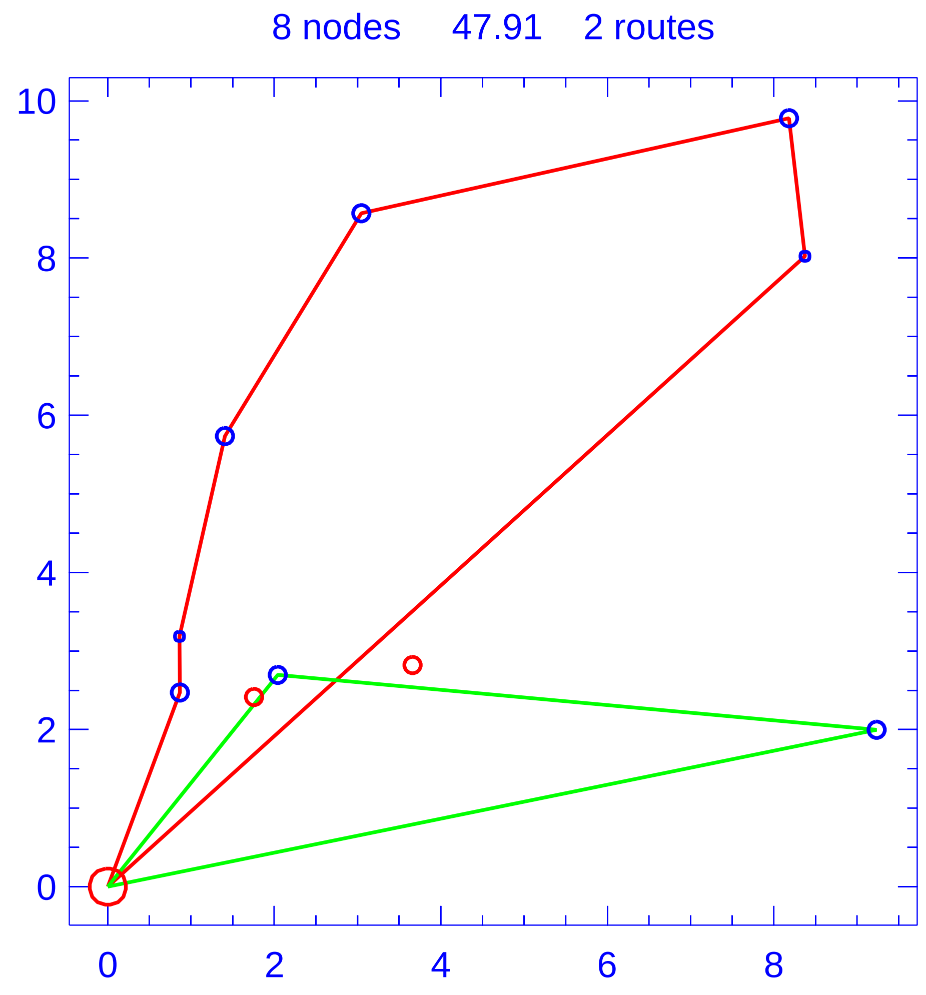


Figure A.2: Solution for day 2 of problem ConVRP-10-3-1

Problem			ConVRP-10-3-1
Vehicle capacity			15
Maximum route length			35.000
Number of nodes			8
Total route length			47.914
Total number of routes			2
Route	Length	Load	Ordering
1	27.853	14	(0, 4, 2, 8, 5, 1, 6, 0)
2	20.061	6	(0, 7, 9, 0)

Table A.2: Details of solution to day 2 of problem ConVRP-10-3-1

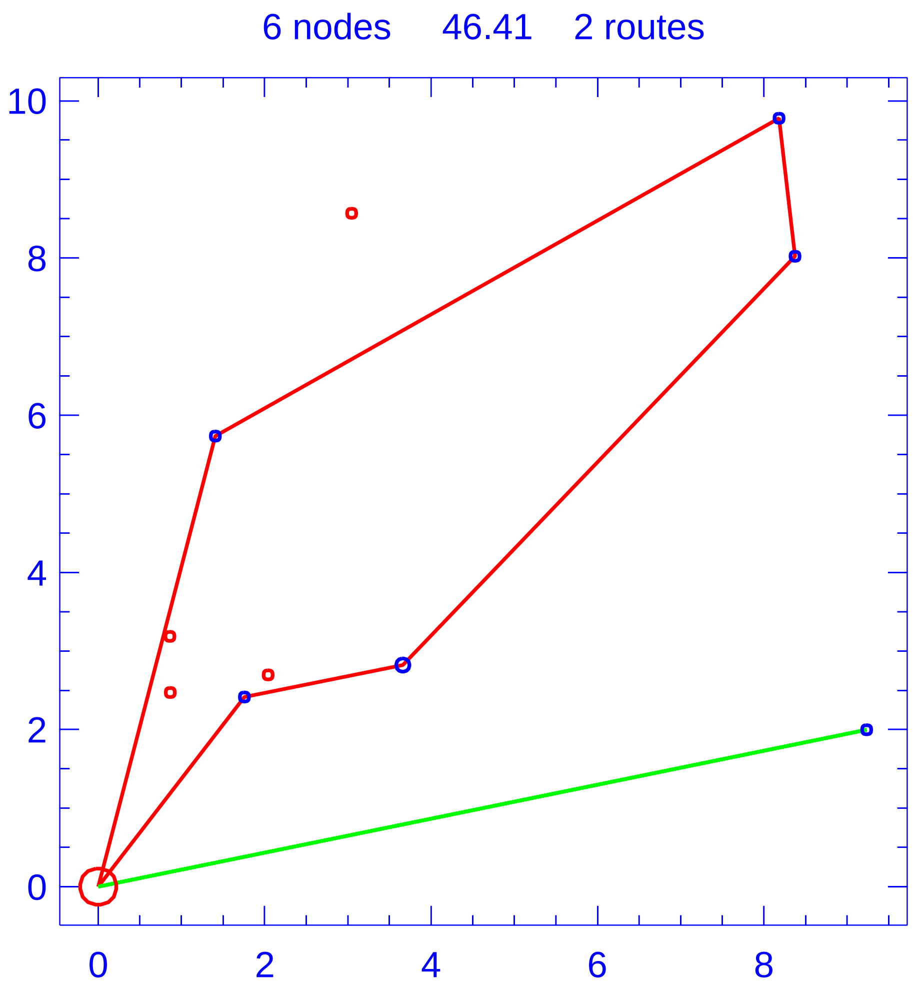


Figure A.3: Solution for day 3 of problem ConVRP-10-3-1

Problem			ConVRP-10-3-1
Vehicle capacity			15
Maximum route length			35.000
Number of nodes			6
Total route length			46.410
Total number of routes			2
Route	Length	Load	Ordering
1	27.511	6	(0, 8, 1, 6, 3, 10, 0)
2	18.899	1	(0, 9, 0)

Table A.3: Details of solution to day 3 of problem ConVRP-10-3-1

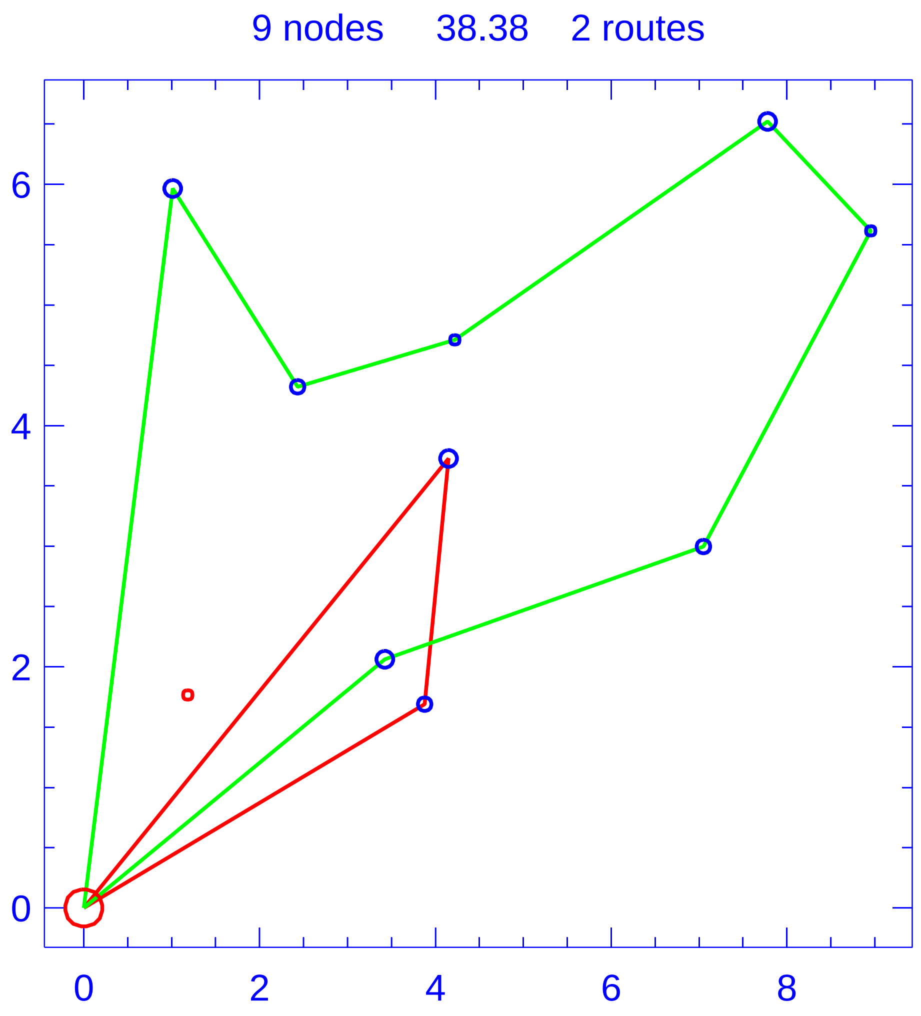


Figure A.4: Solution for day 1 of problem ConVRP-10-3-2

Problem			ConVRP-10-3-2
Vehicle capacity			15
Maximum route length			35.000
Number of nodes			9
Total route length			38.377
Total number of routes			2
Route	Length	Load	Ordering
1	11.869	5	(0, 3, 4, 0)
2	26.508	15	(0, 5, 7, 10, 2, 1, 6, 8, 0)

Table A.4: Details of solution to day 1 of problem ConVRP-10-3-2

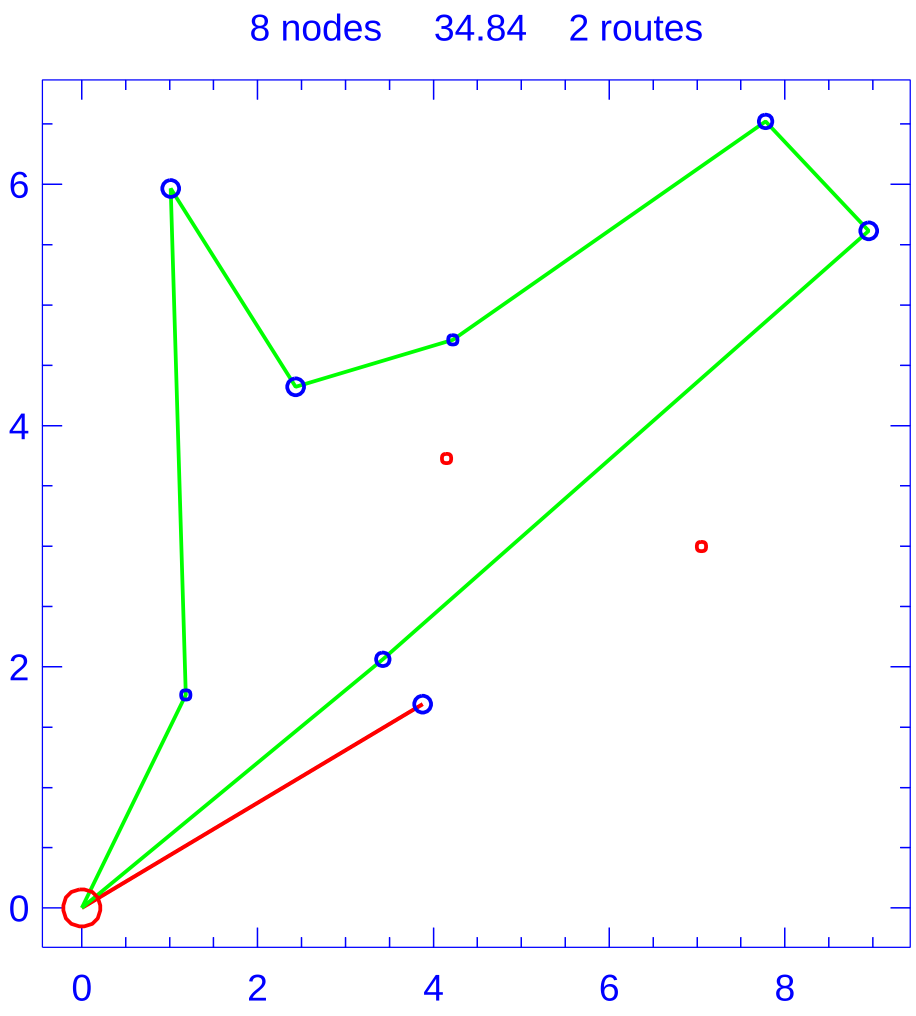


Figure A.5: Solution for day 2 of problem ConVRP-10-3-2

Problem			ConVRP-10-3-2
Vehicle capacity			15
Maximum route length			35.000
Number of nodes			8
Total route length			34.843
Total number of routes			2
Route	Length	Load	Ordering
1	8.464	3	(0, 4, 0)
2	26.379	15	(0, 8, 1, 2, 10, 7, 5, 9, 0)

Table A.5: Details of solution to day 2 of problem ConVRP-10-3-2

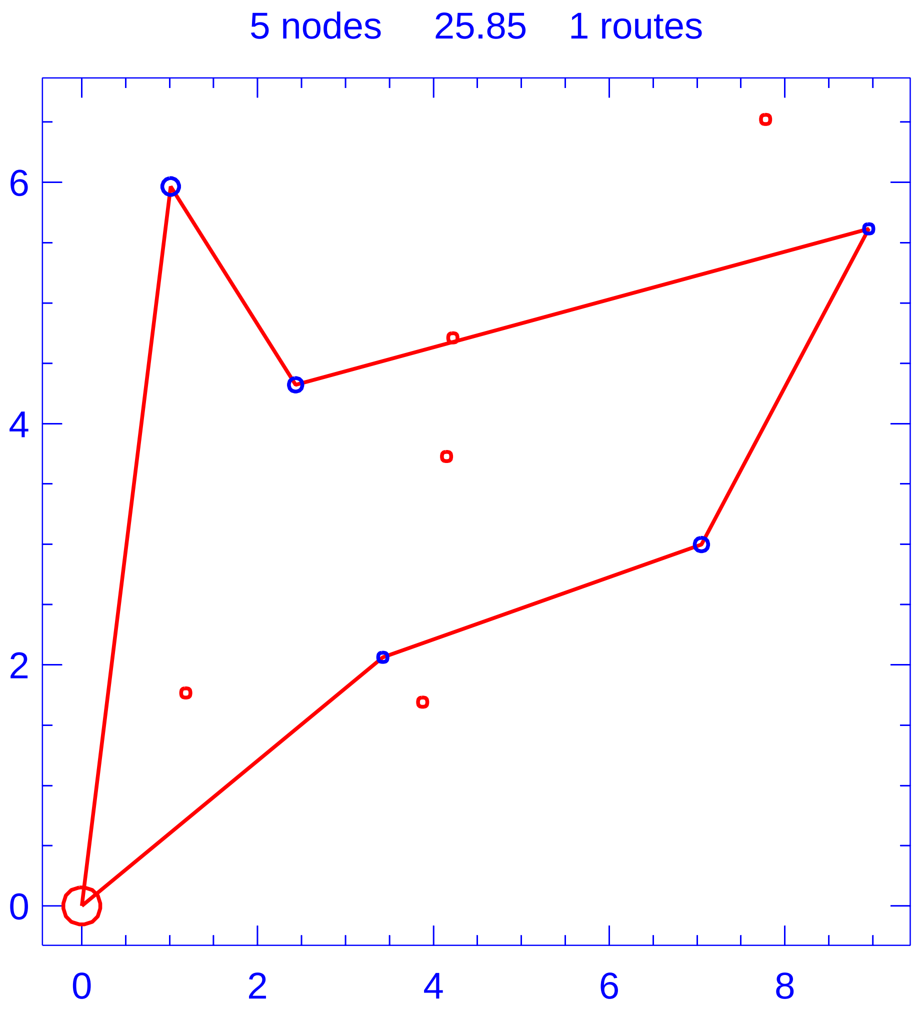


Figure A.6: Solution for day 3 of problem ConVRP-10-3-2

Problem			ConVRP-10-3-2
Vehicle capacity			15
Maximum route length			35.000
Number of nodes			5
Total route length			25.848
Total number of routes			1
Route	Length	Load	Ordering
1	25.848	9	(0, 5, 7, 1, 6, 8, 0)

Table A.6: Details of solution to day 3 of problem ConVRP-10-3-2

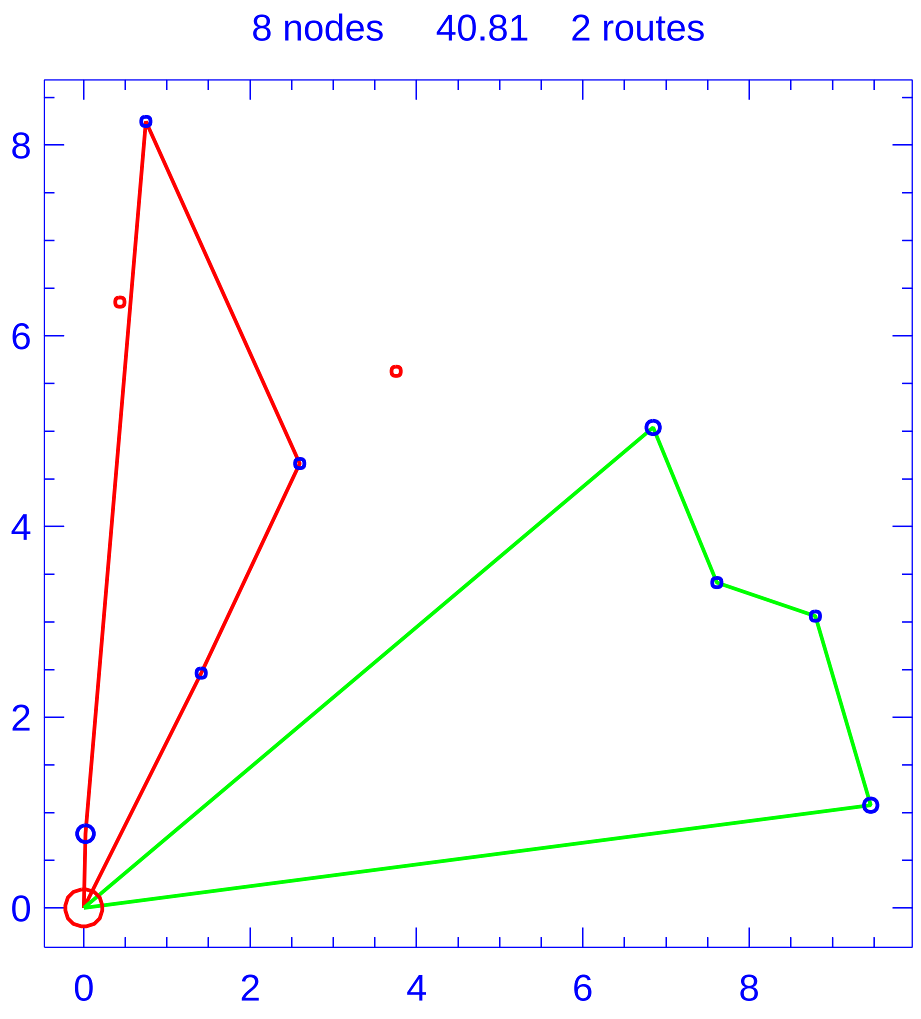


Figure A.7: Solution for day 1 of problem ConVRP-10-3-3

Problem			ConVRP-10-3-3
Vehicle capacity			15
Maximum route length			35.000
Number of nodes			8
Total route length			40.806
Total number of routes			2
Route	Length	Load	Ordering
1	17.657	6	(0, 1, 10, 7, 8, 0)
2	23.149	6	(0, 3, 9, 2, 4, 0)

Table A.7: Details of solution to day 1 of problem ConVRP-10-3-3

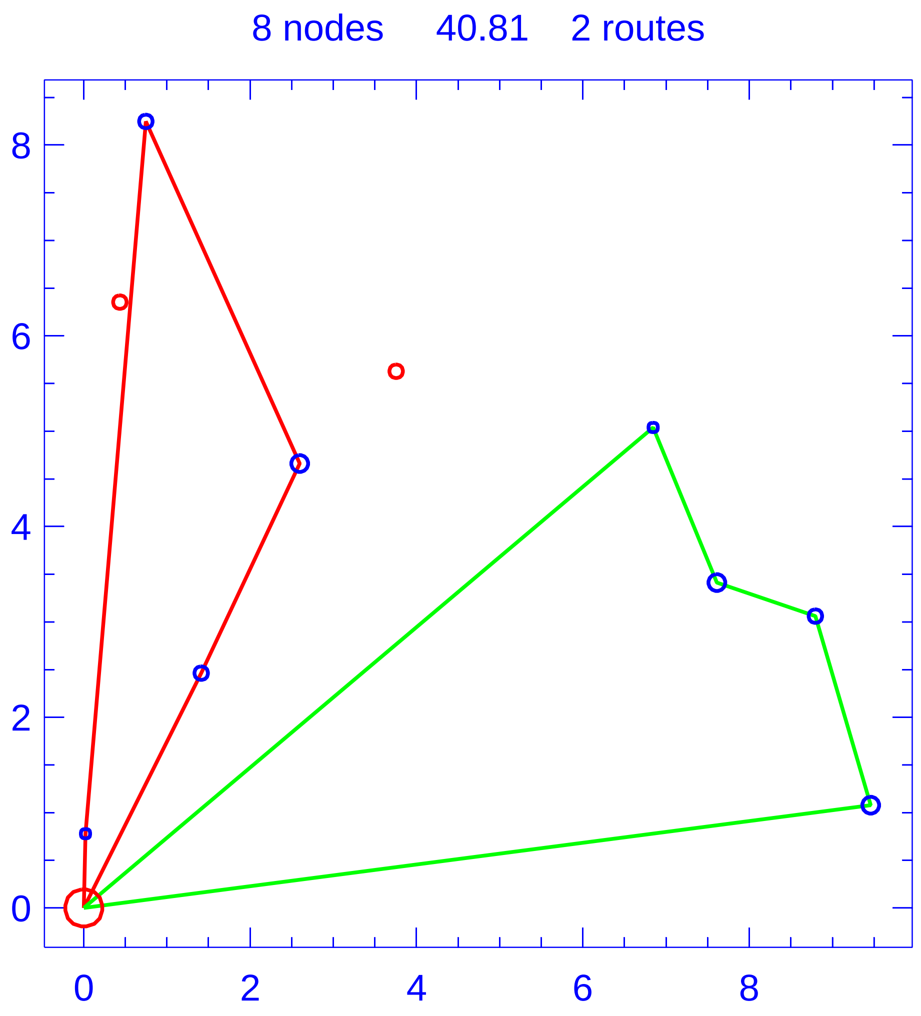


Figure A.8: Solution for day 2 of problem ConVRP-10-3-3

Problem			ConVRP-10-3-3
Vehicle capacity			15
Maximum route length			35.000
Number of nodes			8
Total route length			40.806
Total number of routes			2
Route	Length	Load	Ordering
1	17.657	8	(0, 1, 10, 7, 8, 0)
2	23.149	9	(0, 3, 9, 2, 4, 0)

Table A.8: Details of solution to day 2 of problem ConVRP-10-3-3

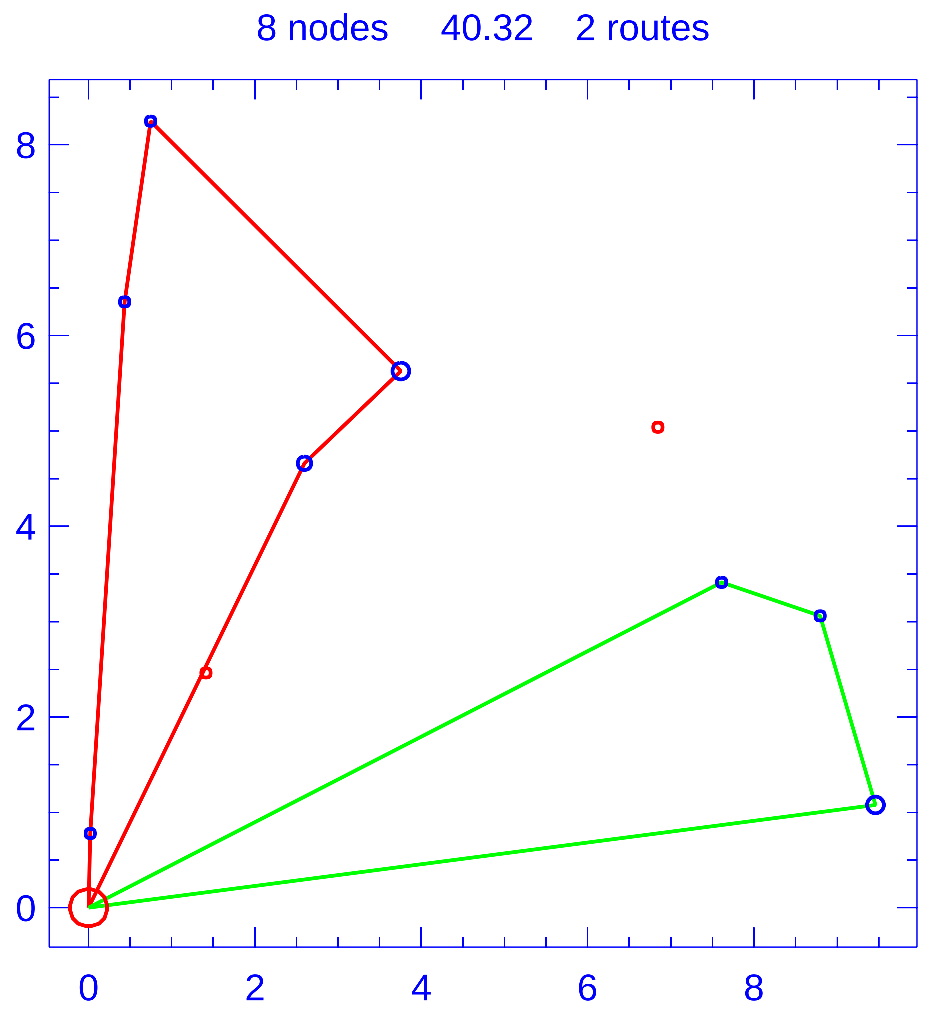


Figure A.9: Solution for day 3 of problem ConVRP-10-3-3

Problem			ConVRP-10-3-3
Vehicle capacity			15
Maximum route length			35.000
Number of nodes			8
Total route length			40.324
Total number of routes			2
Route	Length	Load	Ordering
1	19.132	8	(0, 1, 5, 10, 6, 7, 0)
2	21.192	5	(0, 2, 9, 3, 0)

Table A.9: Details of solution to day 3 of problem ConVRP-10-3-3

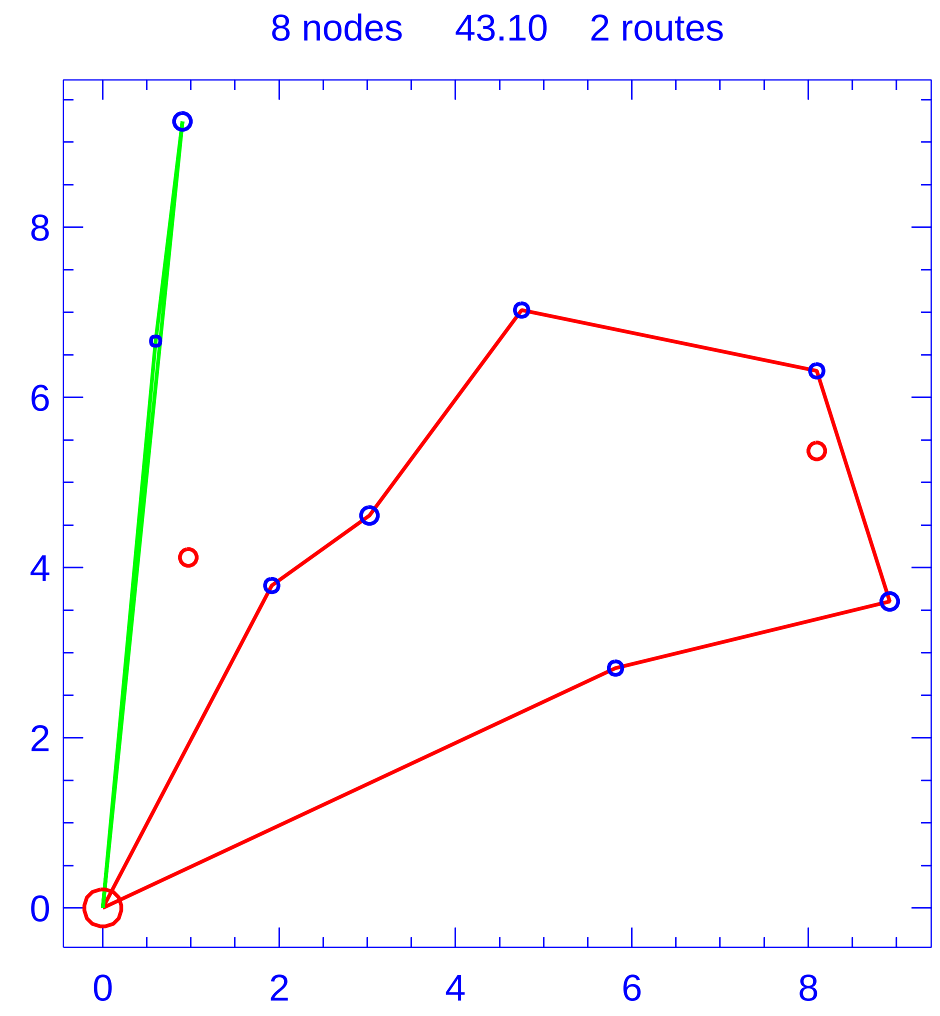


Figure A.10: Solution for day 1 of problem ConVRP-10-3-4

Problem			ConVRP-10-3-4
Vehicle capacity			15
Maximum route length			35.000
Number of nodes			8
Total route length			43.099
Total number of routes			2
Route	Length	Load	Ordering
1	24.520	14	(0, 1, 6, 8, 7, 10, 4, 0)
2	18.578	4	(0, 2, 9, 0)

Table A.10: Details of solution to day 1 of problem ConVRP-10-3-4

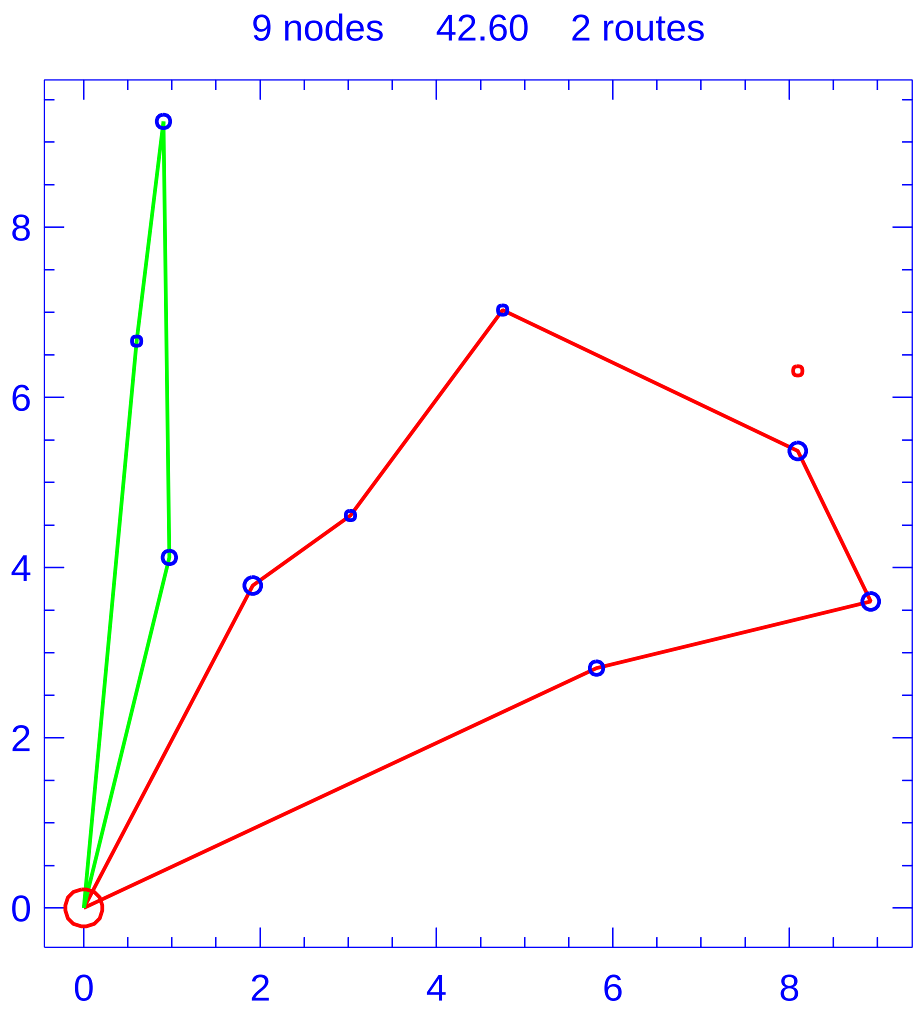


Figure A.11: Solution for day 2 of problem ConVRP-10-3-4

Problem			ConVRP-10-3-4
Vehicle capacity			15
Maximum route length			35.000
Number of nodes			9
Total route length			42.598
Total number of routes			2
Route	Length	Load	Ordering
1	23.950	13	(0, 1, 6, 3, 7, 10, 4, 0)
2	18.647	5	(0, 2, 9, 5, 0)

Table A.11: Details of solution to day 2 of problem ConVRP-10-3-4

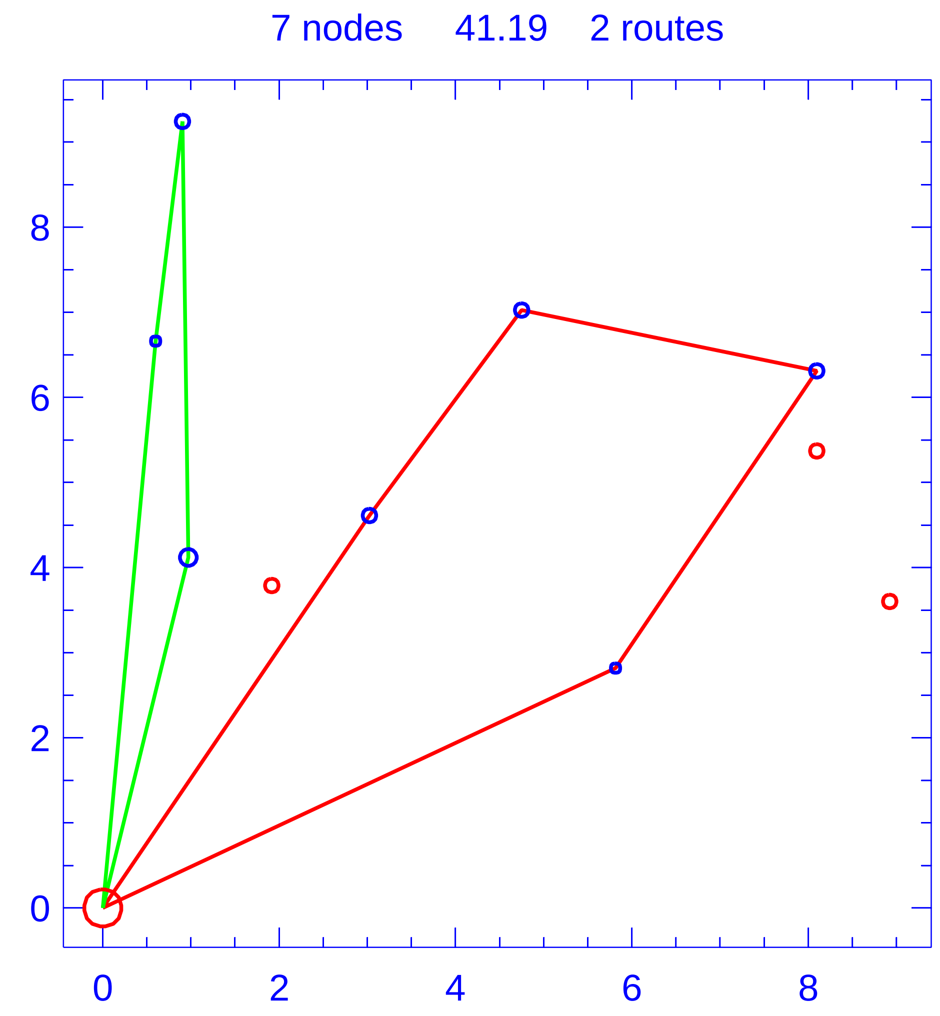


Figure A.12: Solution for day 3 of problem ConVRP-10-3-4

Problem			ConVRP-10-3-4
Vehicle capacity			15
Maximum route length			35.000
Number of nodes			7
Total route length			41.189
Total number of routes			2
Route	Length	Load	Ordering
1	22.542	7	(0, 1, 8, 7, 10, 0)
2	18.647	6	(0, 2, 9, 5, 0)

Table A.12: Details of solution to day 3 of problem ConVRP-10-3-4

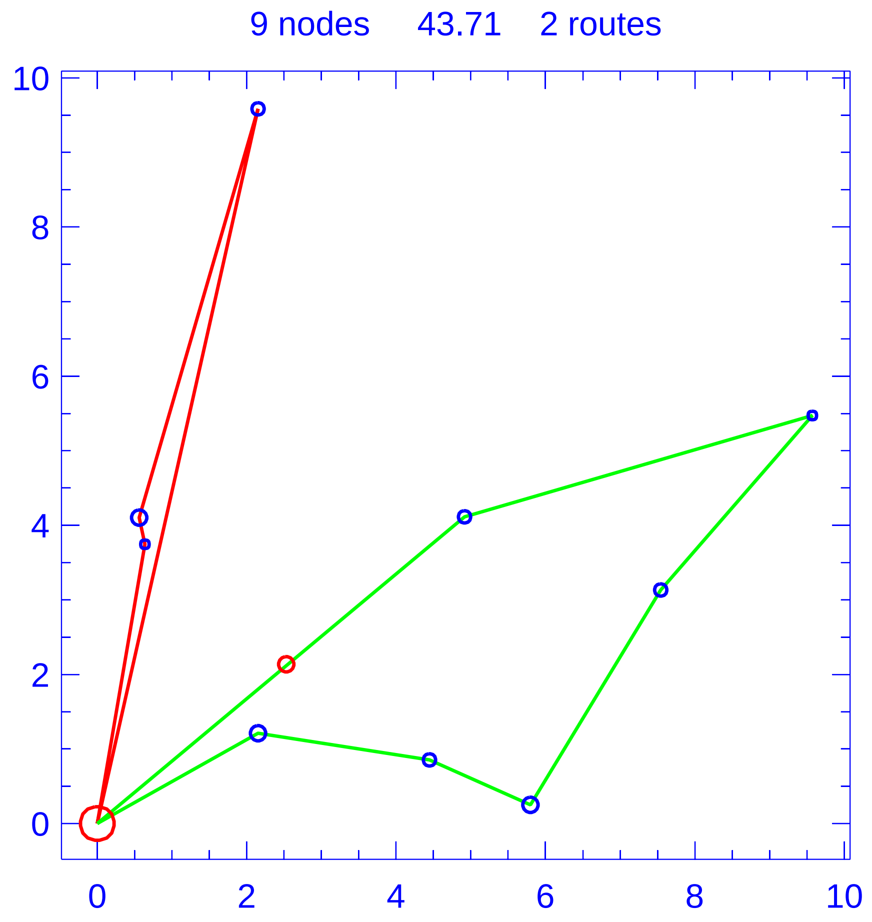


Figure A.13: Solution for day 1 of problem ConVRP-10-3-5

Problem			ConVRP-10-3-5
Vehicle capacity			15
Maximum route length			35.000
Number of nodes			9
Total route length			43.706
Total number of routes			2
Route	Length	Load	Ordering
1	19.700	6	(0, 2, 7, 5, 0)
2	24.005	13	(0, 6, 1, 3, 9, 4, 8, 0)

Table A.13: Details of solution to day 1 of problem ConVRP-10-3-5

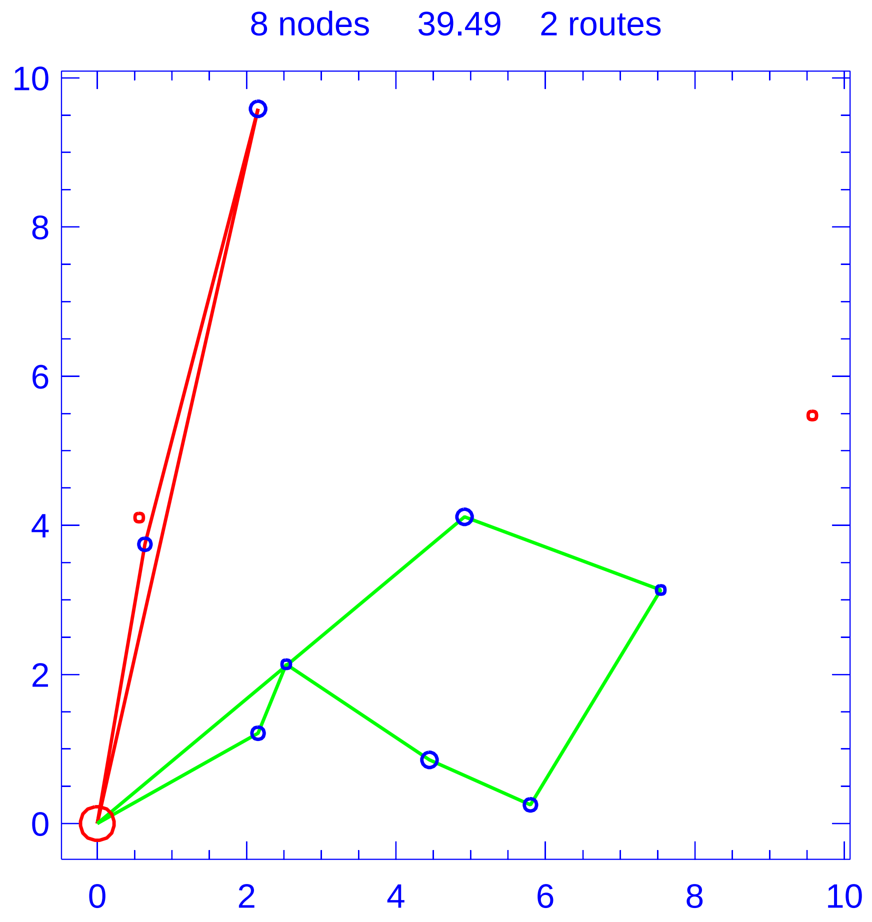


Figure A.14: Solution for day 2 of problem ConVRP-10-3-5

Problem			ConVRP-10-3-5
Vehicle capacity			15
Maximum route length			35.000
Number of nodes			8
Total route length			39.492
Total number of routes			2
Route	Length	Load	Ordering
1	19.660	5	(0, 2, 5, 0)
2	19.832	12	(0, 6, 3, 9, 4, 10, 8, 0)

Table A.14: Details of solution to day 2 of problem ConVRP-10-3-5

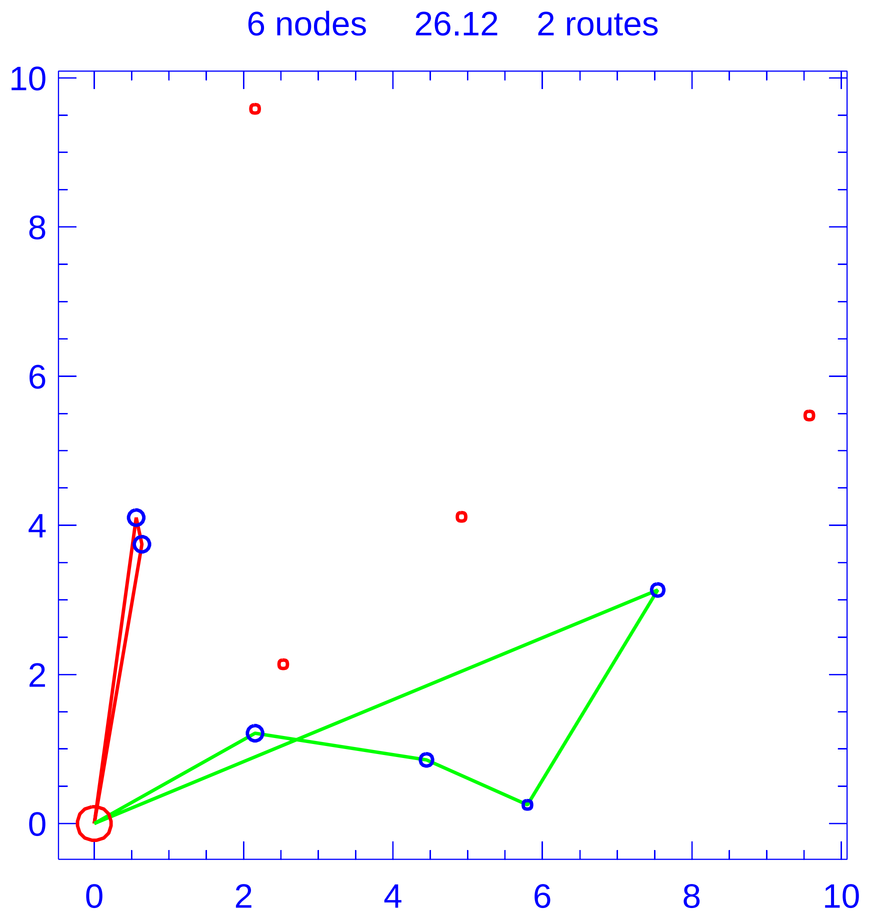


Figure A.15: Solution for day 3 of problem ConVRP-10-3-5

Problem			ConVRP-10-3-5
Vehicle capacity			15
Maximum route length			35.000
Number of nodes			6
Total route length			26.115
Total number of routes			2
Route	Length	Load	Ordering
1	8.308	6	(0, 2, 7, 0)
2	17.807	8	(0, 3, 9, 4, 8, 0)

Table A.15: Details of solution to day 3 of problem ConVRP-10-3-5

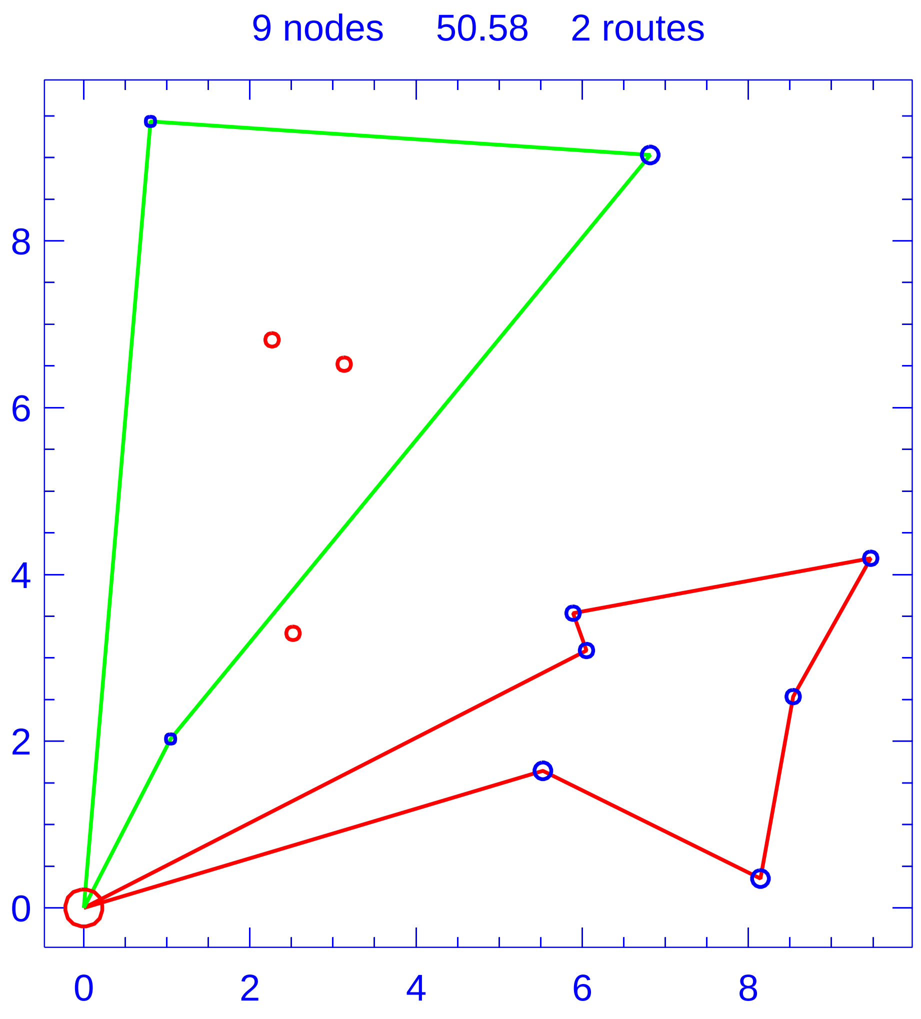


Figure A.16: Solution for day 1 of problem ConVRP-12-3-1

Problem			ConVRP-12-3-1
Vehicle capacity			15
Maximum route length			35.000
Number of nodes			9
Total route length			50.577
Total number of routes			2
Route	Length	Load	Ordering
1	23.724	14	(0, 1, 2, 6, 11, 9, 3, 0)
2	26.853	5	(0, 4, 10, 8, 0)

Table A.16: Details of solution to day 1 of problem ConVRP-12-3-1

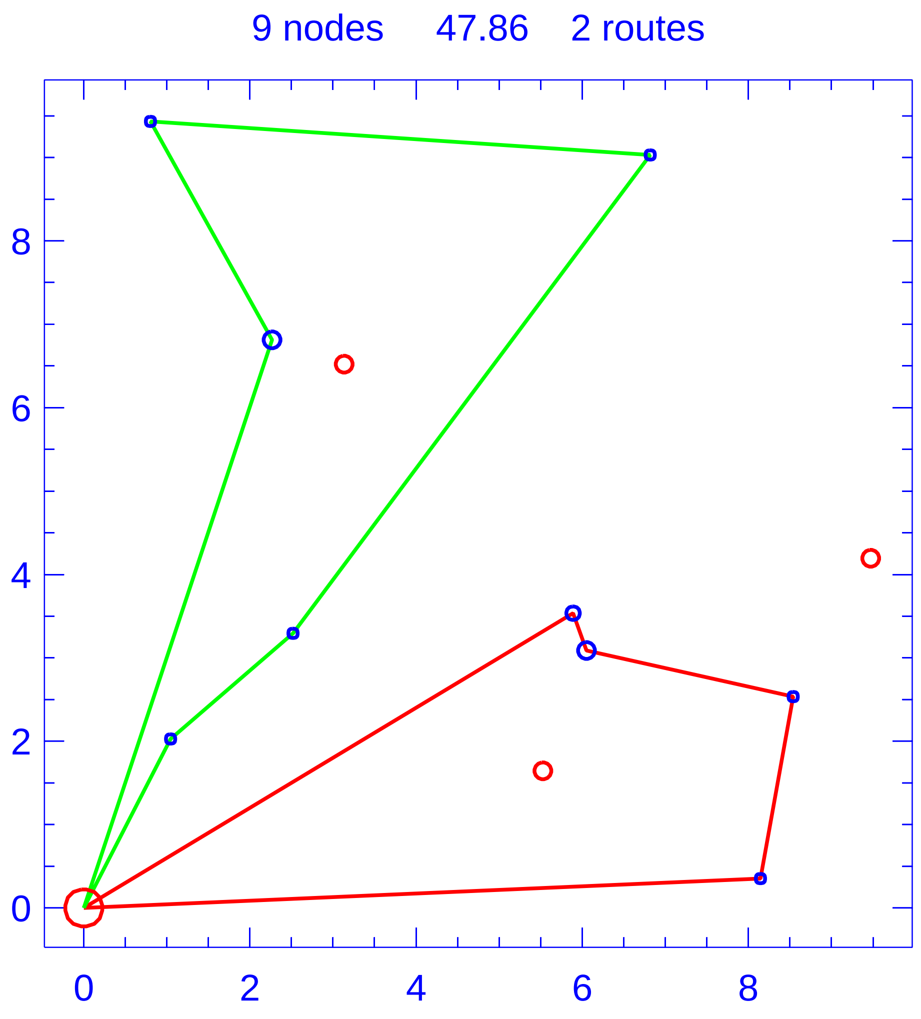


Figure A.17: Solution for day 2 of problem ConVRP-12-3-1

Problem			ConVRP-12-3-1
Vehicle capacity			15
Maximum route length			35.000
Number of nodes			9
Total route length			47.862
Total number of routes			2
Route	Length	Load	Ordering
1	20.259	7	(0, 2, 1, 11, 9, 0)
2	27.603	7	(0, 8, 5, 10, 4, 12, 0)

Table A.17: Details of solution to day 2 of problem ConVRP-12-3-1

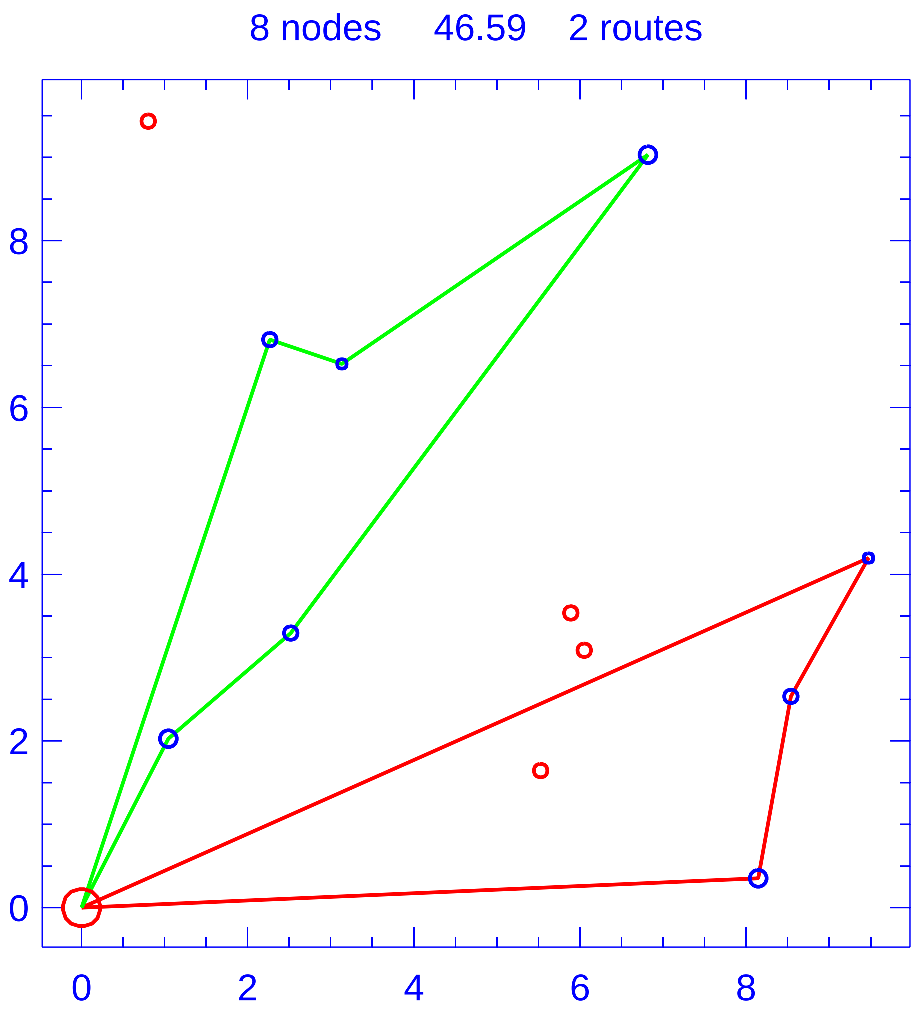


Figure A.18: Solution for day 3 of problem ConVRP-12-3-1

Problem			ConVRP-12-3-1
Vehicle capacity			15
Maximum route length			35.000
Number of nodes			8
Total route length			46.586
Total number of routes			2
Route	Length	Load	Ordering
1	22.639	6	(0, 6, 11, 9, 0)
2	23.947	11	(0, 8, 5, 10, 7, 12, 0)

Table A.18: Details of solution to day 3 of problem ConVRP-12-3-1

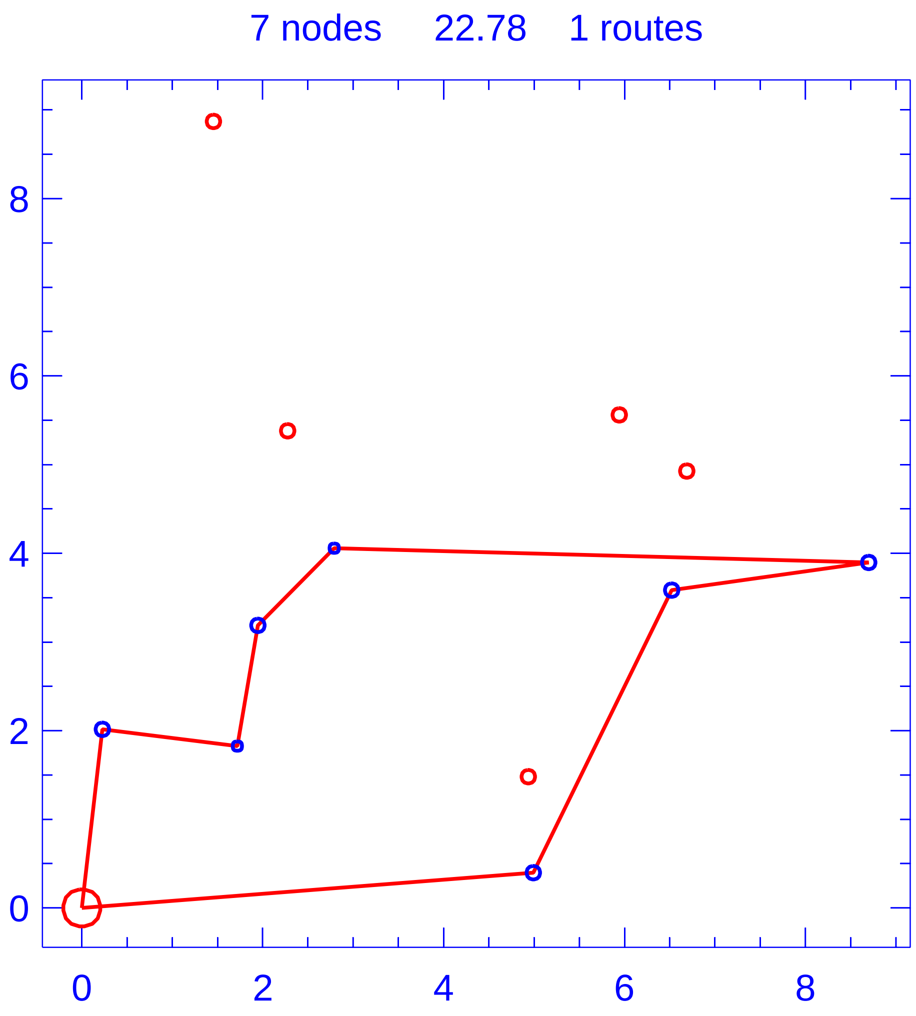


Figure A.19: Solution for day 1 of problem ConVRP-12-3-2

Problem			ConVRP-12-3-2
Vehicle capacity			15
Maximum route length			35.000
Number of nodes			7
Total route length			22.779
Total number of routes			1
Route	Length	Load	Ordering
1	22.779	12	(0, 4, 9, 7, 3, 8, 12, 10, 0)

Table A.19: Details of solution to day 1 of problem ConVRP-12-3-2

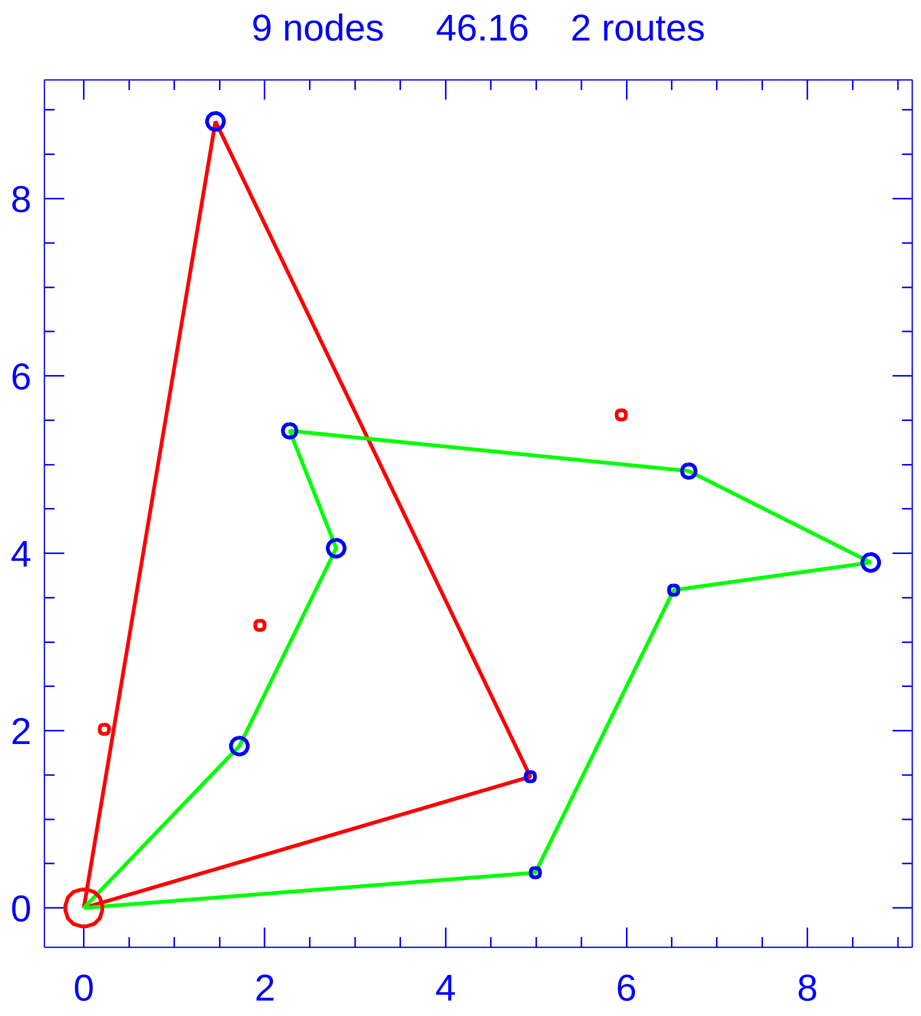


Figure A.20: Solution for day 2 of problem ConVRP-12-3-2

Problem			ConVRP-12-3-2
Vehicle capacity			15
Maximum route length			35.000
Number of nodes			9
Total route length			46.161
Total number of routes			2
Route	Length	Load	Ordering
1	22.314	4	(0, 1, 2, 0)
2	23.847	15	(0, 9, 3, 5, 11, 8, 12, 10, 0)

Table A.20: Details of solution to day 2 of problem ConVRP-12-3-2

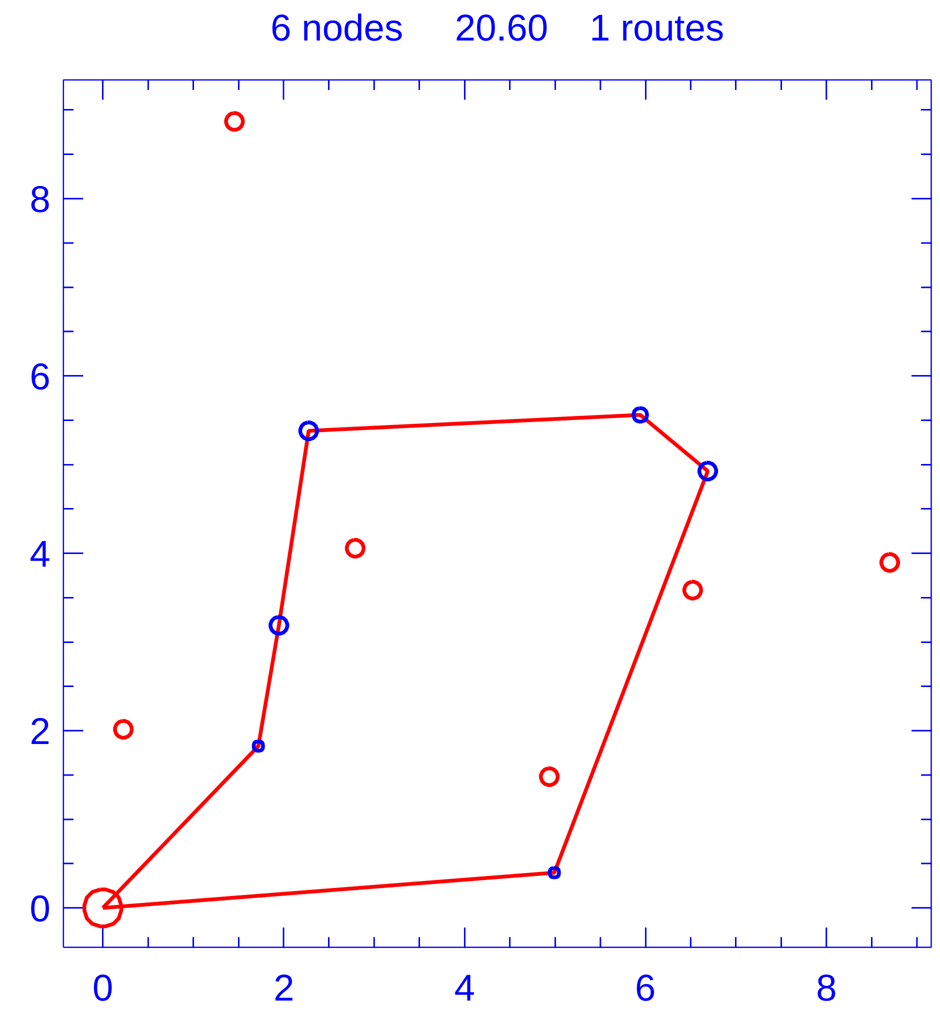


Figure A.21: Solution for day 3 of problem ConVRP-12-3-2

Problem			ConVRP-12-3-2
Vehicle capacity			15
Maximum route length			35.000
Number of nodes			6
Total route length			20.601
Total number of routes			1
Route	Length	Load	Ordering
1	20.601	13	(0, 9, 7, 5, 6, 11, 10, 0)

Table A.21: Details of solution to day 3 of problem ConVRP-12-3-2

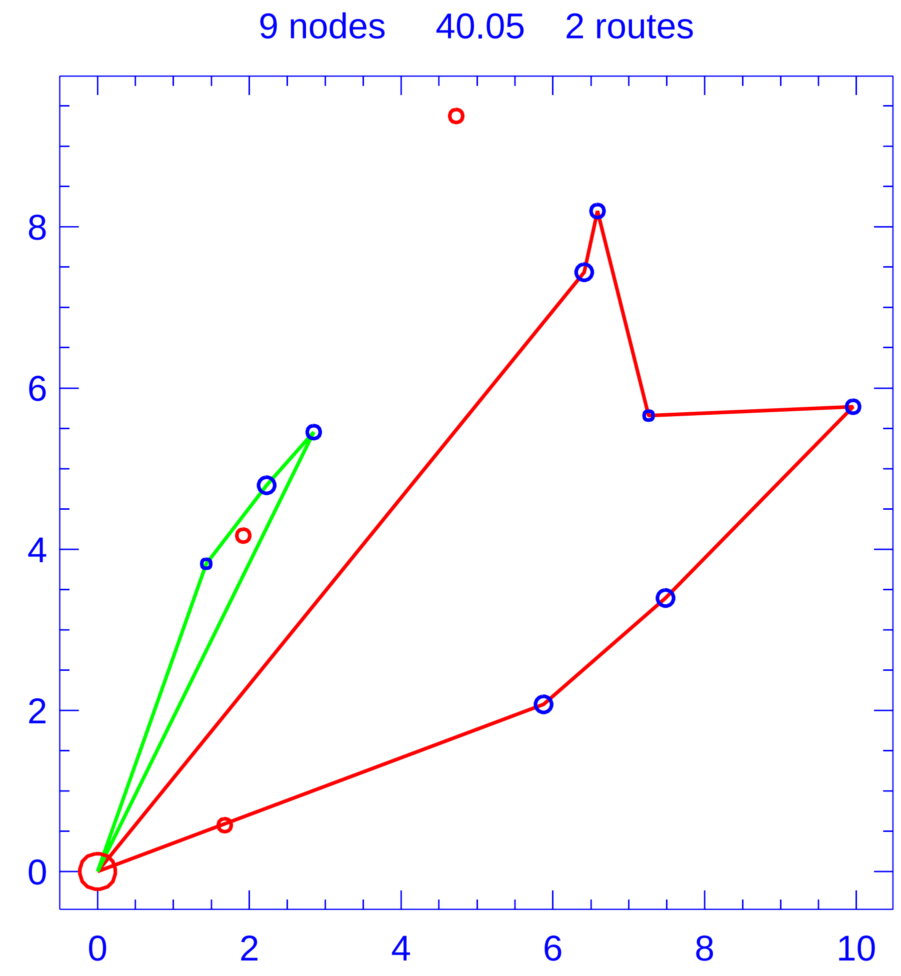


Figure A.22: Solution for day 1 of problem ConVRP-12-3-3

Problem			ConVRP-12-3-3
Vehicle capacity			15
Maximum route length			35.000
Number of nodes			9
Total route length			40.051
Total number of routes			2
Route	Length	Load	Ordering
1	27.658	14	(0, 1, 12, 2, 6, 3, 10, 0)
2	12.393	6	(0, 4, 9, 7, 0)

Table A.22: Details of solution to day 1 of problem ConVRP-12-3-3

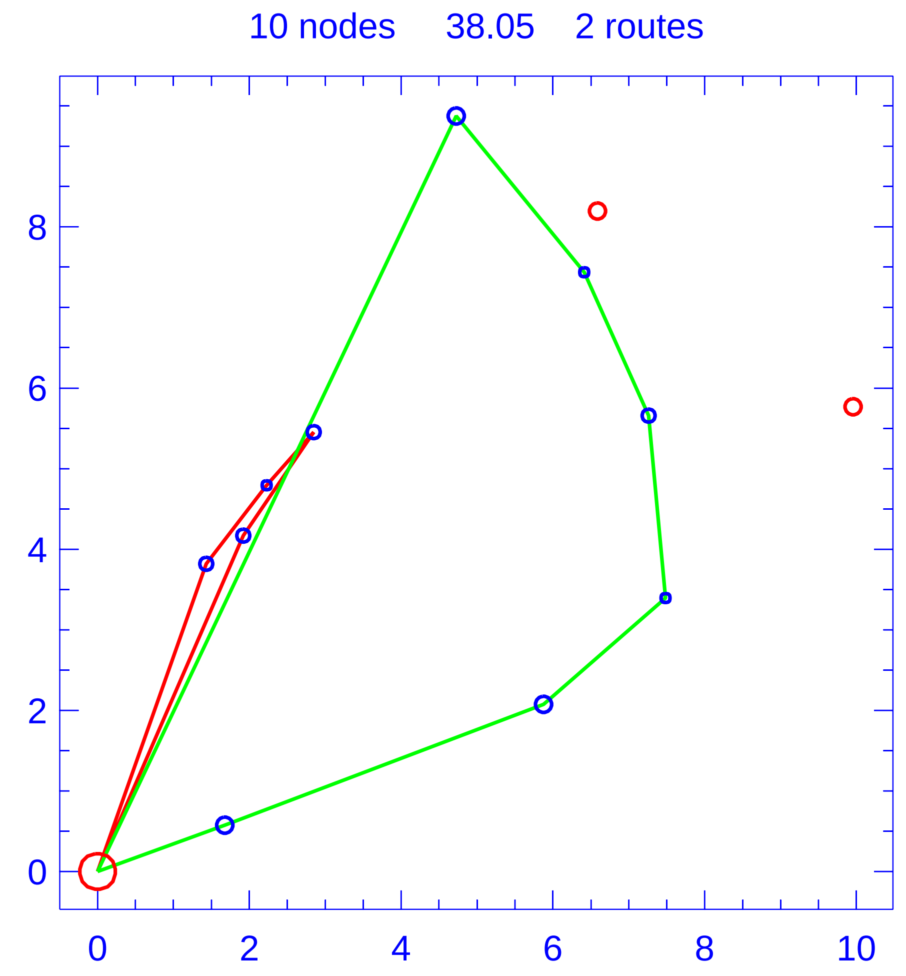


Figure A.23: Solution for day 2 of problem ConVRP-12-3-3

Problem			ConVRP-12-3-3
Vehicle capacity			15
Maximum route length			35.000
Number of nodes			10
Total route length			38.047
Total number of routes			2
Route	Length	Load	Ordering
1	12.414	7	(0, 4, 9, 7, 5, 0)
2	25.632	13	(0, 8, 10, 3, 2, 1, 11, 0)

Table A.23: Details of solution to day 2 of problem ConVRP-12-3-3

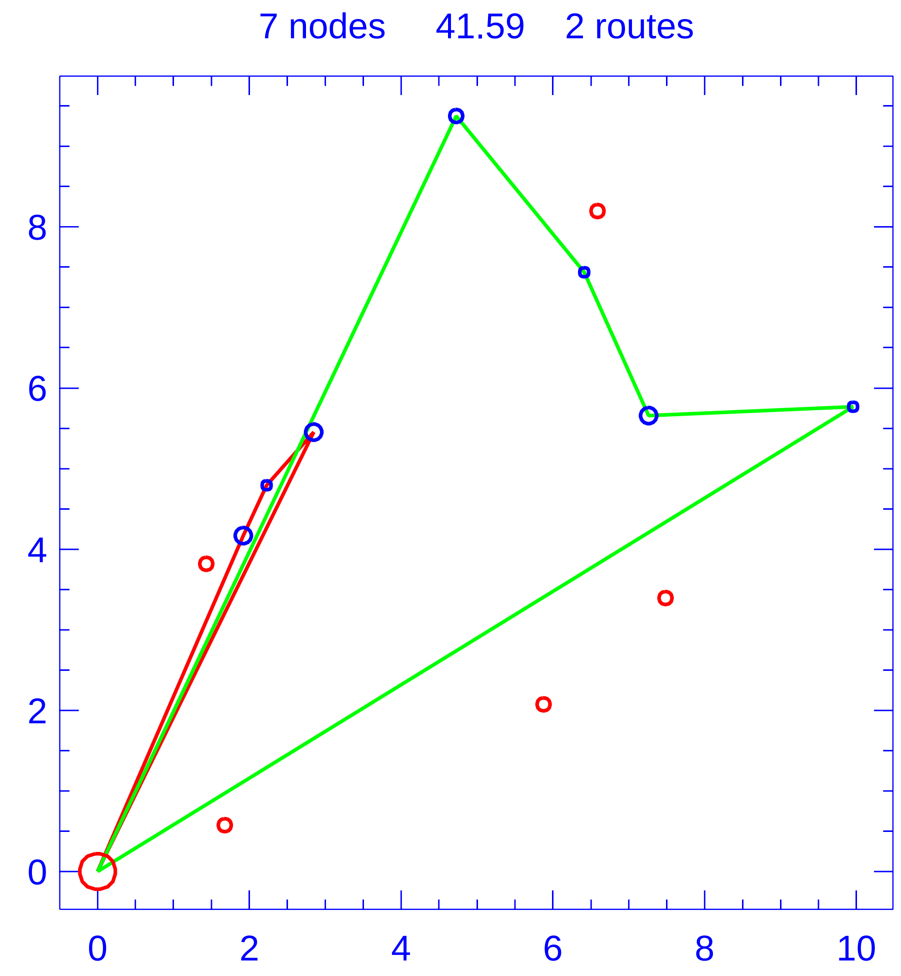


Figure A.24: Solution for day 3 of problem ConVRP-12-3-3

Problem			ConVRP-12-3-3
Vehicle capacity			15
Maximum route length			35.000
Number of nodes			7
Total route length			41.589
Total number of routes			2
Route	Length	Load	Ordering
1	12.342	7	(0, 5, 9, 7, 0)
2	29.247	7	(0, 6, 2, 1, 11, 0)

Table A.24: Details of solution to day 3 of problem ConVRP-12-3-3

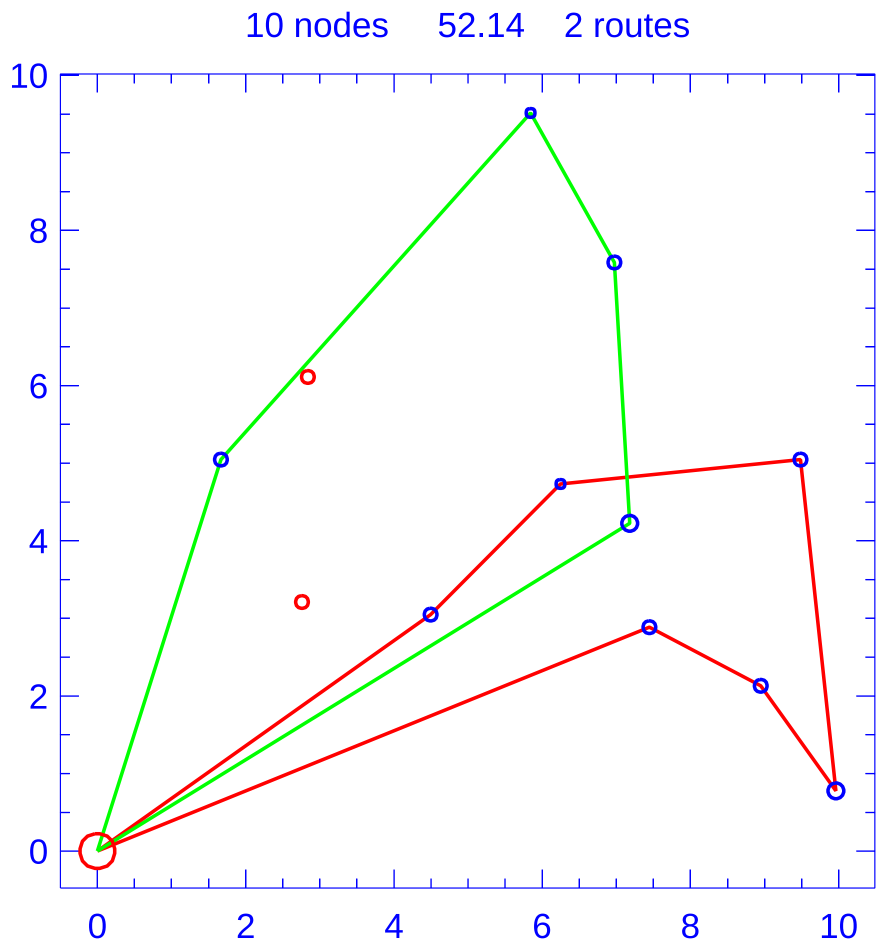


Figure A.25: Solution for day 1 of problem ConVRP-12-3-4

Problem			ConVRP-12-3-4
Vehicle capacity			15
Maximum route length			35.000
Number of nodes			10
Total route length			52.138
Total number of routes			2
Route	Length	Load	Ordering
1	26.773	12	(0, 1, 2, 5, 9, 7, 10, 0)
2	25.365	8	(0, 3, 11, 4, 6, 0)

Table A.25: Details of solution to day 1 of problem ConVRP-12-3-4

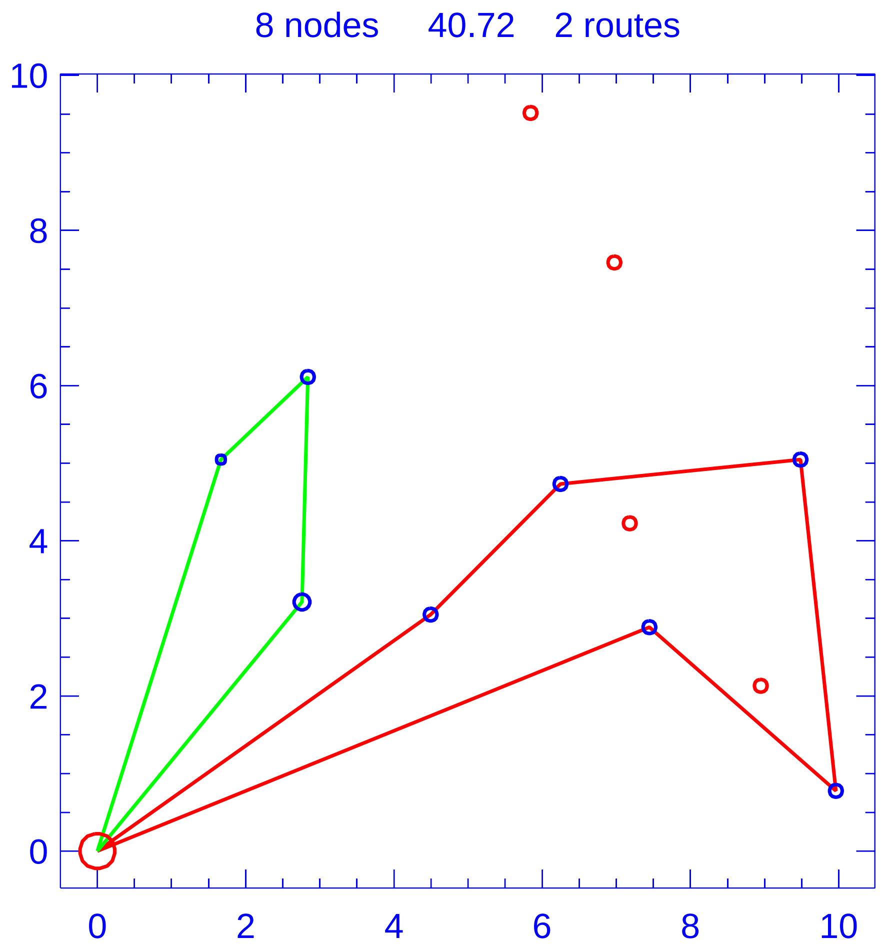


Figure A.26: Solution for day 2 of problem ConVRP-12-3-4

Problem			ConVRP-12-3-4
Vehicle capacity			15
Maximum route length			35.000
Number of nodes			8
Total route length			40.719
Total number of routes			2
Route	Length	Load	Ordering
1	26.684	10	(0, 1, 5, 9, 7, 10, 0)
2	14.035	6	(0, 6, 12, 8, 0)

Table A.26: Details of solution to day 2 of problem ConVRP-12-3-4

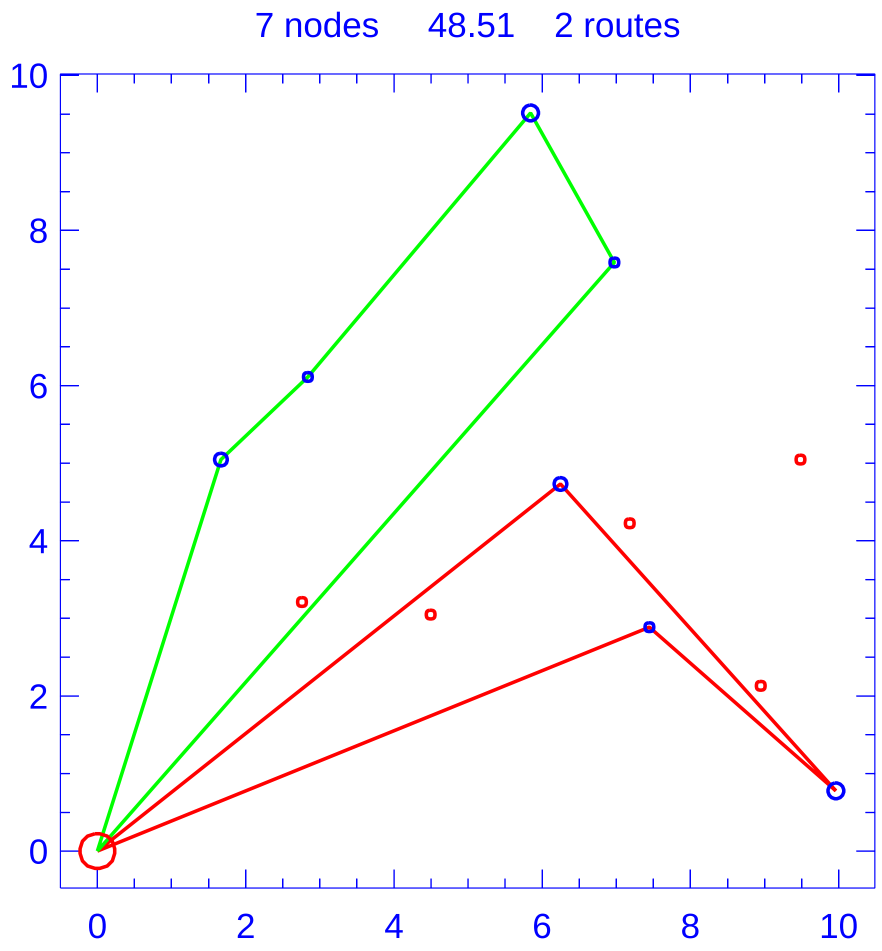


Figure A.27: Solution for day 3 of problem ConVRP-12-3-4

Problem			ConVRP-12-3-4
Vehicle capacity			15
Maximum route length			35.000
Number of nodes			7
Total route length			48.513
Total number of routes			2
Route	Length	Load	Ordering
1	24.538	6	(0, 1, 5, 7, 0)
2	23.975	7	(0, 6, 12, 4, 11, 0)

Table A.27: Details of solution to day 3 of problem ConVRP-12-3-4

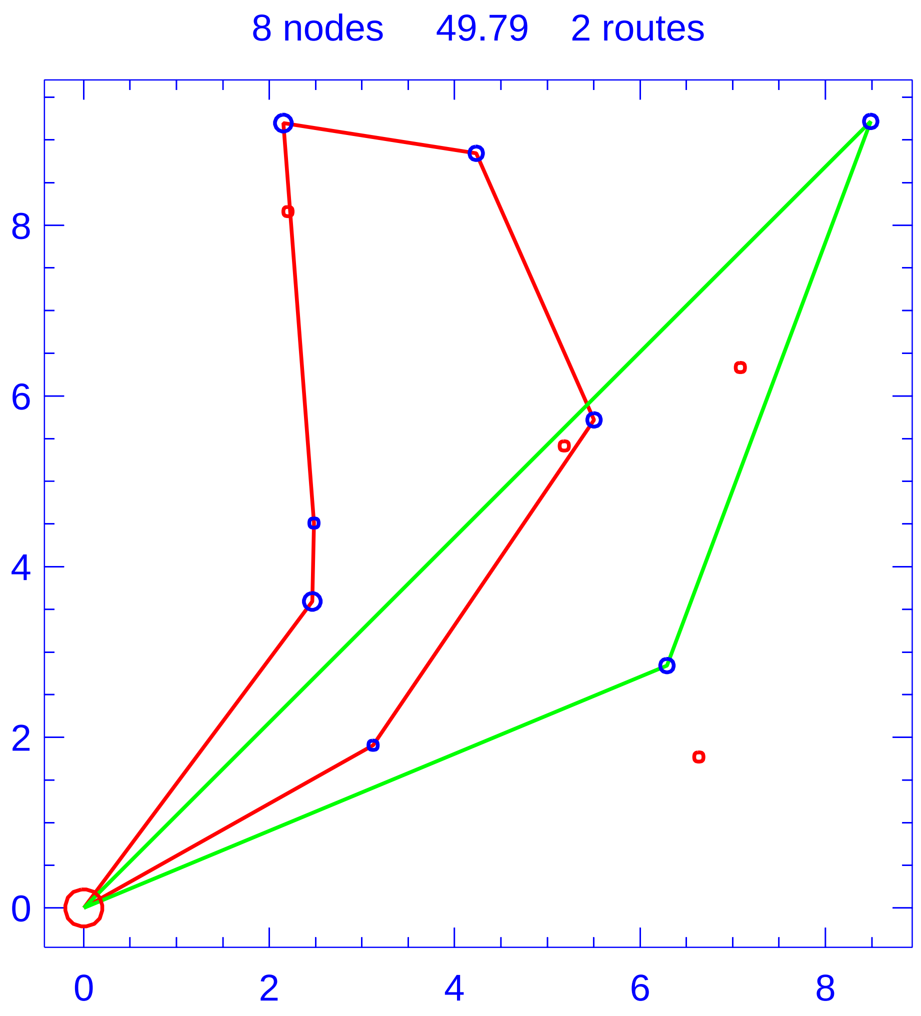


Figure A.28: Solution for day 1 of problem ConVRP-12-3-5

Problem			ConVRP-12-3-5
Vehicle capacity			15
Maximum route length			35.000
Number of nodes			8
Total route length			49.793
Total number of routes			2
Route	Length	Load	Ordering
1	23.612	12	(0, 3, 10, 2, 7, 6, 4, 0)
2	26.181	4	(0, 8, 9, 0)

Table A.28: Details of solution to day 1 of problem ConVRP-12-3-5

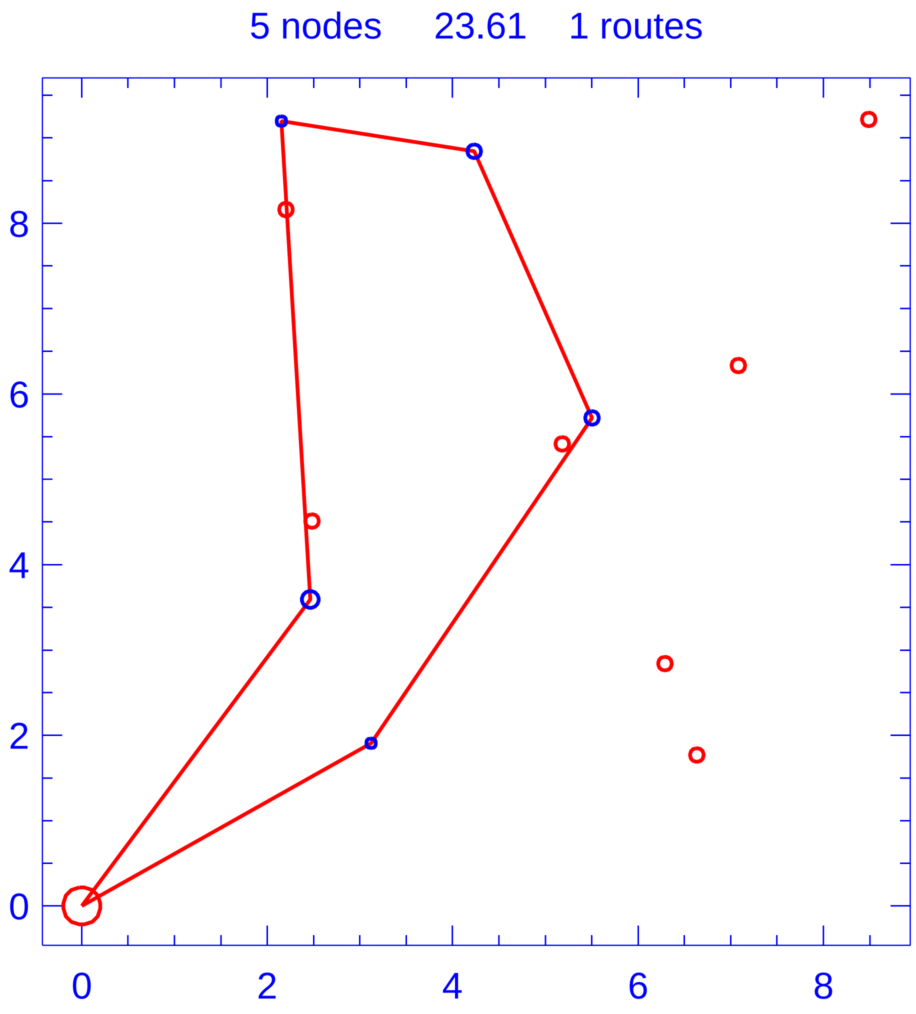


Figure A.29: Solution for day 2 of problem ConVRP-12-3-5

Problem			ConVRP-12-3-5
Vehicle capacity			15
Maximum route length			35.000
Number of nodes			5
Total route length			23.610
Total number of routes			1
Route	Length	Load	Ordering
1	23.610	9	(0, 3, 2, 7, 6, 4, 0)

Table A.29: Details of solution to day 2 of problem ConVRP-12-3-5

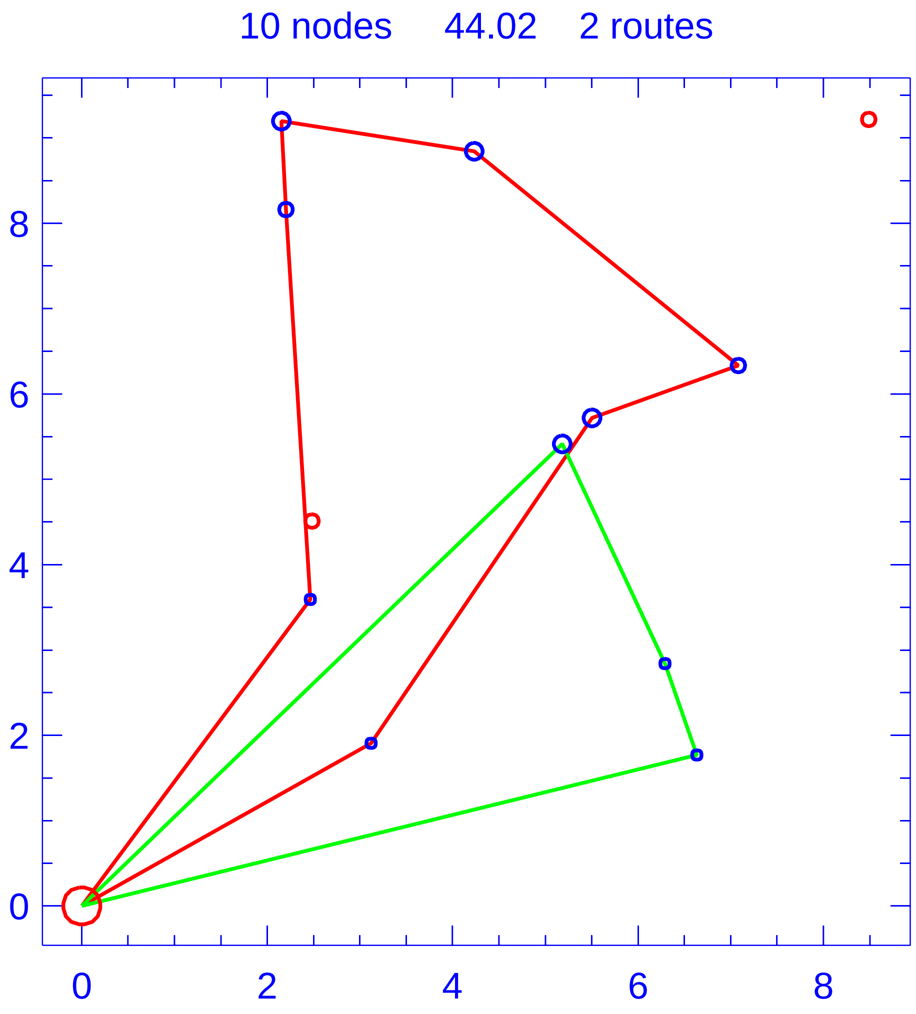


Figure A.30: Solution for day 3 of problem ConVRP-12-3-5

Problem			ConVRP-12-3-5
Vehicle capacity			15
Maximum route length			35.000
Number of nodes			10
Total route length			44.016
Total number of routes			2
Route	Length	Load	Ordering
1	25.728	15	(0, 3, 1, 2, 7, 12, 6, 4, 0)
2	18.288	5	(0, 5, 8, 11, 0)

Table A.30: Details of solution to day 3 of problem ConVRP-12-3-5

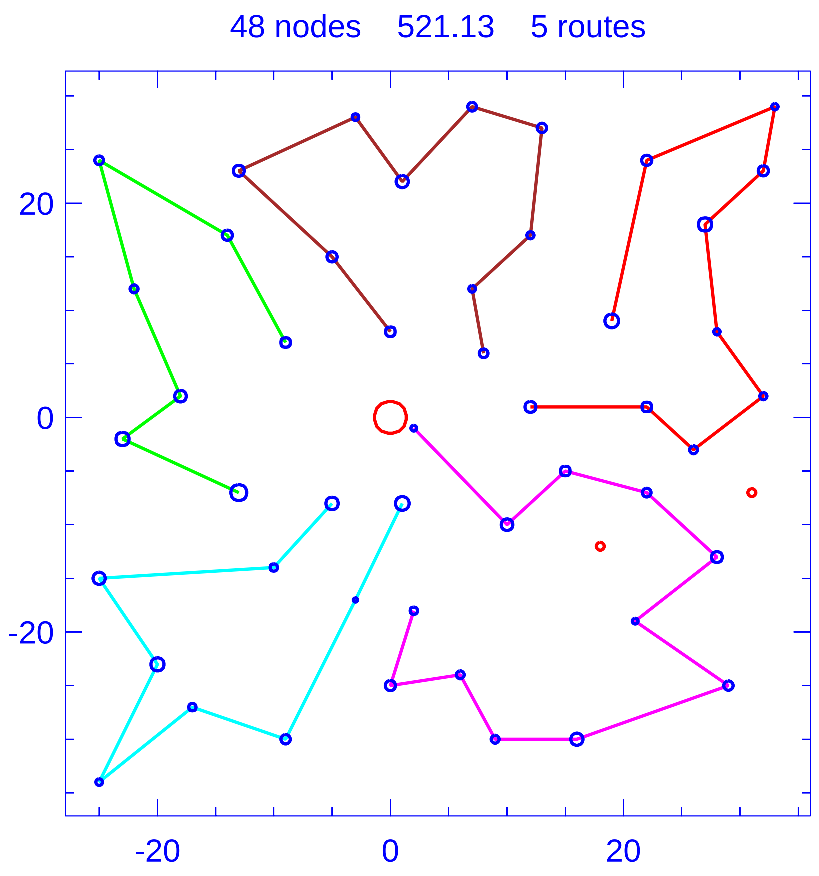


Figure A.31: Template for problem ConVRP-1

Problem			ConVRP-1
Vehicle capacity			160
Maximum route length			N/A
Number of nodes			48
Total route length			521.133
Total number of routes			5
Route	Length	Load	Ordering
1	114.327	155	(0, 2, 3, 36, 35, 20, 29, 21, 50, 16, 11, 0)
2	91.108	142	(0, 6, 23, 43, 24, 14, 25, 18, 0)
3	105.198	145	(0, 12, 17, 42, 19, 40, 41, 13, 4, 47, 0)
4	95.055	133	(0, 27, 48, 7, 26, 8, 31, 28, 22, 1, 32, 0)
5	115.444	158	(0, 37, 44, 15, 45, 33, 39, 10, 30, 9, 38, 5, 46, 0)

Table A.31: Details of template for problem ConVRP-1

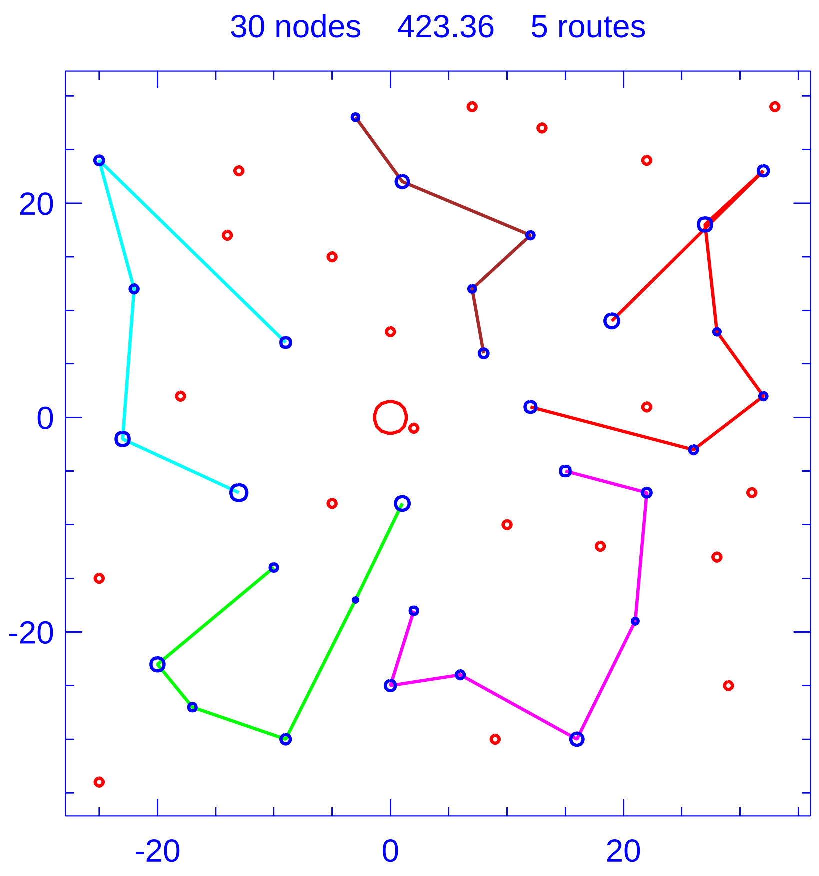


Figure A.32: Solution for day 1 of problem ConVRP-1

Problem			ConVRP-1
Vehicle capacity			160
Maximum route length			N/A
Number of nodes			30
Total route length			423.361
Total number of routes			5
Route	Length	Load	Ordering
1	98.873	118	(0, 2, 35, 20, 29, 21, 50, 11, 0)
2	76.431	90	(0, 4, 41, 19, 42, 17, 12, 0)
3	87.097	105	(0, 6, 43, 24, 25, 18, 0)
4	70.608	57	(0, 26, 8, 22, 1, 32, 0)
5	90.352	89	(0, 37, 44, 15, 33, 10, 9, 38, 0)

Table A.32: Details of solution to day 1 of problem ConVRP-1

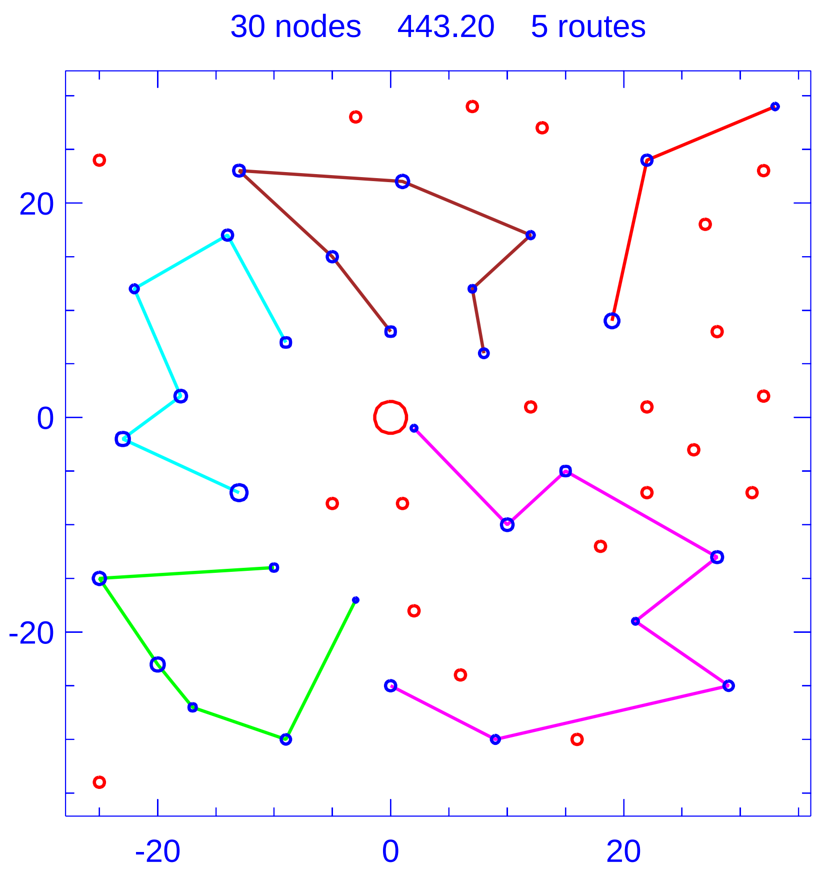


Figure A.33: Solution for day 2 of problem ConVRP-1

Problem			ConVRP-1
Vehicle capacity			160
Maximum route length			N/A
Number of nodes			30
Total route length			443.199
Total number of routes			5
Route	Length	Load	Ordering
1	92.336	52	(0, 2, 3, 36, 0)
2	86.796	84	(0, 4, 13, 41, 19, 42, 17, 0)
3	75.135	131	(0, 6, 23, 24, 14, 25, 18, 0)
4	77.189	101	(0, 27, 48, 7, 8, 22, 1, 32, 0)
5	111.744	105	(0, 44, 45, 39, 10, 30, 38, 5, 46, 0)

Table A.33: Details of solution to day 2 of problem ConVRP-1

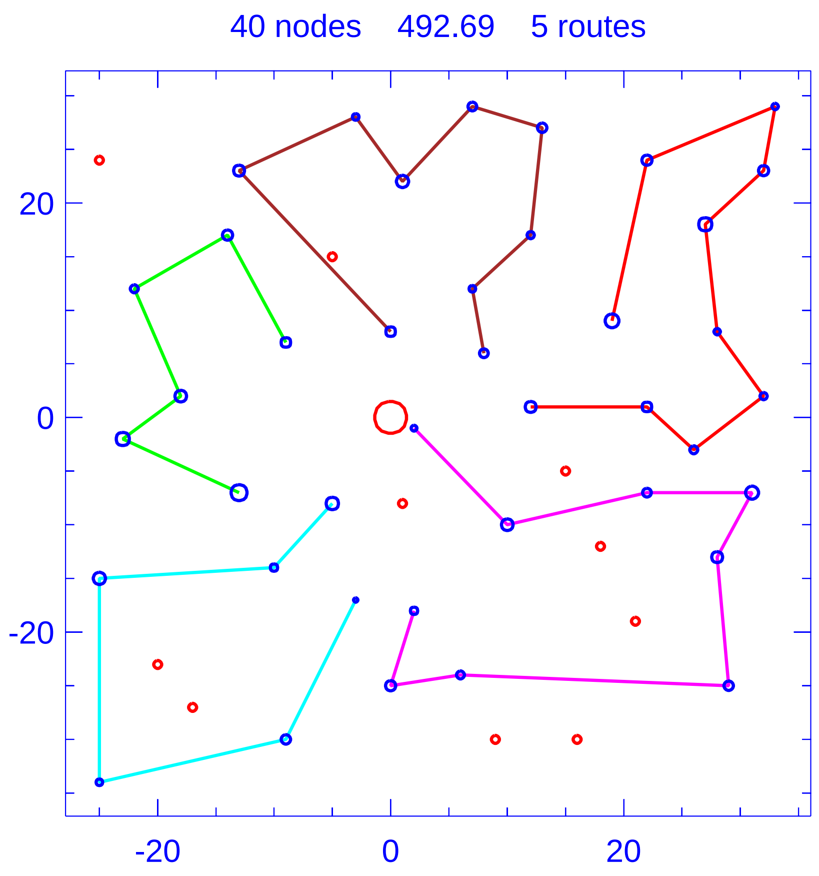


Figure A.34: Solution for day 3 of problem ConVRP-1

Problem			ConVRP-1
Vehicle capacity			160
Maximum route length			N/A
Number of nodes			40
Total route length			492.693
Total number of routes			5
Route	Length	Load	Ordering
1	114.327	155	(0, 2, 3, 36, 35, 20, 29, 21, 50, 16, 11, 0)
2	75.135	131	(0, 6, 23, 24, 14, 25, 18, 0)
3	99.350	80	(0, 17, 42, 40, 13, 4, 47, 0)
4	94.989	116	(0, 27, 7, 26, 8, 31, 28, 22, 1, 32, 0)
5	108.892	131	(0, 37, 44, 15, 39, 30, 34, 9, 5, 46, 0)

Table A.34: Details of solution to day 3 of problem ConVRP-1

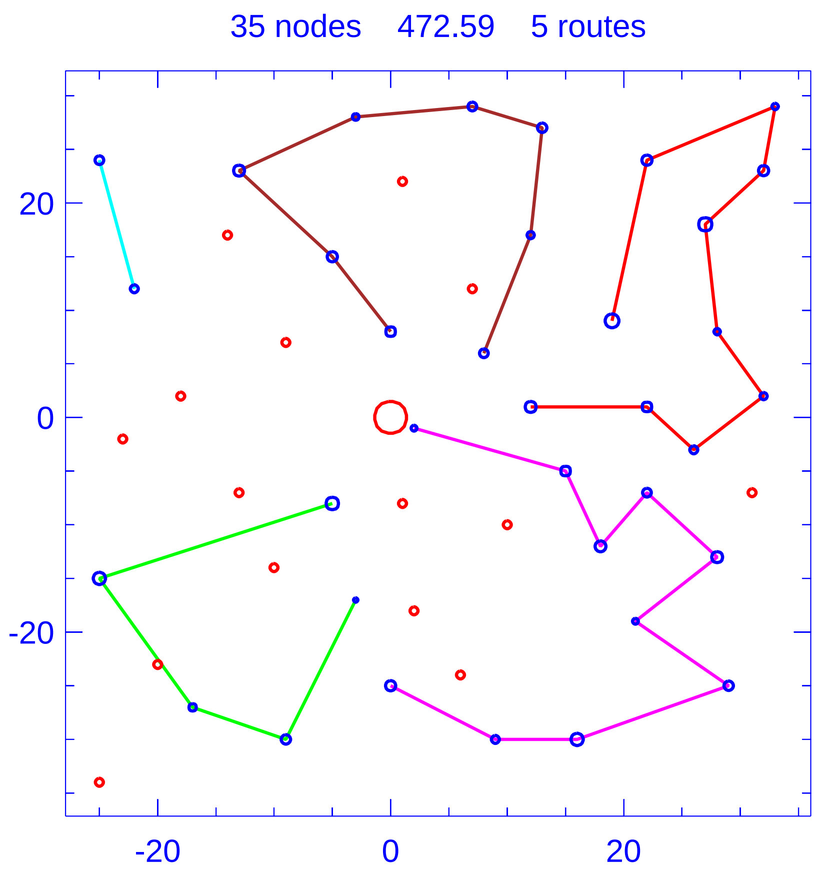


Figure A.35: Solution for day 4 of problem ConVRP-1

Problem			ConVRP-1
Vehicle capacity			160
Maximum route length			N/A
Number of nodes			35
Total route length			472.593
Total number of routes			5
Route	Length	Load	Ordering
1	114.327	155	(0, 2, 3, 36, 35, 20, 29, 21, 50, 16, 11, 0)
2	85.170	73	(0, 17, 42, 19, 13, 47, 0)
3	72.085	21	(0, 24, 43, 0)
4	87.225	103	(0, 27, 48, 7, 26, 31, 28, 22, 32, 0)
5	113.785	136	(0, 44, 45, 33, 39, 10, 30, 9, 49, 38, 46, 0)

Table A.35: Details of solution to day 4 of problem ConVRP-1

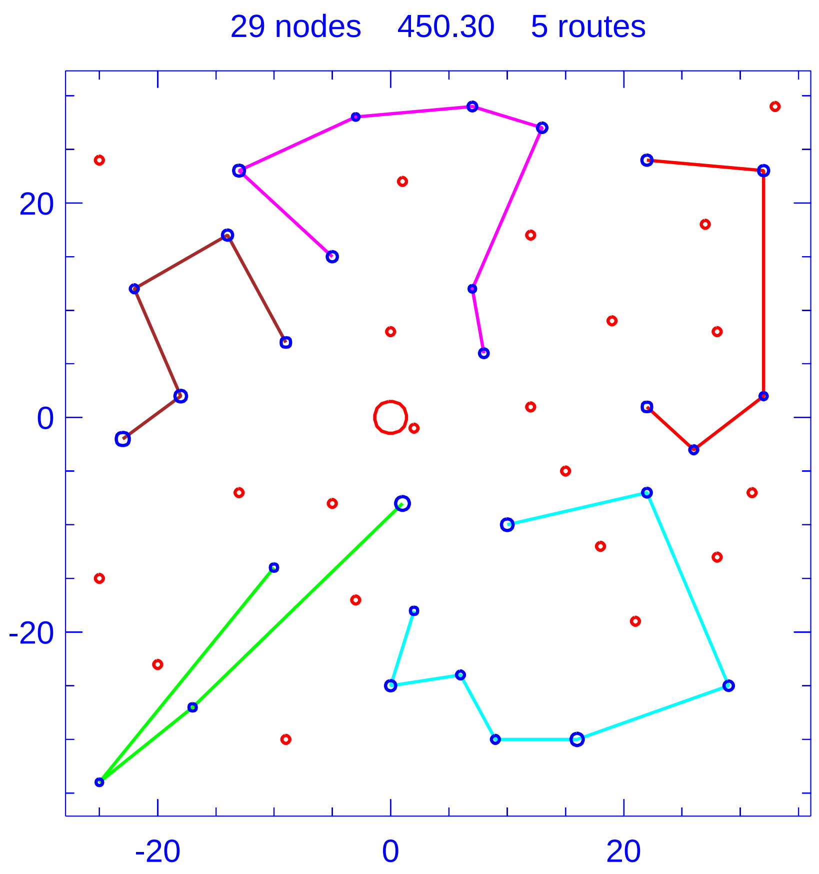


Figure A.36: Solution for day 5 of problem ConVRP-1

Problem			ConVRP-1
Vehicle capacity			160
Maximum route length			N/A
Number of nodes			29
Total route length			450.296
Total number of routes			5
Route	Length	Load	Ordering
1	99.097	66	(0, 3, 35, 21, 50, 16, 0)
2	87.070	54	(0, 4, 40, 19, 12, 0)
3	104.935	114	(0, 5, 9, 39, 33, 45, 15, 44, 37, 0)
4	72.276	90	(0, 6, 23, 24, 14, 25, 0)
5	86.918	87	(0, 32, 1, 28, 31, 26, 7, 48, 0)

Table A.36: Details of solution to day 5 of problem ConVRP-1

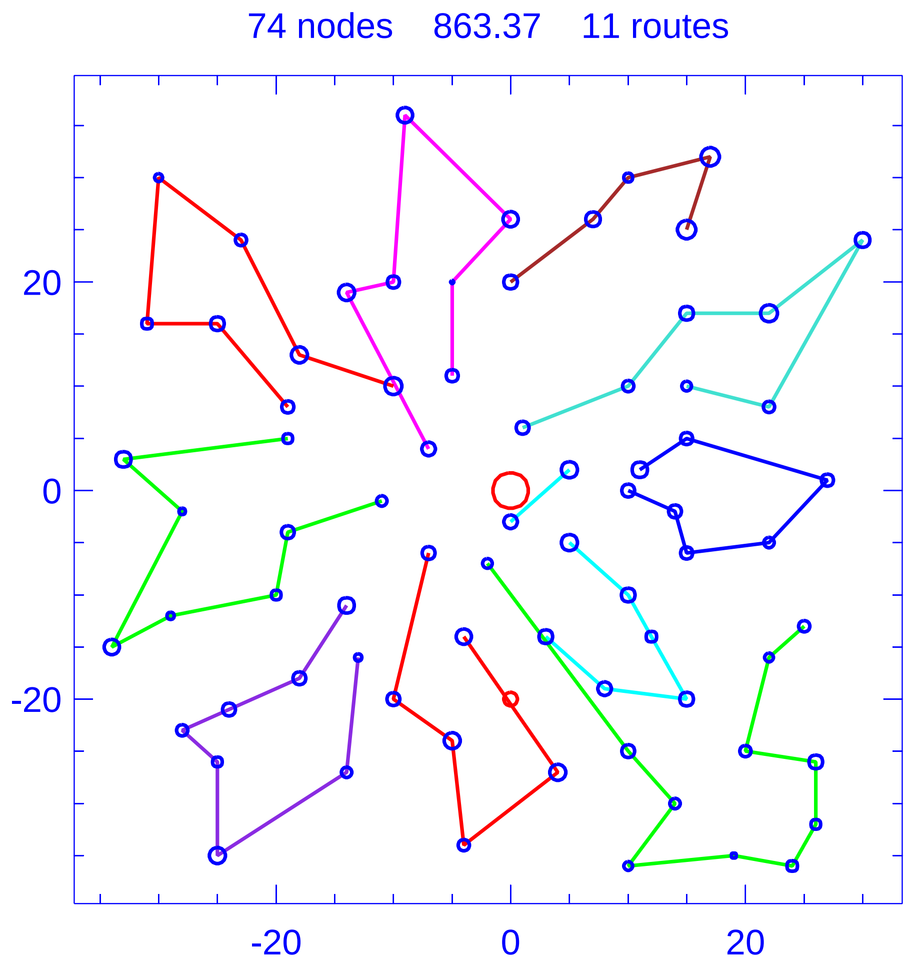


Figure A.37: Template for problem ConVRP-2

Problem			ConVRP-2
Vehicle capacity			140
Maximum route length			N/A
Number of nodes			74
Total route length			863.371
Total number of routes			11
Route	Length	Load	Ordering
1	80.445	135	(0, 2, 21, 61, 28, 62, 6, 0)
2	95.901	117	(0, 3, 24, 49, 56, 23, 63, 16, 51, 0)
3	53.782	127	(0, 4, 45, 29, 5, 48, 30, 0)
4	77.935	128	(0, 11, 66, 65, 38, 58, 0)
5	87.116	133	(0, 12, 72, 10, 31, 39, 9, 17, 0)
6	85.361	135	(0, 26, 7, 53, 14, 59, 19, 35, 0)
7	95.552	135	(0, 33, 1, 43, 41, 42, 64, 22, 73, 0)
8	62.306	126	(0, 34, 52, 27, 13, 54, 8, 46, 0)
9	94.640	134	(0, 40, 32, 25, 55, 18, 50, 44, 0)

Route	Length	Load	Ordering
10	114.877	134	(0, 57, 15, 37, 20, 70, 60, 71, 69, 36, 47, 68, 0)
11	15.456	50	(0, 67, 75, 0)

Table A.37: Details of template for problem ConVRP-2

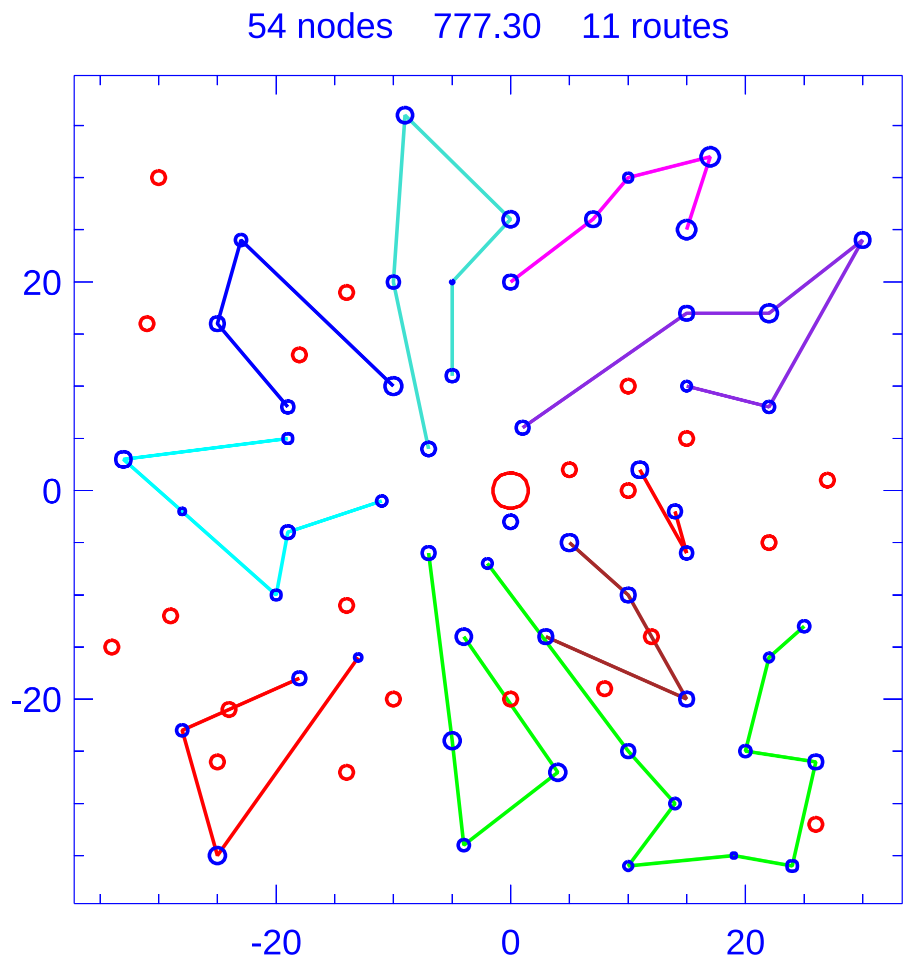


Figure A.38: Solution for day 1 of problem ConVRP-2

Problem			ConVRP-2
Vehicle capacity			140
Maximum route length			N/A
Number of nodes			54
Total route length			777.300
Total number of routes			11
Route	Length	Load	Ordering
1	92.093	67	(0, 1, 41, 64, 73, 0)
2	77.835	117	(0, 2, 21, 61, 28, 6, 0)
3	77.846	85	(0, 3, 24, 49, 63, 16, 51, 0)
4	53.057	94	(0, 4, 45, 5, 30, 0)
5	77.935	128	(0, 11, 66, 65, 38, 58, 0)
6	82.719	104	(0, 12, 72, 10, 31, 39, 17, 0)
7	84.714	120	(0, 26, 53, 14, 59, 19, 35, 0)
8	72.109	86	(0, 40, 25, 50, 44, 0)
9	38.390	63	(0, 46, 27, 52, 0)

Route	Length	Load	Ordering
10	114.603	123	(0, 57, 15, 37, 20, 60, 71, 69, 36, 47, 68, 0)
11	6.000	20	(0, 75, 0)

Table A.38: Details of solution to day 1 of problem ConVRP-2

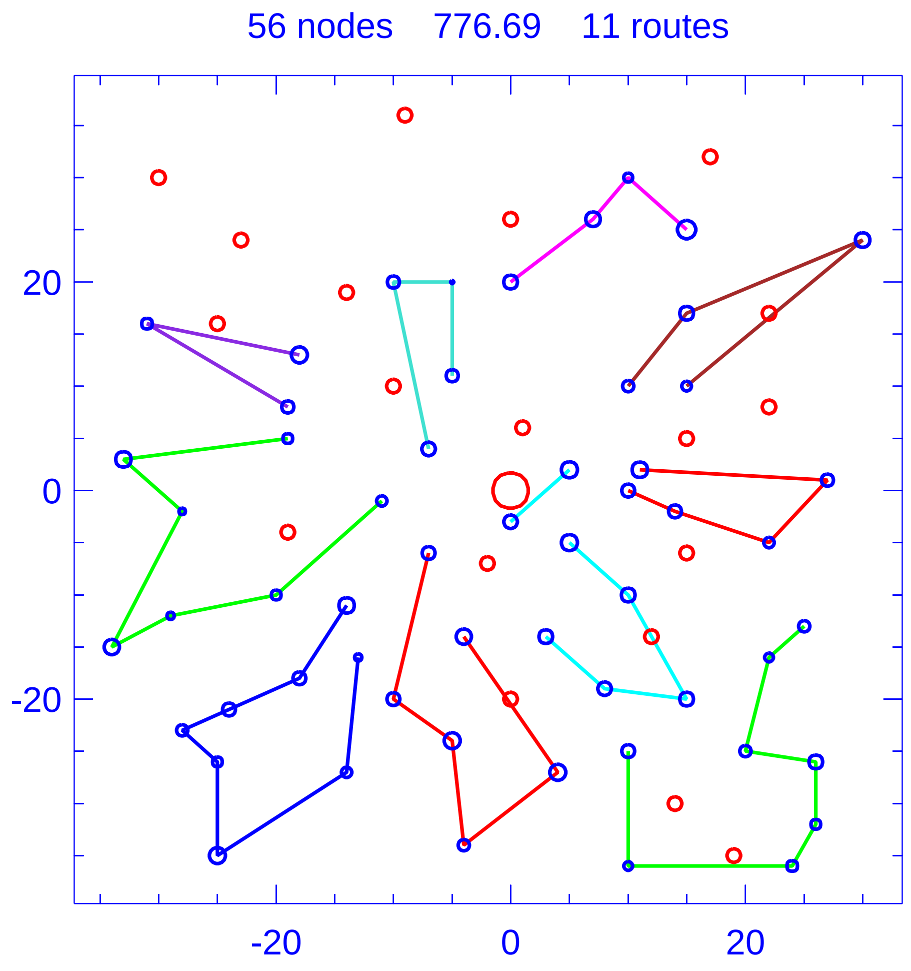


Figure A.39: Solution for day 2 of problem ConVRP-2

Problem			ConVRP-2
Vehicle capacity			140
Maximum route length			N/A
Number of nodes			56
Total route length			776.692
Total number of routes			11
Route	Length	Load	Ordering
1	80.445	135	(0, 2, 21, 61, 28, 62, 6, 0)
2	94.002	98	(0, 3, 24, 49, 56, 23, 63, 51, 0)
3	53.782	114	(0, 4, 45, 5, 48, 30, 0)
4	77.843	71	(0, 7, 53, 59, 35, 0)
5	70.445	91	(0, 11, 65, 38, 58, 0)
6	50.424	53	(0, 12, 72, 39, 17, 0)
7	70.583	58	(0, 32, 18, 44, 0)
8	95.552	135	(0, 33, 1, 43, 41, 42, 64, 22, 73, 0)
9	58.038	93	(0, 34, 52, 13, 54, 46, 0)

Route	Length	Load	Ordering
10	110.121	109	(0, 47, 69, 60, 70, 20, 37, 15, 57, 0)
11	15.456	50	(0, 67, 75, 0)

Table A.39: Details of solution to day 2 of problem ConVRP-2

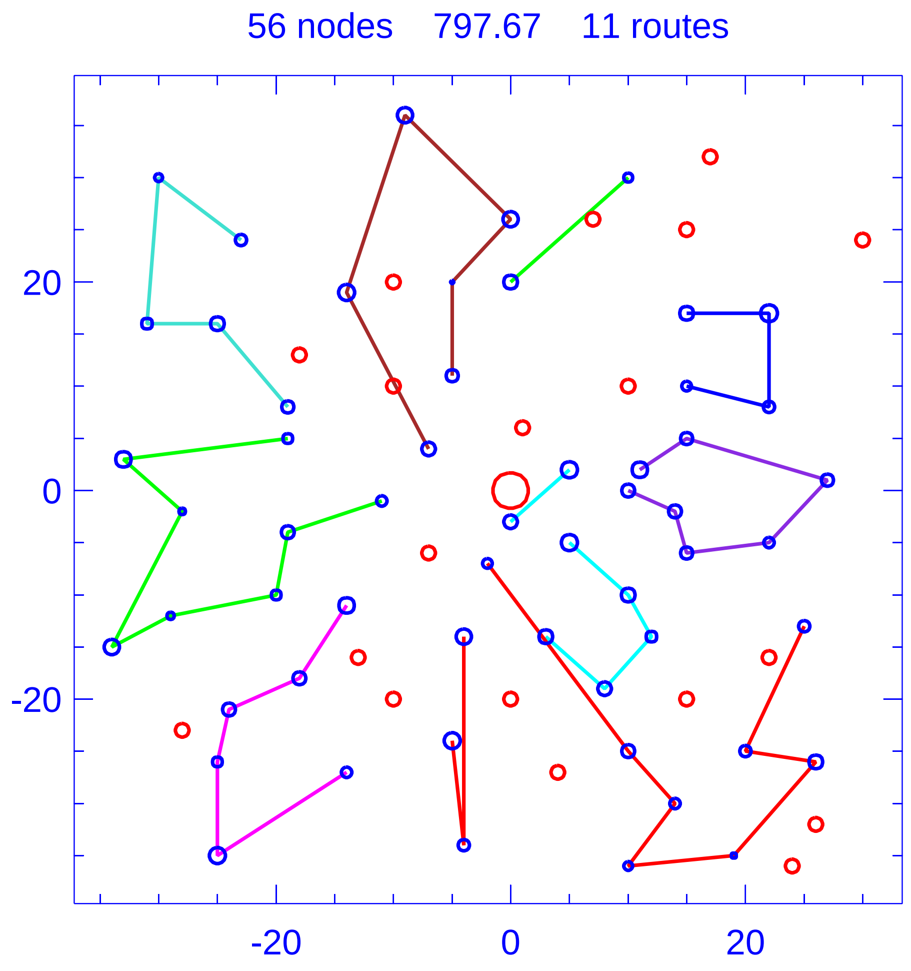


Figure A.40: Solution for day 3 of problem ConVRP-2

Problem			ConVRP-2
Vehicle capacity			140
Maximum route length			N/A
Number of nodes			56
Total route length			797.668
Total number of routes			11
Route	Length	Load	Ordering
1	69.125	70	(0, 2, 61, 28, 0)
2	95.901	117	(0, 3, 24, 49, 56, 23, 63, 16, 51, 0)
3	46.406	106	(0, 4, 45, 29, 48, 30, 0)
4	84.682	117	(0, 12, 72, 10, 31, 9, 17, 0)
5	90.689	114	(0, 22, 64, 42, 43, 1, 33, 0)
6	93.112	73	(0, 25, 55, 18, 50, 44, 0)
7	62.306	126	(0, 34, 52, 27, 13, 54, 8, 46, 0)
8	63.979	78	(0, 35, 19, 14, 53, 0)
9	110.246	102	(0, 57, 37, 20, 71, 69, 36, 47, 68, 0)

Route	Length	Load	Ordering
10	65.765	30	(0, 58, 65, 0)
11	15.456	50	(0, 67, 75, 0)

Table A.40: Details of solution to day 3 of problem ConVRP-2

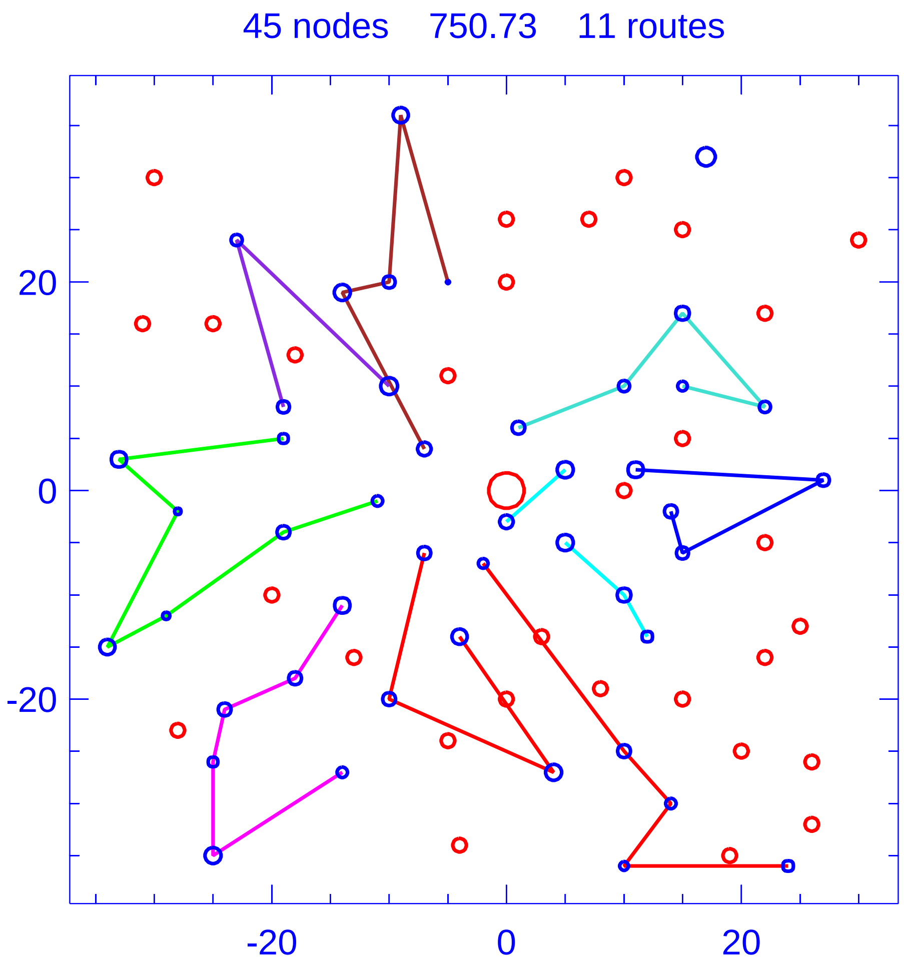


Figure A.41: Solution for day 4 of problem ConVRP-2

Problem			ConVRP-2
Vehicle capacity			140
Maximum route length			N/A
Number of nodes			45
Total route length			750.728
Total number of routes			11
Route	Length	Load	Ordering
1	69.014	91	(0, 2, 21, 62, 6, 0)
2	93.404	106	(0, 3, 24, 49, 56, 23, 16, 51, 0)
3	37.053	64	(0, 4, 45, 29, 0)
4	81.877	91	(0, 17, 9, 39, 31, 72, 0)
5	90.689	114	(0, 22, 64, 42, 43, 1, 33, 0)
6	61.244	80	(0, 26, 7, 53, 19, 35, 0)
7	70.355	64	(0, 40, 25, 44, 0)
8	59.369	79	(0, 46, 54, 27, 52, 0)
9	99.794	62	(0, 60, 69, 36, 47, 68, 0)

Route	Length	Load	Ordering
10	72.471	37	(0, 66, 0)
11	15.456	50	(0, 67, 75, 0)

Table A.41: Details of solution to day 4 of problem ConVRP-2

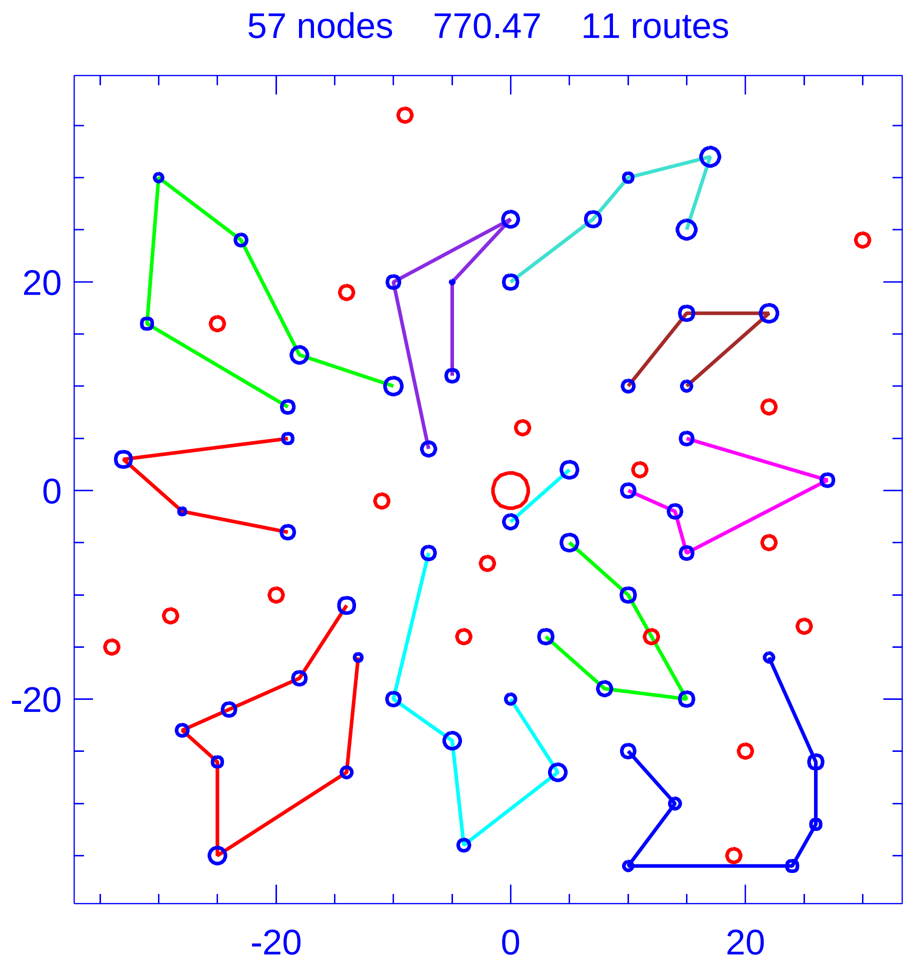


Figure A.42: Solution for day 5 of problem ConVRP-2

Problem			ConVRP-2
Vehicle capacity			140
Maximum route length			N/A
Number of nodes			57
Total route length			770.468
Total number of routes			11
Route	Length	Load	Ordering
1	69.496	62	(0, 3, 24, 49, 16, 0)
2	53.782	114	(0, 4, 45, 5, 48, 30, 0)
3	78.683	119	(0, 6, 62, 28, 61, 21, 74, 0)
4	57.672	78	(0, 7, 53, 14, 35, 0)
5	60.948	87	(0, 8, 54, 27, 52, 34, 0)
6	77.935	128	(0, 11, 66, 65, 38, 58, 0)
7	64.896	79	(0, 12, 72, 10, 39, 17, 0)
8	102.985	93	(0, 15, 20, 70, 60, 69, 36, 47, 0)
9	95.552	135	(0, 33, 1, 43, 41, 42, 64, 22, 73, 0)

Route	Length	Load	Ordering
10	93.062	112	(0, 40, 32, 25, 55, 18, 44, 0)
11	15.456	50	(0, 67, 75, 0)

Table A.42: Details of solution to day 5 of problem ConVRP-2

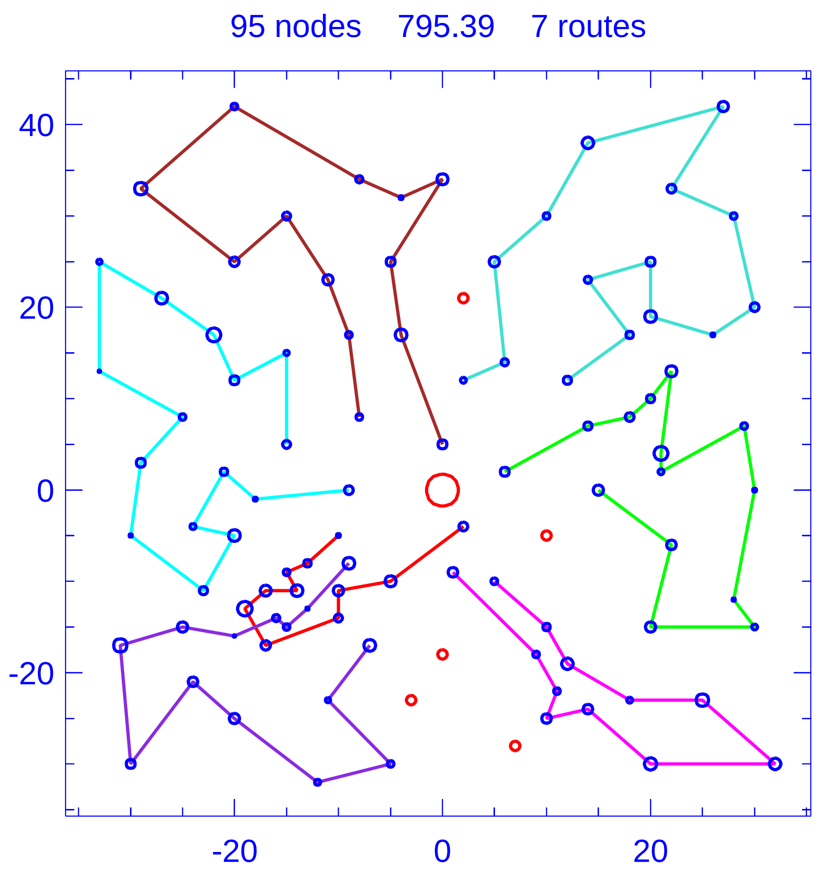


Figure A.43: Template for problem ConVRP-3

Problem			ConVRP-3
Vehicle capacity			200
Maximum route length			N/A
Number of nodes			95
Total route length			795.394
Total number of routes			7
Route	Length	Load	Ordering
1	59.602	200	(0, 6, 96, 99, 59, 93, 85, 100, 97, 95, 13, 53, 0)
2	113.086	197	(0, 12, 54, 4, 25, 55, 24, 29, 80, 68, 79, 3, 77, 76, 28, 0)
3	135.598	204	(0, 18, 7, 82, 48, 47, 36, 46, 8, 45, 17, 61, 5, 84, 83, 60, 89, 0)
4	125.970	200	(0, 27, 31, 10, 32, 90, 63, 64, 49, 19, 11, 62, 88, 52, 0)
5	100.174	207	(0, 40, 21, 72, 56, 39, 67, 23, 75, 22, 74, 73, 58, 0)
6	140.143	207	(0, 50, 33, 51, 9, 81, 78, 34, 35, 71, 65, 66, 20, 30, 1, 69, 0)
7	120.821	202	(0, 87, 42, 15, 43, 14, 44, 38, 86, 16, 91, 98, 37, 92, 94, 0)

Table A.43: Details of template for problem ConVRP-3

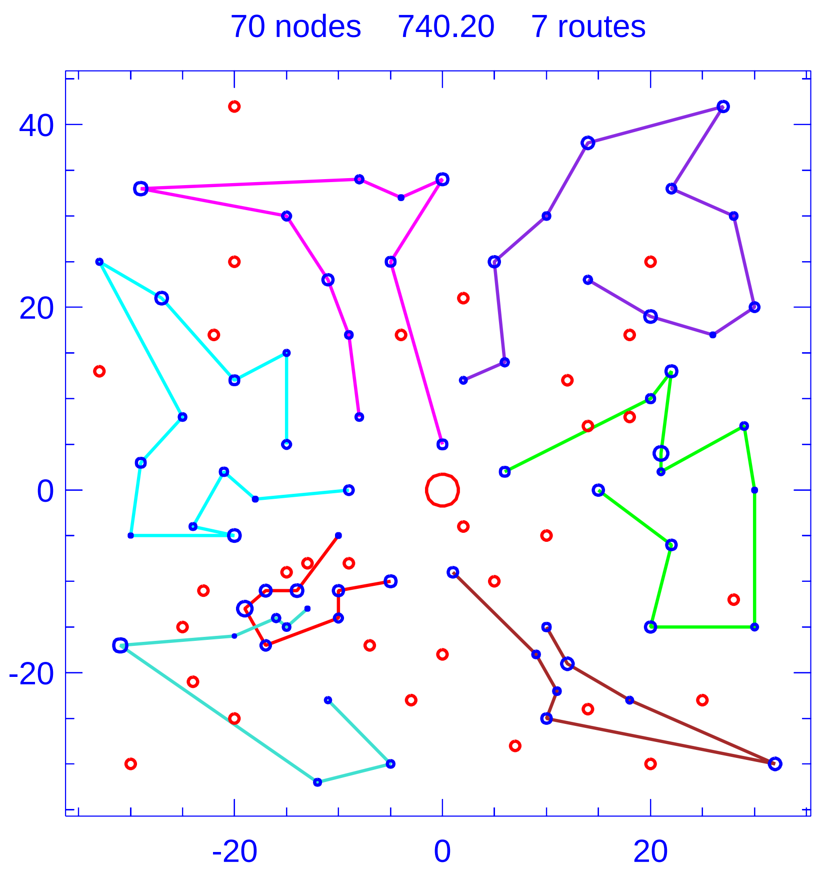


Figure A.44: Solution for day 1 of problem ConVRP-3

Problem			ConVRP-3
Vehicle capacity			200
Maximum route length			N/A
Number of nodes			70
Total route length			740.199
Total number of routes			7
Route	Length	Load	Ordering
1	55.587	166	(0, 6, 59, 93, 85, 100, 97, 95, 13, 0)
2	112.054	168	(0, 12, 54, 4, 25, 24, 29, 80, 68, 79, 3, 28, 0)
3	126.638	154	(0, 18, 7, 82, 47, 36, 8, 45, 17, 5, 84, 83, 60, 89, 0)
4	96.656	120	(0, 21, 72, 56, 67, 22, 74, 73, 58, 0)
5	114.953	147	(0, 27, 10, 32, 90, 63, 49, 11, 62, 88, 52, 0)
6	104.347	76	(0, 42, 15, 43, 86, 91, 98, 37, 92, 0)
7	129.963	167	(0, 51, 81, 78, 34, 35, 71, 65, 66, 20, 30, 1, 69, 0)

Table A.44: Details of solution to day 1 of problem ConVRP-3

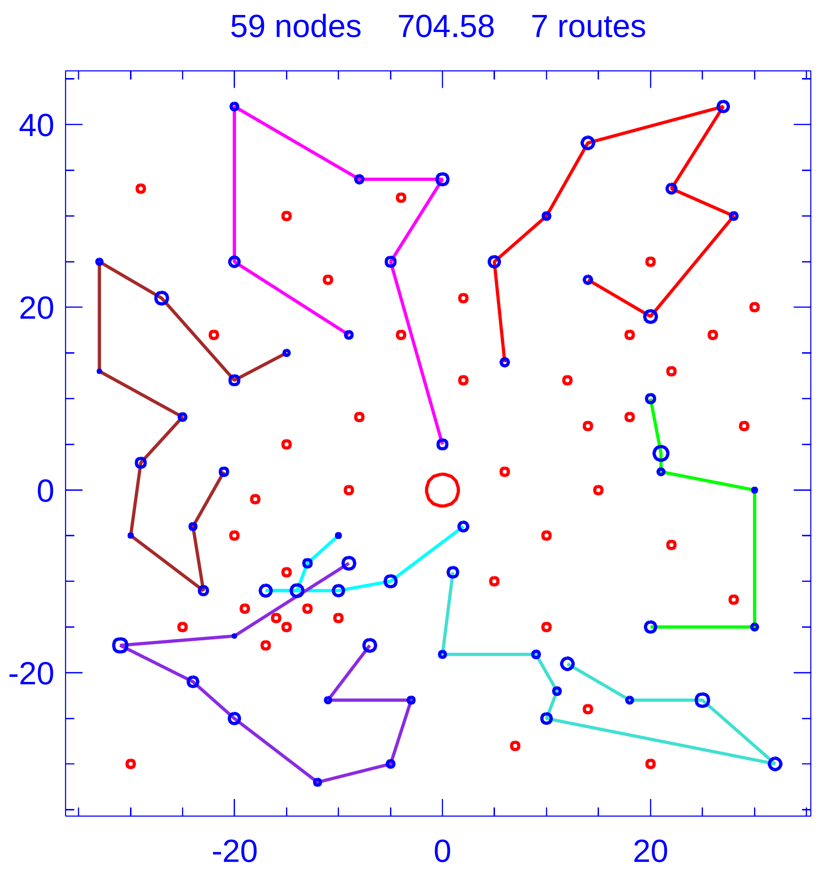


Figure A.45: Solution for day 2 of problem ConVRP-3

Problem			ConVRP-3
Vehicle capacity			200
Maximum route length			N/A
Number of nodes			59
Total route length			704.578
Total number of routes			7
Route	Length	Load	Ordering
1	120.636	144	(0, 1, 30, 20, 66, 65, 71, 35, 81, 51, 0)
2	89.663	83	(0, 3, 68, 80, 24, 25, 4, 0)
3	47.376	121	(0, 6, 96, 59, 93, 95, 13, 53, 0)
4	125.650	112	(0, 7, 82, 47, 36, 46, 8, 45, 17, 61, 84, 83, 0)
5	108.170	100	(0, 27, 10, 32, 63, 64, 19, 88, 0)
6	103.889	147	(0, 58, 2, 73, 74, 22, 67, 39, 56, 72, 0)
7	109.194	154	(0, 87, 42, 57, 15, 43, 14, 44, 86, 91, 94, 0)

Table A.45: Details of solution to day 2 of problem ConVRP-3

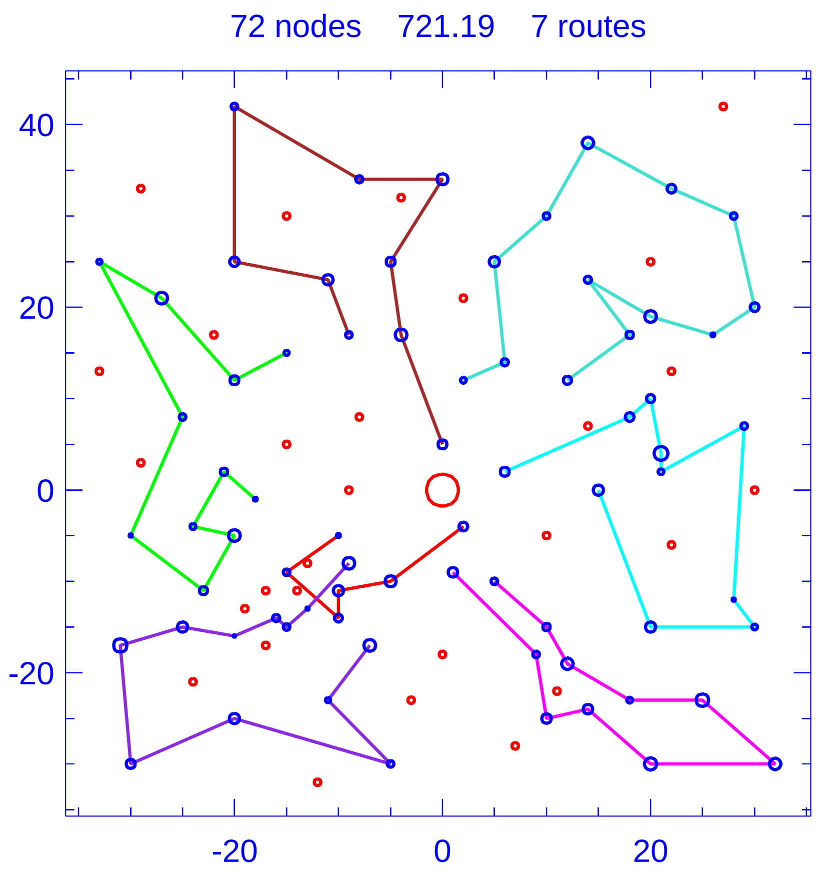


Figure A.46: Solution for day 3 of problem ConVRP-3

Problem			ConVRP-3
Vehicle capacity			200
Maximum route length			N/A
Number of nodes			72
Total route length			721.192
Total number of routes			7
Route	Length	Load	Ordering
1	46.445	81	(0, 6, 99, 97, 95, 13, 53, 0)
2	127.403	124	(0, 7, 82, 47, 36, 8, 17, 61, 5, 84, 83, 60, 0)
3	103.529	140	(0, 12, 4, 25, 55, 29, 80, 68, 3, 77, 28, 0)
4	110.209	146	(0, 27, 31, 10, 32, 63, 64, 19, 62, 88, 0)
5	99.611	199	(0, 40, 21, 72, 56, 39, 67, 23, 75, 22, 73, 58, 0)
6	120.566	171	(0, 50, 33, 51, 81, 78, 34, 35, 71, 66, 20, 30, 1, 69, 0)
7	113.429	177	(0, 87, 42, 15, 14, 38, 86, 16, 91, 98, 37, 92, 94, 0)

Table A.46: Details of solution to day 3 of problem ConVRP-3

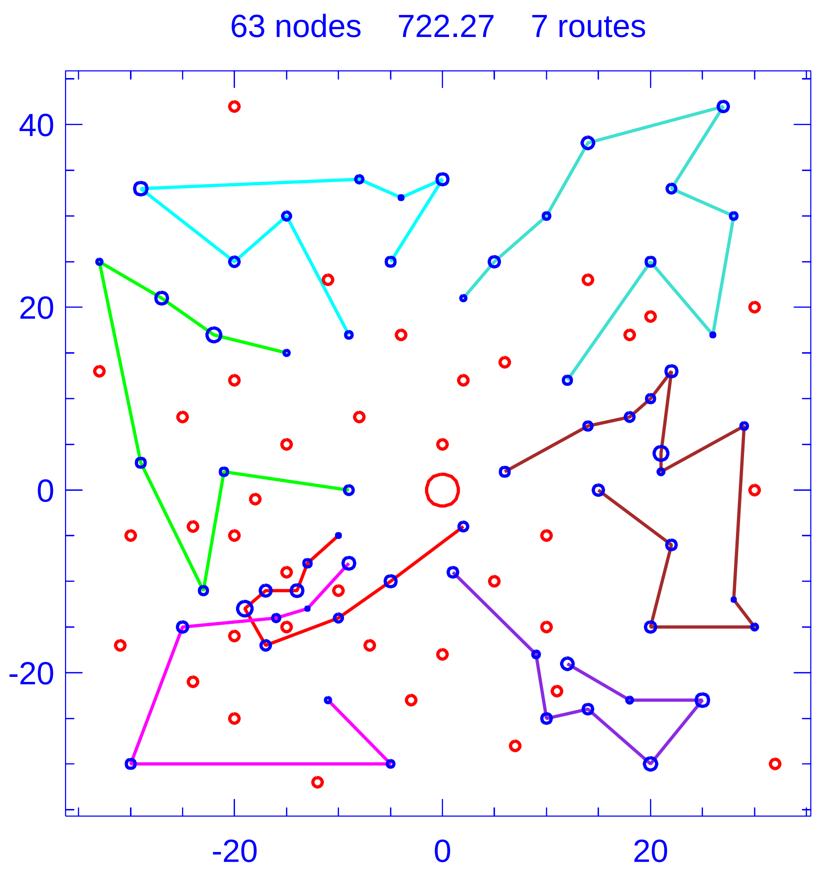


Figure A.47: Solution for day 4 of problem ConVRP-3

Problem			ConVRP-3
Vehicle capacity			200
Maximum route length			N/A
Number of nodes			63
Total route length			722.269
Total number of routes			7
Route	Length	Load	Ordering
1	56.596	171	(0, 6, 96, 59, 93, 85, 100, 97, 13, 53, 0)
2	114.018	128	(0, 7, 48, 47, 36, 45, 61, 83, 89, 0)
3	118.425	120	(0, 10, 32, 90, 63, 49, 19, 11, 88, 0)
4	112.876	194	(0, 12, 54, 4, 25, 55, 29, 80, 68, 79, 3, 77, 76, 28, 0)
5	106.188	87	(0, 42, 15, 38, 16, 98, 92, 94, 0)
6	128.104	135	(0, 50, 9, 78, 35, 71, 65, 66, 20, 30, 70, 0)
7	86.062	154	(0, 58, 73, 22, 75, 23, 39, 56, 72, 0)

Table A.47: Details of solution to day 4 of problem ConVRP-3

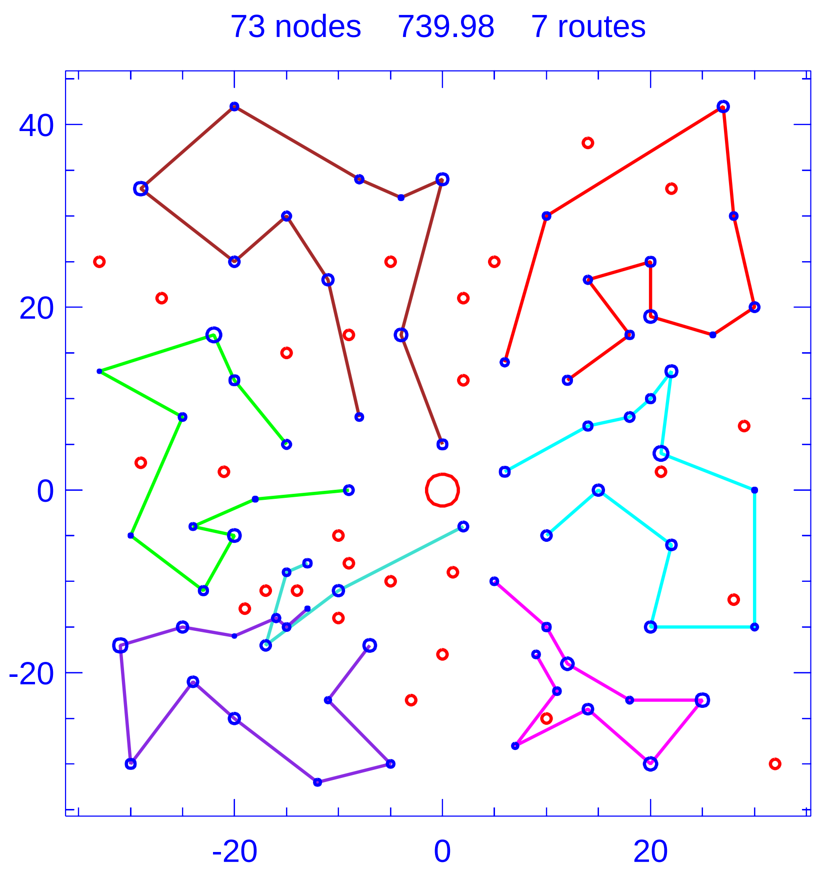


Figure A.48: Solution for day 5 of problem ConVRP-3

Problem			ConVRP-3
Vehicle capacity			200
Maximum route length			N/A
Number of nodes			73
Total route length			739.981
Total number of routes			7
Route	Length	Load	Ordering
1	130.413	140	(0, 1, 20, 65, 35, 34, 78, 81, 9, 51, 33, 50, 0)
2	109.680	140	(0, 18, 82, 48, 46, 8, 17, 61, 5, 84, 60, 89, 0)
3	106.910	197	(0, 26, 12, 54, 4, 25, 24, 68, 79, 3, 77, 76, 28, 0)
4	124.993	175	(0, 27, 31, 32, 90, 63, 64, 49, 19, 11, 62, 52, 0)
5	93.892	151	(0, 40, 21, 72, 56, 39, 23, 75, 41, 74, 73, 0)
6	53.331	71	(0, 53, 95, 100, 99, 96, 0)
7	120.761	175	(0, 87, 42, 15, 43, 14, 44, 38, 86, 16, 91, 98, 37, 92, 0)

Table A.48: Details of solution to day 5 of problem ConVRP-3

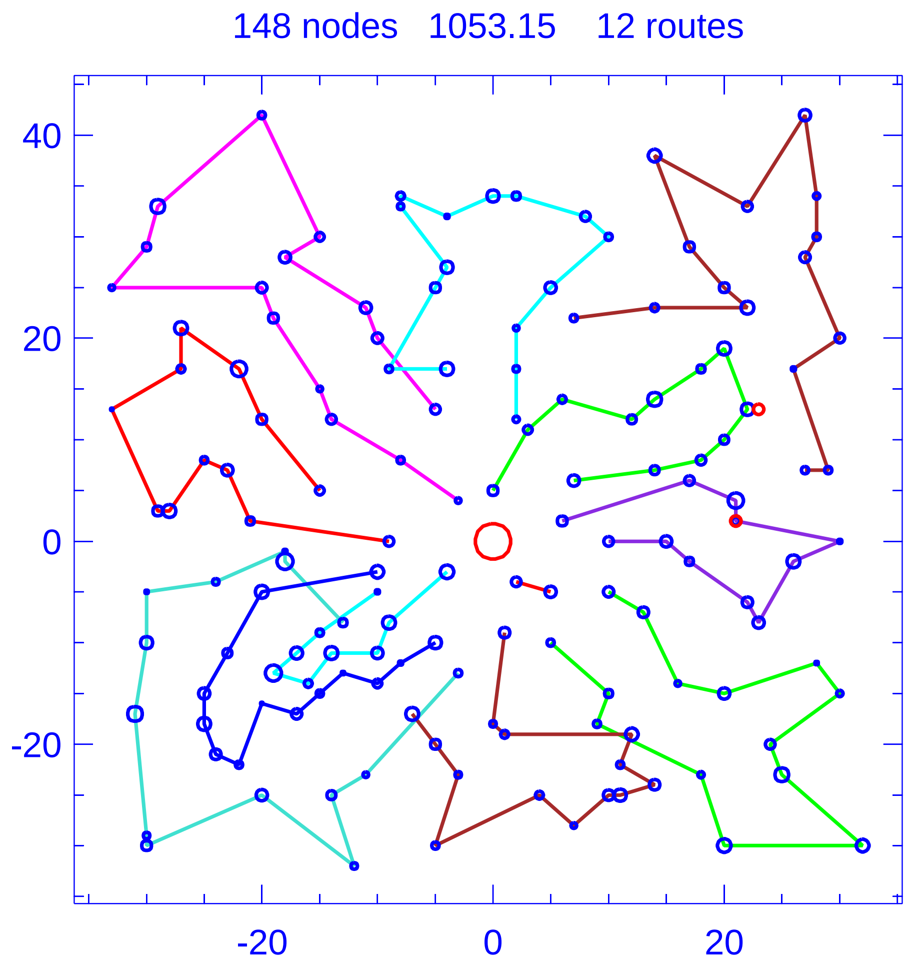


Figure A.49: Template for problem ConVRP-4

Problem			ConVRP-4
Vehicle capacity			200
Maximum route length			N/A
Number of nodes			148
Total route length			1053.149
Total number of routes			12
Route	Length	Load	Ordering
1	14.705	35	(0, 5, 103, 0)
2	66.105	200	(0, 11, 126, 127, 53, 129, 131, 83, 2, 100, 51, 32, 77, 0)
3	49.241	198	(0, 12, 144, 145, 109, 148, 135, 143, 149, 4, 56, 0)
4	143.378	203	(0, 21, 79, 128, 84, 35, 85, 36, 115, 121, 116, 3, 59, 20, 101, 22, 0)
5	118.753	204	(0, 27, 48, 112, 7, 61, 114, 99, 43, 86, 69, 23, 57, 6, 102, 46, 0)
6	120.822	199	(0, 37, 92, 42, 93, 64, 88, 40, 136, 13, 67, 134, 110, 18, 146, 0)
7	71.369	188	(0, 38, 62, 9, 104, 30, 34, 74, 50, 118, 16, 78, 0)
8	76.137	203	(0, 47, 55, 111, 66, 41, 94, 19, 141, 150, 87, 142, 147, 17, 63, 0)
9	93.801	202	(0, 68, 132, 98, 97, 24, 96, 95, 25, 58, 14, 133, 139, 0)

Route	Length	Load	Ordering
10	110.536	201	(0, 76, 49, 10, 54, 105, 75, 39, 89, 117, 73, 106, 123, 71, 90, 0)
11	95.190	191	(0, 81, 138, 60, 8, 26, 113, 140, 82, 31, 28, 70, 80, 120, 1, 119, 0)
12	93.112	199	(0, 108, 52, 15, 122, 124, 125, 33, 72, 91, 45, 65, 107, 44, 137, 0)

Table A.49: Details of template for problem ConVRP-4

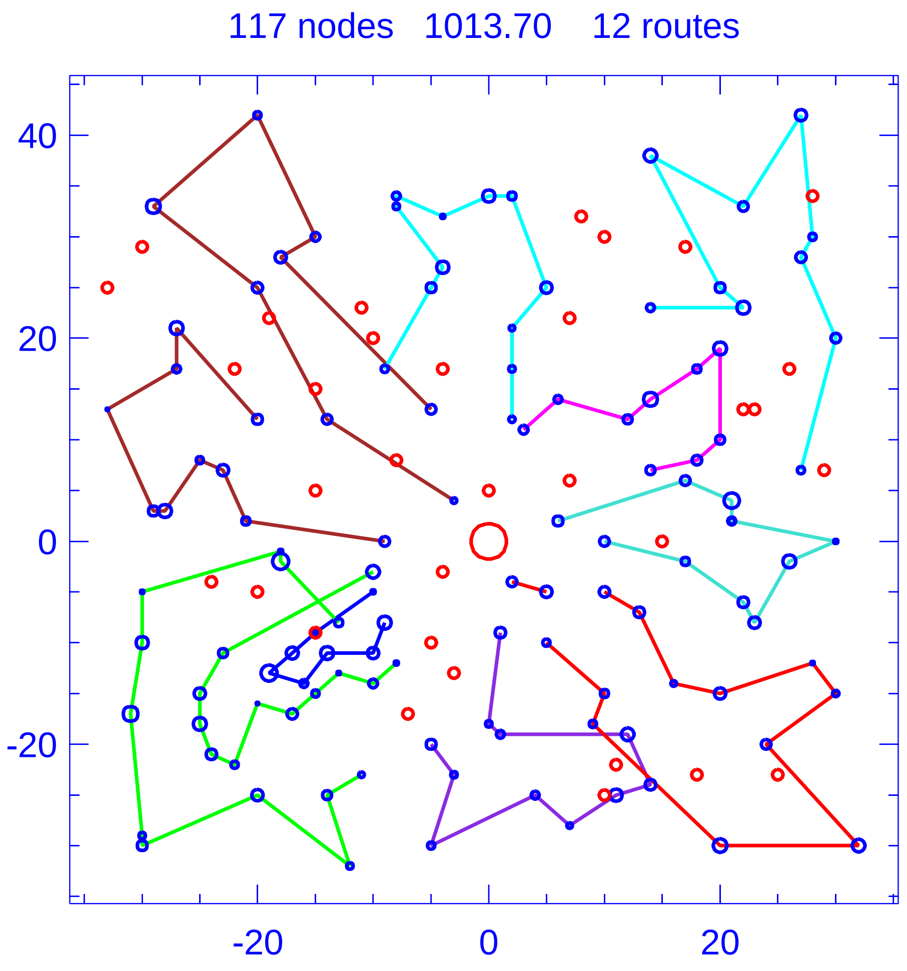


Figure A.50: Solution for day 1 of problem ConVRP-4

Problem			ConVRP-4
Vehicle capacity			200
Maximum route length			N/A
Number of nodes			117
Total route length			1013.700
Total number of routes			12
Route	Length	Load	Ordering
1	14.705	35	(0, 5, 103, 0)
2	74.132	154	(0, 17, 147, 142, 87, 150, 141, 19, 94, 41, 66, 111, 47, 0)
3	135.858	161	(0, 21, 84, 35, 85, 115, 121, 116, 59, 20, 101, 0)
4	108.072	122	(0, 27, 7, 61, 114, 99, 69, 6, 46, 0)
5	64.230	142	(0, 32, 51, 100, 2, 83, 131, 53, 127, 126, 0)
6	70.821	175	(0, 38, 9, 104, 30, 34, 74, 130, 50, 118, 16, 78, 0)
7	90.112	147	(0, 44, 107, 65, 45, 91, 33, 125, 122, 15, 52, 108, 0)
8	49.212	160	(0, 56, 4, 143, 135, 148, 109, 145, 144, 0)
9	108.983	164	(0, 76, 49, 10, 54, 105, 75, 39, 117, 73, 123, 71, 90, 0)

Route	Length	Load	Ordering
10	120.028	183	(0, 92, 42, 93, 64, 88, 40, 136, 13, 67, 110, 18, 146, 0)
11	85.223	141	(0, 119, 1, 120, 80, 31, 82, 140, 113, 26, 8, 60, 138, 0)
12	92.325	154	(0, 132, 97, 24, 96, 95, 25, 58, 14, 133, 139, 0)

Table A.50: Details of solution to day 1 of problem ConVRP-4

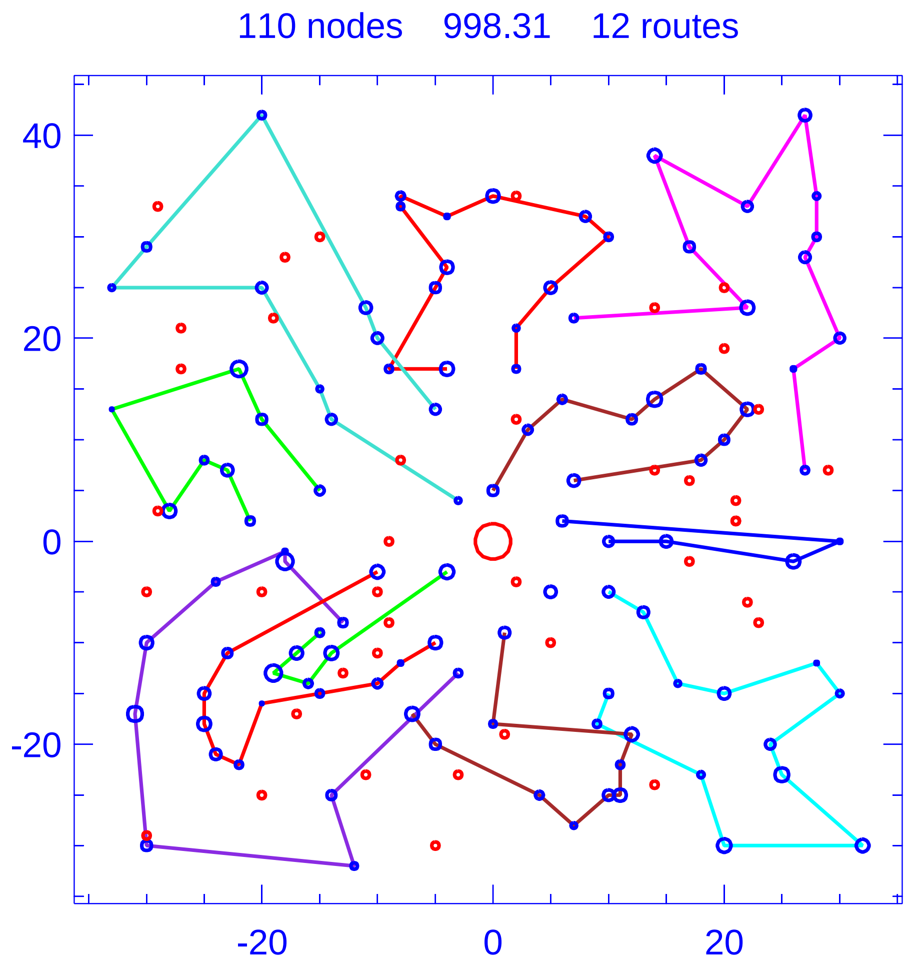


Figure A.51: Solution for day 2 of problem ConVRP-4

Problem			ConVRP-4
Vehicle capacity			200
Maximum route length			N/A
Number of nodes			110
Total route length			998.307
Total number of routes			12
Route	Length	Load	Ordering
1	95.063	174	(0, 1, 120, 80, 70, 28, 82, 140, 113, 26, 8, 60, 138, 81, 0)
2	47.724	148	(0, 4, 149, 143, 135, 148, 109, 12, 0)
3	14.142	21	(0, 5, 0)
4	62.595	161	(0, 11, 127, 53, 129, 83, 2, 100, 51, 32, 77, 0)
5	140.932	168	(0, 21, 128, 84, 35, 85, 36, 115, 121, 116, 3, 20, 22, 0)
6	114.062	118	(0, 27, 48, 112, 114, 43, 86, 69, 57, 6, 46, 0)
7	114.389	165	(0, 37, 42, 93, 88, 136, 13, 134, 110, 18, 146, 0)
8	61.060	79	(0, 38, 62, 34, 74, 78, 0)
9	72.712	158	(0, 47, 111, 66, 41, 94, 19, 141, 87, 147, 17, 63, 0)

Route	Length	Load	Ordering
10	87.231	134	(0, 68, 132, 98, 96, 25, 58, 14, 133, 0)
11	110.312	192	(0, 71, 123, 106, 73, 117, 89, 39, 75, 105, 54, 10, 49, 76, 0)
12	78.086	156	(0, 108, 52, 122, 124, 33, 72, 91, 45, 44, 137, 0)

Table A.51: Details of solution to day 2 of problem ConVRP-4

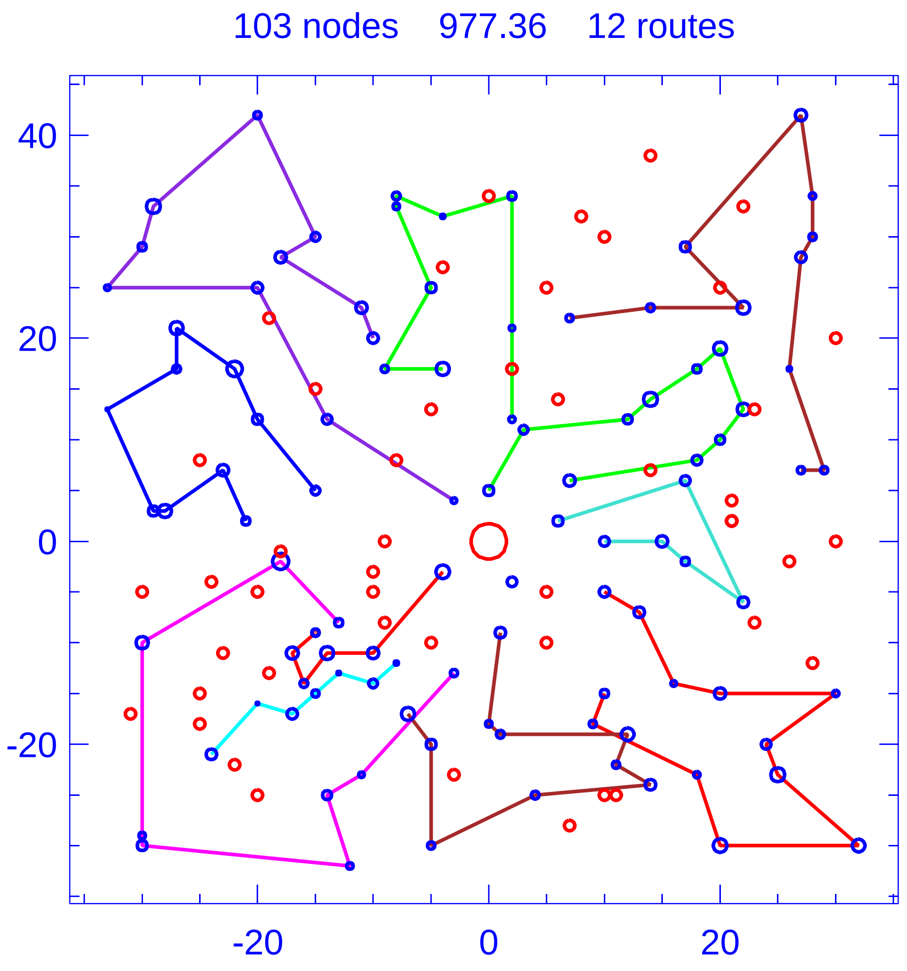


Figure A.52: Solution for day 3 of problem ConVRP-4

Problem			ConVRP-4
Vehicle capacity			200
Maximum route length			N/A
Number of nodes			103
Total route length			977.360
Total number of routes			12
Route	Length	Load	Ordering
1	46.089	127	(0, 4, 149, 143, 148, 109, 145, 12, 0)
2	64.579	177	(0, 11, 127, 53, 129, 131, 83, 2, 100, 32, 77, 0)
3	67.526	61	(0, 17, 147, 142, 87, 150, 141, 94, 0)
4	128.046	133	(0, 21, 79, 128, 35, 85, 36, 115, 3, 20, 101, 22, 0)
5	112.641	132	(0, 37, 92, 42, 93, 88, 40, 13, 18, 146, 0)
6	55.261	94	(0, 38, 62, 9, 104, 16, 78, 0)
7	118.501	159	(0, 46, 6, 69, 86, 43, 99, 114, 61, 7, 112, 48, 0)
8	92.067	178	(0, 68, 132, 98, 97, 24, 96, 95, 25, 14, 133, 0)
9	108.163	190	(0, 71, 123, 106, 73, 117, 89, 39, 75, 54, 10, 49, 76, 0)

Route	Length	Load	Ordering
10	85.915	94	(0, 81, 138, 60, 26, 113, 140, 31, 120, 119, 0)
11	8.944	14	(0, 103, 0)
12	89.629	146	(0, 108, 52, 15, 122, 124, 125, 45, 65, 44, 137, 0)

Table A.52: Details of solution to day 3 of problem ConVRP-4

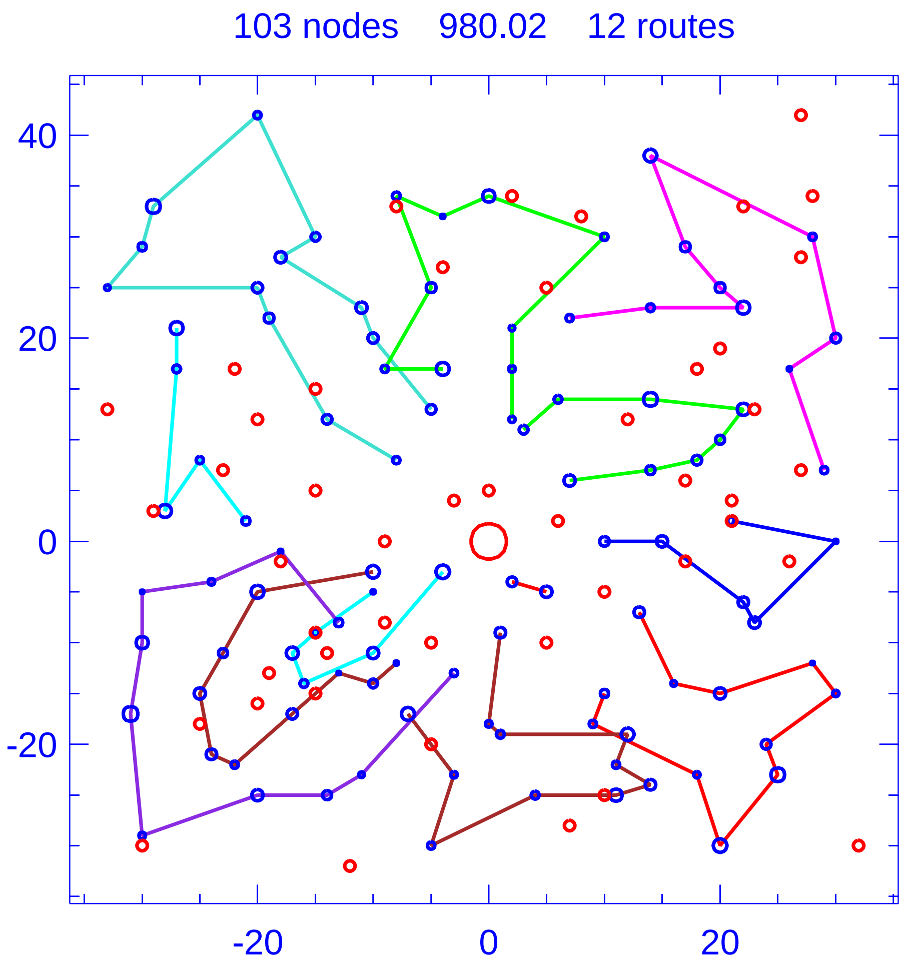


Figure A.53: Solution for day 4 of problem ConVRP-4

Problem			ConVRP-4
Vehicle capacity			200
Maximum route length			N/A
Number of nodes			103
Total route length			980.019
Total number of routes			12
Route	Length	Load	Ordering
1	14.705	35	(0, 5, 103, 0)
2	58.554	134	(0, 11, 126, 127, 53, 129, 2, 51, 32, 0)
3	45.282	93	(0, 12, 145, 148, 143, 149, 56, 0)
4	73.278	144	(0, 17, 147, 142, 150, 19, 94, 66, 111, 55, 47, 0)
5	127.069	137	(0, 22, 101, 20, 59, 3, 116, 85, 84, 128, 79, 0)
6	118.619	194	(0, 27, 48, 112, 7, 61, 114, 99, 43, 86, 69, 23, 6, 102, 0)
7	107.294	135	(0, 37, 92, 42, 64, 40, 136, 13, 67, 134, 110, 146, 0)
8	67.400	84	(0, 38, 62, 104, 30, 74, 50, 0)
9	96.994	150	(0, 49, 10, 54, 105, 75, 39, 89, 73, 106, 123, 71, 0)

Route	Length	Load	Ordering
10	93.817	115	(0, 81, 138, 60, 113, 140, 82, 70, 120, 1, 119, 0)
11	86.378	85	(0, 97, 24, 25, 58, 133, 0)
12	90.627	160	(0, 108, 52, 15, 122, 124, 125, 33, 45, 65, 107, 137, 0)

Table A.53: Details of solution to day 4 of problem ConVRP-4

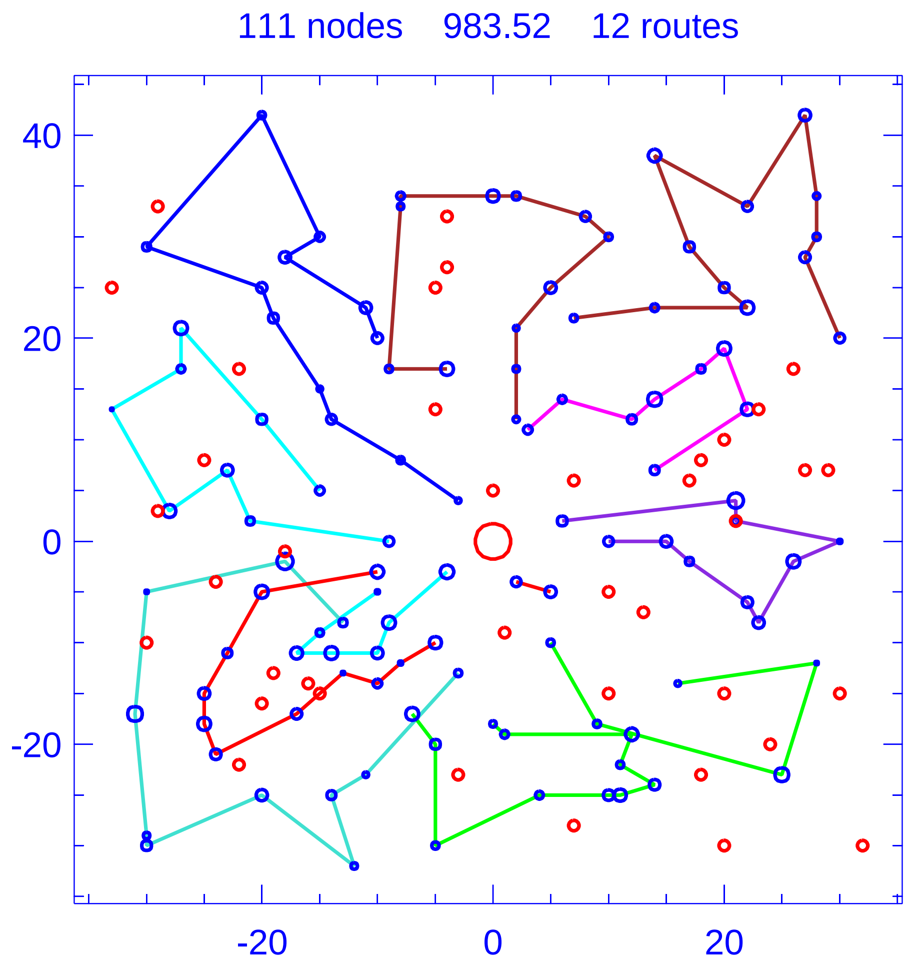


Figure A.54: Solution for day 5 of problem ConVRP-4

Problem			ConVRP-4
Vehicle capacity			200
Maximum route length			N/A
Number of nodes			111
Total route length			983.521
Total number of routes			12
Route	Length	Load	Ordering
1	14.705	35	(0, 5, 103, 0)
2	81.715	56	(0, 10, 105, 89, 123, 90, 0)
3	42.645	147	(0, 12, 144, 145, 109, 143, 149, 4, 56, 0)
4	134.101	183	(0, 22, 101, 20, 59, 3, 116, 121, 115, 36, 85, 35, 84, 0)
5	64.603	138	(0, 32, 51, 100, 2, 83, 131, 129, 126, 0)
6	119.371	166	(0, 37, 92, 42, 93, 64, 88, 40, 136, 67, 18, 146, 0)
7	70.325	173	(0, 38, 62, 9, 104, 30, 34, 74, 50, 118, 78, 0)
8	110.903	154	(0, 46, 102, 6, 57, 23, 69, 43, 114, 61, 7, 112, 48, 0)
9	72.477	185	(0, 47, 55, 111, 66, 41, 94, 150, 142, 147, 17, 63, 0)

Route	Length	Load	Ordering
10	89.630	169	(0, 52, 15, 122, 124, 125, 33, 72, 45, 65, 44, 137, 0)
11	91.161	141	(0, 68, 132, 97, 24, 96, 25, 14, 133, 139, 0)
12	91.885	149	(0, 81, 138, 26, 113, 82, 31, 28, 70, 80, 120, 1, 119, 0)

Table A.54: Details of solution to day 5 of problem ConVRP-4

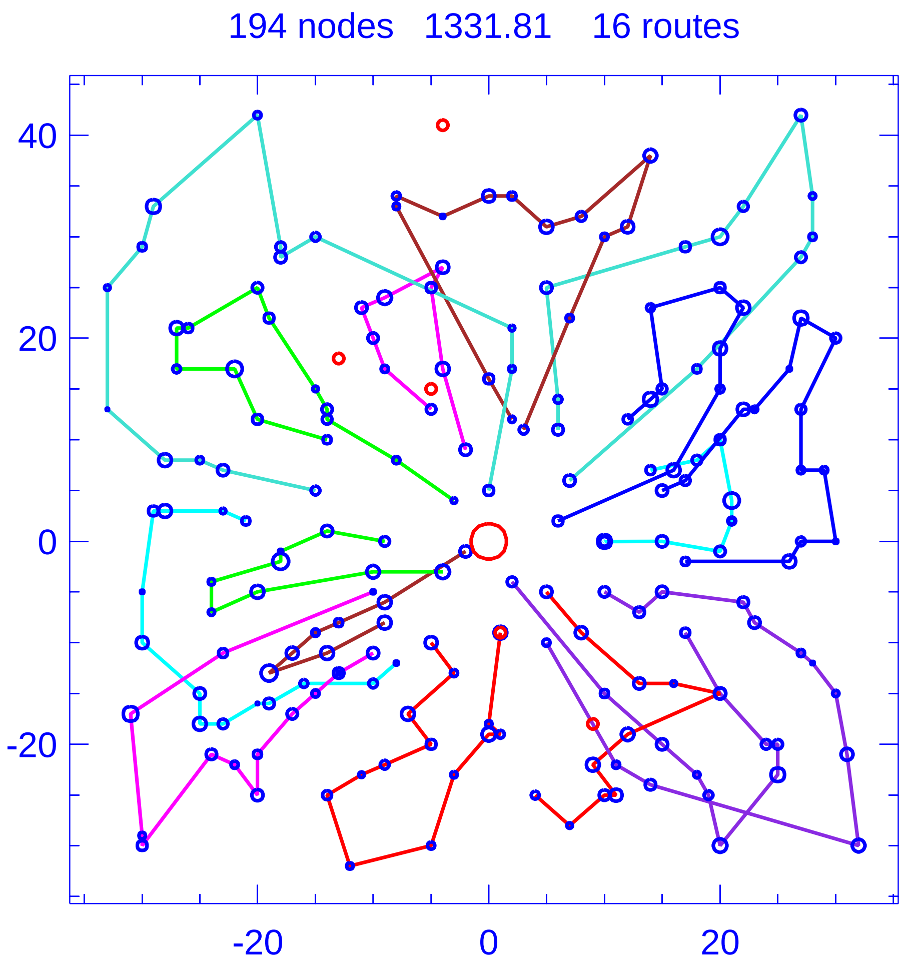


Figure A.55: Template for problem ConVRP-5

Problem			ConVRP-5
Vehicle capacity			200
Maximum route length			N/A
Number of nodes			194
Total route length			1331.805
Total number of routes			16
Route	Length	Load	Ordering
1	81.727	198	(0, 2, 101, 64, 28, 156, 114, 142, 91, 141, 22, 93, 186, 86, 112, 0)
2	79.200	197	(0, 3, 181, 147, 73, 146, 18, 118, 72, 106, 44, 55, 151, 95, 0)
3	53.948	189	(0, 4, 87, 111, 27, 179, 99, 167, 102, 8, 176, 175, 0)
4	46.461	193	(0, 6, 33, 195, 53, 198, 192, 184, 158, 193, 0)
5	61.741	191	(0, 17, 130, 109, 39, 57, 9, 161, 97, 187, 76, 0)
6	109.575	198	(0, 26, 100, 129, 52, 11, 170, 164, 85, 134, 84, 132, 60, 0)
7	81.671	182	(0, 29, 103, 88, 37, 138, 122, 36, 155, 47, 120, 152, 0)
8	95.395	195	(0, 34, 65, 178, 78, 177, 14, 133, 19, 70, 128, 123, 13, 83, 58, 0)
9	77.861	198	(0, 54, 30, 48, 59, 5, 171, 21, 82, 121, 140, 94, 0)

Route	Length	Load	Ordering
10	54.566	171	(0, 61, 96, 104, 23, 183, 67, 159, 16, 188, 0)
11	85.926	190	(0, 66, 196, 197, 43, 190, 41, 90, 115, 62, 116, 144, 74, 49, 182, 0)
12	104.427	194	(0, 81, 71, 119, 38, 165, 77, 10, 80, 131, 189, 162, 75, 12, 168, 0)
13	98.327	197	(0, 105, 160, 185, 89, 137, 143, 68, 113, 42, 199, 136, 191, 1, 194, 0)
14	128.375	198	(0, 117, 63, 107, 24, 145, 135, 92, 148, 163, 25, 56, 110, 169, 50, 126, 0)
15	99.255	191	(0, 125, 98, 45, 153, 79, 15, 154, 124, 20, 166, 174, 173, 139, 0)
16	73.350	191	(0, 127, 46, 35, 180, 69, 108, 150, 7, 51, 149, 0)

Table A.55: Details of template for problem ConVRP-5

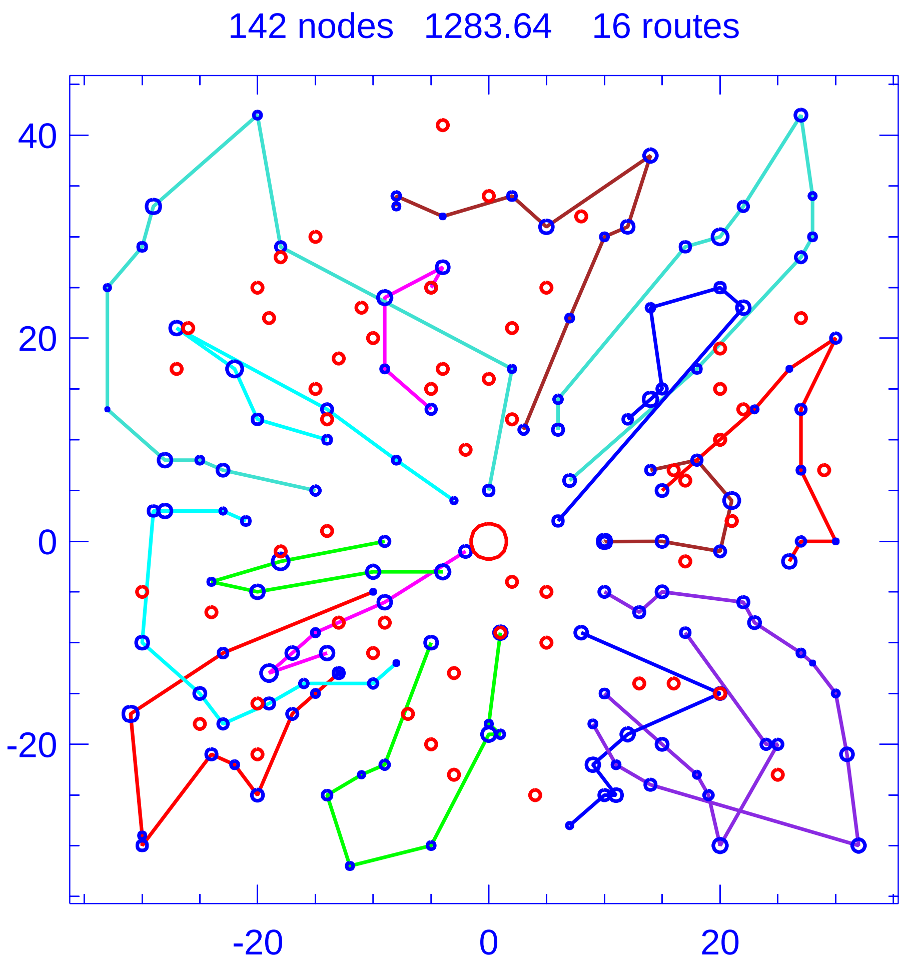


Figure A.56: Solution for day 1 of problem ConVRP-5

Problem			ConVRP-5
Vehicle capacity			200
Maximum route length			N/A
Number of nodes			142
Total route length			1283.639
Total number of routes			16
Route	Length	Load	Ordering
1	97.784	166	(0, 1, 191, 136, 199, 113, 68, 143, 137, 89, 185, 160, 105, 0)
2	76.839	140	(0, 2, 101, 64, 28, 114, 142, 91, 141, 22, 112, 0)
3	69.795	121	(0, 3, 181, 147, 146, 44, 151, 95, 0)
4	49.974	144	(0, 4, 87, 111, 27, 167, 176, 175, 0)
5	46.393	155	(0, 6, 33, 53, 198, 192, 184, 158, 0)
6	104.481	177	(0, 26, 100, 52, 11, 170, 164, 85, 134, 84, 132, 60, 0)
7	81.131	118	(0, 29, 88, 37, 122, 36, 155, 47, 120, 0)
8	76.355	142	(0, 30, 5, 171, 21, 82, 121, 140, 0)
9	89.670	106	(0, 34, 78, 177, 133, 19, 70, 123, 13, 83, 0)

Route	Length	Load	Ordering
10	49.865	143	(0, 61, 96, 104, 183, 67, 188, 0)
11	84.374	160	(0, 66, 196, 197, 43, 41, 115, 62, 144, 74, 49, 182, 0)
12	102.564	135	(0, 75, 162, 189, 80, 10, 165, 38, 119, 71, 81, 0)
13	60.147	92	(0, 76, 187, 9, 57, 109, 0)
14	123.858	162	(0, 117, 63, 107, 24, 145, 135, 92, 148, 163, 25, 50, 126, 0)
15	99.255	191	(0, 125, 98, 45, 153, 79, 15, 154, 124, 20, 166, 174, 173, 172, 0)
16	71.154	128	(0, 127, 69, 108, 150, 7, 51, 149, 0)

Table A.56: Details of solution to day 1 of problem ConVRP-5

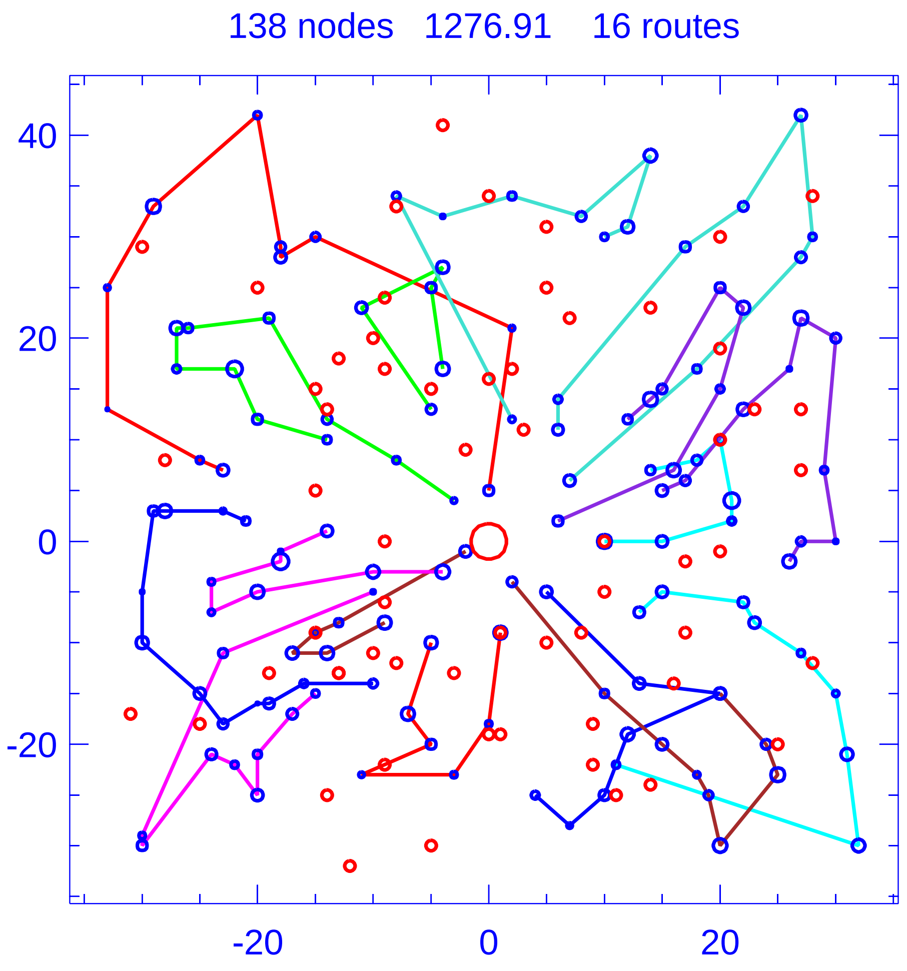


Figure A.57: Solution for day 2 of problem ConVRP-5

Problem			ConVRP-5
Vehicle capacity			200
Maximum route length			N/A
Number of nodes			138
Total route length			1276.913
Total number of routes			16
Route	Length	Load	Ordering
1	60.716	110	(0, 2, 101, 156, 141, 93, 186, 112, 0)
2	75.780	158	(0, 3, 181, 147, 73, 146, 18, 72, 55, 151, 95, 0)
3	52.011	141	(0, 4, 111, 179, 99, 167, 102, 176, 175, 0)
4	41.212	116	(0, 6, 195, 198, 192, 158, 193, 0)
5	54.503	156	(0, 16, 159, 67, 183, 23, 104, 96, 61, 0)
6	104.096	134	(0, 26, 100, 52, 170, 164, 134, 84, 132, 60, 0)
7	92.434	155	(0, 34, 65, 178, 177, 14, 133, 128, 123, 13, 83, 0)
8	75.256	120	(0, 54, 48, 5, 171, 121, 140, 94, 0)
9	127.502	141	(0, 63, 107, 145, 135, 148, 163, 25, 56, 110, 169, 126, 0)

Route	Length	Load	Ordering
10	61.415	116	(0, 76, 161, 57, 39, 109, 130, 0)
11	99.107	145	(0, 98, 45, 153, 79, 15, 124, 20, 166, 173, 0)
12	79.890	155	(0, 103, 88, 138, 122, 36, 155, 47, 120, 152, 0)
13	95.511	122	(0, 105, 160, 89, 137, 143, 68, 113, 42, 199, 136, 0)
14	103.077	102	(0, 119, 38, 165, 77, 80, 189, 162, 168, 0)
15	69.917	155	(0, 127, 46, 35, 69, 108, 7, 51, 149, 0)
16	84.486	160	(0, 182, 49, 74, 144, 116, 62, 115, 41, 190, 43, 197, 196, 0)

Table A.57: Details of solution to day 2 of problem ConVRP-5

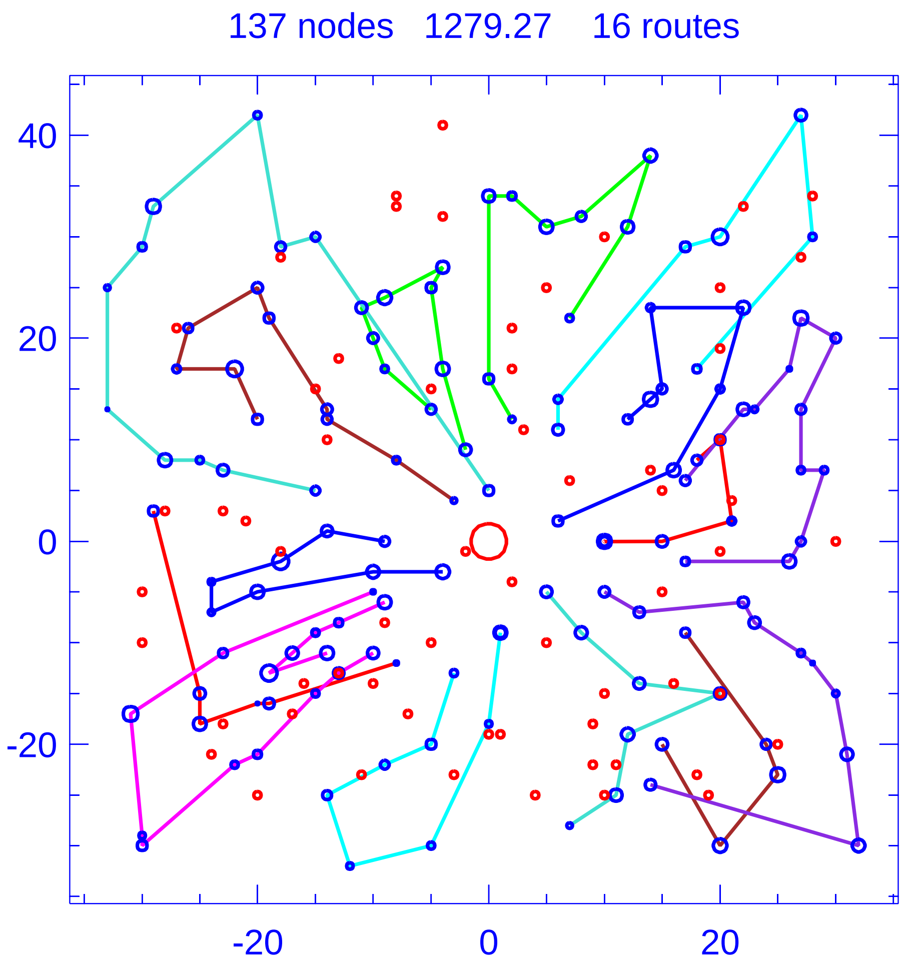


Figure A.58: Solution for day 3 of problem ConVRP-5

Problem			ConVRP-5
Vehicle capacity			200
Maximum route length			N/A
Number of nodes			137
Total route length			1279.269
Total number of routes			16
Route	Length	Load	Ordering
1	51.913	110	(0, 4, 87, 111, 179, 99, 8, 176, 0)
2	61.741	191	(0, 17, 130, 109, 39, 57, 9, 161, 97, 187, 76, 0)
3	104.387	120	(0, 26, 100, 52, 11, 164, 134, 132, 0)
4	80.219	106	(0, 29, 88, 138, 122, 47, 0)
5	46.371	147	(0, 33, 195, 53, 198, 192, 184, 158, 0)
6	75.102	137	(0, 54, 30, 48, 5, 171, 82, 140, 0)
7	92.584	173	(0, 58, 83, 13, 128, 70, 19, 133, 14, 177, 78, 178, 65, 0)
8	54.094	168	(0, 61, 96, 104, 23, 183, 67, 16, 188, 0)
9	83.106	84	(0, 66, 43, 190, 90, 115, 144, 0)

Route	Length	Load	Ordering
10	93.190	153	(0, 71, 38, 165, 77, 10, 80, 131, 12, 168, 0)
11	76.596	116	(0, 86, 93, 22, 91, 142, 114, 101, 2, 157, 0)
12	78.115	154	(0, 95, 151, 55, 44, 72, 118, 18, 73, 147, 181, 0)
13	91.200	140	(0, 105, 160, 185, 89, 137, 68, 42, 136, 1, 194, 0)
14	120.686	167	(0, 117, 63, 107, 24, 145, 135, 92, 148, 163, 25, 110, 126, 0)
15	97.994	153	(0, 125, 98, 153, 79, 15, 154, 124, 20, 166, 174, 0)
16	71.971	149	(0, 127, 46, 35, 69, 150, 7, 51, 149, 0)

Table A.58: Details of solution to day 3 of problem ConVRP-5

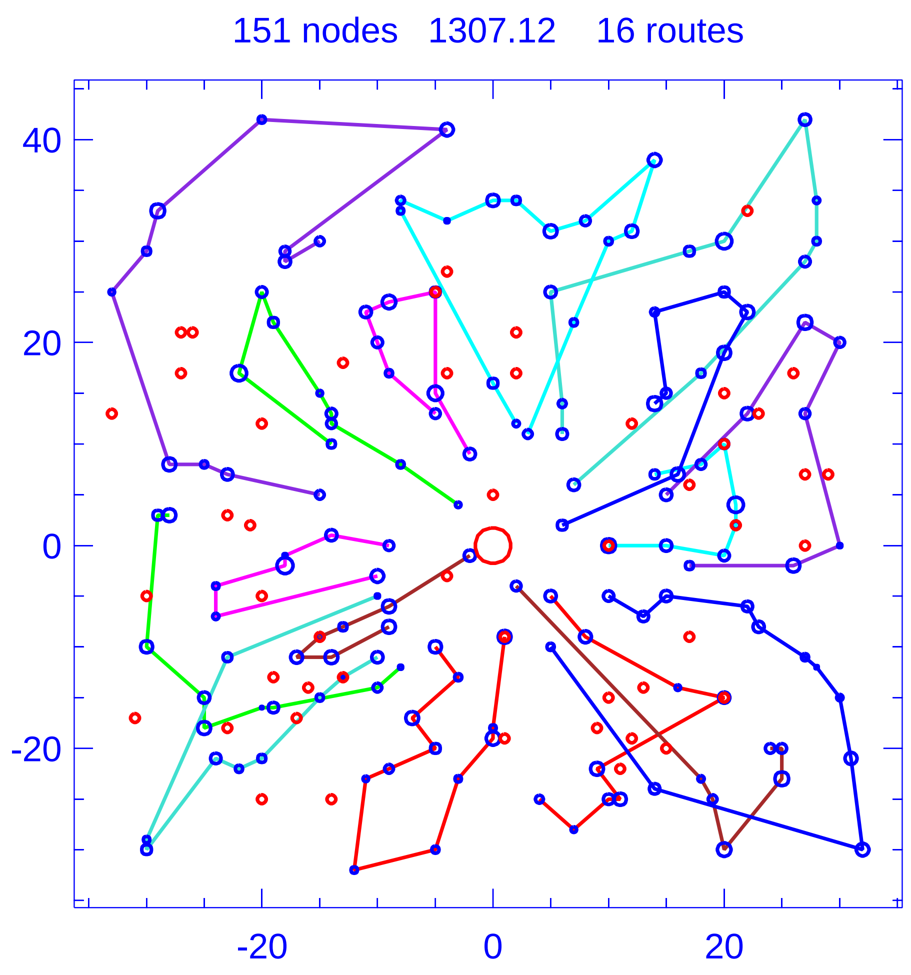


Figure A.59: Solution for day 4 of problem ConVRP-5

Problem			ConVRP-5
Vehicle capacity			200
Maximum route length			N/A
Number of nodes			151
Total route length			1307.124
Total number of routes			16
Route	Length	Load	Ordering
1	78.483	175	(0, 2, 101, 28, 156, 114, 142, 141, 22, 93, 186, 86, 112, 0)
2	69.156	131	(0, 3, 147, 118, 72, 106, 44, 55, 151, 95, 0)
3	53.948	152	(0, 4, 111, 27, 99, 167, 102, 176, 175, 0)
4	41.248	143	(0, 6, 33, 195, 53, 192, 158, 193, 0)
5	58.197	158	(0, 17, 40, 109, 9, 161, 97, 187, 76, 0)
6	109.566	183	(0, 26, 100, 129, 52, 11, 164, 85, 134, 84, 132, 60, 0)
7	91.890	142	(0, 34, 178, 14, 133, 19, 123, 83, 58, 0)
8	73.055	168	(0, 51, 7, 150, 108, 69, 180, 46, 127, 0)
9	77.075	153	(0, 54, 30, 59, 5, 21, 82, 121, 140, 94, 0)

Route	Length	Load	Ordering
10	85.125	147	(0, 66, 196, 43, 190, 90, 115, 62, 144, 74, 0)
11	104.427	194	(0, 81, 71, 119, 38, 165, 77, 10, 80, 131, 189, 162, 75, 12, 168, 0)
12	80.490	120	(0, 88, 37, 138, 122, 36, 155, 152, 0)
13	53.896	116	(0, 96, 23, 183, 67, 159, 16, 188, 0)
14	90.210	107	(0, 105, 160, 89, 137, 143, 68, 42, 136, 191, 194, 0)
15	141.482	194	(0, 110, 56, 25, 31, 163, 148, 92, 135, 24, 107, 63, 117, 0)
16	98.877	183	(0, 125, 98, 45, 153, 79, 15, 154, 124, 20, 166, 174, 139, 0)

Table A.59: Details of solution to day 4 of problem ConVRP-5

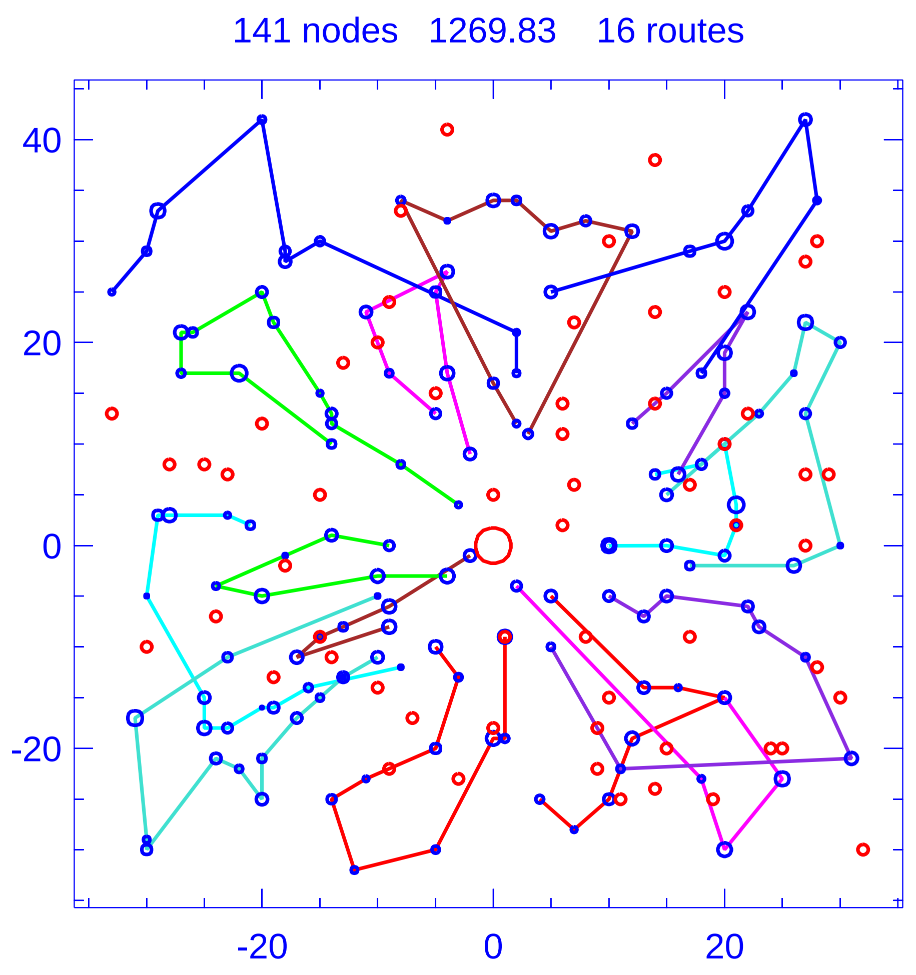


Figure A.60: Solution for day 5 of problem ConVRP-5

Problem			ConVRP-5
Vehicle capacity			200
Maximum route length			N/A
Number of nodes			141
Total route length			1269.830
Total number of routes			16
Route	Length	Load	Ordering
1	79.078	146	(0, 2, 64, 28, 114, 142, 91, 141, 93, 86, 112, 0)
2	78.121	181	(0, 3, 147, 73, 146, 18, 118, 72, 106, 44, 55, 151, 95, 0)
3	53.948	170	(0, 4, 87, 111, 27, 99, 167, 8, 176, 175, 0)
4	40.961	115	(0, 6, 33, 195, 53, 192, 193, 0)
5	61.736	145	(0, 17, 130, 109, 39, 57, 161, 187, 76, 0)
6	92.376	128	(0, 34, 78, 177, 14, 133, 19, 123, 83, 58, 0)
7	66.724	119	(0, 46, 35, 180, 69, 7, 149, 0)
8	121.363	112	(0, 50, 169, 110, 56, 25, 163, 148, 92, 135, 0)
9	75.308	125	(0, 54, 48, 59, 5, 171, 121, 140, 94, 0)

Route	Length	Load	Ordering
10	50.601	124	(0, 61, 96, 104, 183, 159, 16, 188, 0)
11	84.453	155	(0, 66, 197, 43, 190, 41, 90, 115, 116, 144, 74, 49, 182, 0)
12	92.141	145	(0, 81, 38, 77, 10, 80, 131, 189, 162, 12, 168, 0)
13	79.628	99	(0, 103, 138, 122, 155, 152, 0)
14	98.327	197	(0, 105, 160, 185, 89, 137, 143, 68, 113, 42, 199, 136, 191, 1, 194, 0)
15	87.314	140	(0, 125, 98, 45, 153, 79, 15, 20, 173, 139, 0)
16	107.752	126	(0, 129, 52, 11, 170, 164, 85, 132, 0)

Table A.60: Details of solution to day 5 of problem ConVRP-5

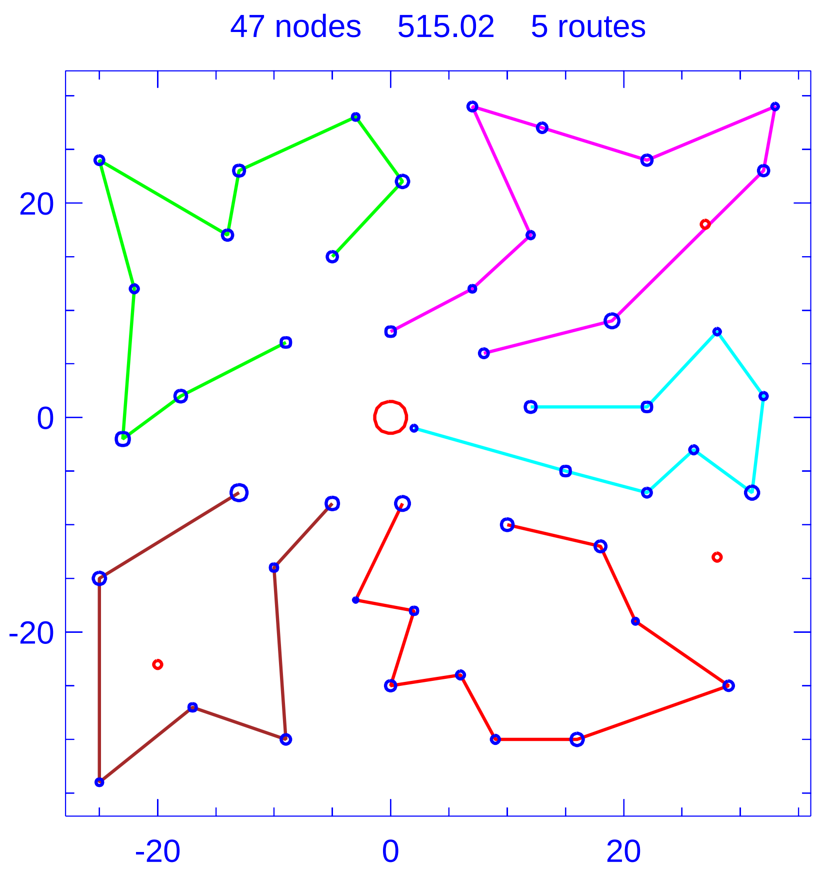


Figure A.61: Template for problem ConVRP-6

Problem			ConVRP-6
Vehicle capacity			160
Maximum route length			200.000
Number of nodes			47
Total route length			515.022
Total number of routes			5
Route	Length	Load	Ordering
1	104.014	158	(0, 5, 49, 10, 39, 33, 45, 15, 44, 37, 17, 12, 0)
2	117.049	167	(0, 6, 14, 25, 24, 43, 23, 7, 26, 8, 48, 0)
3	82.705	115	(0, 11, 16, 29, 21, 34, 50, 9, 38, 46, 0)
4	100.637	127	(0, 18, 13, 40, 19, 42, 4, 47, 0)
5	110.617	136	(0, 27, 1, 22, 31, 28, 3, 36, 35, 2, 32, 0)

Table A.61: Details of template for problem ConVRP-6

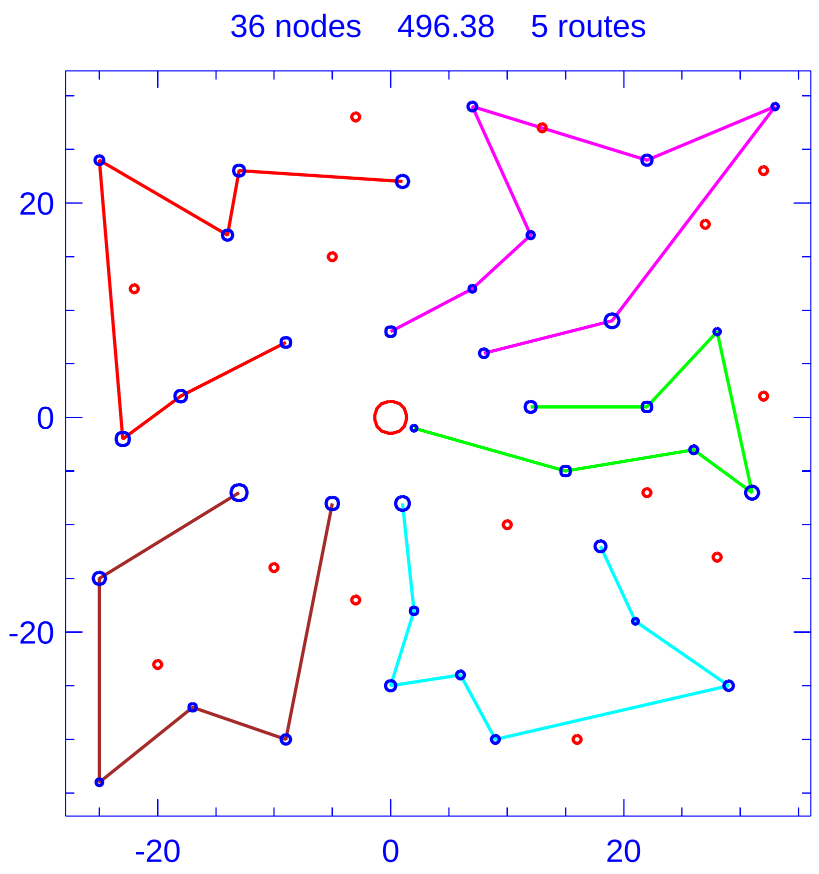


Figure A.62: Solution for day 1 of problem ConVRP-6

Problem			ConVRP-6
Vehicle capacity			160
Maximum route length			200.000
Number of nodes			36
Total route length			496.382
Total number of routes			5
Route	Length	Load	Ordering
1	109.357	133	(0, 6, 14, 25, 43, 23, 7, 8, 0)
2	79.979	96	(0, 11, 16, 29, 34, 50, 38, 46, 0)
3	98.048	111	(0, 12, 37, 44, 15, 45, 39, 10, 49, 0)
4	99.156	118	(0, 18, 13, 40, 19, 42, 47, 0)
5	109.843	105	(0, 27, 1, 22, 31, 3, 36, 2, 32, 0)

Table A.62: Details of solution to day 1 of problem ConVRP-6

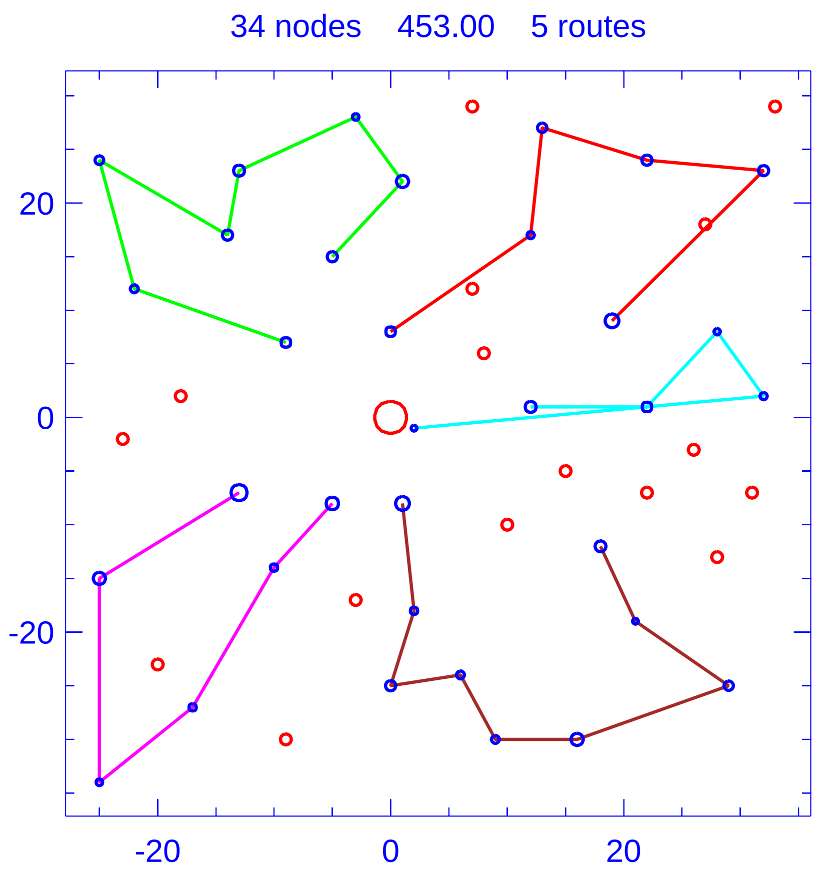


Figure A.63: Solution for day 2 of problem ConVRP-6

Problem			ConVRP-6
Vehicle capacity			160
Maximum route length			200.000
Number of nodes			34
Total route length			453.003
Total number of routes			5
Route	Length	Load	Ordering
1	92.715	100	(0, 2, 35, 3, 28, 22, 27, 0)
2	100.243	118	(0, 6, 24, 43, 23, 7, 26, 8, 48, 0)
3	70.858	53	(0, 11, 16, 29, 21, 46, 0)
4	98.361	134	(0, 12, 37, 44, 15, 45, 33, 39, 10, 49, 0)
5	90.826	114	(0, 18, 13, 40, 19, 4, 47, 0)

Table A.63: Details of solution to day 2 of problem ConVRP-6

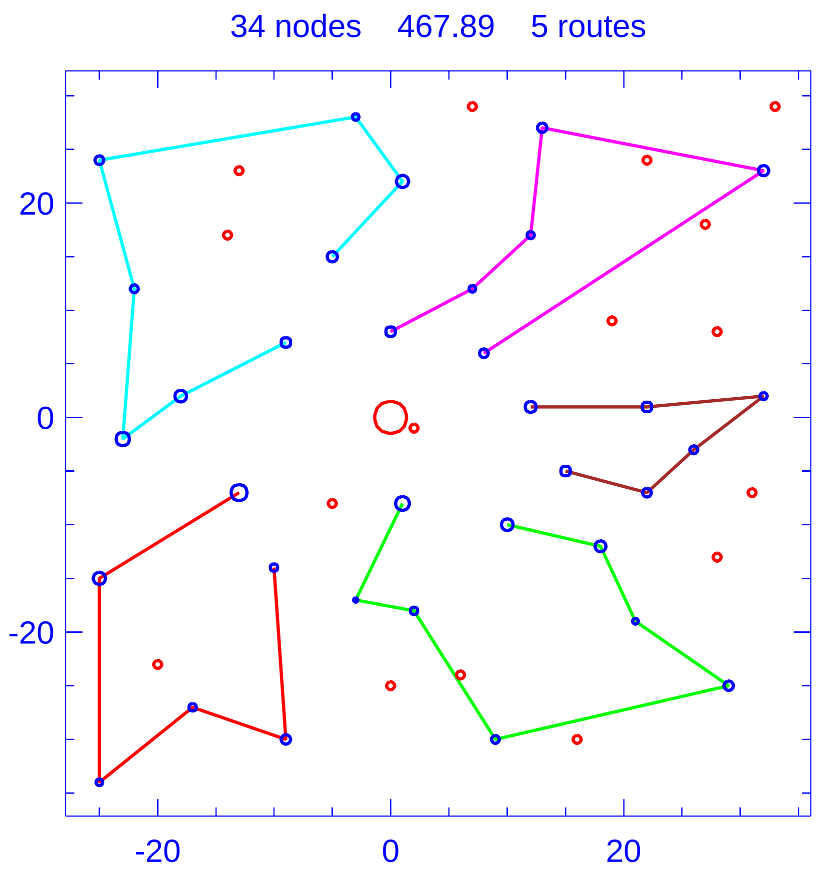


Figure A.64: Solution for day 3 of problem ConVRP-6

Problem			ConVRP-6
Vehicle capacity			160
Maximum route length			200.000
Number of nodes			34
Total route length			467.888
Total number of routes			5
Route	Length	Load	Ordering
1	100.597	102	(0, 4, 42, 19, 40, 13, 18, 0)
2	97.522	109	(0, 5, 49, 10, 39, 45, 37, 17, 12, 0)
3	109.108	132	(0, 6, 14, 25, 24, 43, 26, 8, 48, 0)
4	68.650	78	(0, 11, 16, 21, 50, 9, 38, 0)
5	92.011	73	(0, 27, 1, 22, 28, 35, 32, 0)

Table A.64: Details of solution to day 3 of problem ConVRP-6

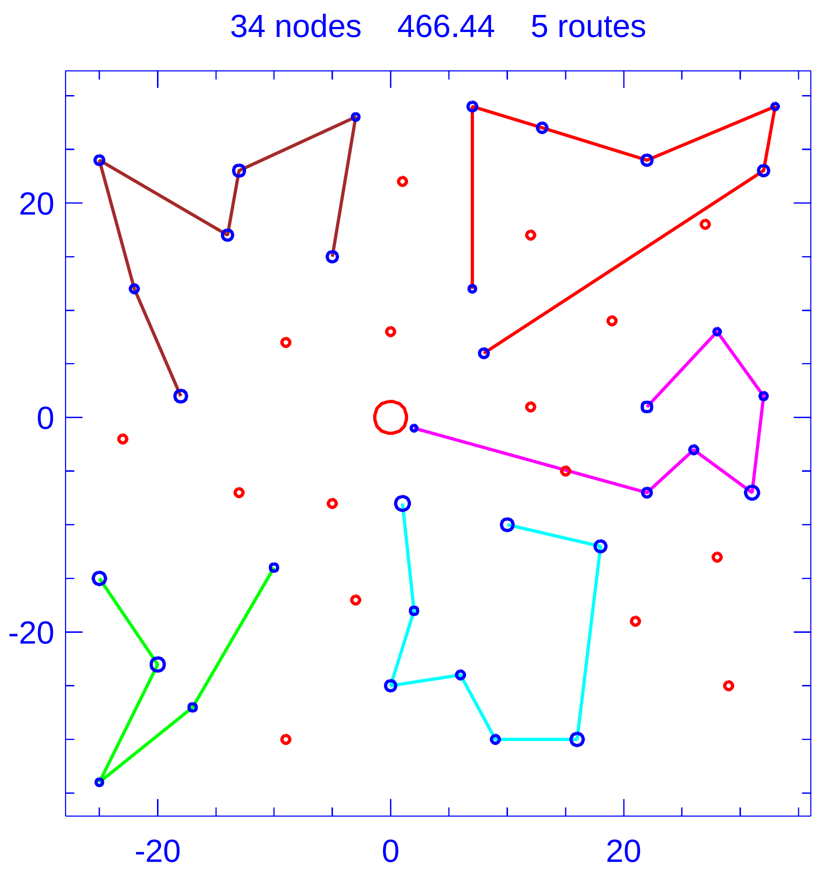


Figure A.65: Solution for day 4 of problem ConVRP-6

Problem			ConVRP-6
Vehicle capacity			160
Maximum route length			200.000
Number of nodes			34
Total route length			466.436
Total number of routes			5
Route	Length	Load	Ordering
1	104.281	83	(0, 1, 31, 28, 3, 36, 35, 32, 0)
2	93.271	75	(0, 4, 19, 40, 41, 13, 0)
3	85.682	136	(0, 5, 49, 33, 45, 15, 44, 37, 12, 0)
4	100.516	101	(0, 14, 24, 43, 23, 7, 26, 48, 0)
5	82.685	81	(0, 16, 29, 21, 34, 50, 9, 46, 0)

Table A.65: Details of solution to day 4 of problem ConVRP-6

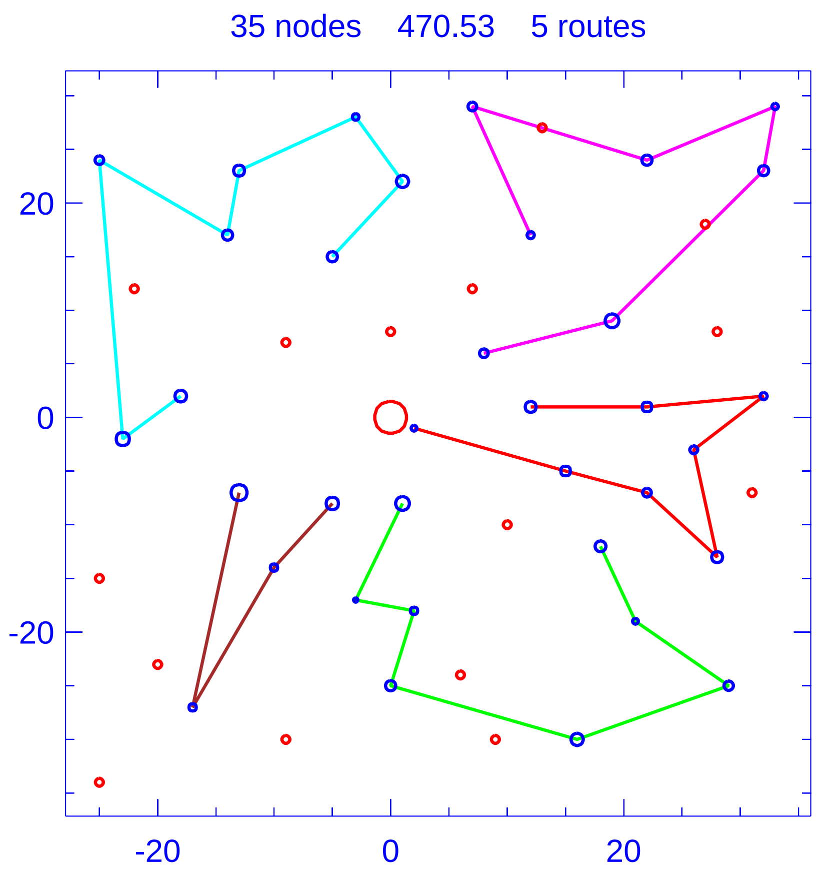


Figure A.66: Solution for day 5 of problem ConVRP-6

Problem			ConVRP-6
Vehicle capacity			160
Maximum route length			200.000
Number of nodes			35
Total route length			470.530
Total number of routes			5
Route	Length	Load	Ordering
1	81.703	102	(0, 11, 16, 21, 50, 30, 9, 38, 46, 0)
2	100.231	117	(0, 12, 17, 37, 44, 33, 39, 10, 49, 0)
3	113.134	142	(0, 14, 25, 43, 23, 7, 26, 8, 48, 0)
4	67.170	84	(0, 18, 19, 4, 47, 0)
5	108.293	100	(0, 22, 31, 3, 36, 35, 2, 32, 0)

Table A.66: Details of solution to day 5 of problem ConVRP-6

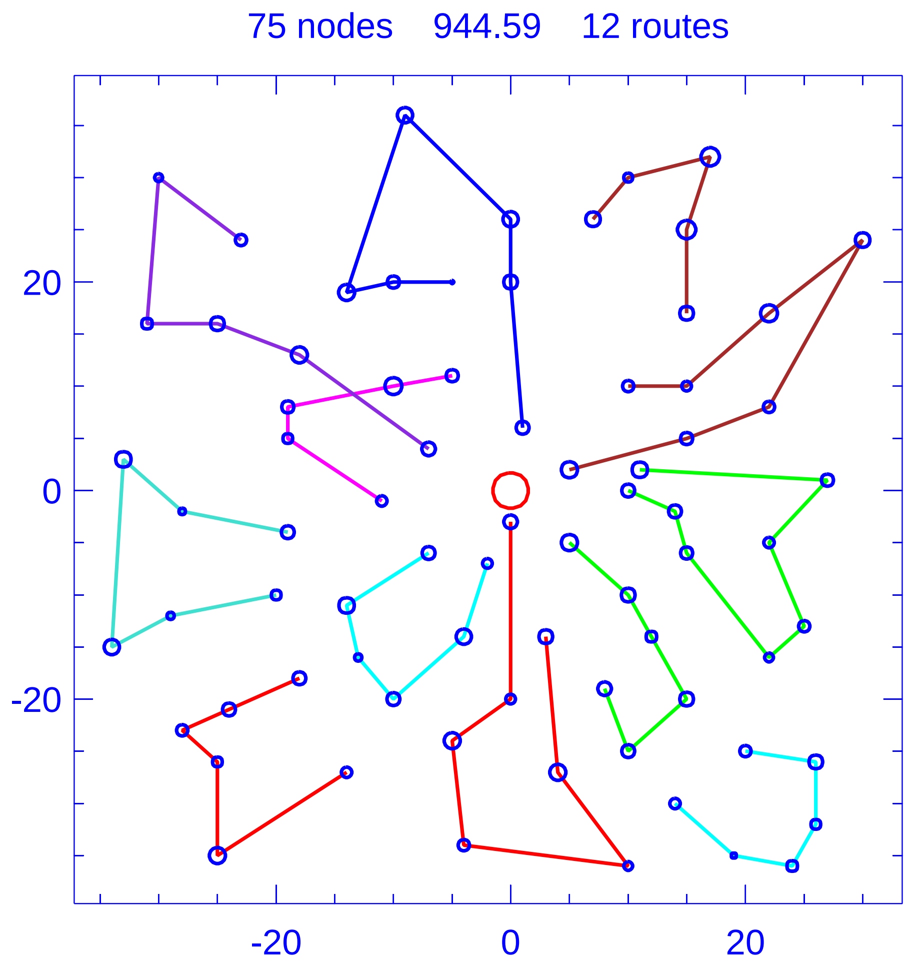


Figure A.67: Template for problem ConVRP-7

Problem			ConVRP-7
Vehicle capacity			140
Maximum route length			160.000
Number of nodes			75
Total route length			944.589
Total number of routes			12
Route	Length	Load	Ordering
1	93.894	102	(0, 1, 43, 41, 42, 64, 22, 0)
2	59.334	124	(0, 4, 45, 29, 5, 47, 48, 0)
3	50.966	106	(0, 6, 33, 73, 62, 2, 68, 0)
4	81.002	141	(0, 7, 35, 14, 59, 19, 8, 67, 0)
5	50.447	89	(0, 12, 40, 44, 3, 51, 0)
6	91.146	94	(0, 16, 49, 24, 56, 23, 63, 0)
7	92.387	104	(0, 17, 32, 50, 18, 55, 25, 0)
8	87.031	136	(0, 26, 58, 10, 31, 9, 39, 72, 0)
9	88.768	132	(0, 30, 21, 69, 61, 28, 74, 75, 0)

Route	Length	Load	Ordering
10	78.610	132	(0, 34, 52, 27, 15, 57, 13, 54, 46, 0)
11	93.846	75	(0, 36, 71, 60, 70, 20, 37, 0)
12	77.158	129	(0, 38, 65, 66, 11, 53, 0)

Table A.67: Details of template for problem ConVRP-7

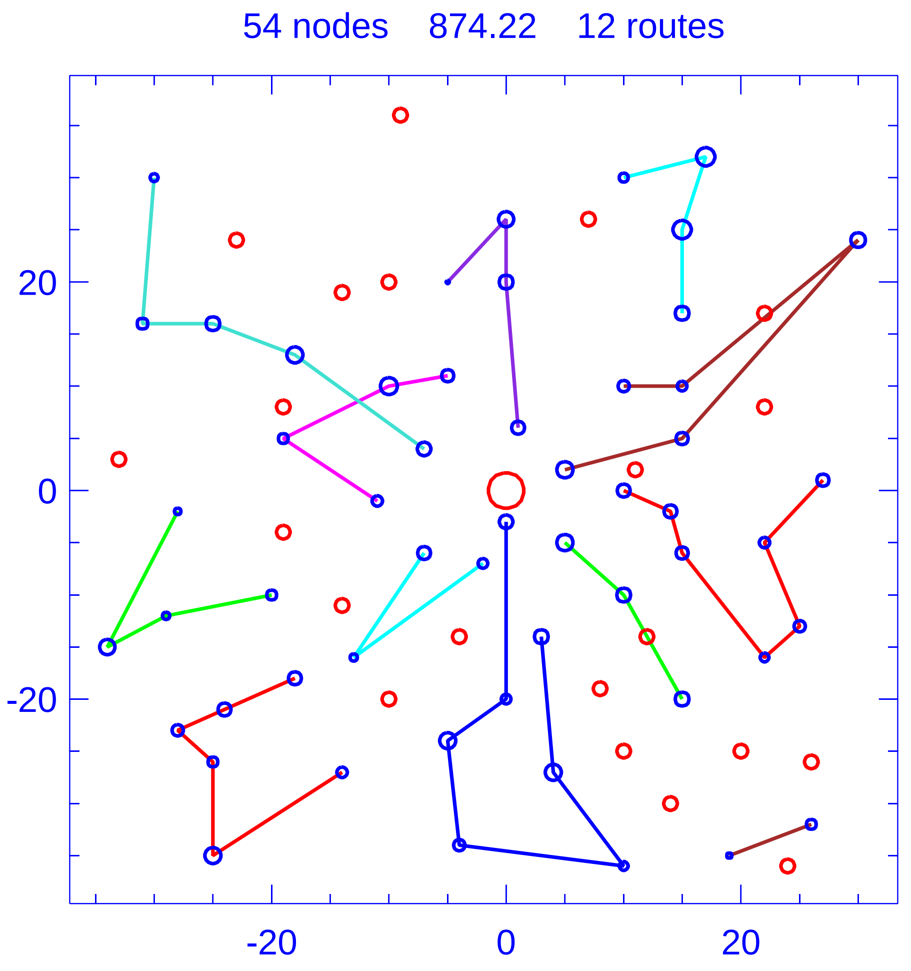


Figure A.68: Solution for day 1 of problem ConVRP-7

Problem			ConVRP-7
Vehicle capacity			140
Maximum route length			160.000
Number of nodes			54
Total route length			874.216
Total number of routes			12
Route	Length	Load	Ordering
1	93.894	102	(0, 1, 43, 41, 42, 64, 22, 0)
2	50.322	72	(0, 4, 45, 5, 0)
3	42.374	35	(0, 6, 73, 68, 0)
4	79.693	95	(0, 7, 35, 59, 8, 67, 0)
5	48.523	72	(0, 12, 40, 3, 51, 0)
6	92.353	90	(0, 17, 32, 50, 18, 55, 0)
7	54.544	66	(0, 26, 58, 10, 72, 0)
8	88.768	132	(0, 30, 21, 69, 61, 28, 74, 75, 0)
9	78.417	105	(0, 34, 52, 27, 15, 57, 13, 54, 0)

Route	Length	Load	Ordering
10	79.800	48	(0, 49, 56, 23, 63, 0)
11	76.855	105	(0, 53, 11, 66, 65, 0)
12	88.671	14	(0, 70, 71, 0)

Table A.68: Details of solution to day 1 of problem ConVRP-7

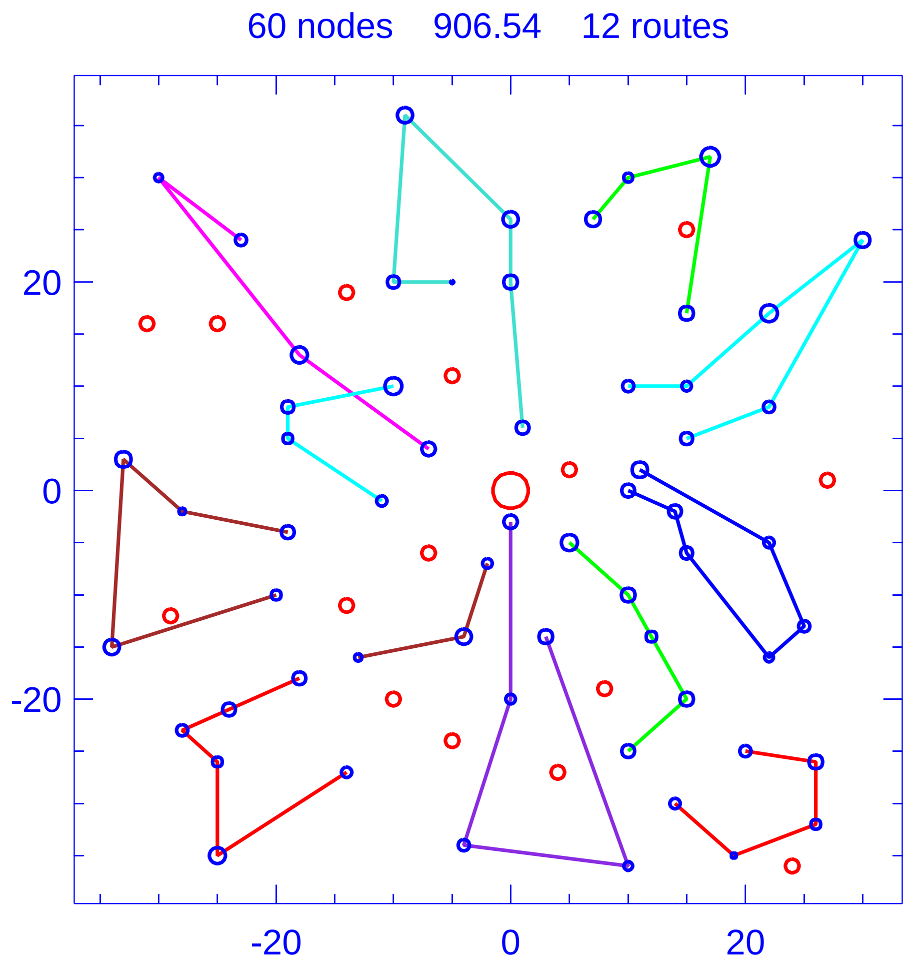


Figure A.69: Solution for day 2 of problem ConVRP-7

Problem			ConVRP-7
Vehicle capacity			140
Maximum route length			160.000
Number of nodes			60
Total route length			906.544
Total number of routes			12
Route	Length	Load	Ordering
1	93.894	102	(0, 1, 43, 41, 42, 64, 22, 0)
2	59.319	104	(0, 4, 45, 29, 5, 47, 0)
3	80.987	111	(0, 7, 35, 14, 59, 19, 8, 0)
4	90.962	88	(0, 16, 49, 24, 56, 63, 0)
5	85.545	69	(0, 17, 32, 55, 25, 0)
6	81.219	107	(0, 26, 58, 10, 31, 39, 72, 0)
7	86.107	75	(0, 30, 69, 61, 74, 75, 0)
8	67.807	116	(0, 34, 52, 27, 15, 57, 13, 46, 0)
9	91.891	62	(0, 36, 71, 70, 20, 37, 0)

Route	Length	Load	Ordering
10	77.010	92	(0, 38, 65, 66, 53, 0)
11	47.407	73	(0, 40, 44, 3, 51, 0)
12	44.395	42	(0, 68, 2, 73, 0)

Table A.69: Details of solution to day 2 of problem ConVRP-7

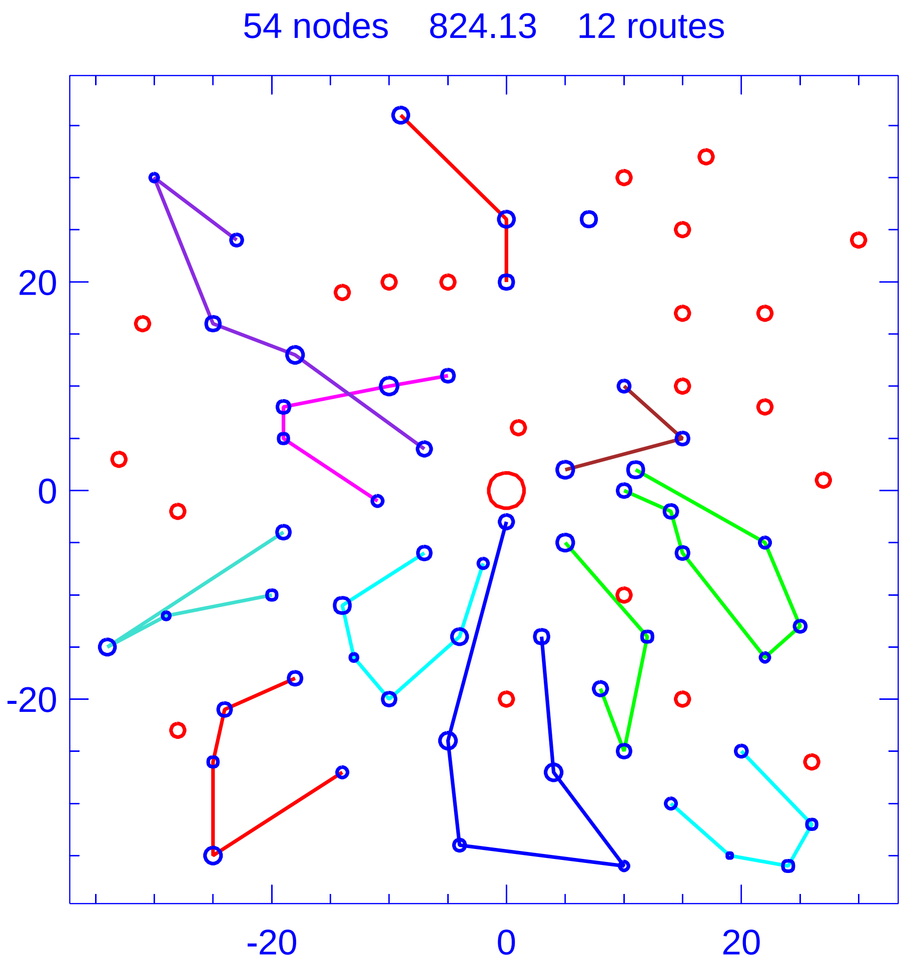


Figure A.70: Solution for day 3 of problem ConVRP-7

Problem			ConVRP-7
Vehicle capacity			140
Maximum route length			160.000
Number of nodes			54
Total route length			824.126
Total number of routes			12
Route	Length	Load	Ordering
1	90.278	87	(0, 1, 43, 42, 64, 22, 0)
2	56.593	82	(0, 4, 29, 47, 48, 0)
3	50.966	106	(0, 6, 33, 73, 62, 2, 68, 0)
4	37.039	61	(0, 7, 8, 67, 0)
5	50.447	89	(0, 12, 40, 44, 3, 51, 0)
6	75.429	62	(0, 16, 56, 23, 63, 0)
7	87.218	91	(0, 17, 32, 50, 55, 25, 0)
8	86.952	122	(0, 30, 21, 69, 61, 28, 75, 0)
9	76.562	72	(0, 31, 10, 58, 0)

Route	Length	Load	Ordering
10	67.807	116	(0, 34, 52, 27, 15, 57, 13, 46, 0)
11	90.983	53	(0, 36, 71, 60, 70, 37, 0)
12	53.852	24	(0, 38, 0)

Table A.70: Details of solution to day 3 of problem ConVRP-7

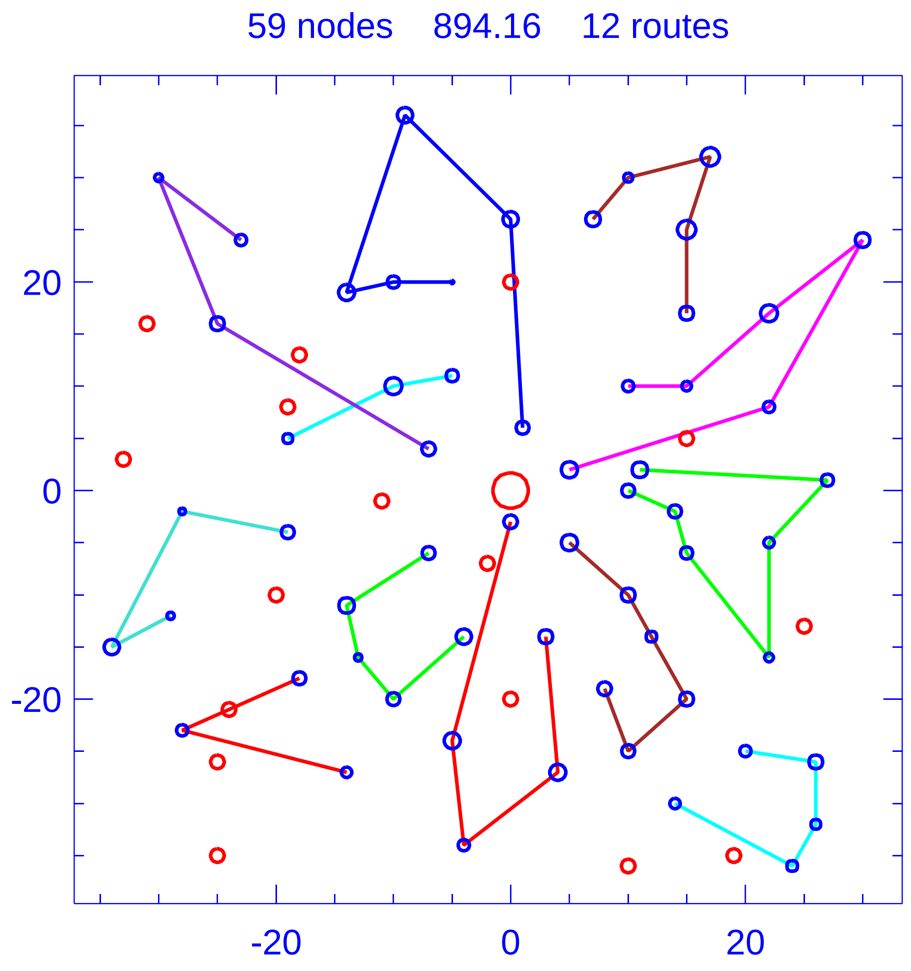


Figure A.71: Solution for day 4 of problem ConVRP-7

Problem			ConVRP-7
Vehicle capacity			140
Maximum route length			160.000
Number of nodes			59
Total route length			894.163
Total number of routes			12
Route	Length	Load	Ordering
1	81.610	45	(0, 1, 41, 22, 0)
2	50.966	96	(0, 2, 62, 73, 33, 6, 0)
3	47.125	60	(0, 3, 40, 12, 0)
4	59.334	124	(0, 4, 45, 29, 5, 47, 48, 0)
5	80.973	125	(0, 7, 35, 14, 59, 19, 67, 0)
6	80.170	56	(0, 16, 49, 56, 23, 0)
7	87.023	63	(0, 17, 50, 55, 25, 0)
8	87.020	115	(0, 26, 10, 31, 9, 39, 72, 0)
9	72.623	114	(0, 30, 21, 61, 28, 75, 0)

Route	Length	Load	Ordering
10	76.824	118	(0, 34, 52, 27, 15, 13, 54, 46, 0)
11	93.338	72	(0, 36, 60, 70, 20, 37, 0)
12	77.158	129	(0, 38, 65, 66, 11, 53, 0)

Table A.71: Details of solution to day 4 of problem ConVRP-7

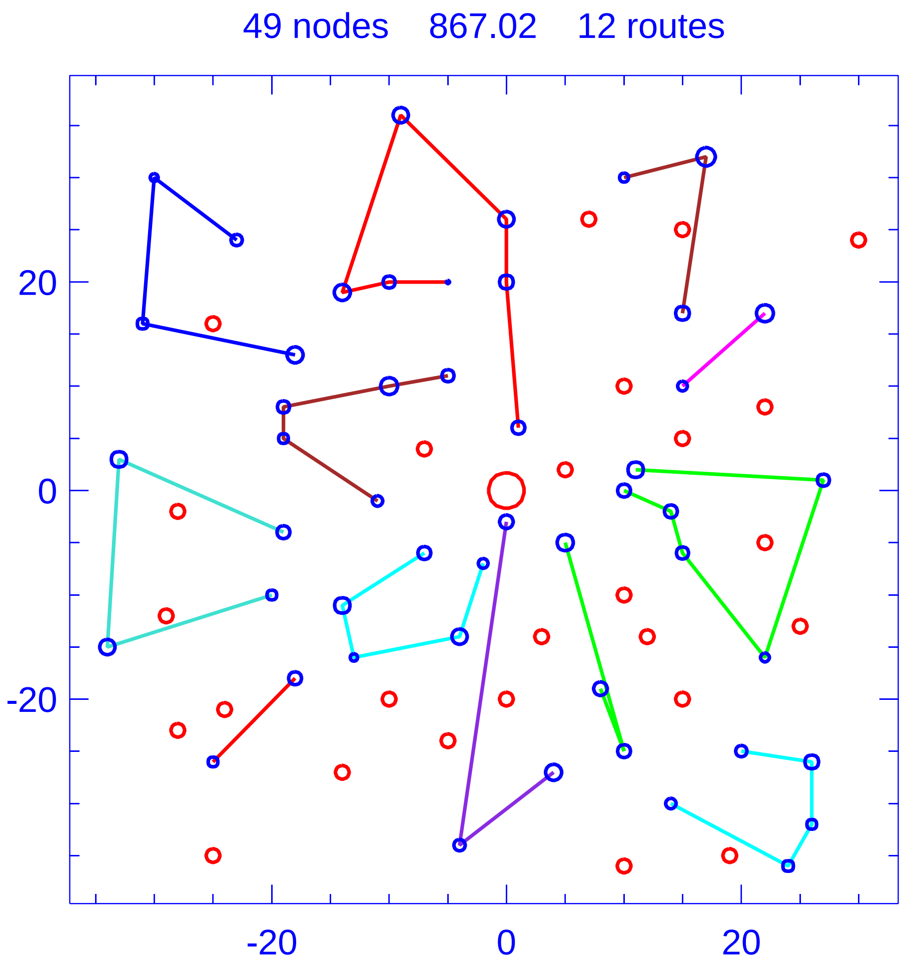


Figure A.72: Solution for day 5 of problem ConVRP-7

Problem			ConVRP-7
Vehicle capacity			140
Maximum route length			160.000
Number of nodes			49
Total route length			867.017
Total number of routes			12
Route	Length	Load	Ordering
1	72.155	29	(0, 1, 42, 0)
2	54.627	69	(0, 4, 47, 48, 0)
3	46.701	88	(0, 6, 33, 73, 2, 68, 0)
4	50.447	89	(0, 12, 40, 44, 3, 51, 0)
5	55.730	41	(0, 14, 35, 0)
6	90.323	83	(0, 16, 24, 56, 63, 0)
7	72.182	63	(0, 21, 61, 75, 0)
8	92.042	62	(0, 25, 55, 18, 32, 0)
9	87.031	136	(0, 26, 58, 10, 31, 9, 39, 72, 0)

Route	Length	Load	Ordering
10	75.733	106	(0, 34, 52, 27, 15, 54, 46, 0)
11	93.338	72	(0, 36, 60, 70, 20, 37, 0)
12	76.707	68	(0, 53, 66, 65, 0)

Table A.72: Details of solution to day 5 of problem ConVRP-7

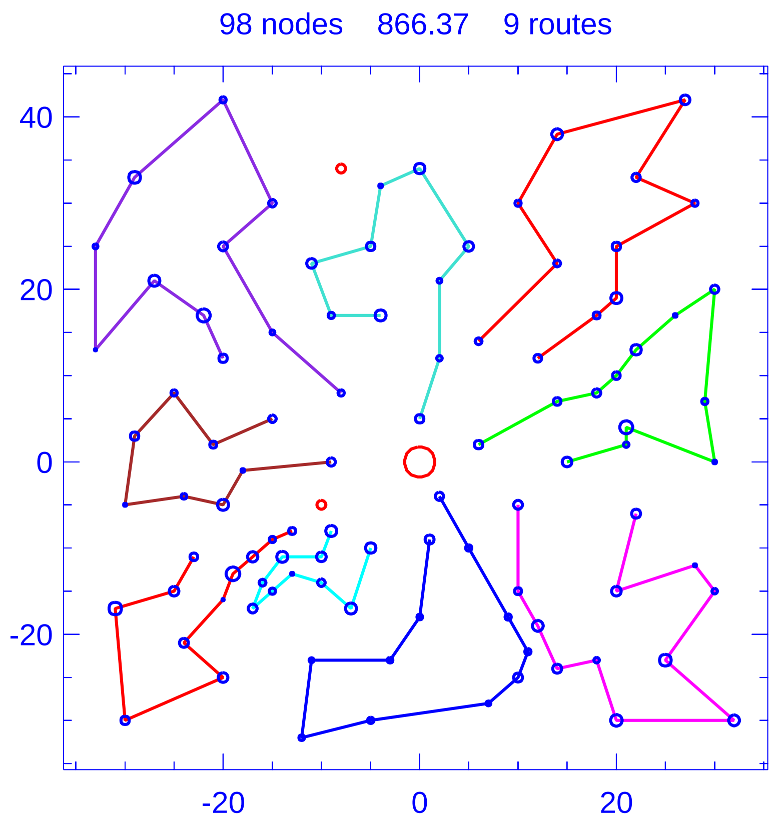


Figure A.73: Template for problem ConVRP-8

Problem			ConVRP-8
Vehicle capacity			200
Maximum route length			230.000
Number of nodes			98
Total route length			866.370
Total number of routes			9
Route	Length	Load	Ordering
1	117.928	163	(0, 1, 51, 20, 66, 65, 71, 35, 9, 81, 33, 50, 0)
2	90.255	169	(0, 12, 80, 68, 24, 29, 34, 78, 79, 3, 77, 76, 28, 0)
3	57.494	173	(0, 13, 87, 97, 92, 37, 100, 98, 59, 95, 94, 0)
4	76.929	101	(0, 18, 83, 8, 45, 17, 84, 5, 60, 89, 0)
5	117.947	207	(0, 26, 21, 72, 75, 56, 23, 67, 39, 25, 55, 4, 54, 0)
6	83.232	145	(0, 27, 69, 70, 30, 32, 90, 10, 62, 88, 31, 0)
7	131.249	167	(0, 52, 7, 19, 11, 64, 49, 36, 46, 47, 48, 82, 0)
8	92.444	115	(0, 53, 40, 73, 74, 22, 41, 15, 43, 42, 57, 2, 58, 0)
9	98.890	205	(0, 61, 16, 86, 38, 14, 44, 91, 85, 93, 99, 96, 0)

Table A.73: Details of template for problem ConVRP-8

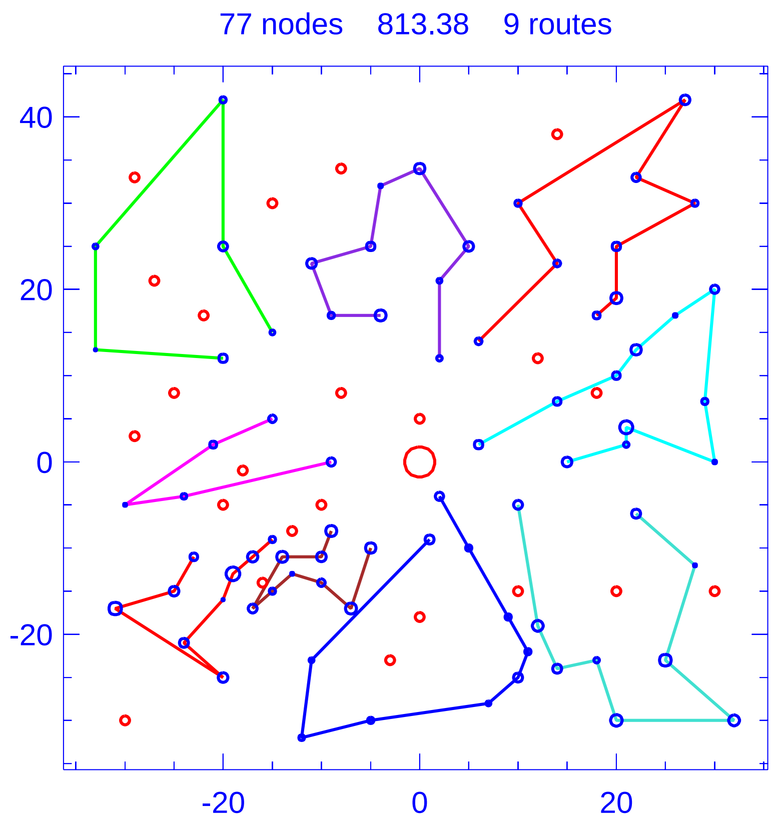


Figure A.74: Solution for day 1 of problem ConVRP-8

Problem			ConVRP-8
Vehicle capacity			200
Maximum route length			230.000
Number of nodes			77
Total route length			813.381
Total number of routes			9
Route	Length	Load	Ordering
1	116.169	125	(0, 1, 51, 20, 65, 71, 35, 9, 81, 33, 0)
2	119.157	53	(0, 7, 19, 64, 36, 46, 82, 0)
3	90.012	155	(0, 12, 80, 68, 24, 29, 34, 78, 79, 3, 76, 28, 0)
4	57.434	163	(0, 13, 87, 97, 92, 37, 100, 59, 95, 94, 0)
5	64.528	47	(0, 18, 83, 17, 84, 89, 0)
6	106.701	171	(0, 26, 72, 75, 56, 23, 67, 39, 55, 54, 0)
7	83.118	129	(0, 31, 88, 62, 10, 90, 32, 30, 70, 69, 0)
8	87.997	101	(0, 53, 40, 73, 74, 22, 41, 15, 43, 42, 58, 0)
9	88.265	178	(0, 61, 16, 86, 14, 44, 91, 85, 93, 99, 0)

Table A.74: Details of solution to day 1 of problem ConVRP-8

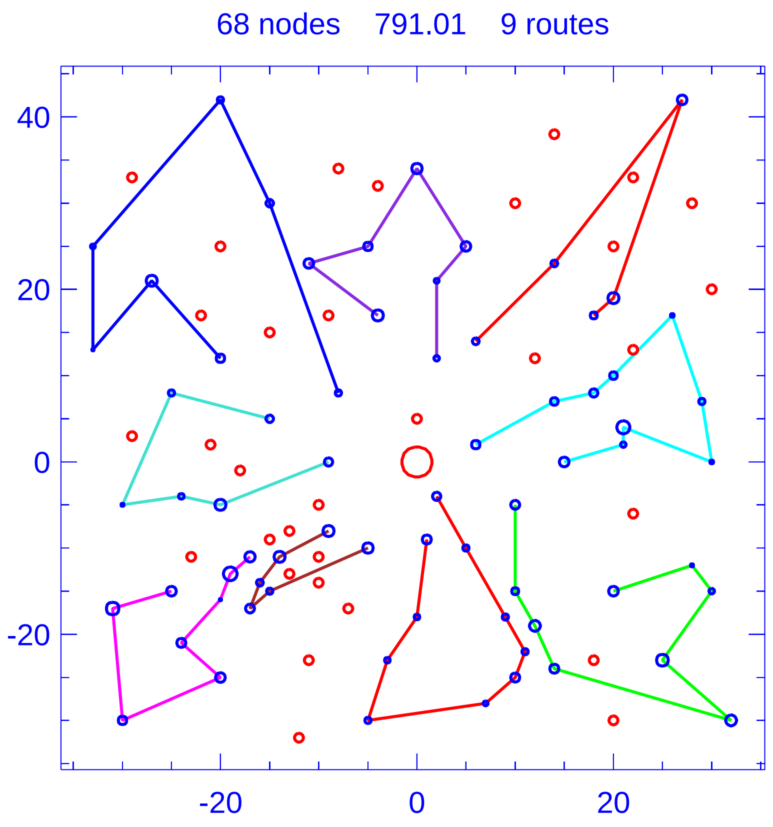


Figure A.75: Solution for day 2 of problem ConVRP-8

Problem			ConVRP-8
Vehicle capacity			200
Maximum route length			230.000
Number of nodes			68
Total route length			791.009
Total number of routes			9
Route	Length	Load	Ordering
1	101.924	77	(0, 1, 51, 65, 81, 33, 0)
2	106.494	154	(0, 4, 55, 25, 39, 67, 75, 72, 21, 26, 0)
3	82.614	132	(0, 12, 80, 68, 24, 29, 78, 3, 77, 76, 28, 0)
4	49.829	113	(0, 13, 37, 100, 98, 59, 94, 0)
5	97.997	172	(0, 16, 86, 38, 14, 44, 91, 85, 93, 0)
6	71.469	71	(0, 18, 8, 17, 84, 5, 89, 0)
7	79.765	117	(0, 31, 62, 10, 32, 30, 70, 69, 0)
8	125.527	79	(0, 52, 11, 64, 36, 46, 47, 82, 0)
9	75.389	103	(0, 53, 40, 73, 74, 22, 41, 15, 57, 2, 58, 0)

Table A.75: Details of solution to day 2 of problem ConVRP-8

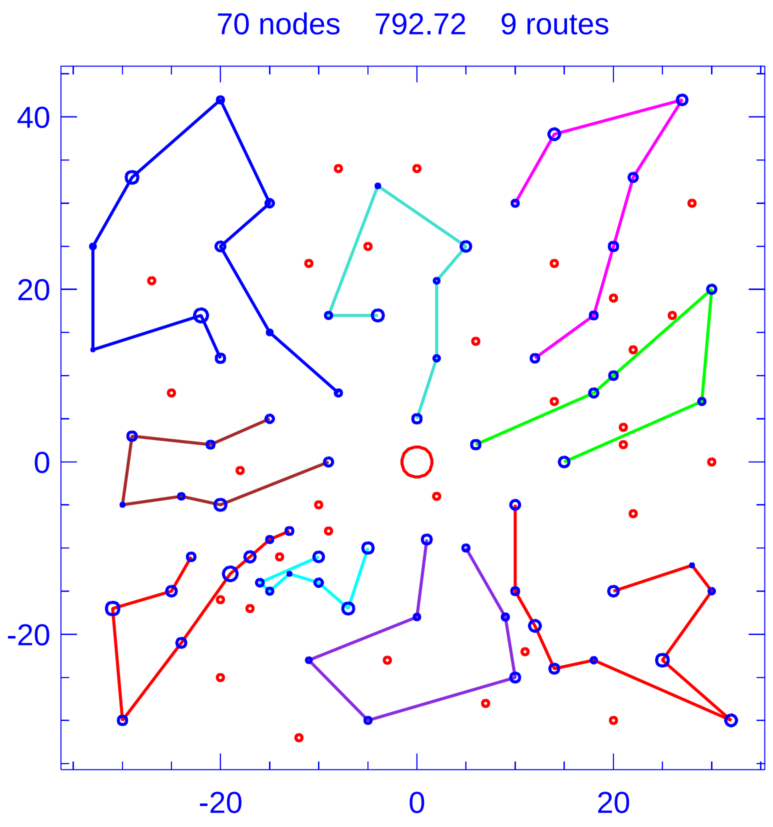


Figure A.76: Solution for day 3 of problem ConVRP-8

Problem			ConVRP-8
Vehicle capacity			200
Maximum route length			230.000
Number of nodes			70
Total route length			792.718
Total number of routes			9
Route	Length	Load	Ordering
1	107.296	160	(0, 4, 55, 25, 39, 67, 56, 75, 72, 21, 26, 0)
2	80.402	85	(0, 12, 29, 34, 3, 77, 28, 0)
3	51.682	101	(0, 13, 87, 97, 92, 37, 98, 95, 0)
4	69.933	89	(0, 18, 83, 45, 17, 84, 5, 89, 0)
5	105.737	109	(0, 20, 66, 65, 71, 9, 33, 50, 0)
6	75.958	87	(0, 27, 69, 70, 30, 90, 88, 31, 0)
7	82.420	74	(0, 40, 73, 22, 15, 42, 2, 58, 0)
8	126.550	140	(0, 52, 7, 19, 11, 64, 49, 36, 46, 48, 82, 0)
9	92.738	184	(0, 61, 16, 86, 38, 44, 85, 93, 99, 96, 0)

Table A.76: Details of solution to day 3 of problem ConVRP-8

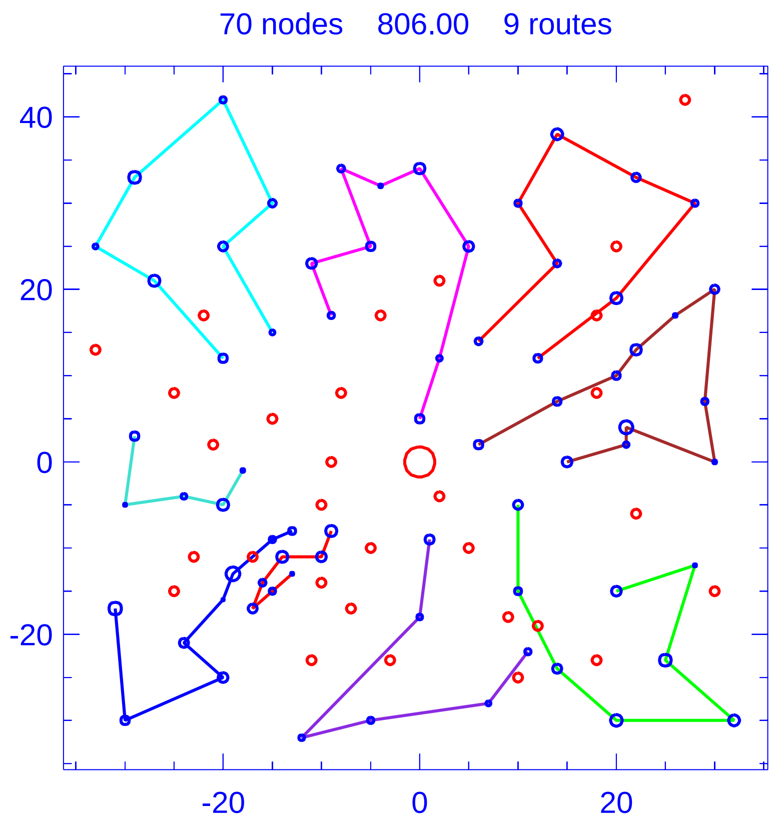


Figure A.77: Solution for day 4 of problem ConVRP-8

Problem			ConVRP-8
Vehicle capacity			200
Maximum route length			230.000
Number of nodes			70
Total route length			805.995
Total number of routes			9
Route	Length	Load	Ordering
1	101.624	116	(0, 1, 51, 20, 66, 71, 35, 81, 50, 0)
2	106.360	152	(0, 4, 55, 39, 67, 23, 75, 21, 26, 0)
3	116.073	121	(0, 7, 19, 11, 64, 49, 36, 47, 82, 0)
4	90.012	155	(0, 12, 80, 68, 24, 29, 34, 78, 79, 3, 76, 28, 0)
5	86.233	123	(0, 27, 69, 30, 32, 90, 63, 10, 62, 88, 0)
6	69.923	54	(0, 45, 17, 84, 5, 60, 0)
7	87.803	53	(0, 58, 2, 43, 15, 41, 74, 0)
8	97.954	151	(0, 86, 38, 14, 44, 91, 85, 99, 96, 0)
9	50.013	112	(0, 92, 37, 100, 98, 59, 95, 94, 0)

Table A.77: Details of solution to day 4 of problem ConVRP-8

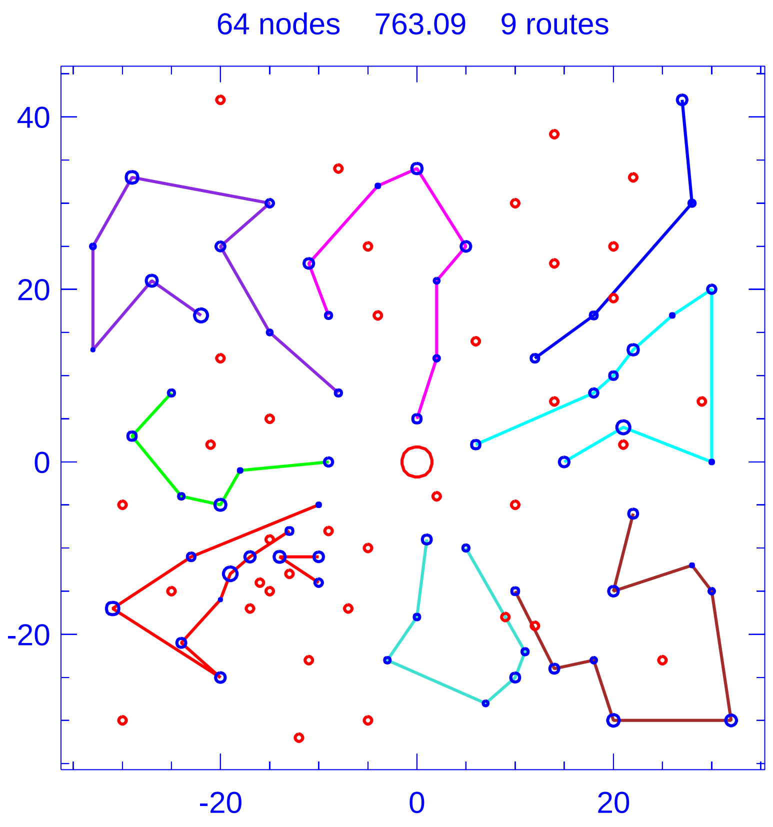


Figure A.78: Solution for day 5 of problem ConVRP-8

Problem			ConVRP-8
Vehicle capacity			200
Maximum route length			230.000
Number of nodes			64
Total route length			763.087
Total number of routes			9
Route	Length	Load	Ordering
1	87.415	164	(0, 6, 61, 86, 14, 44, 91, 85, 93, 96, 0)
2	67.905	76	(0, 8, 45, 84, 5, 60, 89, 0)
3	88.892	141	(0, 12, 68, 24, 34, 78, 79, 3, 77, 28, 0)
4	110.585	134	(0, 21, 75, 56, 23, 67, 25, 55, 4, 54, 0)
5	78.010	102	(0, 27, 69, 70, 30, 32, 90, 62, 88, 0)
6	67.124	72	(0, 40, 74, 22, 41, 57, 2, 58, 0)
7	118.933	142	(0, 48, 47, 46, 36, 49, 11, 19, 7, 52, 0)
8	103.154	52	(0, 50, 33, 35, 65, 0)
9	41.071	60	(0, 95, 59, 97, 0)

Table A.78: Details of solution to day 5 of problem ConVRP-8

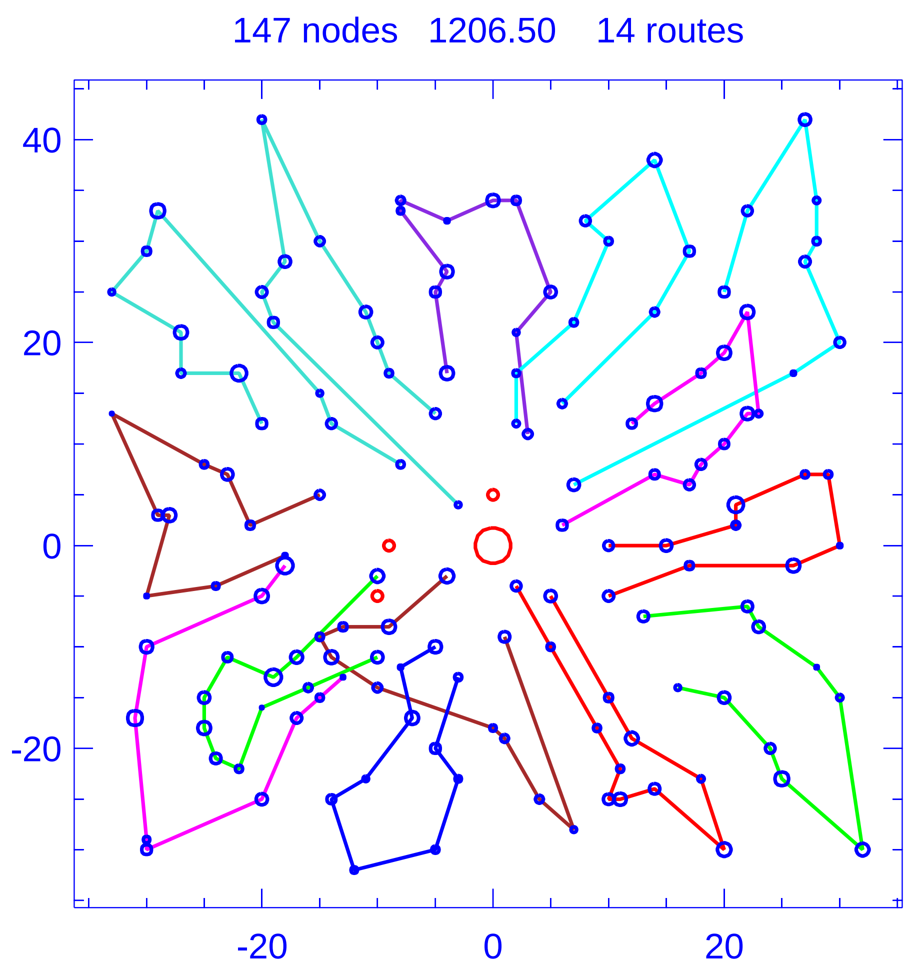


Figure A.79: Template for problem ConVRP-9

Problem			ConVRP-9
Vehicle capacity			200
Maximum route length			200.000
Number of nodes			147
Total route length			1206.497
Total number of routes			14
Route	Length	Load	Ordering
1	77.621	191	(0, 5, 71, 122, 106, 73, 125, 33, 72, 124, 123, 90, 103, 0)
2	96.046	157	(0, 10, 54, 39, 89, 117, 75, 105, 30, 104, 49, 0)
3	109.574	118	(0, 11, 128, 84, 35, 85, 36, 115, 121, 59, 0)
4	77.659	175	(0, 12, 144, 146, 4, 149, 109, 147, 52, 15, 45, 91, 108, 0)
5	96.775	195	(0, 18, 55, 13, 136, 40, 88, 64, 150, 87, 142, 0)
6	96.965	138	(0, 27, 138, 48, 112, 61, 114, 7, 69, 23, 46, 0)
7	82.856	158	(0, 32, 120, 80, 31, 82, 140, 113, 26, 8, 60, 81, 0)
8	76.769	117	(0, 37, 44, 107, 65, 93, 42, 92, 137, 17, 63, 0)
9	71.372	160	(0, 38, 62, 130, 50, 118, 21, 79, 74, 34, 9, 76, 0)

Route	Length	Load	Ordering
10	73.612	205	(0, 47, 143, 135, 111, 66, 41, 94, 19, 141, 148, 145, 0)
11	87.562	105	(0, 51, 101, 3, 116, 28, 70, 22, 1, 119, 0)
12	90.410	110	(0, 68, 133, 14, 58, 96, 95, 25, 67, 134, 110, 0)
13	70.740	208	(0, 78, 126, 16, 127, 53, 129, 29, 20, 131, 83, 2, 100, 0)
14	98.534	164	(0, 102, 6, 57, 99, 43, 86, 97, 24, 98, 132, 0)

Table A.79: Details of template for problem ConVRP-9

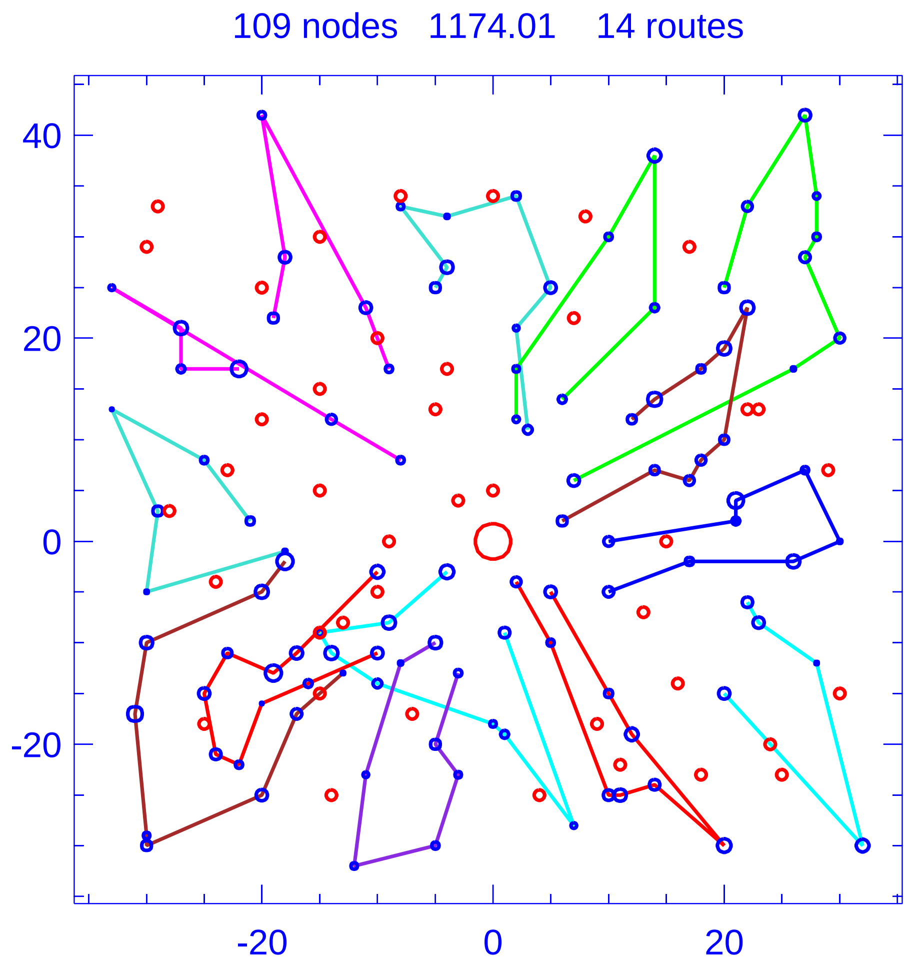


Figure A.80: Solution for day 1 of problem ConVRP-9

Problem			ConVRP-9
Vehicle capacity			200
Maximum route length			200.000
Number of nodes			109
Total route length			1174.013
Total number of routes			14
Route	Length	Load	Ordering
1	75.964	168	(0, 5, 71, 122, 73, 125, 33, 72, 90, 103, 0)
2	109.574	118	(0, 11, 128, 84, 35, 85, 36, 115, 121, 59, 0)
3	77.371	145	(0, 12, 144, 4, 109, 147, 52, 15, 91, 108, 0)
4	96.775	187	(0, 18, 55, 13, 136, 40, 88, 64, 150, 142, 0)
5	95.878	72	(0, 23, 7, 114, 112, 138, 0)
6	81.328	98	(0, 32, 120, 80, 31, 140, 26, 8, 60, 0)
7	74.031	78	(0, 37, 44, 107, 65, 93, 92, 17, 63, 0)
8	69.773	132	(0, 38, 130, 50, 118, 21, 74, 34, 9, 76, 0)
9	73.533	178	(0, 47, 143, 135, 111, 66, 94, 19, 141, 148, 145, 0)

Route	Length	Load	Ordering
10	83.647	67	(0, 51, 101, 116, 70, 1, 119, 0)
11	94.091	83	(0, 54, 117, 105, 30, 104, 0)
12	69.238	179	(0, 78, 126, 16, 127, 53, 20, 131, 83, 2, 100, 0)
13	85.561	102	(0, 98, 24, 97, 86, 6, 102, 0)
14	87.250	42	(0, 110, 67, 95, 96, 58, 133, 0)

Table A.80: Details of solution to day 1 of problem ConVRP-9

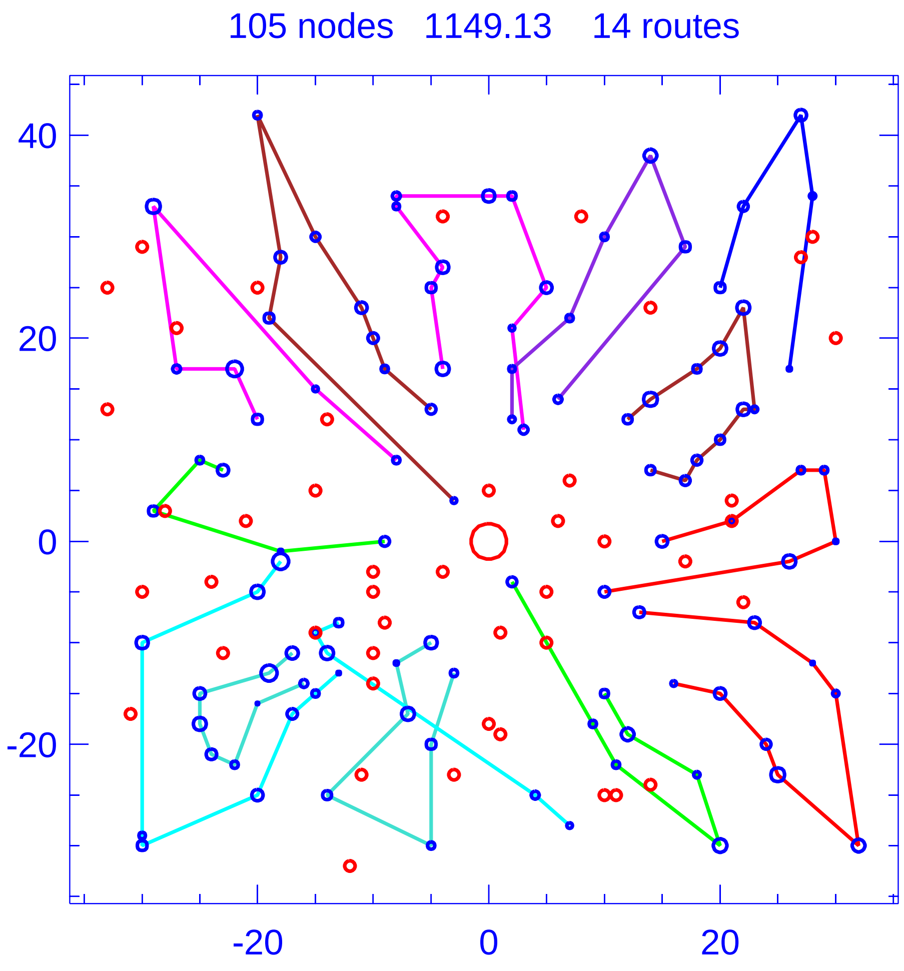


Figure A.81: Solution for day 2 of problem ConVRP-9

Problem			ConVRP-9
Vehicle capacity			200
Maximum route length			200.000
Number of nodes			105
Total route length			1149.131
Total number of routes			14
Route	Length	Load	Ordering
1	94.804	139	(0, 10, 54, 39, 89, 117, 75, 105, 30, 49, 0)
2	62.441	64	(0, 14, 58, 95, 110, 139, 0)
3	96.663	160	(0, 18, 55, 13, 40, 88, 64, 150, 87, 142, 0)
4	96.280	121	(0, 27, 138, 48, 112, 61, 114, 7, 23, 46, 0)
5	81.912	155	(0, 32, 120, 80, 31, 82, 113, 26, 8, 60, 81, 0)
6	71.432	98	(0, 37, 44, 65, 42, 137, 17, 63, 0)
7	85.044	81	(0, 51, 3, 116, 70, 22, 1, 119, 0)
8	106.801	60	(0, 59, 121, 115, 36, 128, 0)
9	70.137	88	(0, 62, 130, 21, 79, 74, 34, 76, 0)

Route	Length	Load	Ordering
10	73.629	102	(0, 71, 122, 106, 73, 124, 123, 103, 0)
11	75.644	63	(0, 91, 45, 109, 149, 146, 0)
12	70.634	192	(0, 100, 2, 83, 131, 20, 29, 129, 53, 127, 16, 126, 0)
13	93.850	106	(0, 102, 57, 99, 24, 98, 132, 0)
14	69.857	147	(0, 143, 135, 66, 41, 94, 19, 141, 148, 0)

Table A.81: Details of solution to day 2 of problem ConVRP-9

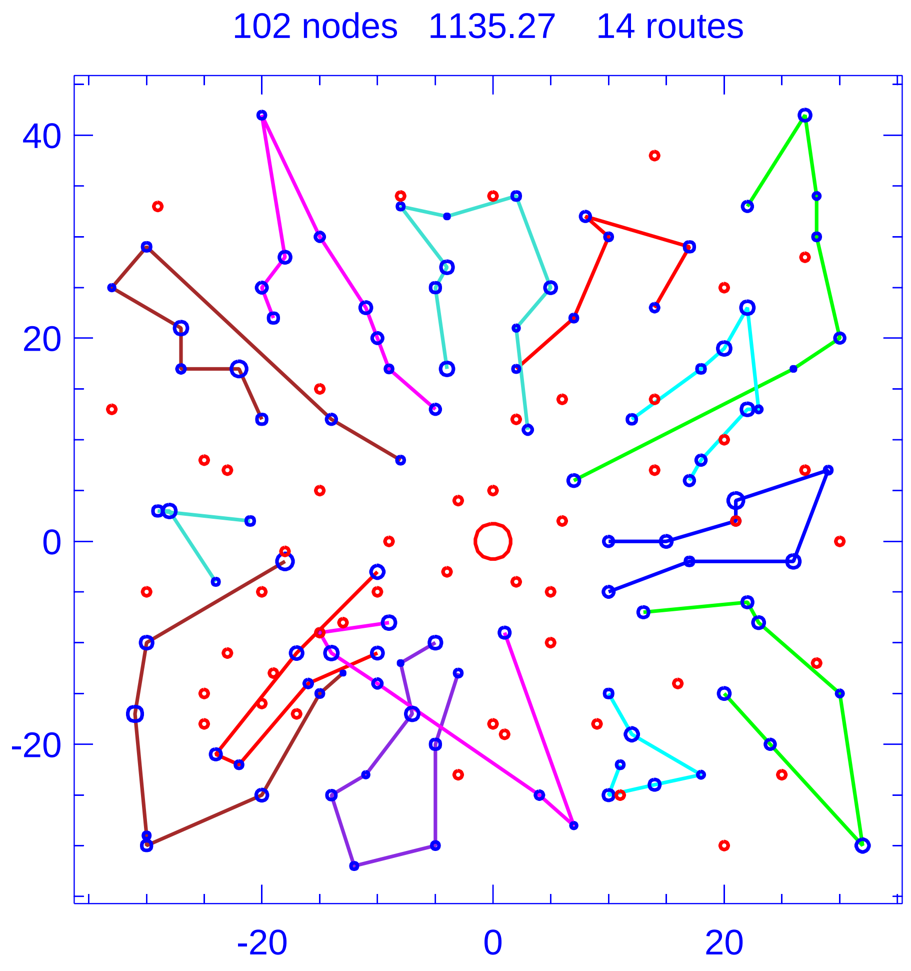


Figure A.82: Solution for day 3 of problem ConVRP-9

Problem			ConVRP-9
Vehicle capacity			200
Maximum route length			200.000
Number of nodes			102
Total route length			1135.266
Total number of routes			14
Route	Length	Load	Ordering
1	78.682	64	(0, 1, 22, 70, 28, 3, 101, 0)
2	108.391	85	(0, 11, 128, 84, 85, 36, 115, 121, 0)
3	69.798	136	(0, 16, 127, 129, 29, 20, 131, 83, 100, 0)
4	96.220	152	(0, 18, 13, 136, 40, 88, 64, 87, 142, 0)
5	96.951	133	(0, 23, 69, 7, 114, 61, 112, 48, 138, 27, 0)
6	81.360	125	(0, 32, 120, 80, 31, 140, 26, 8, 60, 81, 0)
7	75.884	110	(0, 37, 44, 65, 93, 42, 92, 137, 17, 63, 0)
8	69.152	143	(0, 38, 62, 50, 118, 79, 34, 9, 76, 0)
9	67.087	104	(0, 47, 143, 94, 19, 148, 145, 0)

Route	Length	Load	Ordering
10	95.298	119	(0, 49, 104, 30, 75, 117, 39, 54, 0)
11	65.716	86	(0, 71, 122, 106, 125, 72, 124, 0)
12	91.790	129	(0, 102, 6, 43, 86, 97, 24, 98, 132, 0)
13	76.388	109	(0, 108, 91, 45, 147, 109, 4, 144, 0)
14	62.551	62	(0, 133, 95, 25, 134, 0)

Table A.82: Details of solution to day 3 of problem ConVRP-9

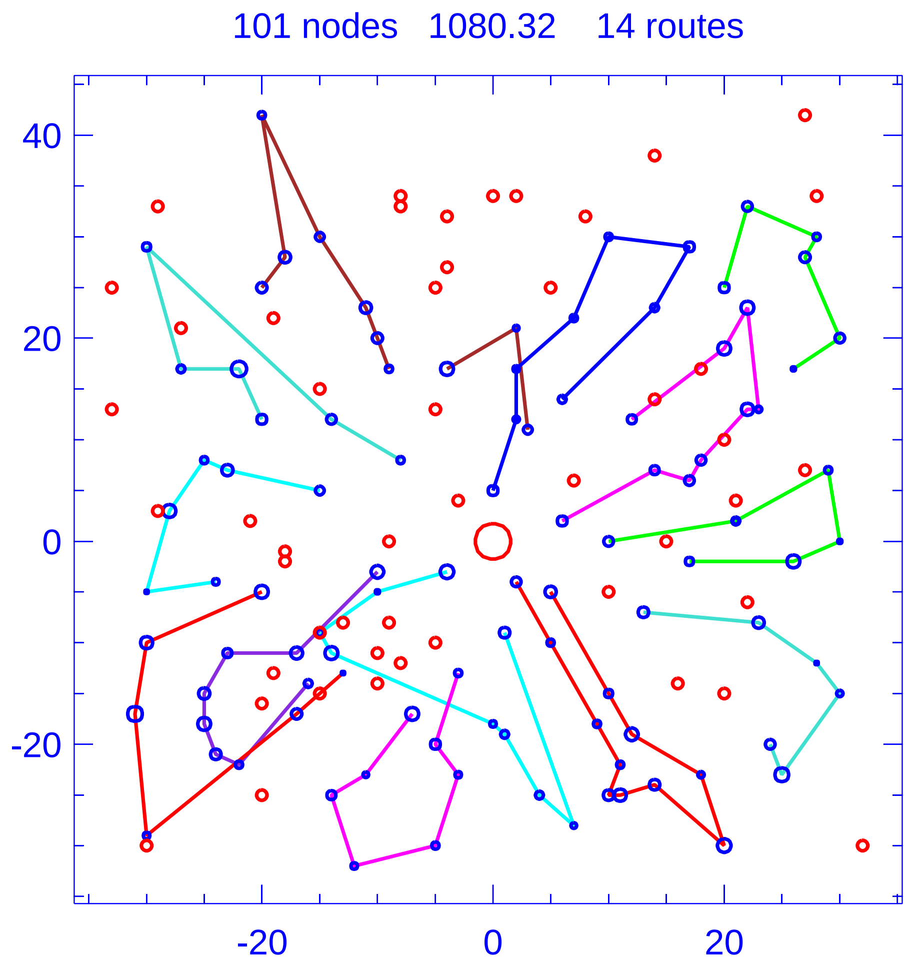


Figure A.83: Solution for day 4 of problem ConVRP-9

Problem			ConVRP-9
Vehicle capacity			200
Maximum route length			200.000
Number of nodes			101
Total route length			1080.321
Total number of routes			14
Route	Length	Load	Ordering
1	77.621	191	(0, 5, 71, 122, 106, 73, 125, 33, 72, 124, 123, 90, 103, 0)
2	68.275	80	(0, 9, 34, 74, 79, 50, 130, 38, 0)
3	76.962	119	(0, 12, 56, 149, 109, 52, 15, 45, 91, 108, 0)
4	46.127	44	(0, 32, 120, 81, 0)
5	75.269	91	(0, 37, 44, 107, 65, 93, 42, 92, 137, 0)
6	78.661	90	(0, 39, 89, 75, 105, 30, 49, 0)
7	71.201	143	(0, 47, 143, 111, 66, 41, 94, 19, 148, 0)
8	73.948	82	(0, 51, 101, 3, 70, 22, 1, 119, 77, 0)
9	92.642	110	(0, 55, 13, 136, 40, 150, 142, 0)

Route	Length	Load	Ordering
10	93.815	73	(0, 59, 121, 85, 35, 84, 128, 0)
11	70.785	79	(0, 68, 14, 58, 25, 67, 134, 0)
12	96.386	102	(0, 69, 7, 114, 61, 112, 48, 138, 0)
13	70.683	154	(0, 78, 126, 16, 127, 129, 29, 20, 131, 100, 0)
14	87.948	97	(0, 102, 6, 43, 24, 98, 132, 0)

Table A.83: Details of solution to day 4 of problem ConVRP-9

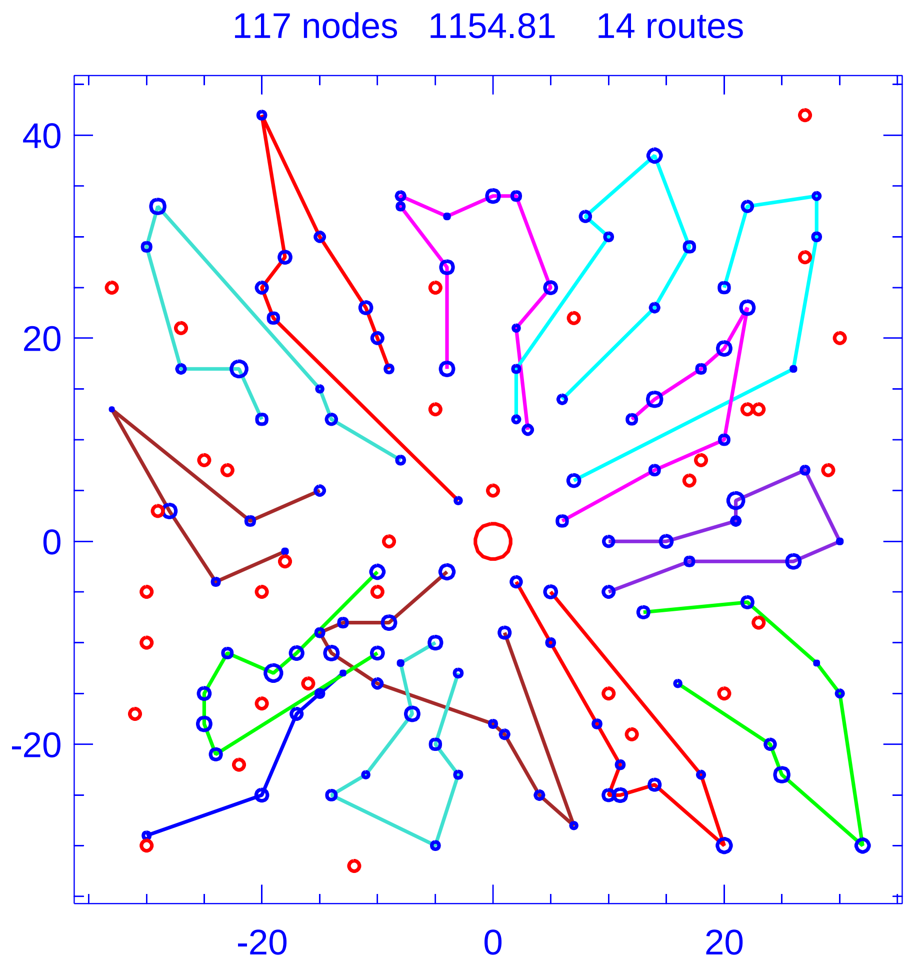


Figure A.84: Solution for day 5 of problem ConVRP-9

Problem			ConVRP-9
Vehicle capacity			200
Maximum route length			200.000
Number of nodes			117
Total route length			1154.810
Total number of routes			14
Route	Length	Load	Ordering
1	76.961	155	(0, 5, 106, 73, 125, 33, 72, 124, 123, 90, 103, 0)
2	95.366	119	(0, 10, 39, 89, 117, 75, 105, 104, 49, 0)
3	94.672	67	(0, 11, 128, 85, 36, 121, 59, 0)
4	77.659	175	(0, 12, 144, 146, 4, 149, 109, 147, 52, 15, 45, 91, 108, 0)
5	82.558	142	(0, 32, 120, 80, 31, 82, 140, 113, 26, 8, 81, 0)
6	72.505	110	(0, 37, 44, 107, 65, 42, 92, 137, 17, 63, 0)
7	69.917	151	(0, 38, 62, 130, 50, 118, 21, 74, 34, 9, 76, 0)
8	85.081	54	(0, 40, 64, 150, 87, 142, 0)
9	96.615	123	(0, 46, 23, 69, 7, 114, 61, 112, 48, 138, 0)

Route	Length	Load	Ordering
10	71.076	185	(0, 47, 143, 135, 111, 66, 41, 94, 145, 0)
11	87.212	97	(0, 51, 101, 3, 116, 28, 70, 1, 119, 0)
12	82.777	62	(0, 68, 133, 96, 25, 134, 110, 0)
13	67.719	150	(0, 78, 126, 53, 20, 131, 83, 2, 100, 0)
14	94.692	132	(0, 102, 6, 57, 99, 43, 24, 98, 132, 0)

Table A.84: Details of solution to day 5 of problem ConVRP-9

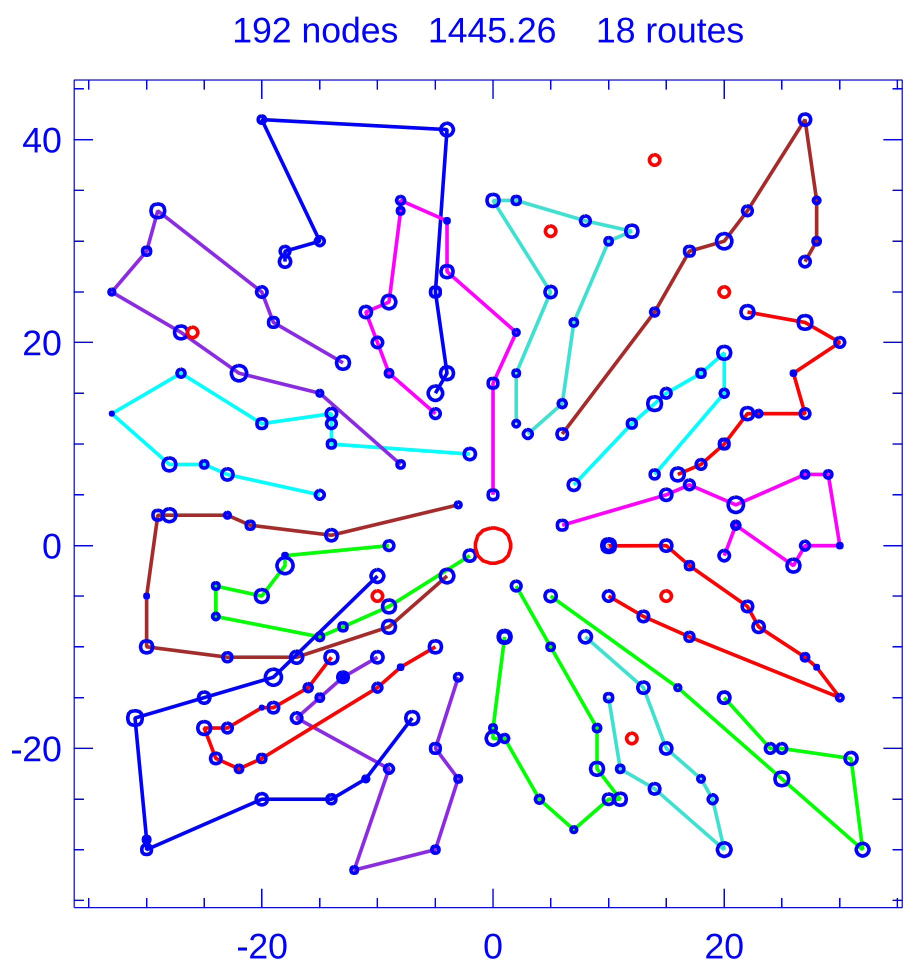


Figure A.85: Template for problem ConVRP-10

Problem			ConVRP-10
Vehicle capacity			200
Maximum route length			200.000
Number of nodes			192
Total route length			1445.264
Total number of routes			18
Route	Length	Load	Ordering
1	70.063	176	(0, 4, 87, 111, 58, 153, 79, 15, 154, 124, 29, 98, 125, 0)
2	56.550	173	(0, 6, 33, 195, 198, 53, 23, 183, 104, 67, 159, 188, 0)
3	82.522	159	(0, 17, 3, 55, 44, 181, 73, 145, 24, 107, 63, 117, 0)
4	103.919	147	(0, 26, 150, 52, 11, 170, 164, 85, 134, 84, 0)
5	73.125	177	(0, 27, 99, 179, 83, 13, 123, 128, 70, 167, 65, 34, 127, 0)
6	74.205	145	(0, 30, 48, 47, 155, 36, 122, 174, 173, 120, 0)
7	95.849	184	(0, 32, 72, 118, 148, 92, 135, 146, 147, 106, 151, 0)
8	108.621	171	(0, 40, 130, 109, 39, 31, 163, 110, 25, 56, 0)
9	80.790	190	(0, 46, 176, 102, 8, 178, 78, 19, 177, 133, 14, 69, 0)

Route	Length	Load	Ordering
10	91.452	172	(0, 54, 59, 138, 166, 20, 37, 88, 5, 103, 0)
11	57.359	137	(0, 60, 149, 51, 7, 132, 180, 35, 175, 0)
12	78.457	200	(0, 61, 193, 192, 160, 62, 116, 144, 74, 49, 182, 16, 95, 0)
13	77.544	169	(0, 76, 187, 97, 161, 9, 75, 162, 189, 57, 169, 12, 126, 0)
14	82.940	145	(0, 81, 100, 71, 119, 38, 77, 80, 131, 129, 50, 168, 0)
15	82.790	124	(0, 86, 93, 156, 114, 142, 22, 199, 136, 1, 191, 194, 0)
16	95.966	207	(0, 96, 184, 115, 185, 89, 137, 113, 91, 141, 186, 0)
17	69.076	175	(0, 112, 66, 196, 42, 68, 143, 90, 41, 190, 43, 197, 158, 0)
18	64.034	206	(0, 152, 139, 172, 21, 82, 121, 140, 94, 64, 28, 101, 2, 157, 0)

Table A.85: Details of template for problem ConVRP-10

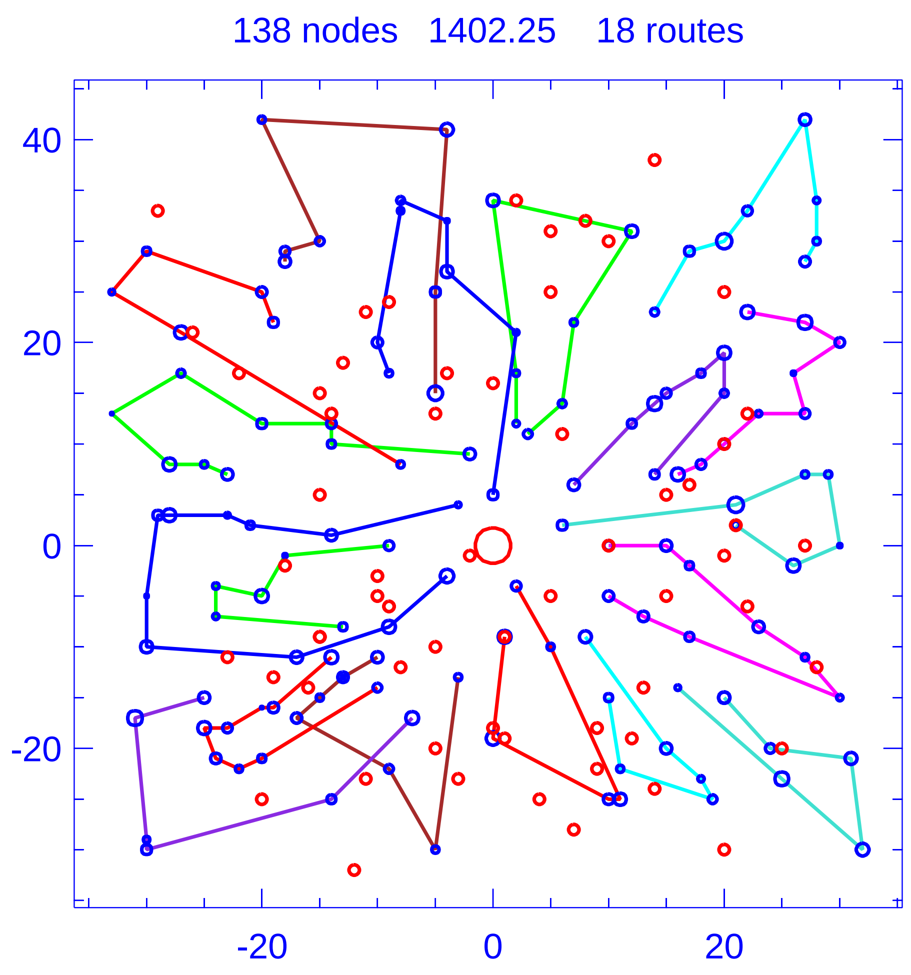


Figure A.86: Solution for day 1 of problem ConVRP-10

Problem			ConVRP-10
Vehicle capacity			200
Maximum route length			200.000
Number of nodes			138
Total route length			1402.254
Total number of routes			18
Route	Length	Load	Ordering
1	59.103	119	(0, 2, 28, 121, 82, 139, 152, 0)
2	81.423	130	(0, 17, 3, 55, 181, 73, 145, 24, 107, 63, 0)
3	65.202	78	(0, 30, 47, 155, 36, 173, 120, 0)
4	108.323	144	(0, 40, 109, 39, 31, 163, 110, 25, 56, 0)
5	80.427	154	(0, 46, 176, 8, 78, 19, 177, 133, 14, 69, 0)
6	91.417	137	(0, 59, 138, 166, 20, 88, 5, 103, 0)
7	57.359	137	(0, 60, 149, 51, 7, 132, 180, 35, 175, 0)
8	78.425	187	(0, 61, 193, 192, 62, 116, 144, 74, 49, 182, 16, 95, 0)
9	89.548	85	(0, 72, 118, 92, 135, 146, 151, 0)

Route	Length	Load	Ordering
10	80.654	90	(0, 81, 100, 71, 38, 131, 50, 168, 0)
11	103.893	129	(0, 84, 134, 85, 164, 170, 11, 52, 150, 0)
12	72.966	94	(0, 86, 114, 22, 199, 136, 1, 191, 194, 0)
13	69.890	126	(0, 87, 111, 58, 79, 15, 124, 29, 98, 125, 0)
14	69.207	108	(0, 99, 83, 123, 128, 70, 167, 127, 0)
15	94.299	116	(0, 115, 185, 89, 137, 91, 186, 0)
16	75.633	90	(0, 126, 169, 57, 189, 162, 75, 97, 187, 0)
17	68.527	139	(0, 158, 43, 190, 41, 90, 143, 68, 42, 196, 0)
18	55.960	68	(0, 188, 159, 104, 183, 23, 195, 0)

Table A.86: Details of solution to day 1 of problem ConVRP-10

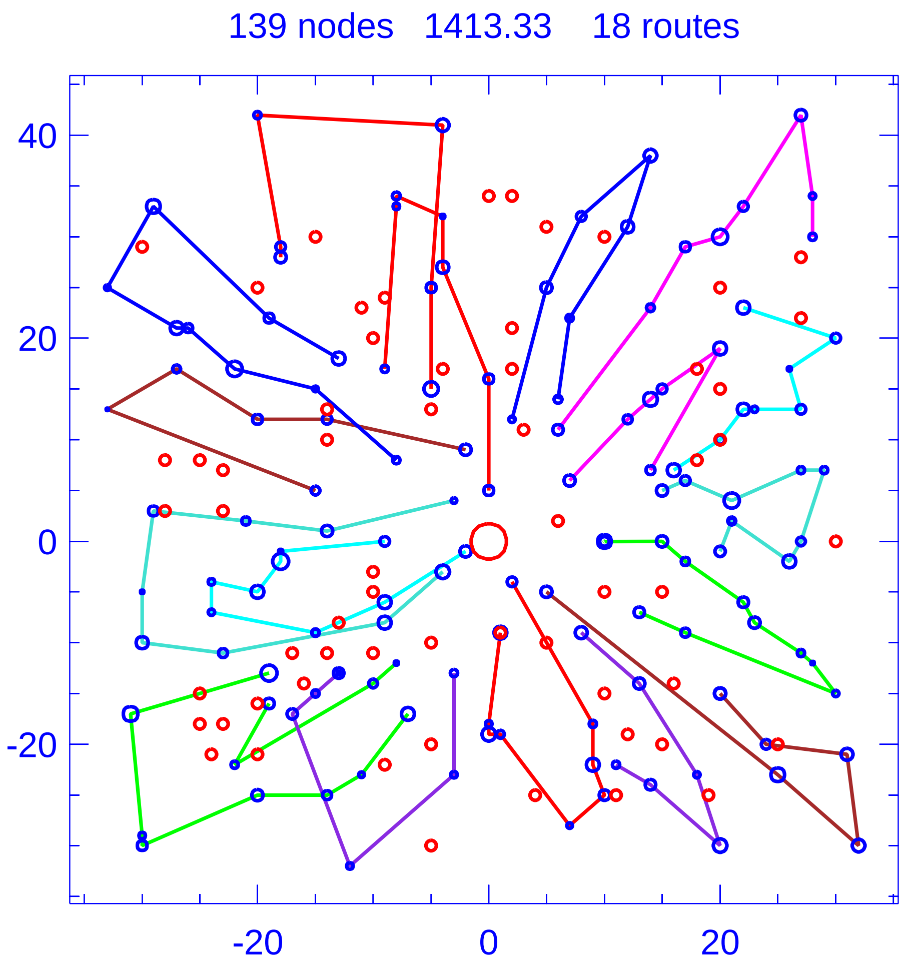


Figure A.87: Solution for day 2 of problem ConVRP-10

Problem			ConVRP-10
Vehicle capacity			200
Maximum route length			200.000
Number of nodes			139
Total route length			1413.326
Total number of routes			18
Route	Length	Load	Ordering
1	62.457	146	(0, 2, 101, 28, 64, 140, 121, 21, 172, 152, 0)
2	70.042	159	(0, 4, 87, 111, 58, 153, 79, 15, 154, 124, 29, 98, 0)
3	56.550	162	(0, 6, 33, 198, 53, 23, 183, 104, 67, 159, 188, 0)
4	78.911	74	(0, 17, 55, 181, 73, 145, 117, 0)
5	103.823	130	(0, 26, 150, 52, 11, 170, 164, 85, 134, 0)
6	70.334	158	(0, 27, 99, 179, 83, 13, 128, 70, 167, 65, 34, 0)
7	73.376	103	(0, 30, 48, 155, 122, 174, 173, 0)
8	95.586	169	(0, 32, 72, 148, 135, 146, 18, 147, 106, 151, 0)
9	105.313	132	(0, 40, 109, 39, 31, 163, 25, 56, 0)

Route	Length	Load	Ordering
10	63.221	42	(0, 43, 68, 196, 66, 0)
11	80.565	129	(0, 46, 102, 178, 78, 19, 177, 133, 69, 0)
12	91.407	153	(0, 54, 138, 166, 20, 88, 5, 103, 0)
13	56.744	116	(0, 60, 149, 51, 7, 180, 175, 0)
14	78.057	145	(0, 61, 193, 160, 62, 116, 144, 182, 16, 95, 0)
15	75.923	68	(0, 86, 156, 142, 199, 136, 1, 191, 0)
16	82.478	108	(0, 100, 71, 38, 165, 77, 129, 168, 0)
17	73.443	84	(0, 126, 12, 57, 189, 162, 75, 187, 0)
18	95.094	163	(0, 184, 185, 89, 137, 113, 91, 141, 186, 0)

Table A.87: Details of solution to day 2 of problem ConVRP-10

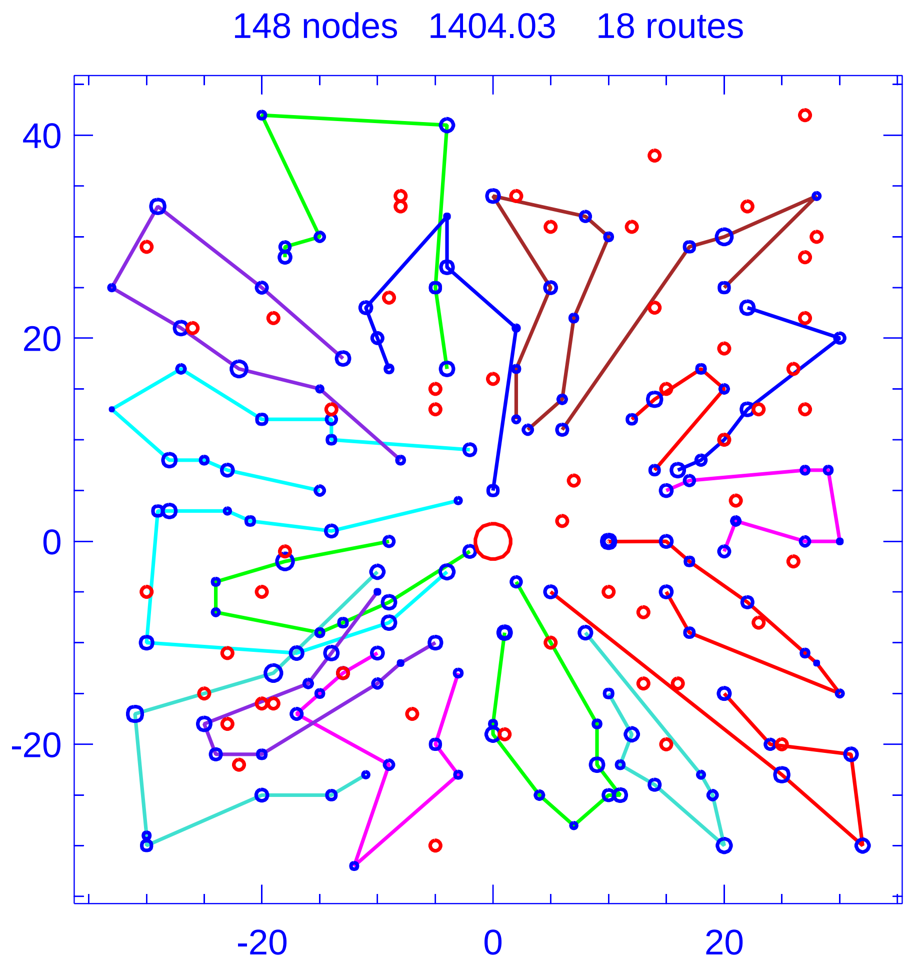


Figure A.88: Solution for day 3 of problem ConVRP-10

Problem			ConVRP-10
Vehicle capacity			200
Maximum route length			200.000
Number of nodes			148
Total route length			1404.027
Total number of routes			18
Route	Length	Load	Ordering
1	70.924	143	(0, 4, 87, 111, 58, 153, 15, 154, 124, 29, 45, 0)
2	54.310	144	(0, 6, 33, 195, 198, 53, 23, 183, 67, 188, 0)
3	81.439	142	(0, 17, 3, 55, 181, 73, 145, 24, 107, 63, 117, 0)
4	89.789	93	(0, 26, 52, 11, 85, 108, 0)
5	69.680	99	(0, 27, 99, 179, 13, 123, 128, 70, 65, 34, 0)
6	74.334	131	(0, 30, 155, 36, 122, 174, 173, 171, 120, 0)
7	95.197	157	(0, 32, 118, 148, 135, 146, 147, 106, 151, 0)
8	77.136	122	(0, 46, 176, 8, 178, 133, 69, 0)
9	91.407	153	(0, 54, 138, 166, 20, 88, 5, 103, 0)

Route	Length	Load	Ordering
10	108.038	138	(0, 56, 25, 110, 163, 31, 39, 109, 130, 0)
11	78.401	185	(0, 61, 193, 192, 62, 144, 74, 49, 182, 16, 95, 0)
12	79.330	110	(0, 81, 100, 71, 119, 77, 131, 129, 50, 168, 0)
13	80.958	114	(0, 86, 93, 156, 142, 22, 199, 136, 1, 194, 0)
14	95.866	162	(0, 96, 184, 185, 89, 137, 113, 91, 141, 0)
15	68.829	135	(0, 105, 158, 197, 90, 143, 42, 196, 66, 112, 0)
16	71.571	92	(0, 126, 169, 57, 189, 161, 97, 187, 0)
17	53.280	77	(0, 149, 51, 132, 35, 175, 0)
18	63.537	187	(0, 152, 172, 21, 82, 121, 140, 94, 28, 101, 2, 157, 0)

Table A.88: Details of solution to day 3 of problem ConVRP-10

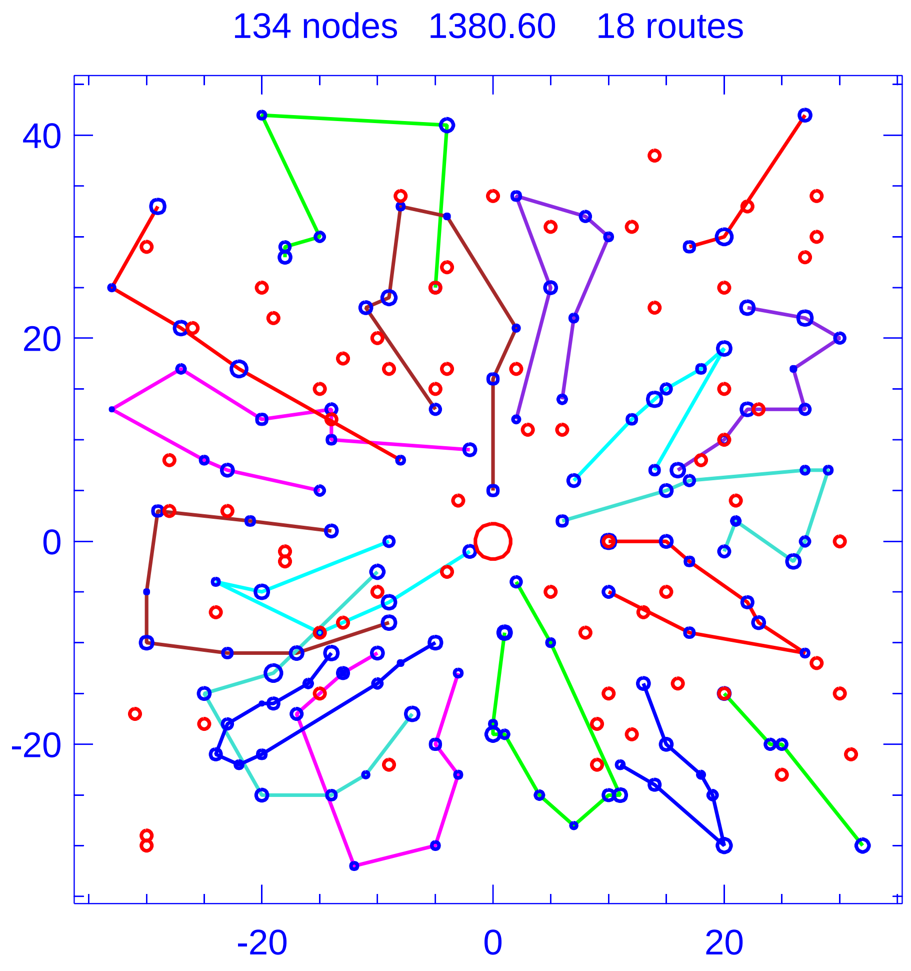


Figure A.89: Solution for day 4 of problem ConVRP-10

Problem			ConVRP-10
Vehicle capacity			200
Maximum route length			200.000
Number of nodes			134
Total route length			1380.605
Total number of routes			18
Route	Length	Load	Ordering
1	60.908	135	(0, 4, 111, 58, 153, 79, 15, 29, 125, 0)
2	88.473	74	(0, 5, 88, 37, 166, 0)
3	53.048	103	(0, 6, 33, 198, 183, 104, 188, 0)
4	75.888	133	(0, 16, 182, 144, 116, 62, 160, 192, 193, 0)
5	81.885	117	(0, 17, 3, 44, 181, 73, 145, 107, 63, 117, 0)
6	69.204	138	(0, 27, 99, 179, 83, 13, 128, 70, 65, 34, 127, 0)
7	80.725	154	(0, 46, 102, 178, 19, 177, 133, 14, 69, 0)
8	73.695	112	(0, 48, 47, 155, 36, 122, 174, 173, 0)
9	100.600	73	(0, 52, 11, 164, 0)

Route	Length	Load	Ordering
10	108.006	95	(0, 56, 25, 110, 163, 31, 109, 0)
11	56.775	127	(0, 60, 149, 51, 7, 132, 180, 175, 0)
12	74.920	110	(0, 76, 161, 9, 75, 189, 169, 12, 126, 0)
13	78.727	104	(0, 86, 93, 156, 114, 142, 199, 1, 191, 194, 0)
14	76.600	149	(0, 96, 184, 115, 113, 91, 141, 186, 0)
15	75.985	79	(0, 100, 71, 119, 77, 80, 129, 168, 0)
16	67.076	148	(0, 112, 66, 196, 42, 68, 143, 41, 190, 43, 197, 158, 0)
17	94.447	107	(0, 148, 135, 146, 147, 151, 0)
18	63.640	169	(0, 152, 139, 82, 121, 140, 94, 64, 28, 101, 2, 157, 0)

Table A.89: Details of solution to day 4 of problem ConVRP-10

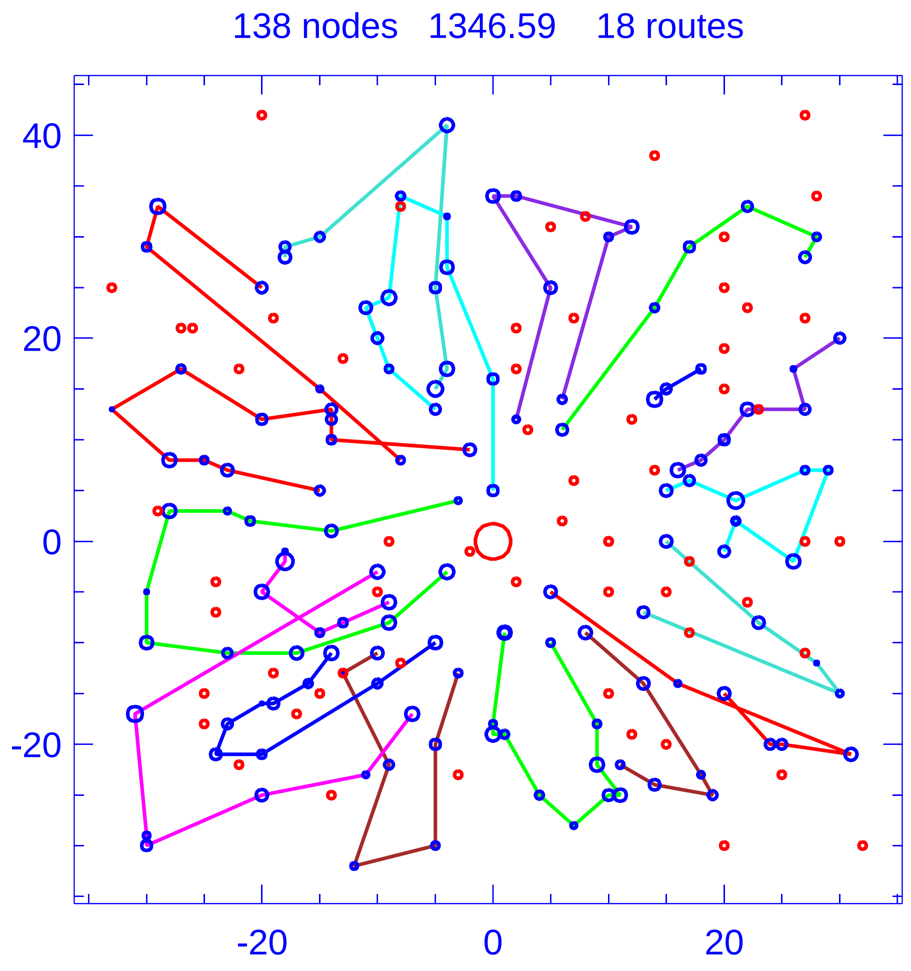


Figure A.90: Solution for day 5 of problem ConVRP-10

Problem			ConVRP-10
Vehicle capacity			200
Maximum route length			200.000
Number of nodes			138
Total route length			1346.590
Total number of routes			18
Route	Length	Load	Ordering
1	82.522	159	(0, 17, 3, 55, 44, 181, 73, 145, 24, 107, 63, 117, 0)
2	87.905	84	(0, 26, 150, 52, 170, 134, 84, 0)
3	70.305	146	(0, 27, 99, 179, 83, 128, 70, 167, 65, 34, 0)
4	64.946	86	(0, 30, 48, 155, 36, 174, 173, 0)
5	46.561	126	(0, 33, 195, 198, 53, 104, 67, 159, 0)
6	95.146	162	(0, 40, 130, 109, 39, 31, 110, 25, 56, 0)
7	76.313	125	(0, 46, 176, 102, 8, 178, 19, 177, 133, 0)
8	49.578	56	(0, 51, 7, 132, 0)
9	76.323	116	(0, 54, 59, 20, 37, 88, 5, 103, 0)

Route	Length	Load	Ordering
10	77.641	184	(0, 61, 193, 192, 160, 62, 116, 74, 49, 182, 16, 95, 0)
11	75.373	157	(0, 76, 187, 97, 161, 9, 162, 189, 57, 12, 126, 0)
12	76.663	74	(0, 86, 93, 114, 142, 22, 191, 194, 0)
13	94.717	134	(0, 96, 185, 89, 137, 113, 141, 186, 0)
14	69.876	64	(0, 98, 124, 154, 79, 111, 0)
15	82.203	104	(0, 100, 119, 38, 80, 131, 129, 168, 0)
16	66.573	136	(0, 112, 196, 42, 143, 41, 190, 43, 197, 158, 0)
17	89.912	72	(0, 118, 148, 92, 106, 151, 0)
18	64.034	192	(0, 139, 172, 21, 82, 121, 140, 94, 64, 28, 101, 2, 157, 0)

Table A.90: Details of solution to day 5 of problem ConVRP-10

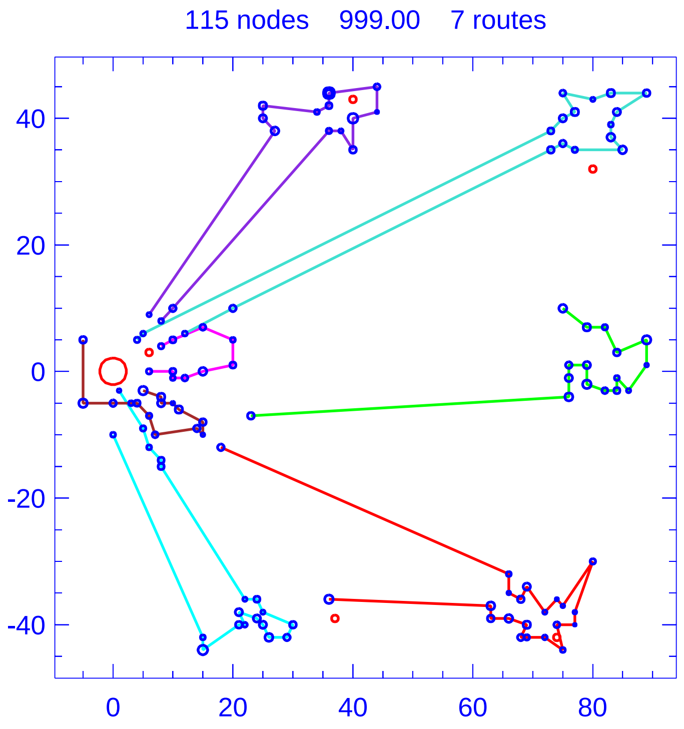


Figure A.91: Template for problem ConVRP-11

Problem			ConVRP-11
Vehicle capacity			200
Maximum route length			N/A
Number of nodes			115
Total route length			999.005
Total number of routes			7
Route	Length	Load	Ordering
1	212.552	204	(0, 8, 17, 16, 19, 25, 22, 24, 27, 33, 31, 34, 36, 29, 35, 32, 28, 26, 23, 20, 21, 108, 0)
2	200.638	204	(0, 40, 43, 45, 48, 51, 50, 49, 47, 46, 44, 41, 42, 39, 38, 37, 109, 0)
3	122.061	207	(0, 81, 2, 1, 3, 4, 5, 10, 11, 15, 14, 13, 9, 7, 6, 83, 113, 117, 112, 88, 0)
4	60.427	182	(0, 87, 92, 89, 91, 90, 114, 118, 18, 84, 85, 86, 111, 82, 119, 120, 0)
5	47.212	102	(0, 95, 96, 93, 94, 97, 115, 110, 116, 99, 101, 0)
6	212.230	206	(0, 100, 98, 52, 54, 57, 65, 61, 62, 64, 66, 63, 60, 56, 58, 55, 53, 106, 105, 0)
7	143.885	205	(0, 104, 103, 73, 76, 68, 77, 79, 80, 75, 72, 74, 71, 70, 69, 67, 107, 0)

Table A.91: Details of template for problem ConVRP-11

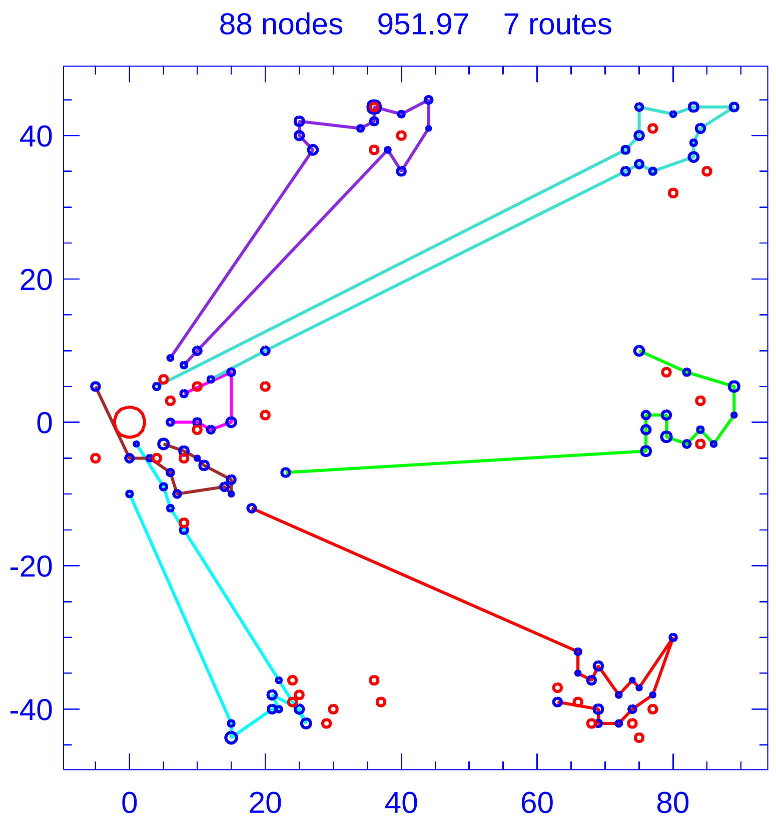


Figure A.92: Solution for day 1 of problem ConVRP-11

Problem			ConVRP-11
Vehicle capacity			200
Maximum route length			N/A
Number of nodes			88
Total route length			951.974
Total number of routes			7
Route	Length	Load	Ordering
1	199.106	129	(0, 16, 25, 24, 27, 31, 36, 29, 35, 32, 28, 26, 23, 20, 21, 108, 0)
2	196.505	164	(0, 40, 45, 51, 50, 49, 47, 44, 41, 42, 39, 38, 37, 109, 0)
3	114.088	140	(0, 81, 2, 1, 3, 4, 5, 11, 15, 6, 83, 117, 112, 88, 0)
4	55.620	136	(0, 87, 92, 91, 90, 114, 118, 18, 84, 85, 111, 82, 120, 0)
5	38.958	66	(0, 95, 96, 94, 97, 116, 101, 0)
6	205.803	167	(0, 100, 98, 52, 54, 57, 61, 62, 64, 66, 63, 60, 56, 55, 53, 105, 0)
7	141.894	162	(0, 104, 103, 76, 68, 79, 80, 78, 75, 74, 71, 70, 69, 67, 107, 0)

Table A.92: Details of solution to day 1 of problem ConVRP-11

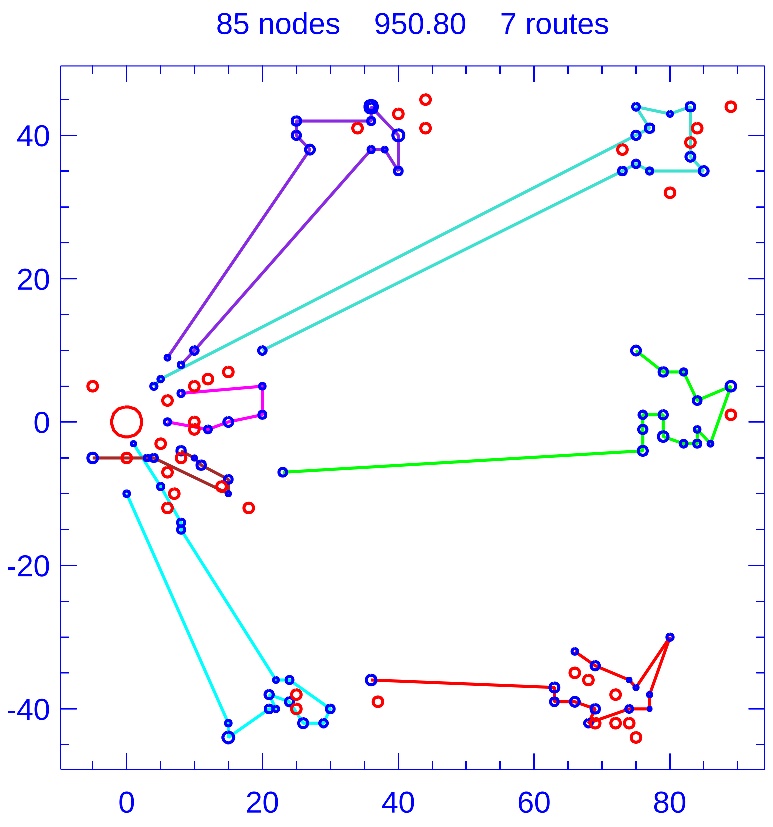


Figure A.93: Solution for day 2 of problem ConVRP-11

Problem			ConVRP-11
Vehicle capacity			200
Maximum route length			N/A
Number of nodes			85
Total route length			950.799
Total number of routes			7
Route	Length	Load	Ordering
1	200.553	149	(0, 8, 17, 16, 19, 25, 22, 31, 34, 36, 29, 35, 32, 26, 21, 0)
2	200.182	200	(0, 40, 43, 45, 48, 51, 49, 47, 46, 44, 41, 42, 39, 38, 37, 109, 0)
3	121.447	180	(0, 81, 2, 1, 3, 4, 5, 10, 15, 14, 13, 7, 6, 83, 113, 112, 88, 0)
4	47.221	91	(0, 92, 91, 90, 114, 118, 86, 111, 119, 0)
5	45.330	63	(0, 95, 94, 97, 115, 110, 101, 0)
6	203.002	154	(0, 98, 52, 54, 57, 65, 61, 63, 60, 56, 58, 55, 106, 105, 0)
7	133.065	185	(0, 104, 103, 73, 76, 68, 77, 75, 72, 74, 70, 69, 67, 107, 0)

Table A.93: Details of solution to day 2 of problem ConVRP-11

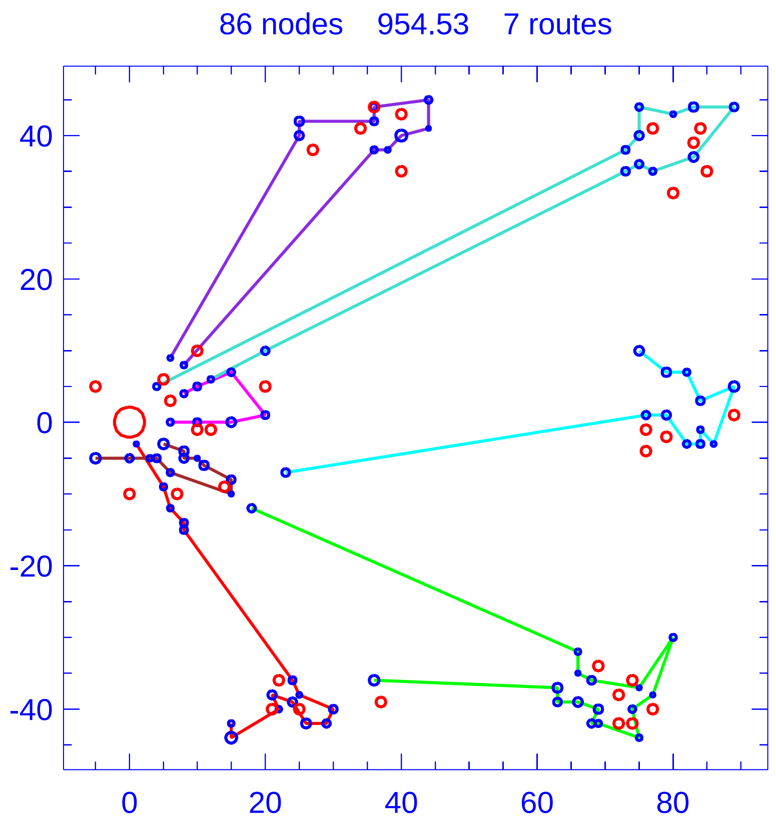


Figure A.94: Solution for day 3 of problem ConVRP-11

Problem			ConVRP-11
Vehicle capacity			200
Maximum route length			N/A
Number of nodes			86
Total route length			954.527
Total number of routes			7
Route	Length	Load	Ordering
1	120.287	167	(0, 2, 1, 4, 5, 10, 15, 14, 13, 9, 7, 83, 113, 117, 112, 88, 0)
2	206.469	173	(0, 8, 17, 16, 19, 25, 22, 24, 33, 31, 36, 29, 35, 23, 20, 21, 108, 0)
3	194.535	148	(0, 40, 43, 45, 48, 51, 49, 47, 46, 44, 42, 39, 109, 0)
4	48.266	147	(0, 87, 92, 89, 91, 90, 114, 118, 85, 86, 111, 82, 119, 0)
5	44.475	78	(0, 95, 96, 97, 115, 116, 99, 101, 0)
6	204.955	146	(0, 100, 98, 52, 54, 57, 61, 66, 63, 60, 56, 55, 53, 105, 0)
7	135.540	121	(0, 104, 73, 76, 77, 79, 80, 72, 74, 70, 69, 107, 0)

Table A.94: Details of solution to day 3 of problem ConVRP-11

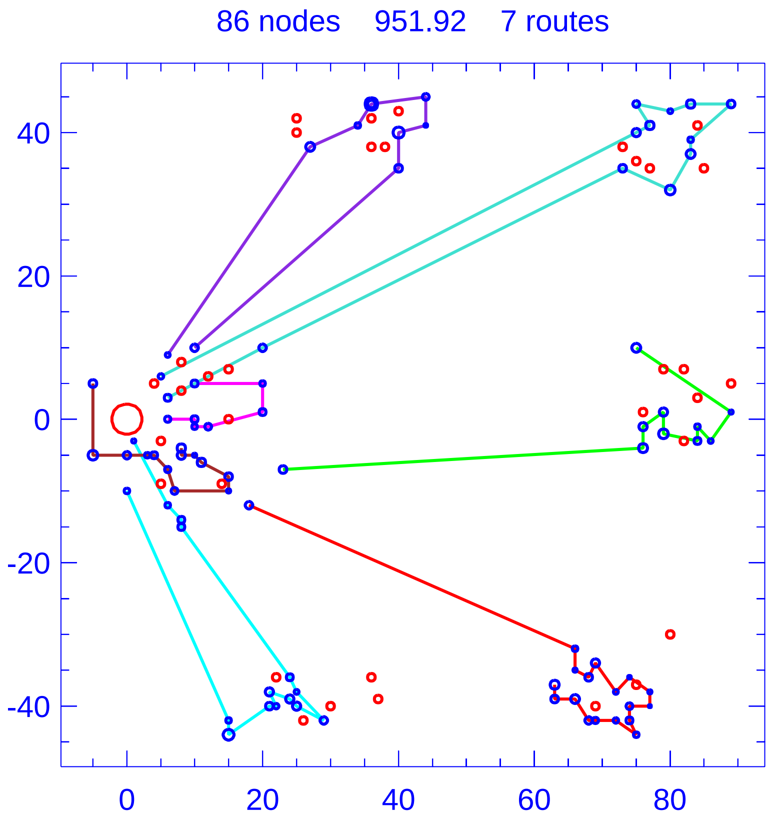


Figure A.95: Solution for day 4 of problem ConVRP-11

Problem			ConVRP-11
Vehicle capacity			200
Maximum route length			N/A
Number of nodes			86
Total route length			951.924
Total number of routes			7
Route	Length	Load	Ordering
1	191.048	167	(0, 17, 16, 19, 22, 24, 27, 33, 30, 31, 34, 36, 32, 28, 26, 23, 20, 21, 108, 0)
2	193.967	123	(0, 40, 50, 49, 47, 46, 41, 42, 38, 37, 109, 0)
3	118.417	163	(0, 81, 2, 1, 3, 4, 5, 10, 11, 14, 9, 7, 83, 113, 117, 88, 0)
4	59.892	149	(0, 92, 89, 91, 90, 114, 118, 84, 85, 86, 111, 82, 119, 120, 0)
5	46.427	66	(0, 95, 96, 93, 94, 115, 110, 99, 0)
6	209.951	159	(0, 102, 98, 52, 59, 61, 62, 66, 63, 60, 56, 58, 55, 106, 0)
7	132.222	143	(0, 103, 68, 77, 79, 80, 75, 72, 71, 67, 107, 0)

Table A.95: Details of solution to day 4 of problem ConVRP-11

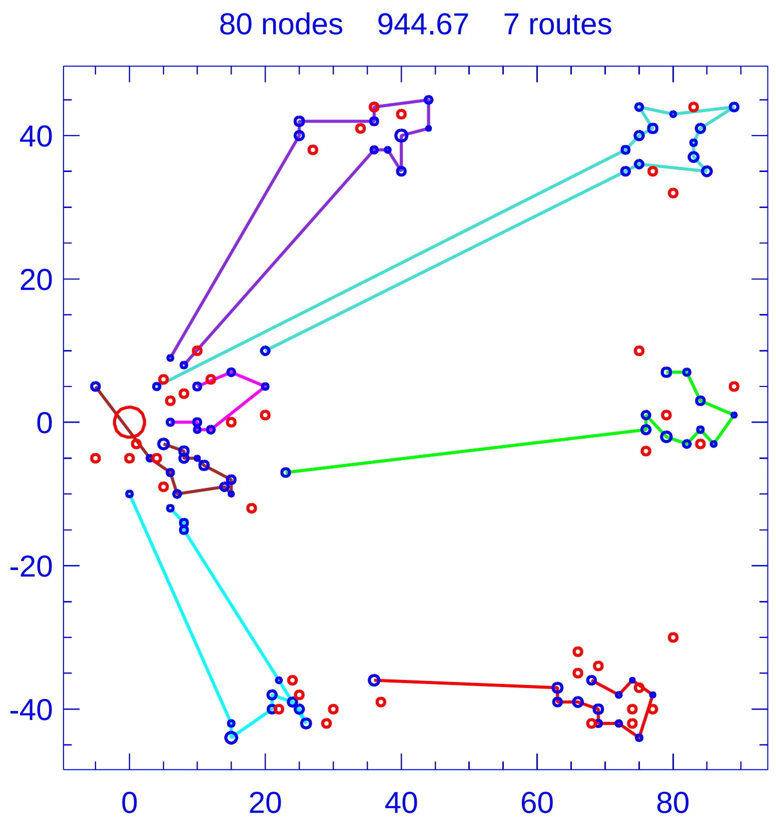


Figure A.96: Solution for day 5 of problem ConVRP-11

Problem			ConVRP-11
Vehicle capacity			200
Maximum route length			N/A
Number of nodes			80
Total route length			944.672
Total number of routes			7
Route	Length	Load	Ordering
1	188.870	129	(0, 8, 17, 16, 19, 25, 24, 27, 33, 36, 32, 28, 23, 0)
2	189.609	123	(0, 43, 45, 48, 50, 49, 47, 44, 41, 39, 38, 109, 0)
3	113.060	148	(0, 81, 2, 1, 3, 5, 10, 11, 15, 6, 83, 113, 117, 0)
4	55.010	139	(0, 87, 92, 89, 91, 90, 114, 118, 18, 84, 85, 111, 120, 0)
5	44.951	65	(0, 95, 96, 93, 94, 110, 116, 99, 0)
6	211.855	169	(0, 98, 52, 54, 65, 61, 62, 64, 66, 60, 56, 58, 55, 53, 105, 0)
7	141.317	134	(0, 104, 73, 76, 68, 77, 79, 80, 72, 74, 70, 69, 107, 0)

Table A.96: Details of solution to day 5 of problem ConVRP-11

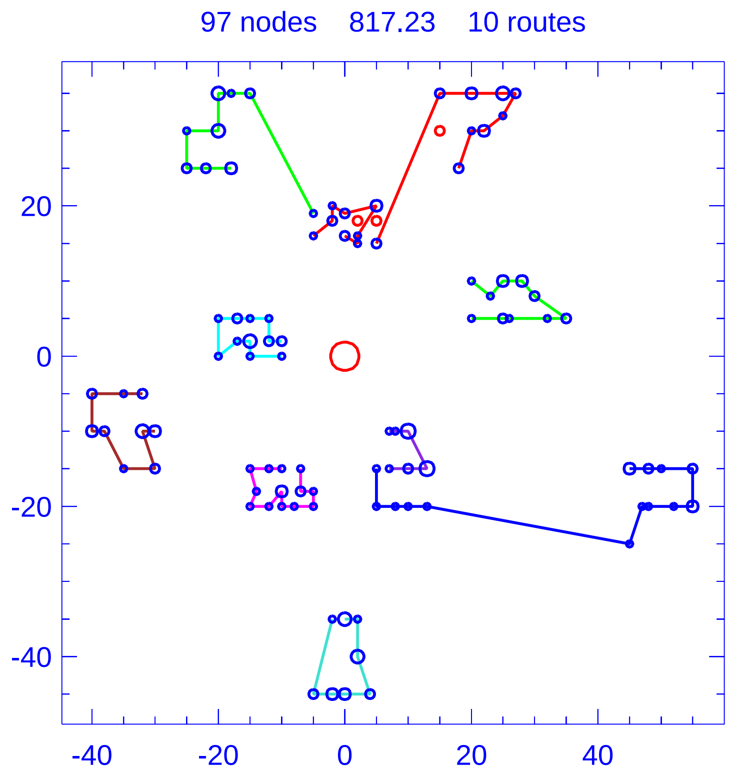


Figure A.97: Template for problem ConVRP-12

Problem			ConVRP-12
Vehicle capacity			200
Maximum route length			N/A
Number of nodes			97
Total route length			817.234
Total number of routes			10
Route	Length	Load	Ordering
1	53.940	130	(0, 7, 5, 3, 2, 6, 9, 8, 10, 0)
2	96.321	200	(0, 11, 12, 14, 16, 15, 19, 18, 17, 13, 0)
3	50.804	170	(0, 20, 24, 25, 27, 29, 30, 28, 26, 23, 22, 21, 0)
4	97.227	200	(0, 32, 33, 31, 35, 37, 38, 39, 36, 34, 0)
5	64.807	160	(0, 43, 42, 41, 40, 44, 45, 46, 48, 51, 50, 52, 49, 47, 0)
6	101.883	200	(0, 57, 55, 54, 53, 56, 58, 60, 59, 0)
7	43.590	150	(0, 66, 62, 74, 63, 65, 67, 0)
8	137.019	200	(0, 69, 68, 64, 61, 72, 80, 79, 77, 73, 70, 71, 76, 78, 81, 0)
9	95.574	200	(0, 75, 100, 97, 93, 92, 94, 95, 96, 98, 0)
10	76.070	170	(0, 90, 87, 86, 83, 82, 84, 85, 88, 89, 91, 0)

Table A.97: Details of template for problem ConVRP-12

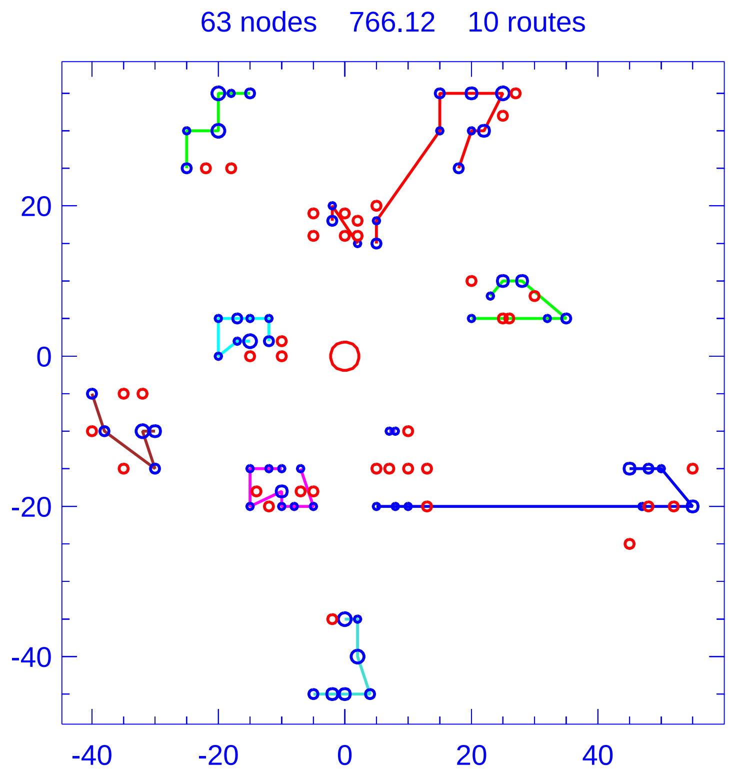


Figure A.98: Solution for day 1 of problem ConVRP-12

Problem			ConVRP-12
Vehicle capacity			200
Maximum route length			N/A
Number of nodes			63
Total route length			766.121
Total number of routes			10
Route	Length	Load	Ordering
1	41.647	40	(0, 5, 9, 8, 0)
2	93.434	140	(0, 12, 14, 16, 15, 19, 18, 0)
3	48.904	130	(0, 22, 23, 26, 28, 30, 29, 27, 25, 0)
4	94.138	130	(0, 32, 33, 31, 37, 39, 0)
5	62.351	110	(0, 43, 40, 44, 45, 46, 51, 52, 49, 47, 0)
6	101.662	190	(0, 57, 55, 54, 53, 56, 58, 60, 0)
7	26.013	20	(0, 65, 67, 0)
8	130.121	130	(0, 68, 64, 61, 79, 70, 76, 78, 81, 0)
9	93.454	190	(0, 75, 1, 99, 100, 97, 93, 95, 96, 98, 0)
10	74.398	110	(0, 89, 88, 85, 82, 83, 90, 0)

Table A.98: Details of solution to day 1 of problem ConVRP-12

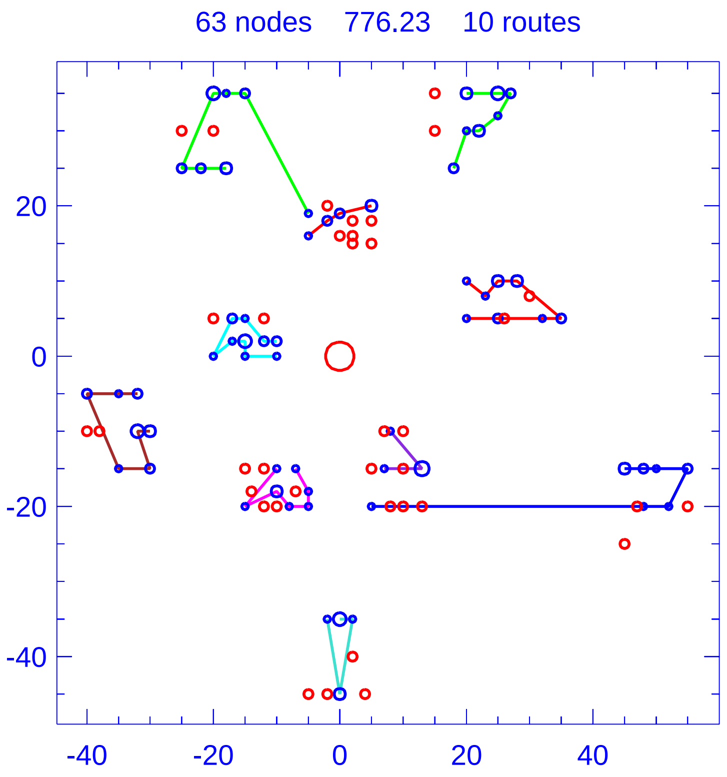


Figure A.99: Solution for day 2 of problem ConVRP-12

Problem			ConVRP-12
Vehicle capacity			200
Maximum route length			N/A
Number of nodes			63
Total route length			776.235
Total number of routes			10
Route	Length	Load	Ordering
1	48.319	80	(0, 2, 6, 8, 10, 0)
2	92.501	150	(0, 11, 12, 14, 16, 18, 17, 13, 0)
3	46.877	150	(0, 20, 24, 25, 27, 29, 28, 26, 22, 21, 0)
4	95.577	150	(0, 32, 33, 31, 35, 39, 36, 34, 0)
5	58.471	90	(0, 43, 41, 40, 44, 46, 51, 47, 0)
6	92.453	90	(0, 57, 55, 56, 59, 0)
7	42.430	70	(0, 65, 74, 66, 0)
8	130.881	110	(0, 68, 77, 73, 71, 76, 78, 81, 0)
9	76.013	140	(0, 90, 87, 83, 82, 85, 88, 89, 91, 0)
10	92.713	160	(0, 97, 93, 92, 94, 95, 96, 98, 0)

Table A.99: Details of solution to day 2 of problem ConVRP-12

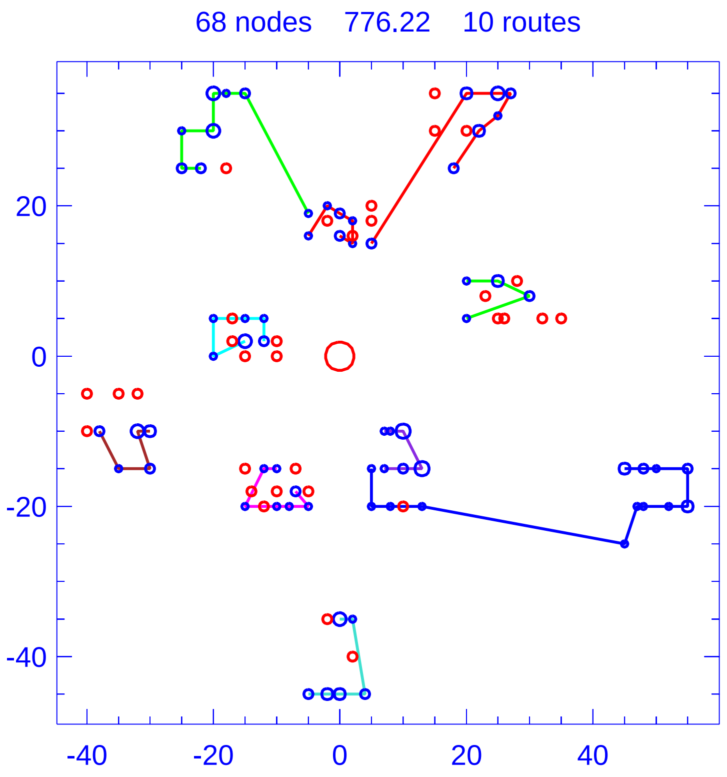


Figure A.100: Solution for day 3 of problem ConVRP-12

Problem			ConVRP-12
Vehicle capacity			200
Maximum route length			N/A
Number of nodes			68
Total route length			776.222
Total number of routes			10
Route	Length	Load	Ordering
1	47.471	80	(0, 7, 5, 4, 6, 9, 10, 0)
2	94.816	170	(0, 11, 12, 14, 16, 15, 19, 18, 17, 0)
3	48.683	100	(0, 22, 23, 26, 30, 29, 25, 0)
4	89.133	120	(0, 32, 33, 31, 35, 37, 0)
5	58.000	80	(0, 42, 40, 44, 45, 51, 49, 47, 0)
6	101.475	150	(0, 57, 55, 53, 56, 58, 60, 0)
7	43.590	150	(0, 66, 62, 74, 63, 65, 67, 0)
8	137.019	190	(0, 69, 68, 64, 72, 80, 79, 77, 73, 70, 71, 76, 78, 81, 0)
9	92.231	170	(0, 75, 97, 93, 92, 94, 95, 98, 0)
10	63.802	70	(0, 90, 84, 88, 91, 0)

Table A.100: Details of solution to day 3 of problem ConVRP-12

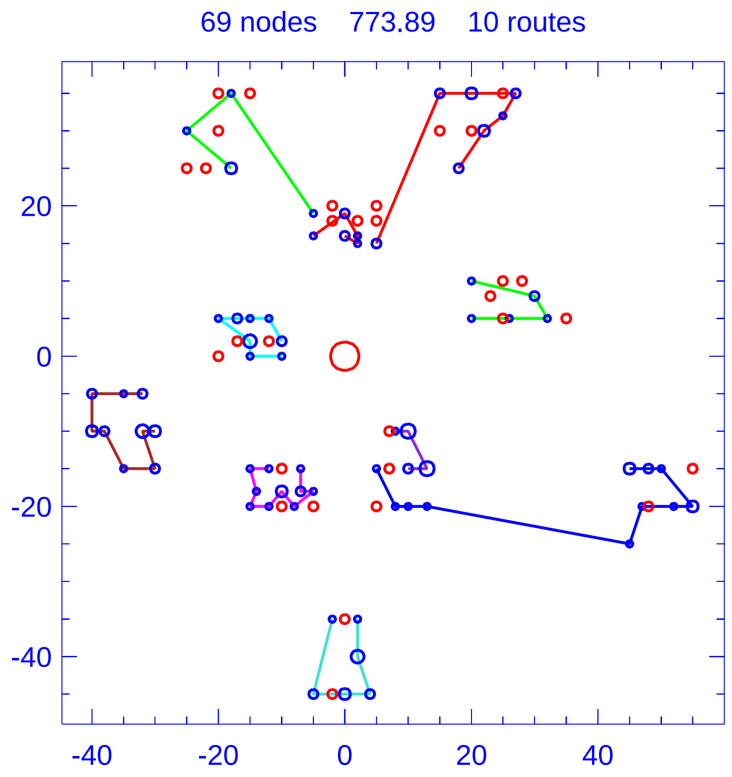


Figure A.101: Solution for day 4 of problem ConVRP-12

Problem			ConVRP-12
Vehicle capacity			200
Maximum route length			N/A
Number of nodes			69
Total route length			773.891
Total number of routes			10
Route	Length	Load	Ordering
1	45.436	70	(0, 7, 5, 3, 6, 10, 0)
2	88.273	60	(0, 11, 14, 19, 13, 0)
3	44.635	130	(0, 20, 24, 25, 30, 28, 26, 23, 21, 0)
4	97.227	200	(0, 32, 33, 31, 35, 37, 38, 39, 36, 34, 0)
5	61.423	130	(0, 43, 42, 41, 44, 46, 48, 51, 50, 52, 49, 0)
6	99.940	130	(0, 55, 54, 53, 56, 60, 59, 0)
7	41.665	130	(0, 62, 74, 63, 65, 0)
8	131.921	160	(0, 69, 64, 61, 72, 80, 79, 73, 70, 76, 78, 81, 0)
9	94.592	150	(0, 75, 100, 97, 92, 94, 95, 98, 0)
10	68.780	60	(0, 90, 86, 83, 84, 91, 0)

Table A.101: Details of solution to day 4 of problem ConVRP-12

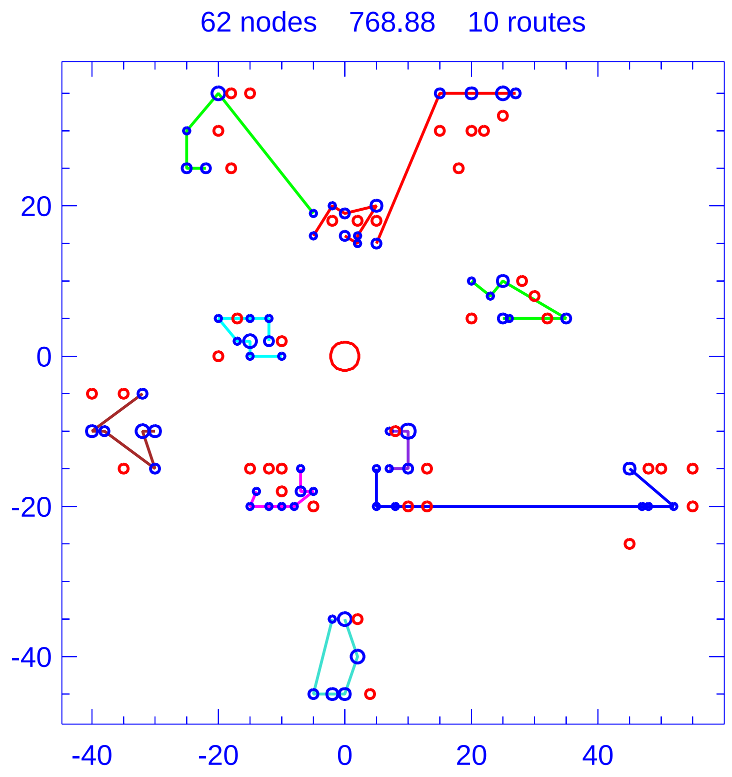


Figure A.102: Solution for day 5 of problem ConVRP-12

Problem			ConVRP-12
Vehicle capacity			200
Maximum route length			N/A
Number of nodes			62
Total route length			768.877
Total number of routes			10
Route	Length	Load	Ordering
1	53.334	110	(0, 7, 5, 3, 2, 6, 9, 10, 0)
2	89.951	100	(0, 11, 16, 19, 18, 17, 0)
3	46.408	120	(0, 20, 24, 25, 27, 30, 26, 23, 22, 0)
4	92.264	160	(0, 32, 33, 31, 37, 38, 34, 0)
5	57.198	90	(0, 43, 42, 41, 44, 45, 48, 51, 50, 0)
6	96.268	170	(0, 57, 54, 56, 58, 60, 59, 0)
7	39.760	90	(0, 66, 62, 63, 67, 0)
8	123.848	90	(0, 69, 68, 64, 79, 77, 73, 81, 0)
9	94.376	130	(0, 75, 100, 97, 93, 92, 0)
10	75.470	100	(0, 87, 86, 82, 88, 89, 91, 0)

Table A.102: Details of solution to day 5 of problem ConVRP-12

Appendix B

Additional Information Related to the Parallel Algorithm

This Appendix contains additional information and data related to our parallel solver for the VRP. We first provide the results produced by our algorithm on the standard benchmark problems when using a single computer equipped with a modern quad-core processor. Next, we provide computational results produced by our algorithm on two newly proposed benchmark problems containing 2400 and 3600 nodes. We then provide the best solutions we have found to the published benchmark instances for the classical VRP. For each problem, we provide a plot of the solution we found, we list the tours traveled by each vehicle in the solution, the total length of each route, the total load carried by each vehicle, and the sum of the lengths of all routes. The final section of this Appendix contains additional data from the computational experiments discussed in Section 4.4.

B.1. Running the parallel solver on a single computer

Multi-core processors are becoming more and more prevalent in the personal computing marketplace. Loosely speaking, these processors allow one to execute multiple threads or processes concurrently on a single CPU, effectively allowing for parallel processing. We ran our parallel solver on a quad-core processor using the standard *mpirun* command, specifying four processors with a single master process, two heuristic solvers, and a single set covering solver. We made a single run of our algorithm on the four sets of standard benchmark problems discussed in Chapter 4. We set a time limit for our procedure that is based on the number of nodes in the problem; this time limit is comparable to the average amount of time required by other metaheuristic solvers from the literature. For problems with less than 100 nodes, we set a time limit of 60 seconds, for problems with 100-199 nodes, we used 100 seconds, for problems with 200-500 nodes, we used 200 seconds, and we used a time limit of 300 seconds for problems containing more than 500 nodes.

We present the results of this experiment in Tables B.1–B.4. For each of the four sets of standard benchmark problems, we compare the solutions found by a single run of our algorithm with those found by the most powerful serial algorithms in the literature[1]. When comparing our results with those found by others, we chose to restrict our attention to those that publish the results of a single run rather than those that publish the best result found in multiple runs. For all four problem sets, the solutions produced by our parallel algorithm when run on a single machine are very competitive. In terms of average deviation from the best-known solutions, only the AGES algorithm of Mester and Bräysy[54] is able to produce better solutions than our algorithm.

[1]We became aware of the 2009 publication of Prins [65] just one week before the dissertation defense date.

Problem	Number of Nodes, Routes	D-Ants [69]	SEPAS [78]	AGES [54]	Prins [65]	Our solution	Num. secs.
1	50, 5	524.61	524.61	524.61	524.61	524.61	60
2	75,10	840.61	835.26	835.26	835.26	835.26	60
3	100, 8	828.21	826.14	826.14	826.14	826.14	100
4	150,12	1037.57	1028.42	1028.42	1029.48	1028.42	100
5	199,16	1306.91	1311.48	1291.29	1294.09	1291.45	100
6	50, 6	555.43	555.43	555.43	555.43	555.43	60
7	75,11	917.50	909.68	909.68	909.68	909.68	60
8	100, 9	865.94	865.94	865.94	865.94	865.94	100
9	150,14	1173.94	1162.55	1162.55	1162.55	1162.99	100
10	199,18	1415.53	1407.21	1401.12	1401.46	1398.52	100
11	120, 7	1043.46	1042.11	1042.11	1042.11	1042.11	100
12	100,10	819.56	819.56	819.56	819.56	819.56	100
13	120,11	1546.84	1544.01	1541.14	1545.43	1542.86	100
14	100,11	866.37	866.37	866.37	866.37	866.37	100
Avg. deviation from best known solution		0.48%	0.20%	0.03%	0.03%	0.03%	

Table B.1: Solutions to the problems of Christofides et al. [19, 20] found using a single machine

Problem	Number of Nodes, Routes	AGES [53]	Our solution	Num. secs.
100A	100,11	2041.34	2041.34	100
100B	100,11	1939.90	1940.61	100
100C	100,11	1406.20	1406.20	100
100D	100,11	1581.25	1581.25	100
150A	150,15	3055.23	3055.23	100
150B	150,14	2727.67	2727.67	100
150C	150,15	2343.11	2362.39	100
150D	150,14	2645.40	2659.12	100
385	385,48	24855.32	24491.78	200
Avg. deviation from best known solution		0.24%	0.22%	

Table B.2: Solutions to the problems of Taillard [76] found using a single machine

Problem	Number of Nodes, Routes	SEPAS [78]	AGES [54]	Prins [65]	Our solution	Num. secs.
1	240, 9	5676.97	5627.54	5644.52	5644.44	200
2	320,10	8459.91	8447.92	8444.50	8447.92	200
3	400,10	11036.22	11036.22	11036.22	11036.22	200
4	480,10	13637.53	13624.52	13624.52	13624.52	200
5	200, 5	6460.98	6460.98	6460.98	6460.98	200
6	280, 7	8414.28	8412.88	8412.90	8412.90	200
7	360, 9	10216.50	10195.56	10195.59	10195.59	200
8	440,10	11936.16	11663.55	11643.90	11691.76	200
9	255,14	585.43	583.39	586.18	584.57	200
10	323,16	746.56	741.56	744.36	741.09	200
11	399,18	923.17	918.45	922.40	919.40	200
12	483,19	1130.40	1107.19	1116.12	1120.31	200
13	252,26	865.01	859.11	860.55	860.40	200
14	320,30	1086.07	1081.31	1084.82	1080.55	200
15	396,33	1353.91	1345.23	1352.39	1340.82	200
16	480,37	1634.74	1622.69	1634.27	1622.35	200
17	240,22	708.74	707.79	707.79	707.79	200
18	300,27	1006.90	998.73	1002.15	1007.65	200
19	360,33	1371.01	1366.86	1371.67	1366.07	200
20	420,38	1837.67	1820.09	1830.98	1823.41	200
Avg. deviation from best known solution		0.84%	0.23%	0.49%	0.37%	

Table B.3: Solutions to the problems of Golden et al. [36] found using a single machine

B.2. Additional Large Benchmark Problems

In Kytöjoki et al. [45], the authors use the VRP problem generator given in Li et al. [51] to produce large benchmark VRP instances containing between 2400 and 20,000 nodes. We ran our parallel algorithm on the 2400 and 3600 node problems and varied the number of processors devoted to solving the set covering problem. Our interest in this experiment was to test the parallel algorithm's performance on even larger problems and to examine the importance of the set covering solvers for these very large problems.

Problem	Number of Nodes, Routes	VRTR [51]	GVNS [45]	AGES [54]	Our solution	Num. secs.
21	560,10	16602.99	16221.22	16212.74	16212.83	300
22	600,15	14651.27	14654.87	14597.18	14634.04	300
23	640,10	18838.62	18810.72	18801.12	18801.13	300
24	720,10	21616.25	21401.31	21389.33	21389.43	300
25	760,19	17146.41	17358.18	17095.27	16849.08	300
26	800,10	24009.74	23996.86	23971.74	23977.73	300
27	840,20	17823.40	18233.93	17488.74	17764.78	300
28	880,10	26606.11	26592.05	26565.92	26566.03	300
29	960,10	29181.12	29166.32	29160.33	29154.34	300
30	1040,10	31961.58	31805.28	31742.51	31935.56	300
31	1120,10	35355.50	34352.48	34330.84	35322.07	300
32	1200,11	37410.84	37025.37	36928.70	37268.98	300
Avg. deviation from best known solution		1.17%	0.80%	0.20%	0.60%	

Table B.4: Solutions to the problems of Li et al. [51] found using a single machine

We used 65 processors for 600 seconds and varied the number of set covering solvers, using 1,2,4, or 8 set covering solvers. We ran each configuration five times and present the results of this experiment in Table B.5. For each configuration, we provide the mean final solution value obtained over the five runs.

Configuration	2400 node problem	3600 node problem
1 Set Covering Solver	75748.45	114579.49
2 Set Covering Solvers	75747.57	114578.73
4 Set Covering Solvers	75748.26	114580.31
8 Set Covering Solvers	75749.38	114579.19
Solution from Kytojoki et al. [45]	75754.55	114579.08
Visually estimated solution from [45]	75743.77	114568.30

Table B.5: Performance of the parallel algorithm on two very large problems from Kytöjoki et al. [45]

For both problems, all four configurations perform equally well and the solutions we find are within 0.01% of the visually estimated solution. There appears to be no advantage of using the set covering solvers for problems of this size. In fact, when examining the logs of these computations, we found that the set covering solvers were never able to combine routes from different solutions into a better solution. Intuitively, this makes sense since the routes in these problems each contain several hundred nodes so that it is more difficult to find a collection of routes from different solutions that contain all the nodes. Although we find solutions that are slightly superior to those in Kytöjoki et al. [45], it is important to note that their running times are very low. Their algorithm requires less than two minutes on a single machine for each problem.

B.3. Christofides and Eilon Benchmarks

This section contains the best solutions we found for the 14 problem instances of Christofides et al. [19, 20].

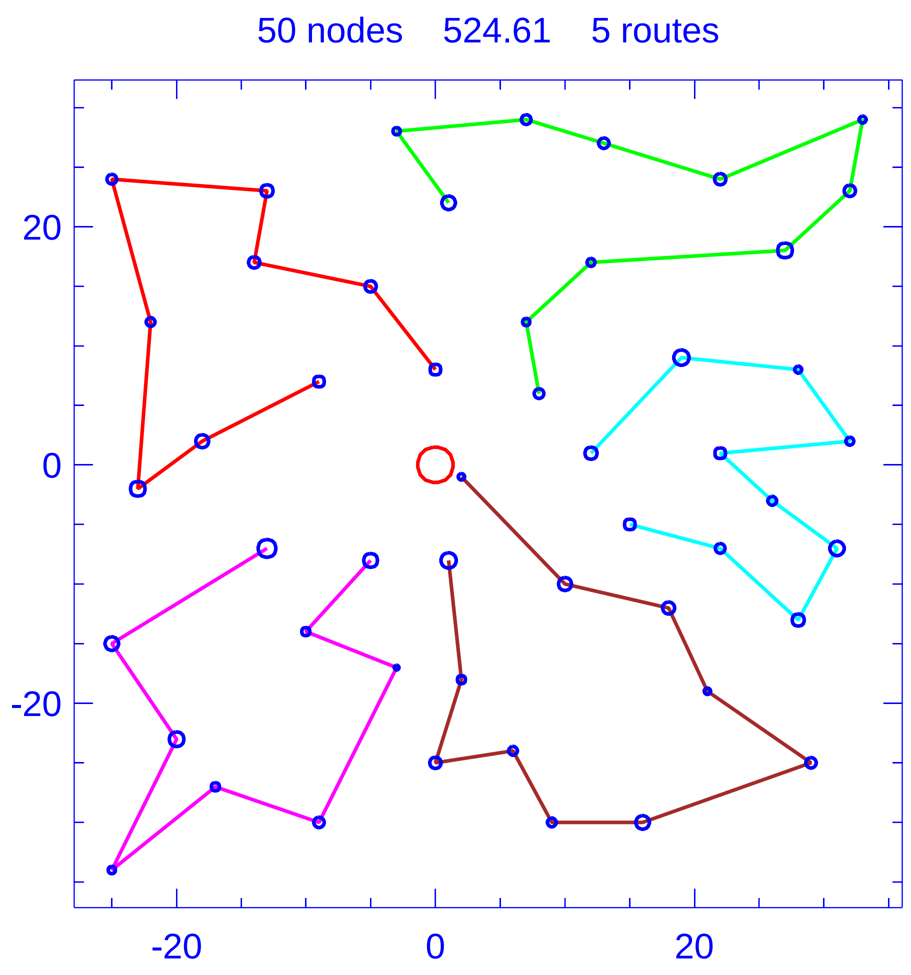


Figure B.1: Solution for problem Christofides-1

Problem			Christofides-1
Vehicle capacity			160
Maximum route length			N/A
Number of nodes			50
Total route length			524.611
Total number of routes			5
Route	Length	Load	Ordering
1	98.452	152	(0, 6, 14, 25, 24, 43, 7, 23, 48, 27, 0)
2	118.519	149	(0, 8, 26, 31, 28, 3, 36, 35, 20, 22, 1, 32, 0)
3	99.333	159	(0, 11, 2, 29, 21, 16, 50, 34, 30, 9, 38, 0)
4	99.251	160	(0, 12, 37, 44, 15, 45, 33, 39, 10, 49, 5, 46, 0)
5	109.056	157	(0, 18, 13, 41, 40, 19, 42, 17, 4, 47, 0)

Table B.6: Details of solution to problem Christofides-1

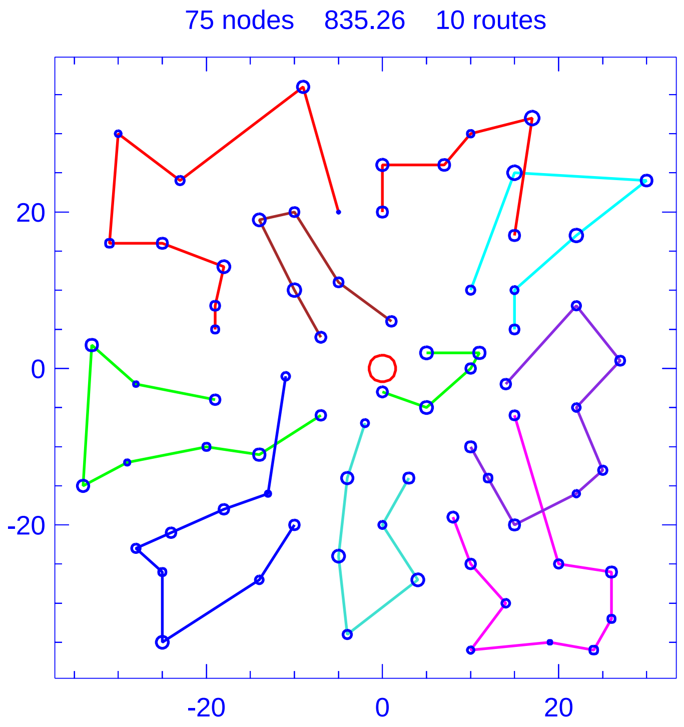


Figure B.2: Solution for problem Christofides-2

Problem			Christofides-2
Vehicle capacity			140
Maximum route length			N/A
Number of nodes			75
Total route length			835.262
Total number of routes			10
Route	Length	Load	Ordering
1	120.164	138	(0, 3, 44, 32, 50, 18, 55, 25, 31, 72, 0)
2	92.690	140	(0, 6, 33, 63, 23, 56, 24, 49, 16, 0)
3	86.328	133	(0, 7, 11, 59, 14, 35, 8, 0)
4	52.931	132	(0, 17, 40, 9, 39, 12, 26, 0)
5	107.066	139	(0, 27, 37, 20, 70, 60, 71, 69, 36, 47, 48, 0)
6	74.378	140	(0, 30, 74, 21, 61, 28, 2, 68, 0)
7	89.532	139	(0, 45, 29, 5, 15, 57, 13, 54, 19, 52, 0)
8	100.011	138	(0, 51, 73, 1, 43, 41, 42, 64, 22, 62, 0)
9	83.084	139	(0, 53, 66, 65, 38, 10, 58, 0)
10	29.077	126	(0, 67, 46, 34, 4, 75, 0)

Table B.7: Details of solution to problem Christofides-2

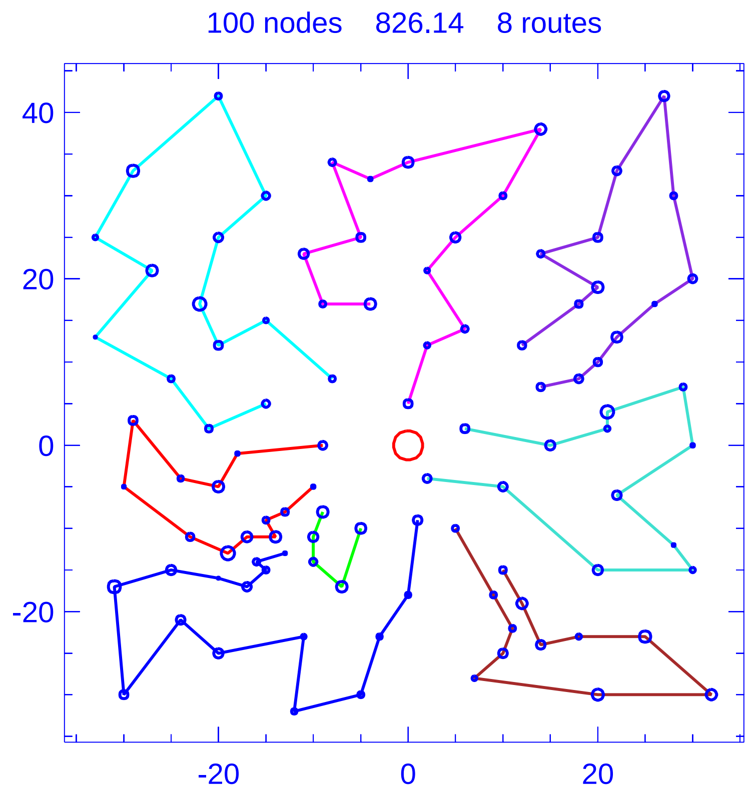


Figure B.3: Solution for problem Christofides-3

Problem			Christofides-3
Vehicle capacity			200
Maximum route length			N/A
Number of nodes			100
Total route length			826.137
Total number of routes			8
Route	Length	Load	Ordering
1	82.730	196	(0, 6, 96, 99, 59, 93, 85, 61, 17, 45, 84, 5, 60, 89, 0)
2	40.907	108	(0, 13, 87, 97, 95, 94, 0)
3	138.795	199	(0, 18, 83, 8, 46, 47, 36, 49, 64, 11, 19, 48, 82, 7, 52, 0)
4	106.062	194	(0, 21, 72, 75, 56, 39, 67, 23, 41, 22, 74, 73, 40, 0)
5	113.935	199	(0, 27, 69, 1, 70, 30, 20, 66, 32, 90, 63, 10, 62, 88, 31, 0)
6	98.251	165	(0, 28, 12, 80, 68, 29, 24, 54, 55, 25, 4, 26, 53, 0)
7	118.793	199	(0, 50, 33, 81, 51, 9, 71, 65, 35, 34, 78, 79, 3, 77, 76, 0)
8	126.664	198	(0, 58, 2, 57, 15, 43, 42, 14, 44, 38, 86, 16, 91, 100, 37, 98, 92, 0)

Table B.8: Details of solution to problem Christofides-3

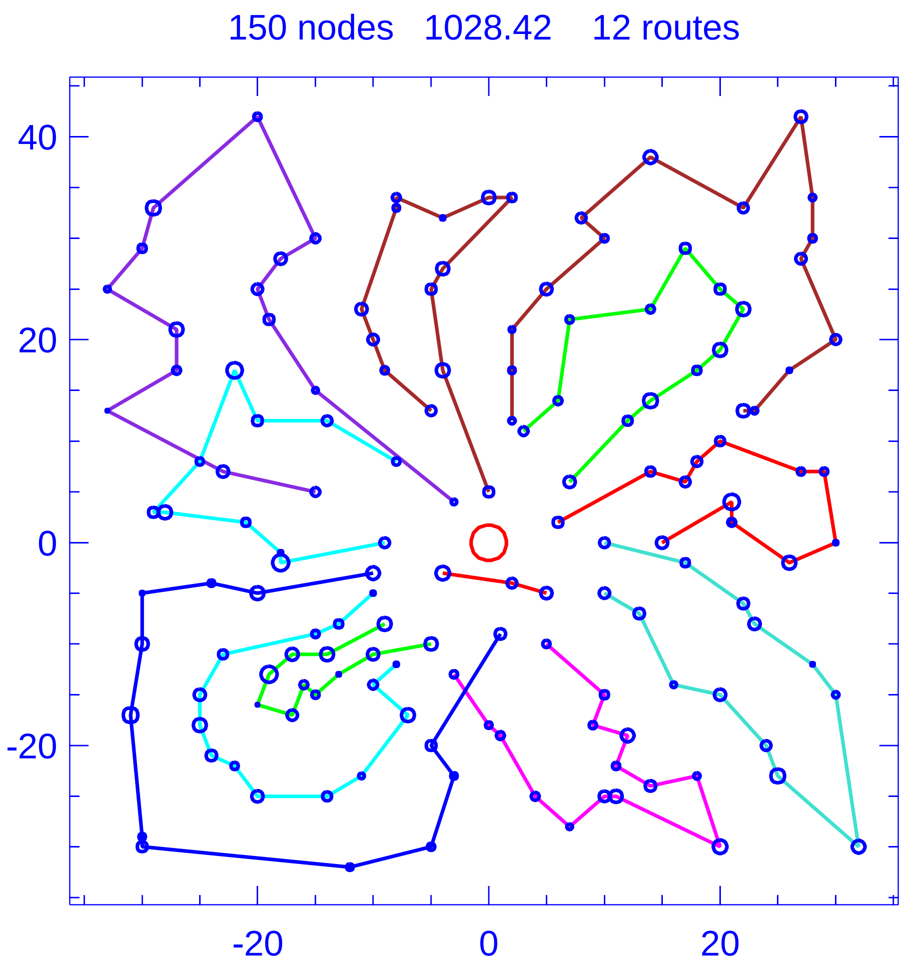


Figure B.4: Solution for problem Christofides-4

Problem			Christofides-4
Vehicle capacity			200
Maximum route length			N/A
Number of nodes			150
Total route length			1028.424
Total number of routes			12
Route	Length	Load	Ordering
1	21.316	64	(0, 5, 103, 12, 0)
2	77.473	199	(0, 11, 100, 2, 83, 131, 20, 59, 3, 101, 22, 51, 32, 0)
3	80.691	197	(0, 17, 147, 137, 92, 42, 64, 19, 94, 41, 66, 111, 4, 149, 146, 56, 0)
4	85.461	196	(0, 27, 138, 48, 112, 26, 113, 140, 82, 31, 8, 60, 81, 77, 0)
5	89.823	197	(0, 37, 52, 15, 45, 91, 72, 33, 73, 106, 125, 124, 122, 123, 71, 90, 0)
6	97.050	200	(0, 38, 9, 104, 30, 105, 75, 117, 89, 39, 54, 10, 49, 76, 0)

Route	Length	Load	Ordering
7	128.707	200	(0, 46, 57, 23, 69, 7, 61, 114, 99, 43, 86, 97, 24, 96, 14, 68, 0)
8	113.819	197	(0, 47, 55, 134, 67, 13, 136, 40, 88, 93, 65, 107, 44, 108, 0)
9	75.759	188	(0, 62, 118, 50, 130, 34, 74, 79, 21, 53, 127, 16, 126, 78, 0)
10	57.315	199	(0, 63, 145, 142, 87, 148, 150, 141, 135, 143, 109, 144, 0)
11	77.333	199	(0, 102, 6, 132, 98, 58, 95, 25, 133, 110, 18, 139, 0)
12	123.676	199	(0, 119, 1, 120, 80, 70, 28, 116, 121, 115, 36, 85, 35, 84, 128, 29, 129, 0)

Table B.9: Details of solution to problem Christofides-4

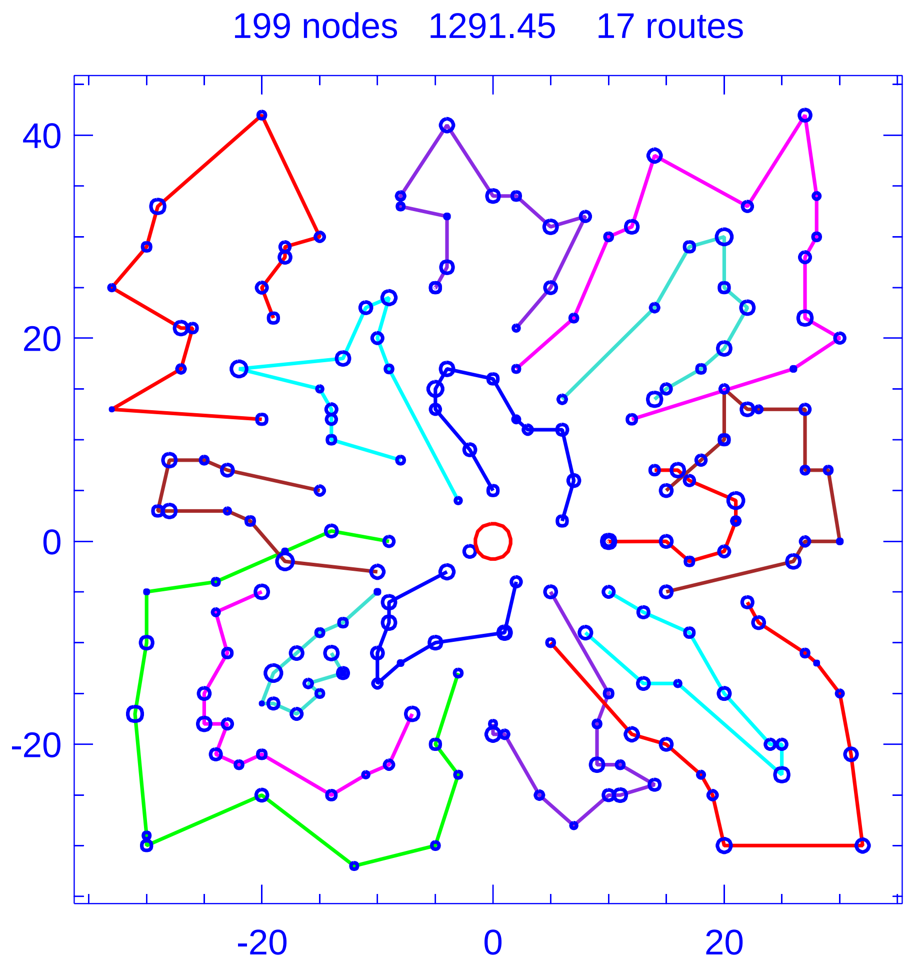


Figure B.5: Solution for problem Christofides-5

Problem			Christofides-5
Vehicle capacity			200
Maximum route length			N/A
Number of nodes			199
Total route length			1291.445
Total number of routes			17
Route	Length	Load	Ordering
1	49.692	199	(0, 4, 87, 111, 58, 27, 179, 99, 167, 65, 46, 175, 0)
2	4.472	19	(0, 6, 0)
3	71.210	194	(0, 30, 48, 59, 138, 37, 88, 103, 5, 29, 98, 125, 0)
4	83.231	195	(0, 34, 176, 102, 8, 35, 178, 78, 19, 70, 128, 123, 13, 83, 45, 0)
5	122.719	200	(0, 50, 71, 119, 38, 165, 170, 164, 85, 134, 84, 14, 133, 177, 149, 0)
6	77.091	199	(0, 51, 7, 132, 180, 69, 108, 11, 52, 150, 100, 0)
7	70.375	197	(0, 54, 120, 172, 21, 173, 174, 82, 121, 140, 94, 64, 28, 101, 0)
8	41.081	199	(0, 61, 33, 193, 194, 196, 66, 112, 157, 2, 152, 0)
9	130.758	200	(0, 72, 118, 56, 25, 110, 163, 148, 92, 135, 146, 18, 73, 145, 181, 0)

Route	Length	Load	Ordering
10	117.073	194	(0, 86, 93, 156, 114, 142, 113, 137, 89, 185, 62, 116, 183, 159, 16, 188, 0)
11	73.434	200	(0, 95, 187, 97, 9, 161, 32, 147, 106, 44, 55, 3, 151, 0)
12	66.131	195	(0, 96, 67, 182, 49, 74, 144, 24, 107, 63, 117, 0)
13	83.140	200	(0, 104, 23, 160, 115, 90, 41, 143, 68, 42, 91, 141, 22, 186, 0)
14	57.160	197	(0, 105, 195, 198, 53, 192, 184, 190, 43, 199, 136, 197, 1, 191, 158, 0)
15	97.095	200	(0, 109, 39, 57, 189, 75, 162, 31, 131, 80, 10, 77, 129, 169, 0)
16	47.264	198	(0, 126, 17, 76, 40, 130, 12, 168, 81, 26, 60, 127, 0)
17	99.520	200	(0, 139, 171, 47, 155, 36, 122, 166, 20, 124, 154, 15, 79, 153, 0)

Table B.10: Details of solution to problem Christofides-5

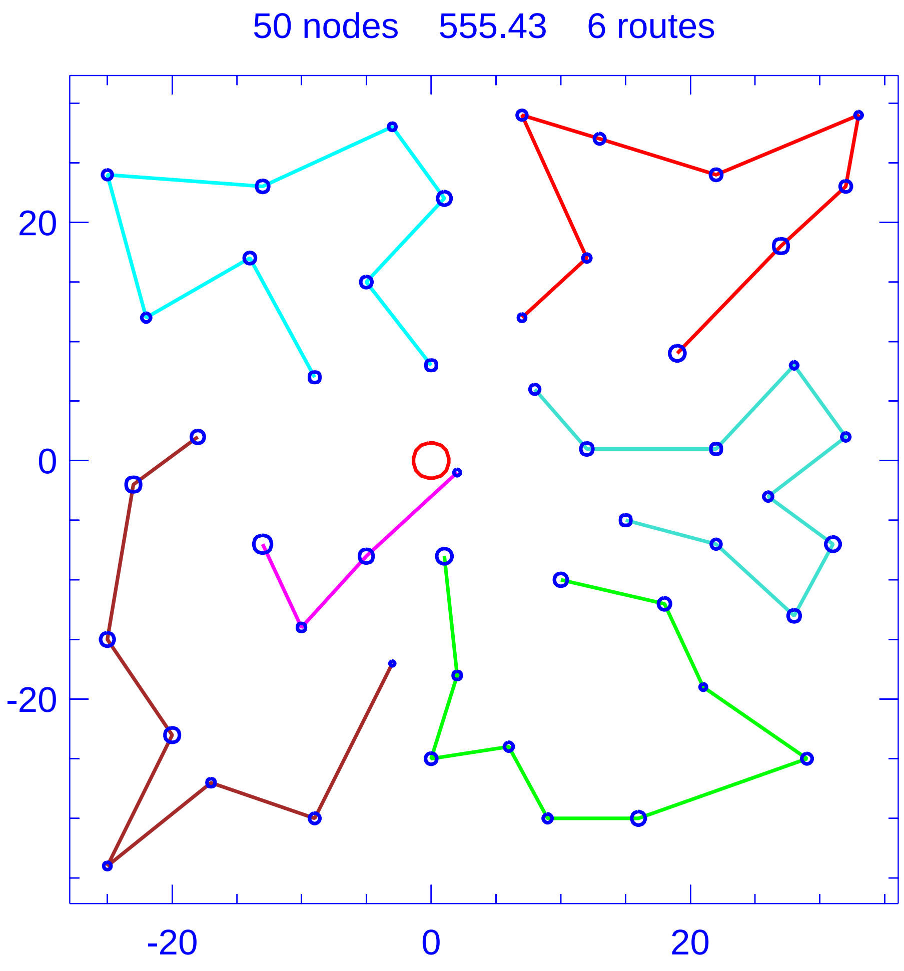


Figure B.6: Solution for problem Christofides-6

Problem			Christofides-6
Vehicle capacity			160
Maximum route length			200.000
Number of nodes			50
Total route length			555.430
Total number of routes			6
Route	Length	Load	Ordering
1	198.077	137	(0, 1, 22, 31, 28, 3, 36, 35, 20, 2, 0)
2	199.116	155	(0, 5, 49, 10, 39, 33, 45, 15, 44, 37, 12, 0)
3	190.640	133	(0, 6, 23, 24, 43, 7, 26, 8, 48, 27, 0)
4	189.939	131	(0, 14, 25, 13, 41, 40, 19, 42, 17, 0)
5	82.326	80	(0, 18, 4, 47, 46, 0)
6	195.332	141	(0, 32, 11, 16, 29, 21, 50, 34, 30, 9, 38, 0)

Table B.11: Details of solution to problem Christofides-6

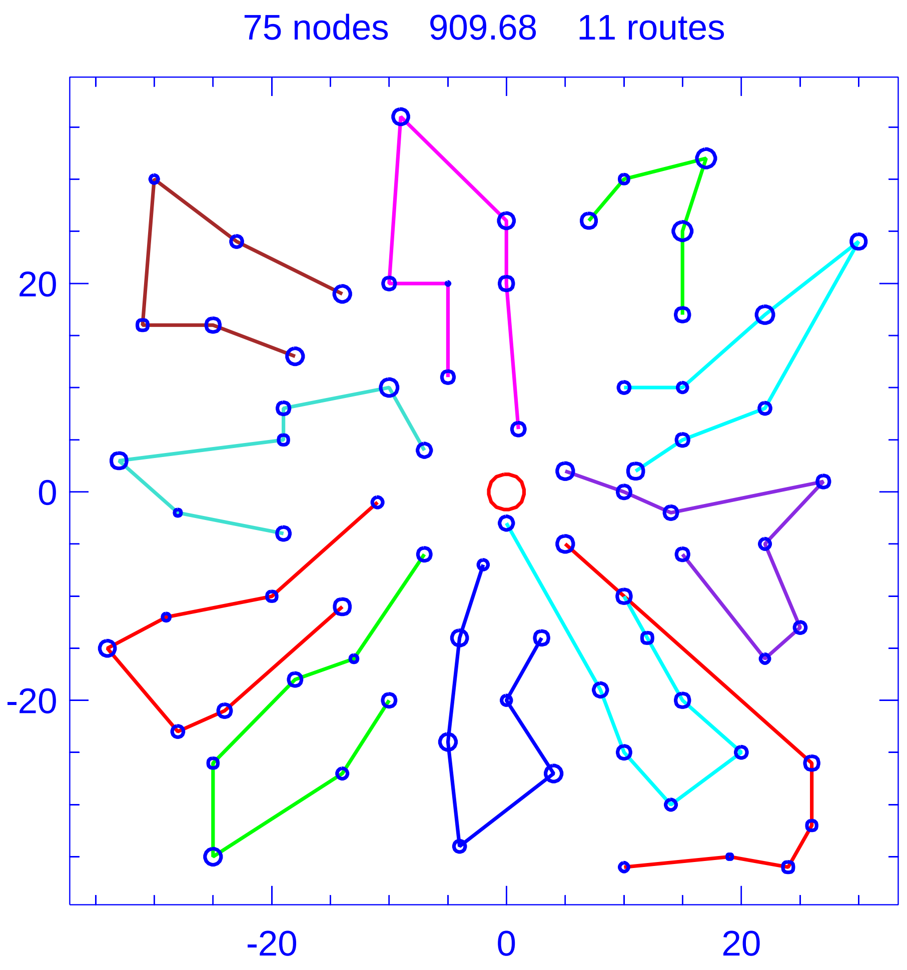


Figure B.7: Solution for problem Christofides-7

Problem			Christofides-7
Vehicle capacity			140
Maximum route length			160.000
Number of nodes			75
Total route length			909.675
Total number of routes			11
Route	Length	Load	Ordering
1	158.759	87	(0, 4, 20, 70, 60, 71, 69, 0)
2	159.921	112	(0, 6, 73, 1, 42, 64, 22, 62, 0)
3	151.356	138	(0, 7, 35, 14, 59, 19, 8, 46, 0)
4	152.971	113	(0, 9, 25, 55, 18, 50, 32, 0)
5	151.686	123	(0, 12, 72, 39, 31, 10, 58, 26, 0)
6	146.839	132	(0, 16, 49, 24, 3, 44, 40, 17, 0)

Route	Length	Load	Ordering
7	157.543	135	(0, 27, 15, 57, 13, 54, 52, 34, 67, 0)
8	144.378	140	(0, 30, 74, 21, 61, 28, 2, 68, 0)
9	155.243	115	(0, 33, 43, 41, 56, 23, 63, 51, 0)
10	127.158	129	(0, 38, 65, 66, 11, 53, 0)
11	153.820	140	(0, 45, 29, 5, 37, 36, 47, 48, 75, 0)

Table B.12: Details of solution to problem Christofides-7

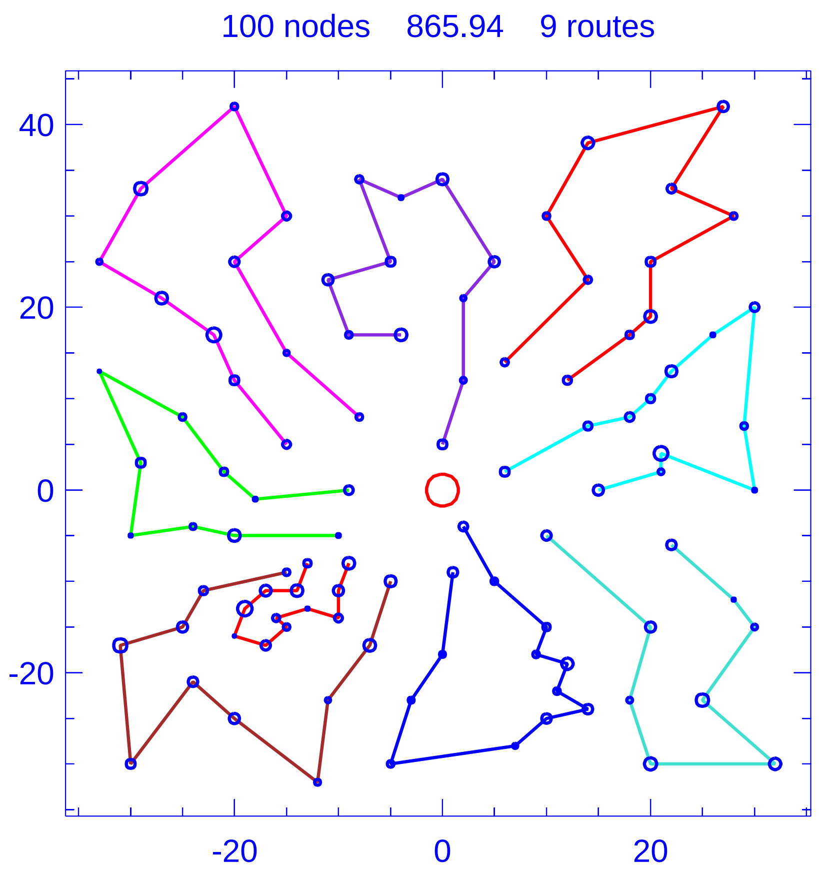


Figure B.8: Solution for problem Christofides-8

Problem			Christofides-8
Vehicle capacity			200
Maximum route length			230.000
Number of nodes			100
Total route length			865.945
Total number of routes			9
Route	Length	Load	Ordering
1	227.928	163	(0, 1, 51, 20, 66, 65, 71, 35, 9, 81, 33, 50, 0)
2	189.162	93	(0, 6, 5, 84, 17, 45, 46, 8, 83, 60, 89, 0)
3	210.255	169	(0, 12, 80, 68, 24, 29, 34, 78, 79, 3, 77, 76, 28, 0)
4	221.405	191	(0, 13, 87, 42, 43, 14, 44, 38, 86, 16, 61, 99, 0)
5	227.550	178	(0, 18, 82, 48, 47, 36, 49, 64, 11, 19, 7, 52, 0)
6	197.077	153	(0, 26, 4, 56, 23, 67, 39, 25, 55, 54, 0)
7	200.120	155	(0, 27, 69, 70, 30, 32, 90, 63, 10, 62, 88, 31, 0)
8	213.097	157	(0, 53, 40, 21, 73, 72, 74, 75, 22, 41, 15, 57, 2, 58, 0)
9	179.351	199	(0, 94, 95, 97, 92, 98, 37, 100, 91, 85, 93, 59, 96, 0)

Table B.13: Details of solution to problem Christofides-8

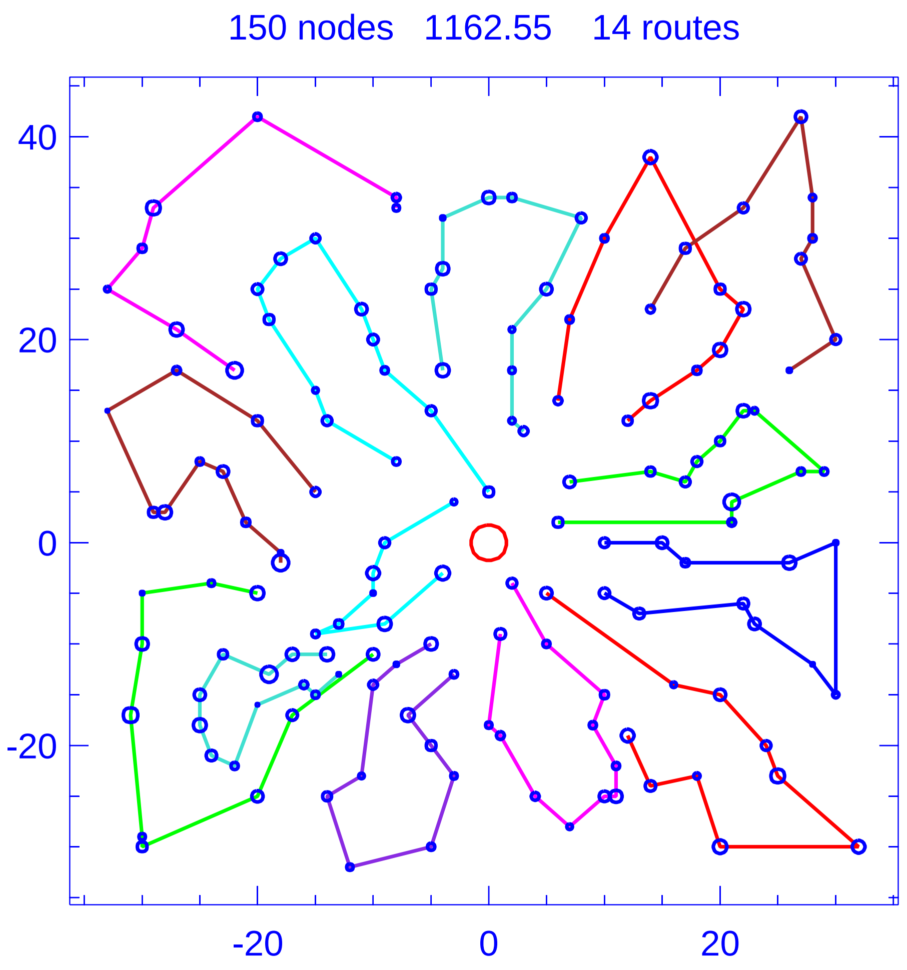


Figure B.9: Solution for problem Christofides-9

Problem			Christofides-9
Vehicle capacity			200
Maximum route length			200.000
Number of nodes			150
Total route length			1162.547
Total number of routes			14
Route	Length	Load	Ordering
1	196.132	193	(0, 5, 10, 54, 39, 89, 117, 73, 106, 125, 122, 0)
2	199.641	188	(0, 11, 126, 16, 127, 53, 129, 29, 79, 21, 118, 130, 50, 78, 0)
3	132.006	133	(0, 12, 144, 149, 4, 146, 56, 47, 139, 46, 0)
4	198.803	168	(0, 18, 110, 133, 14, 58, 25, 95, 96, 24, 132, 68, 0)
5	192.646	135	(0, 26, 113, 114, 99, 43, 86, 97, 98, 0)
6	199.991	168	(0, 32, 119, 1, 120, 80, 28, 31, 82, 140, 8, 60, 81, 0)

Route	Length	Load	Ordering
7	188.325	129	(0, 37, 137, 44, 107, 65, 93, 42, 92, 147, 17, 63, 0)
8	192.387	154	(0, 38, 62, 9, 34, 74, 75, 105, 30, 104, 49, 76, 0)
9	190.028	176	(0, 51, 22, 70, 116, 59, 20, 131, 83, 2, 100, 0)
10	199.744	173	(0, 55, 134, 67, 13, 136, 40, 88, 64, 150, 145, 0)
11	194.600	169	(0, 77, 27, 138, 48, 112, 61, 7, 69, 23, 57, 6, 102, 0)
12	199.240	109	(0, 101, 3, 121, 115, 36, 85, 35, 84, 128, 0)
13	184.604	142	(0, 103, 90, 71, 123, 124, 33, 72, 91, 45, 15, 52, 108, 0)
14	194.400	198	(0, 109, 143, 135, 111, 66, 41, 94, 19, 141, 148, 87, 142, 0)

Table B.14: Details of solution to problem Christofides-9

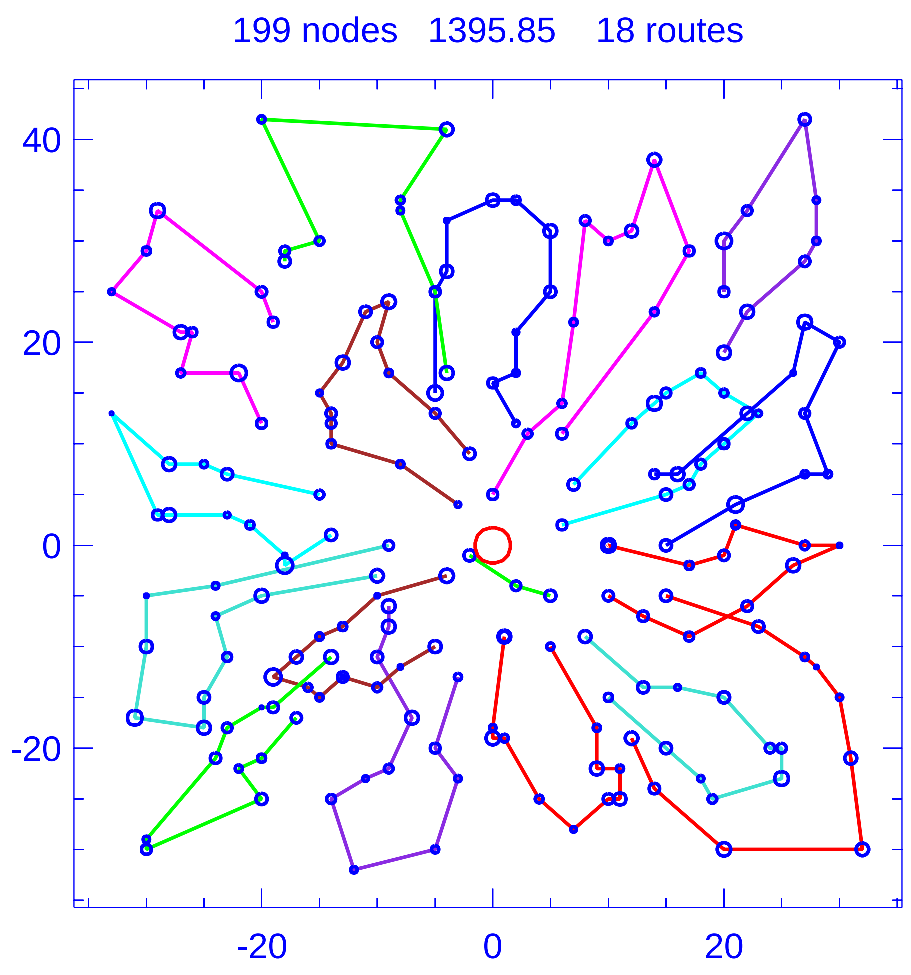


Figure B.10: Solution for problem Christofides-10

Problem			Christofides-10
Vehicle capacity			200
Maximum route length			200.000
Number of nodes			199
Total route length			1395.853
Total number of routes			18
Route	Length	Load	Ordering
1	198.147	196	(0, 4, 87, 58, 27, 99, 179, 13, 123, 83, 153, 29, 98, 125, 0)
2	47.469	54	(0, 6, 152, 54, 0)
3	199.649	193	(0, 16, 67, 159, 182, 49, 74, 144, 145, 24, 107, 63, 117, 0)
4	191.352	199	(0, 17, 76, 187, 97, 9, 161, 32, 106, 44, 55, 3, 151, 95, 0)
5	199.555	162	(0, 26, 150, 52, 165, 38, 119, 77, 71, 100, 81, 126, 0)
6	194.541	194	(0, 30, 48, 59, 103, 5, 88, 37, 138, 36, 155, 47, 120, 0)
7	199.982	177	(0, 33, 193, 194, 186, 22, 141, 91, 142, 114, 156, 93, 86, 0)
8	197.636	190	(0, 40, 109, 57, 189, 131, 80, 10, 129, 169, 50, 12, 168, 0)
9	197.856	175	(0, 45, 79, 15, 154, 124, 20, 166, 122, 174, 171, 0)

Route	Length	Load	Ordering
10	199.613	139	(0, 56, 25, 110, 163, 31, 162, 75, 39, 130, 0)
11	188.667	197	(0, 60, 149, 51, 7, 132, 35, 78, 102, 8, 176, 65, 34, 127, 0)
12	191.642	200	(0, 61, 105, 195, 53, 198, 192, 184, 197, 136, 1, 191, 196, 66, 112, 0)
13	199.439	181	(0, 72, 118, 148, 92, 135, 146, 18, 73, 147, 181, 0)
14	195.466	198	(0, 96, 104, 23, 160, 115, 90, 185, 62, 116, 183, 188, 0)
15	194.345	173	(0, 108, 11, 170, 164, 85, 134, 84, 69, 180, 0)
16	195.359	198	(0, 111, 167, 70, 128, 19, 133, 14, 177, 178, 46, 175, 0)
17	195.429	200	(0, 139, 172, 21, 173, 82, 121, 140, 94, 64, 28, 101, 2, 157, 0)
18	199.707	160	(0, 158, 43, 190, 41, 143, 89, 137, 113, 68, 42, 199, 0)

Table B.15: Details of solution to problem Christofides-10

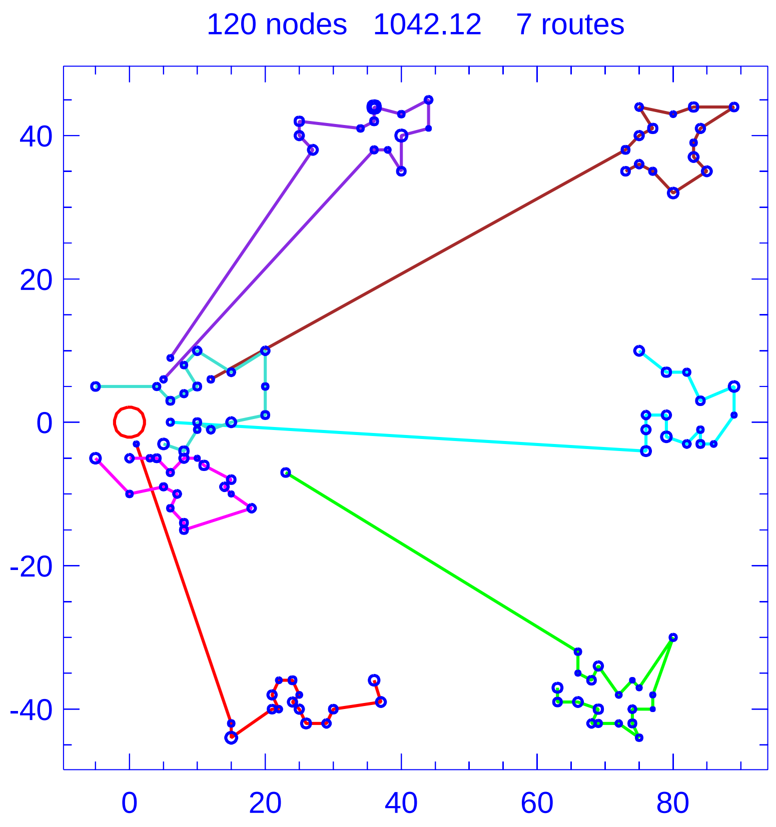


Figure B.11: Solution for problem Christofides-11

Problem			Christofides-11
Vehicle capacity			200
Maximum route length			N/A
Number of nodes			120
Total route length			1042.115
Total number of routes			7
Route	Length	Load	Ordering
1	134.964	199	(0, 8, 12, 13, 14, 15, 11, 10, 9, 7, 6, 5, 4, 3, 1, 2, 88, 0)
2	207.944	197	(0, 17, 16, 19, 25, 22, 24, 27, 33, 30, 31, 34, 36, 29, 35, 32, 28, 26, 23, 20, 21, 109, 0)
3	199.626	200	(0, 40, 43, 45, 48, 51, 50, 49, 46, 47, 44, 41, 42, 39, 38, 37, 95, 0)
4	213.630	199	(0, 52, 54, 57, 59, 65, 61, 62, 64, 66, 63, 60, 56, 58, 55, 53, 100, 0)
5	66.959	188	(0, 82, 111, 86, 85, 89, 91, 90, 114, 18, 118, 108, 83, 113, 117, 84, 112, 81, 119, 0)
6	74.564	193	(0, 87, 92, 93, 96, 94, 97, 115, 110, 98, 116, 103, 104, 99, 101, 102, 105, 120, 0)
7	144.429	199	(0, 106, 73, 76, 68, 77, 79, 80, 78, 75, 72, 74, 71, 70, 69, 67, 107, 0)

Table B.16: Details of solution to problem Christofides-11

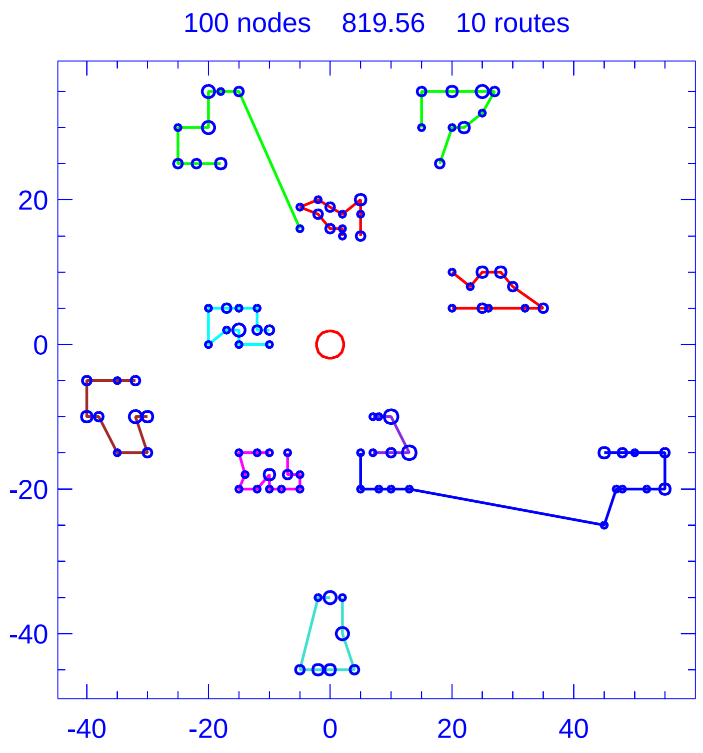


Figure B.12: Solution for problem Christofides-12

Problem			Christofides-12
Vehicle capacity			200
Maximum route length			N/A
Number of nodes			100
Total route length			819.558
Total number of routes			10
Route	Length	Load	Ordering
1	56.175	170	(0, 5, 3, 7, 8, 11, 9, 6, 4, 2, 1, 75, 0)
2	96.040	200	(0, 10, 12, 14, 16, 15, 19, 18, 17, 13, 0)
3	50.804	170	(0, 20, 24, 25, 27, 29, 30, 28, 26, 23, 22, 21, 0)
4	97.227	200	(0, 32, 33, 31, 35, 37, 38, 39, 36, 34, 0)
5	64.807	160	(0, 43, 42, 41, 40, 44, 45, 46, 48, 51, 50, 52, 49, 47, 0)
6	101.883	200	(0, 55, 54, 53, 56, 58, 60, 59, 57, 0)
7	43.590	150	(0, 66, 62, 74, 63, 65, 67, 0)
8	137.019	200	(0, 69, 68, 64, 61, 72, 80, 79, 77, 73, 70, 71, 76, 78, 81, 0)
9	76.070	170	(0, 90, 87, 86, 83, 82, 84, 85, 88, 89, 91, 0)
10	95.943	190	(0, 98, 96, 95, 94, 92, 93, 97, 100, 99, 0)

Table B.17: Details of solution to problem Christofides-12

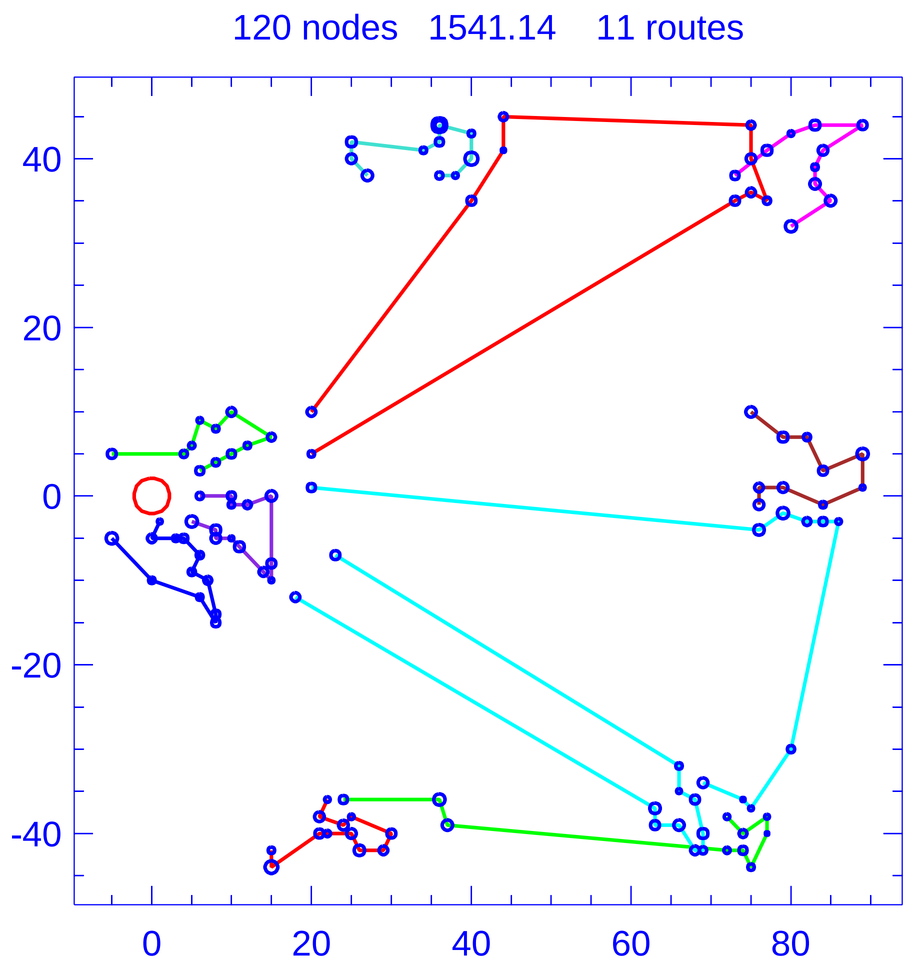


Figure B.13: Solution for problem Christofides-13

Problem			Christofides-13
Vehicle capacity			200
Maximum route length			720.000
Number of nodes			120
Total route length			1541.142
Total number of routes			11
Route	Length	Load	Ordering
1	719.669	148	(0, 2, 1, 3, 4, 11, 15, 14, 13, 9, 10, 5, 6, 0)
2	692.112	92	(0, 7, 8, 12, 27, 30, 33, 34, 36, 31, 28, 0)
3	706.998	106	(0, 26, 32, 35, 29, 49, 46, 44, 41, 37, 115, 0)
4	689.298	127	(0, 38, 39, 42, 47, 50, 51, 48, 45, 43, 40, 0)
5	704.955	135	(0, 53, 58, 60, 63, 66, 64, 62, 61, 65, 59, 0)
6	679.032	162	(0, 67, 69, 70, 71, 74, 75, 72, 78, 77, 76, 73, 0)

Route	Length	Load	Ordering
7	695.227	155	(0, 87, 92, 89, 91, 90, 18, 118, 114, 97, 94, 93, 96, 95, 0)
8	645.894	116	(0, 88, 82, 111, 86, 85, 112, 84, 113, 83, 117, 81, 119, 0)
9	691.978	102	(0, 98, 68, 79, 80, 56, 55, 57, 54, 52, 110, 0)
10	598.122	99	(0, 102, 101, 99, 100, 116, 103, 104, 107, 106, 105, 120, 0)
11	717.857	133	(0, 108, 17, 16, 19, 22, 24, 25, 23, 20, 21, 109, 0)

Table B.18: Details of solution to problem Christofides-13

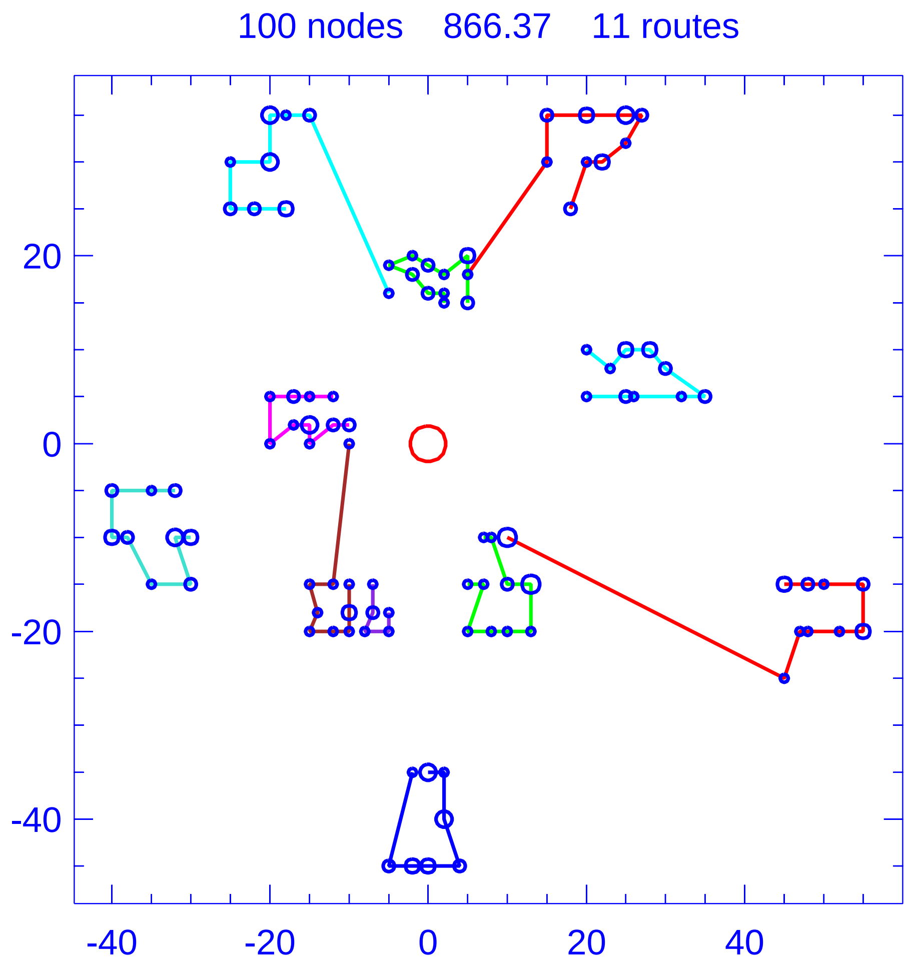


Figure B.14: Solution for problem Christofides-14

Problem			Christofides-14
Vehicle capacity			200
Maximum route length			1040.000
Number of nodes			100
Total route length			866.365
Total number of routes			11
Route	Length	Load	Ordering
1	996.704	200	(0, 1, 99, 100, 97, 93, 92, 94, 95, 96, 98, 0)
2	956.175	160	(0, 5, 3, 7, 8, 11, 9, 6, 4, 2, 75, 0)
3	906.040	200	(0, 10, 12, 14, 16, 15, 19, 18, 17, 13, 0)
4	871.559	110	(0, 20, 49, 52, 50, 51, 48, 45, 46, 47, 0)
5	949.409	160	(0, 21, 22, 24, 25, 27, 29, 30, 28, 26, 23, 0)
6	907.227	200	(0, 32, 33, 31, 35, 37, 38, 39, 36, 34, 0)
7	495.471	60	(0, 41, 40, 44, 42, 43, 0)
8	821.883	200	(0, 57, 55, 54, 53, 56, 58, 60, 59, 0)
9	1028.040	200	(0, 63, 80, 79, 77, 73, 70, 71, 76, 78, 81, 0)

Route	Length	Load	Ordering
10	957.788	150	(0, 67, 65, 62, 74, 72, 61, 64, 68, 66, 69, 0)
11	976.070	170	(0, 90, 87, 86, 83, 82, 84, 85, 88, 89, 91, 0)

Table B.19: Details of solution to problem Christofides-14

B.4. Golden Benchmarks

This section contains the best solutions we found for the 20 problem instances of Golden et al. [36].

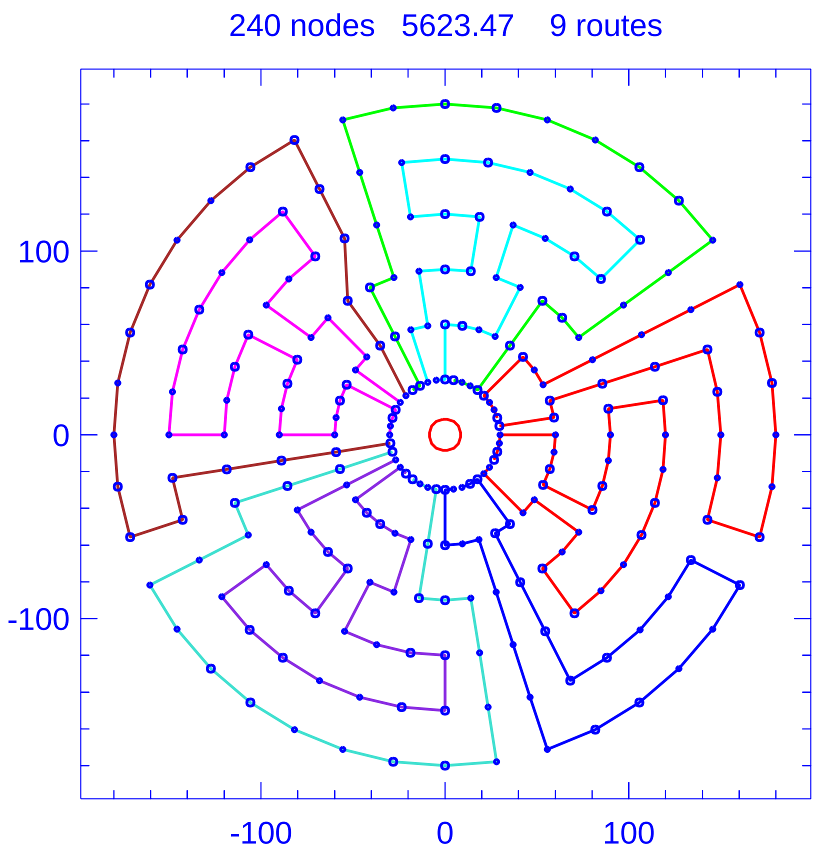


Figure B.15: Solution for problem Golden-1

Problem			Golden-1
Vehicle capacity			550
Maximum route length			650.000
Number of nodes			240
Total route length			5623.468
Total number of routes			9
Route	Length	Load	Ordering
1	637.745	540	(0, 2, 42, 43, 83, 123, 163, 162, 161, 200, 199, 239, 240, 201, 202, 203, 204, 164, 124, 84, 44, 45, 46, 6, 5, 4, 3, 0)
2	647.160	520	(0, 10, 9, 8, 7, 47, 87, 86, 85, 125, 165, 205, 206, 207, 208, 209, 210, 211, 212, 213, 173, 133, 93, 94, 54, 14, 15, 0)
3	620.113	540	(0, 12, 11, 51, 50, 49, 48, 88, 89, 129, 128, 127, 126, 166, 167, 168, 169, 170, 171, 172, 132, 131, 130, 90, 91, 92, 52, 53, 13, 0)
4	642.621	530	(0, 16, 55, 95, 134, 174, 214, 215, 216, 217, 218, 219, 220, 221, 222, 223, 183, 182, 142, 102, 62, 22, 0)
5	643.651	550	(0, 17, 57, 56, 96, 97, 137, 136, 135, 175, 176, 177, 178, 179, 180, 181, 141, 140, 139, 138, 98, 99, 100, 101, 61, 60, 59, 58, 18, 19, 20, 21, 0)
6	647.160	510	(0, 23, 63, 103, 143, 144, 184, 224, 225, 226, 227, 228, 229, 230, 231, 232, 192, 152, 112, 111, 110, 70, 30, 29, 28, 27, 0)

Route	Length	Load	Ordering
7	634.235	550	(0, 24, 64, 104, 105, 106, 107, 147, 146, 145, 185, 186, 187, 188, 189, 190, 191, 151, 150, 149, 148, 108, 109, 69, 68, 67, 66, 65, 25, 26, 0)
8	637.745	540	(0, 32, 31, 71, 72, 73, 113, 153, 193, 233, 234, 235, 236, 237, 238, 198, 197, 196, 195, 194, 154, 114, 74, 75, 35, 34, 33, 0)
9	513.037	520	(0, 37, 36, 76, 77, 117, 116, 115, 155, 156, 157, 158, 159, 160, 121, 122, 82, 81, 120, 119, 118, 78, 79, 80, 41, 1, 40, 39, 38, 0)

Table B.20: Details of solution to problem Golden-1

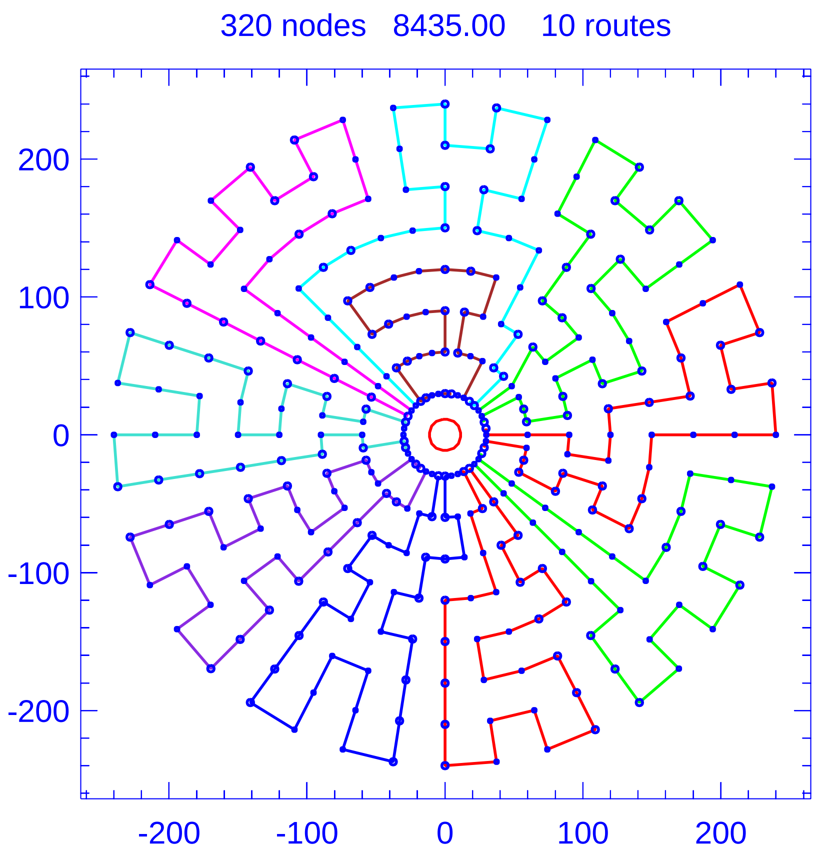


Figure B.16: Solution for problem Golden-2

Problem			Golden-2
Vehicle capacity			700
Maximum route length			900.000
Number of nodes			320
Total route length			8434.997
Total number of routes			10
Route	Length	Load	Ordering
1	888.358	700	(0, 2, 1, 41, 81, 120, 160, 121, 122, 162, 202, 203, 204, 244, 284, 283, 243, 242, 282, 281, 241, 201, 161, 200, 199, 198, 158, 159, 119, 118, 78, 79, 80, 40, 39, 0)
2	899.913	700	(0, 3, 4, 44, 43, 42, 82, 83, 84, 124, 123, 163, 164, 165, 166, 206, 205, 245, 285, 286, 246, 247, 287, 288, 248, 208, 207, 167, 127, 126, 125, 85, 86, 45, 5, 0)
3	897.773	620	(0, 7, 6, 46, 47, 87, 88, 128, 168, 169, 170, 210, 209, 249, 289, 290, 250, 251, 291, 292, 252, 212, 211, 171, 172, 173, 174, 175, 176, 136, 96, 56, 16, 0)
4	508.330	620	(0, 9, 8, 48, 49, 50, 90, 89, 129, 130, 131, 132, 133, 134, 135, 95, 94, 93, 92, 91, 51, 52, 53, 54, 55, 15, 14, 13, 12, 11, 10, 0)
5	891.868	560	(0, 17, 57, 97, 137, 177, 217, 216, 215, 214, 213, 253, 293, 294, 254, 255, 295, 296, 256, 257, 297, 298, 258, 218, 178, 138, 98, 58, 18, 0)

Route	Length	Load	Ordering
6	854.207	680	(0, 21, 20, 19, 59, 60, 100, 99, 139, 140, 141, 181, 180, 179, 219, 259, 299, 300, 260, 220, 221, 261, 301, 302, 262, 222, 182, 142, 102, 101, 61, 62, 22, 23, 24, 0)
7	869.528	690	(0, 27, 26, 25, 65, 64, 63, 103, 104, 105, 145, 144, 143, 183, 184, 224, 223, 263, 303, 304, 264, 265, 305, 306, 266, 226, 225, 185, 186, 146, 106, 66, 67, 68, 28, 29, 0)
8	868.330	670	(0, 30, 70, 69, 109, 108, 107, 147, 148, 188, 187, 227, 267, 307, 308, 268, 228, 229, 269, 309, 310, 270, 230, 190, 189, 149, 150, 110, 111, 112, 72, 71, 31, 32, 0)
9	860.113	670	(0, 33, 34, 74, 73, 113, 153, 152, 151, 191, 231, 271, 311, 312, 272, 273, 313, 314, 274, 234, 233, 232, 192, 193, 194, 195, 155, 154, 114, 115, 75, 35, 0)
10	896.575	490	(0, 36, 76, 116, 156, 196, 236, 235, 275, 315, 316, 276, 277, 317, 318, 278, 279, 319, 320, 280, 240, 239, 238, 237, 197, 157, 117, 77, 37, 38, 0)

Table B.21: Details of solution to problem Golden-2

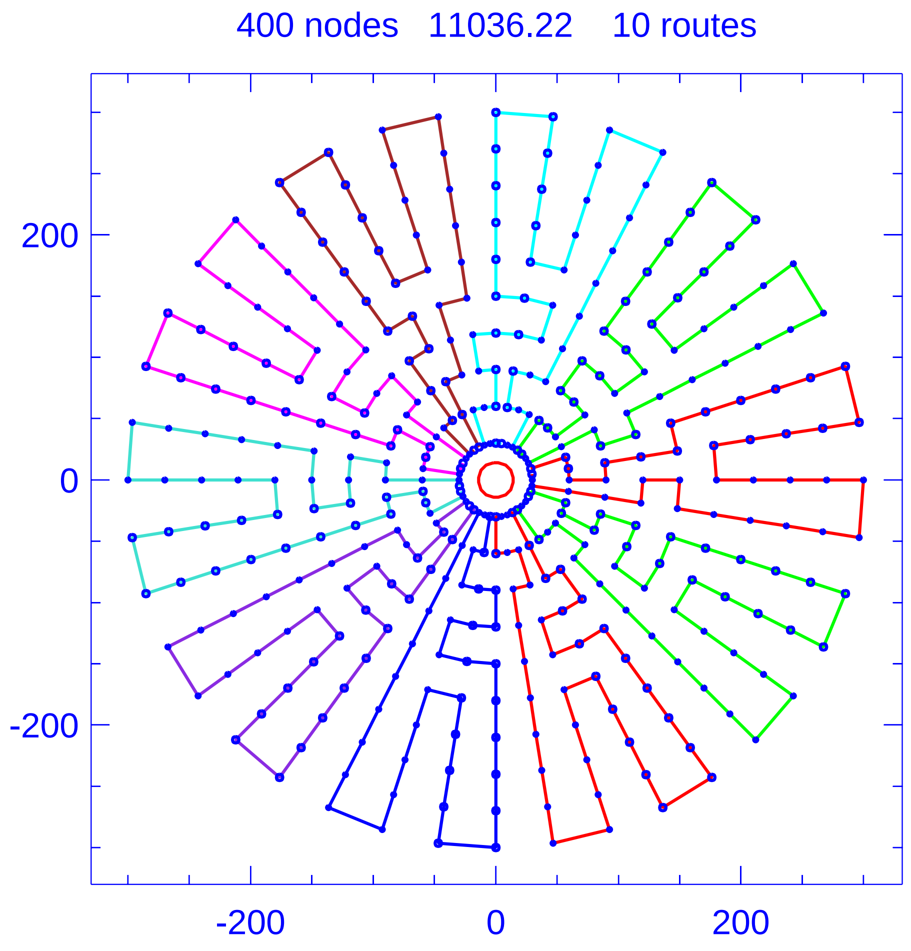


Figure B.17: Solution for problem Golden-3

Problem			Golden-3
Vehicle capacity			900
Maximum route length			1200.000
Number of nodes			400
Total route length			11036.222
Total number of routes			10
Route	Length	Load	Ordering
1	1070.669	740	(0, 2, 1, 40, 80, 120, 160, 121, 161, 200, 240, 280, 320, 360, 400, 361, 321, 281, 241, 201, 202, 242, 282, 322, 362, 363, 323, 283, 243, 203, 163, 162, 122, 82, 81, 41, 42, 43, 3, 0)
2	1136.575	860	(0, 4, 44, 84, 83, 123, 124, 164, 204, 244, 284, 324, 364, 365, 325, 285, 245, 205, 206, 246, 286, 326, 366, 367, 327, 287, 247, 207, 167, 166, 165, 125, 126, 127, 87, 86, 85, 45, 46, 47, 7, 6, 5, 0)
3	1164.820	860	(0, 9, 8, 48, 49, 50, 90, 89, 88, 128, 168, 208, 248, 288, 328, 368, 369, 329, 289, 249, 209, 210, 250, 290, 330, 370, 371, 331, 291, 251, 211, 171, 170, 169, 129, 130, 131, 132, 92, 91, 51, 52, 53, 13, 12, 11, 10, 0)

Route	Length	Load	Ordering
4	1056.547	760	(0, 14, 54, 94, 93, 133, 173, 172, 212, 252, 292, 332, 372, 373, 333, 293, 253, 213, 214, 254, 294, 334, 374, 375, 335, 295, 255, 215, 175, 174, 134, 135, 95, 55, 56, 16, 15, 0)
5	1103.622	800	(0, 17, 57, 97, 96, 136, 137, 138, 178, 177, 176, 216, 256, 296, 336, 376, 377, 337, 297, 257, 217, 218, 258, 298, 338, 378, 379, 339, 299, 259, 219, 179, 139, 99, 98, 58, 59, 60, 20, 19, 18, 0)
6	1103.622	800	(0, 22, 21, 61, 101, 100, 140, 141, 142, 182, 181, 180, 220, 260, 300, 340, 380, 381, 341, 301, 261, 221, 222, 262, 302, 342, 382, 383, 343, 303, 263, 223, 183, 143, 103, 102, 62, 63, 64, 24, 23, 0)
7	1089.500	780	(0, 25, 65, 66, 106, 105, 104, 144, 184, 224, 264, 304, 344, 384, 385, 345, 305, 265, 225, 226, 266, 306, 346, 386, 387, 347, 307, 267, 227, 187, 186, 185, 145, 146, 147, 107, 67, 27, 26, 0)
8	1089.500	740	(0, 28, 68, 108, 148, 188, 228, 268, 308, 348, 388, 389, 349, 309, 269, 229, 230, 270, 310, 350, 390, 391, 351, 311, 271, 231, 191, 190, 189, 149, 150, 151, 111, 110, 109, 69, 70, 30, 29, 0)
9	1103.622	800	(0, 32, 31, 71, 72, 73, 113, 112, 152, 192, 232, 272, 312, 352, 392, 393, 353, 313, 273, 233, 234, 274, 314, 354, 394, 395, 355, 315, 275, 235, 195, 194, 193, 153, 154, 155, 115, 114, 74, 34, 33, 0)
10	1117.745	860	(0, 37, 36, 35, 75, 76, 77, 117, 116, 156, 196, 236, 276, 316, 356, 396, 397, 357, 317, 277, 237, 238, 278, 318, 358, 398, 399, 359, 319, 279, 239, 199, 198, 197, 157, 158, 159, 119, 118, 78, 79, 39, 38, 0)

Table B.22: Details of solution to problem Golden-3

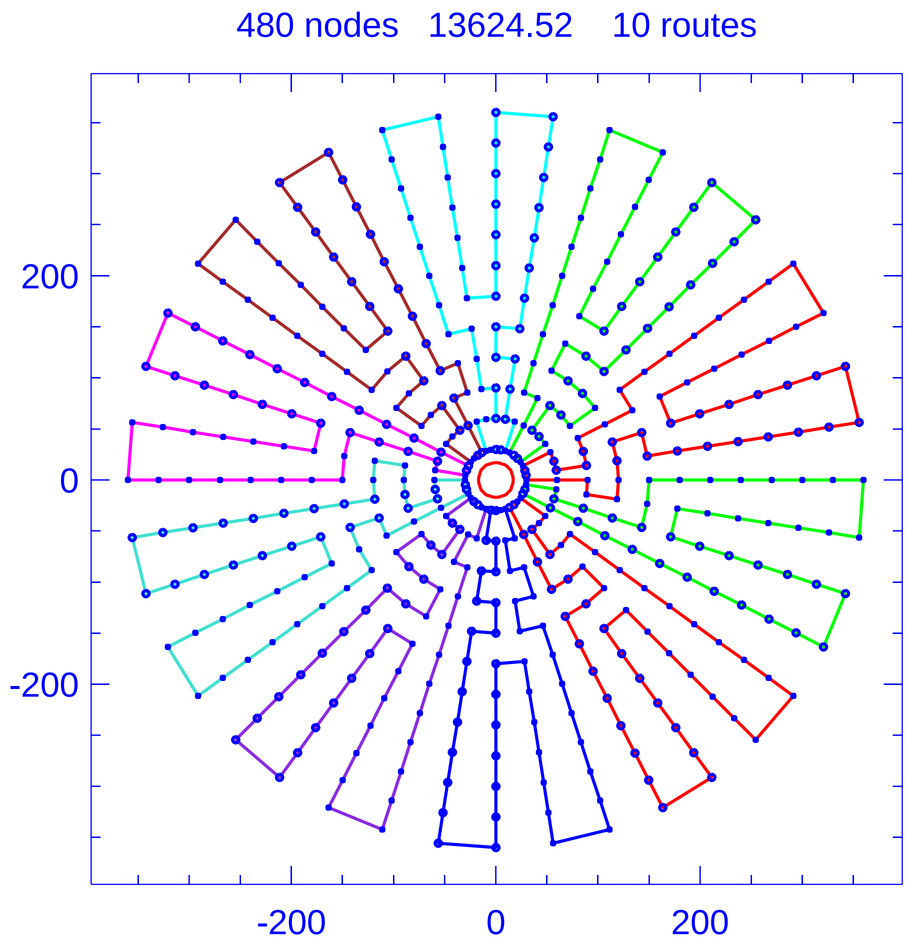


Figure B.18: Solution for problem Golden-4

Problem			Golden-4
Vehicle capacity			1000
Maximum route length			1600.000
Number of nodes			480
Total route length			13624.524
Total number of routes			10
Route	Length	Load	Ordering
1	1395.405	980	(0, 2, 1, 41, 81, 120, 160, 121, 122, 123, 163, 162, 202, 242, 282, 322, 362, 402, 442, 443, 403, 363, 323, 283, 243, 203, 204, 244, 284, 324, 364, 404, 444, 445, 405, 365, 325, 285, 245, 205, 165, 164, 124, 84, 83, 82, 42, 43, 44, 4, 3, 0)
2	1395.405	980	(0, 7, 6, 5, 45, 46, 47, 87, 86, 85, 125, 126, 127, 128, 168, 167, 166, 206, 246, 286, 326, 366, 406, 446, 447, 407, 367, 327, 287, 247, 207, 208, 248, 288, 328, 368, 408, 448, 449, 409, 369, 329, 289, 249, 209, 169, 129, 89, 88, 48, 8, 0)

Route	Length	Load	Ordering
3	1343.622	960	(0, 12, 11, 10, 9, 49, 50, 90, 130, 131, 171, 170, 210, 250, 290, 330, 370, 410, 450, 451, 411, 371, 331, 291, 251, 211, 212, 252, 292, 332, 372, 412, 452, 453, 413, 373, 333, 293, 253, 213, 173, 172, 132, 92, 91, 51, 52, 53, 13, 0)
4	1395.405	980	(0, 14, 54, 94, 93, 133, 134, 174, 214, 254, 294, 334, 374, 414, 454, 455, 415, 375, 335, 295, 255, 215, 216, 256, 296, 336, 376, 416, 456, 457, 417, 377, 337, 297, 257, 217, 177, 176, 175, 135, 136, 137, 97, 96, 95, 55, 56, 57, 17, 16, 15, 0)
5	1282.424	900	(0, 18, 58, 98, 138, 178, 218, 258, 298, 338, 378, 418, 458, 459, 419, 379, 339, 299, 259, 219, 220, 260, 300, 340, 380, 420, 460, 461, 421, 381, 341, 301, 261, 221, 181, 180, 179, 139, 99, 59, 60, 20, 19, 0)
6	1395.405	980	(0, 22, 21, 61, 62, 63, 103, 102, 101, 100, 140, 141, 142, 182, 222, 262, 302, 342, 382, 422, 462, 463, 423, 383, 343, 303, 263, 223, 224, 264, 304, 344, 384, 424, 464, 465, 425, 385, 345, 305, 265, 225, 185, 184, 183, 143, 144, 104, 64, 24, 23, 0)
7	1409.528	1000	(0, 27, 26, 25, 65, 66, 67, 107, 106, 105, 145, 146, 147, 148, 188, 187, 186, 226, 266, 306, 346, 386, 426, 466, 467, 427, 387, 347, 307, 267, 227, 228, 268, 308, 348, 388, 428, 468, 469, 429, 389, 349, 309, 269, 229, 189, 149, 109, 108, 68, 69, 29, 28, 0)
8	1362.452	960	(0, 32, 31, 30, 70, 71, 111, 110, 150, 151, 191, 190, 230, 270, 310, 350, 390, 430, 470, 471, 431, 391, 351, 311, 271, 231, 232, 272, 312, 352, 392, 432, 472, 473, 433, 393, 353, 313, 273, 233, 193, 192, 152, 153, 113, 112, 72, 73, 33, 0)
9	1362.452	960	(0, 34, 74, 114, 154, 155, 156, 196, 195, 194, 234, 274, 314, 354, 394, 434, 474, 475, 435, 395, 355, 315, 275, 235, 236, 276, 316, 356, 396, 436, 476, 477, 437, 397, 357, 317, 277, 237, 197, 157, 117, 116, 115, 75, 76, 77, 37, 36, 35, 0)
10	1282.424	900	(0, 38, 78, 118, 158, 198, 238, 278, 318, 358, 398, 438, 478, 479, 439, 399, 359, 319, 279, 239, 240, 280, 320, 360, 400, 440, 480, 441, 401, 361, 321, 281, 241, 201, 161, 200, 199, 159, 119, 79, 80, 40, 39, 0)

Table B.23: Details of solution to problem Golden-4

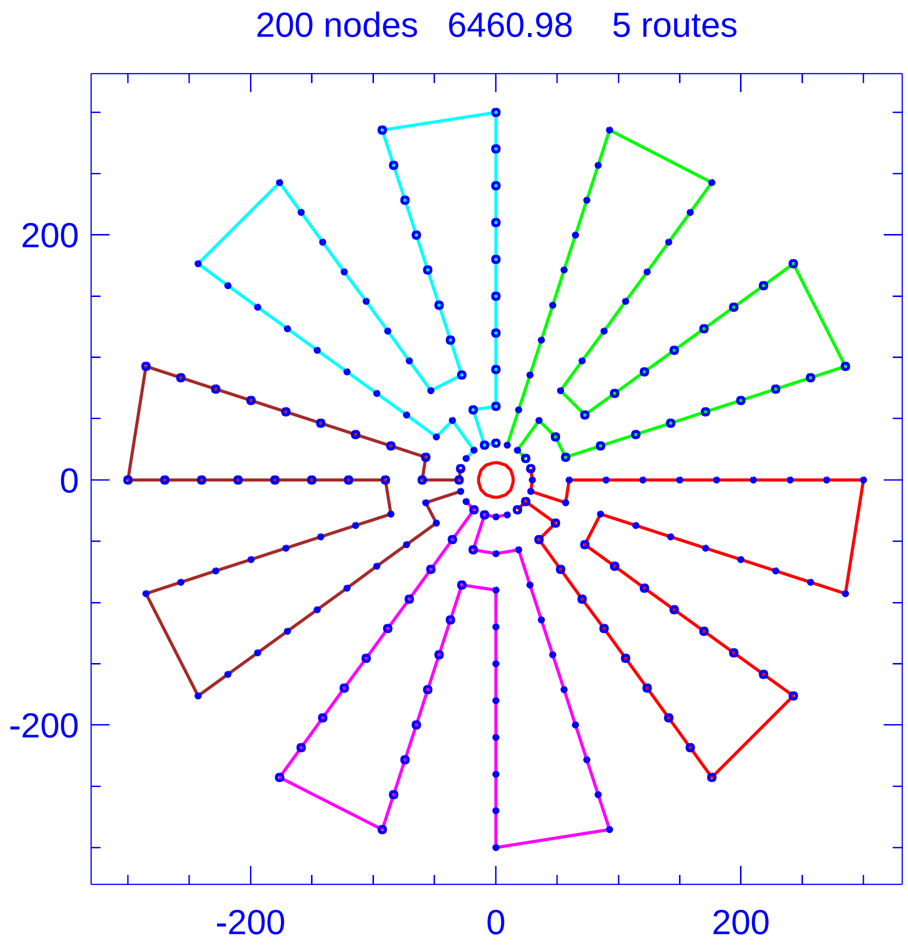


Figure B.19: Solution for problem Golden-5

Problem			Golden-5
Vehicle capacity Maximum route length Number of nodes Total route length Total number of routes			900 1800.000 200 6460.980 5
Route	Length	Load	Ordering
1	1301.582	830	(0, 2, 1, 20, 40, 21, 41, 61, 81, 101, 121, 141, 161, 181, 200, 180, 160, 140, 120, 100, 80, 60, 59, 79, 99, 119, 139, 159, 179, 199, 198, 178, 158, 138, 118, 98, 78, 58, 38, 39, 19, 18, 0)
2	1282.810	770	(0, 3, 4, 24, 23, 22, 42, 62, 82, 102, 122, 142, 162, 182, 183, 163, 143, 123, 103, 83, 63, 43, 44, 64, 84, 104, 124, 144, 164, 184, 185, 165, 145, 125, 105, 85, 65, 45, 25, 5, 0)
3	1292.196	800	(0, 6, 7, 27, 26, 46, 66, 86, 106, 126, 146, 166, 186, 187, 167, 147, 127, 107, 87, 67, 47, 48, 68, 88, 108, 128, 148, 168, 188, 189, 169, 149, 129, 109, 89, 69, 49, 29, 28, 8, 9, 0)

Route	Length	Load	Ordering
4	1282.810	790	(0, 10, 11, 31, 30, 50, 70, 90, 110, 130, 150, 170, 190, 191, 171, 151, 131, 111, 91, 71, 51, 52, 72, 92, 112, 132, 152, 172, 192, 193, 173, 153, 133, 113, 93, 73, 53, 33, 32, 12, 0)
5	1301.582	810	(0, 13, 14, 34, 54, 74, 94, 114, 134, 154, 174, 194, 195, 175, 155, 135, 115, 95, 75, 55, 56, 76, 96, 116, 136, 156, 176, 196, 197, 177, 157, 137, 117, 97, 77, 57, 37, 36, 35, 15, 16, 17, 0)

Table B.24: Details of solution to problem Golden-5

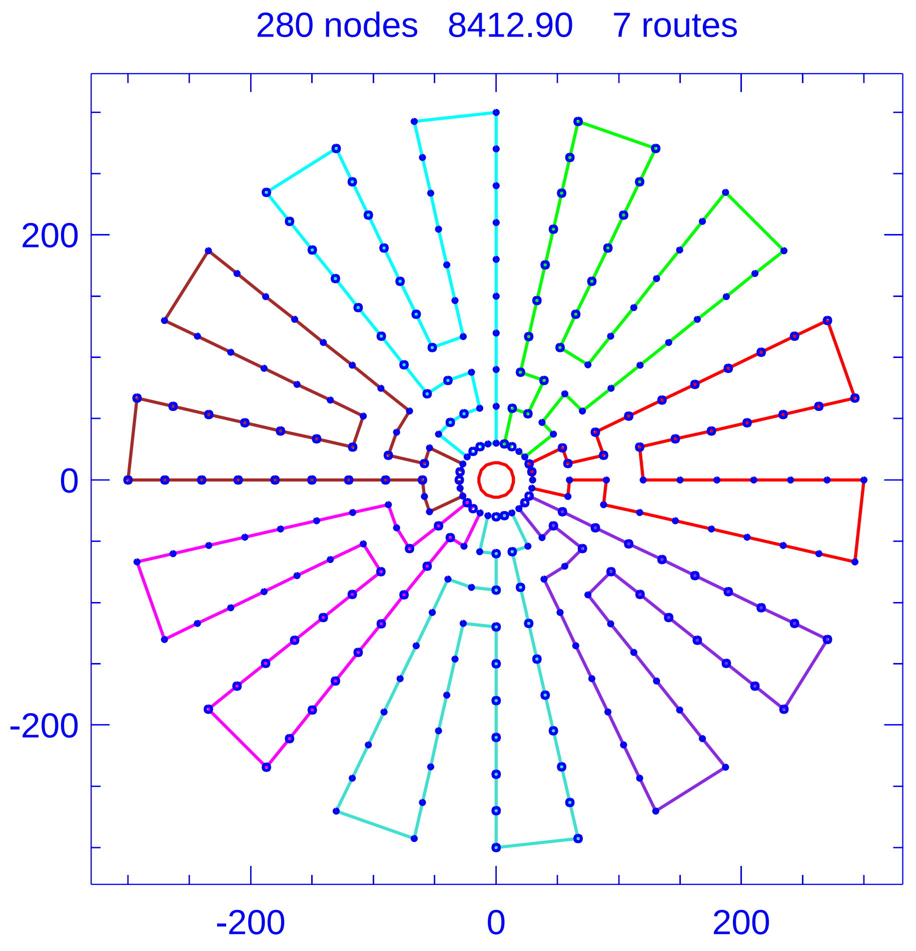


Figure B.20: Solution for problem Golden-6

Problem			Golden-6
Vehicle capacity			900
Maximum route length			1500.000
Number of nodes			280
Total route length			8412.901
Total number of routes			7
Route	Length	Load	Ordering
1	1201.843	800	(0, 1, 2, 3, 31, 30, 58, 59, 87, 115, 143, 171, 199, 227, 255, 254, 226, 198, 170, 142, 114, 86, 85, 113, 141, 169, 197, 225, 253, 280, 252, 224, 196, 168, 140, 112, 84, 57, 29, 56, 28, 0)
2	1201.843	800	(0, 5, 4, 32, 33, 61, 60, 88, 116, 144, 172, 200, 228, 256, 257, 229, 201, 173, 145, 117, 89, 90, 118, 146, 174, 202, 230, 258, 259, 231, 203, 175, 147, 119, 91, 63, 62, 34, 35, 7, 6, 0)
3	1221.997	820	(0, 10, 9, 8, 36, 64, 92, 120, 148, 176, 204, 232, 260, 261, 233, 205, 177, 149, 121, 93, 94, 122, 150, 178, 206, 234, 262, 263, 235, 207, 179, 151, 123, 95, 67, 66, 65, 37, 38, 39, 40, 12, 11, 0)

Route	Length	Load	Ordering
4	1221.997	820	(0, 14, 13, 41, 42, 70, 69, 68, 96, 124, 152, 180, 208, 236, 264, 265, 237, 209, 181, 153, 125, 97, 98, 126, 154, 182, 210, 238, 266, 267, 239, 211, 183, 155, 127, 99, 71, 43, 44, 45, 17, 16, 15, 0)
5	1181.689	780	(0, 19, 18, 46, 74, 73, 72, 100, 128, 156, 184, 212, 240, 268, 269, 241, 213, 185, 157, 129, 101, 102, 130, 158, 186, 214, 242, 270, 271, 243, 215, 187, 159, 131, 103, 75, 47, 48, 20, 0)
6	1201.843	800	(0, 21, 49, 50, 78, 77, 76, 104, 132, 160, 188, 216, 244, 272, 273, 245, 217, 189, 161, 133, 105, 106, 134, 162, 190, 218, 246, 274, 275, 247, 219, 191, 163, 135, 107, 79, 51, 52, 24, 23, 22, 0)
7	1181.689	780	(0, 26, 25, 53, 54, 82, 81, 80, 108, 136, 164, 192, 220, 248, 276, 277, 249, 221, 193, 165, 137, 109, 110, 138, 166, 194, 222, 250, 278, 279, 251, 223, 195, 167, 139, 111, 83, 55, 27, 0)

Table B.25: Details of solution to problem Golden-6

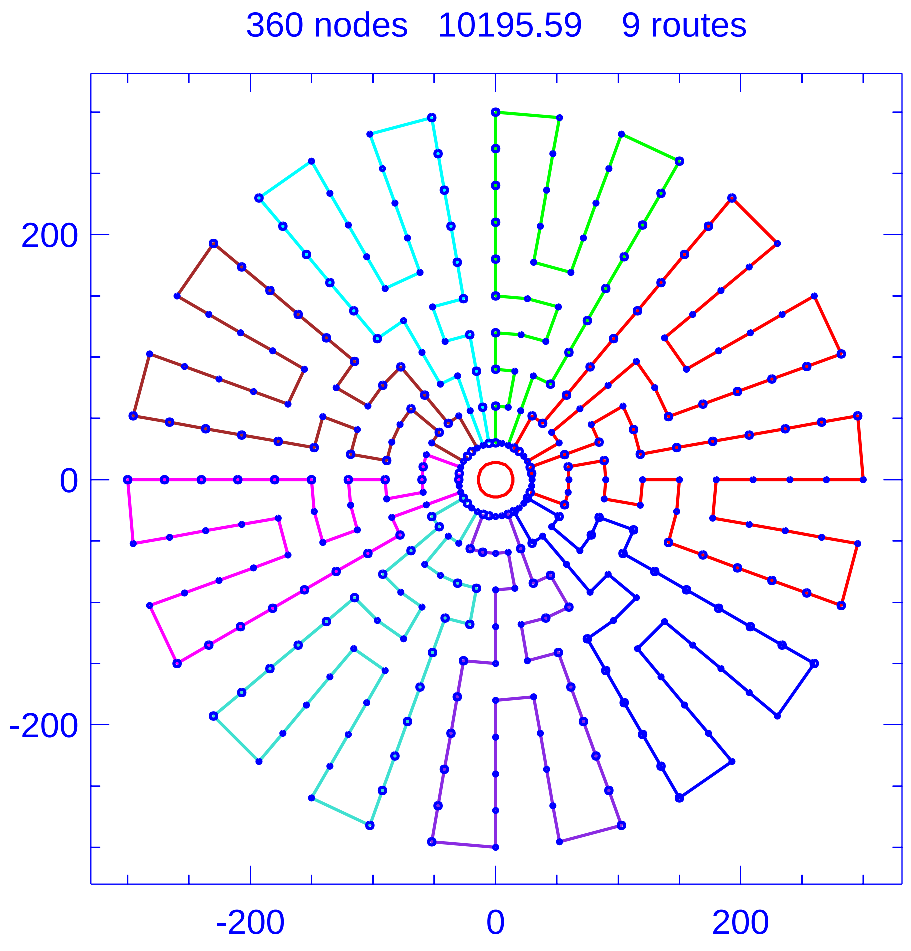


Figure B.21: Solution for problem Golden-7

Problem			Golden-7
Vehicle capacity			900
Maximum route length			1300.000
Number of nodes			360
Total route length			10195.586
Total number of routes			9
Route	Length	Load	Ordering
1	1059.632	720	(0, 5, 4, 40, 41, 77, 113, 149, 148, 147, 183, 219, 255, 291, 327, 328, 292, 256, 220, 184, 185, 221, 257, 293, 329, 330, 294, 258, 222, 186, 150, 114, 78, 42, 43, 7, 6, 0)
2	1117.155	740	(0, 8, 44, 80, 79, 115, 151, 187, 223, 259, 295, 331, 332, 296, 260, 224, 188, 189, 225, 261, 297, 333, 334, 298, 262, 226, 190, 154, 153, 152, 116, 117, 118, 82, 81, 45, 46, 10, 9, 0)
3	1064.862	660	(0, 11, 47, 83, 119, 120, 156, 155, 191, 227, 263, 299, 335, 336, 300, 264, 228, 192, 193, 229, 265, 301, 337, 338, 302, 266, 230, 194, 158, 157, 121, 85, 84, 48, 12, 0)

Route	Length	Load	Ordering
4	1169.448	860	(0, 14, 13, 49, 50, 86, 122, 123, 124, 160, 159, 195, 231, 267, 303, 339, 340, 304, 268, 232, 196, 197, 233, 269, 305, 341, 342, 306, 270, 234, 198, 162, 161, 125, 126, 90, 89, 88, 87, 51, 52, 16, 15, 0)
5	1148.531	820	(0, 17, 53, 54, 55, 56, 92, 91, 127, 128, 129, 165, 164, 163, 199, 235, 271, 307, 343, 344, 308, 272, 236, 200, 201, 237, 273, 309, 345, 346, 310, 274, 238, 202, 166, 130, 94, 93, 57, 21, 20, 19, 18, 0)
6	1169.448	860	(0, 23, 22, 58, 59, 95, 131, 132, 133, 169, 168, 167, 203, 239, 275, 311, 347, 348, 312, 276, 240, 204, 205, 241, 277, 313, 349, 350, 314, 278, 242, 206, 170, 134, 135, 99, 98, 97, 96, 60, 61, 25, 24, 0)
7	1148.531	860	(0, 26, 62, 63, 64, 65, 101, 100, 136, 172, 171, 207, 243, 279, 315, 351, 352, 316, 280, 244, 208, 209, 245, 281, 317, 353, 354, 318, 282, 246, 210, 174, 173, 137, 138, 139, 103, 102, 66, 30, 29, 28, 27, 0)
8	1132.843	800	(0, 32, 31, 67, 68, 104, 140, 141, 177, 176, 175, 211, 247, 283, 319, 355, 356, 320, 284, 248, 212, 213, 249, 285, 321, 357, 358, 322, 286, 250, 214, 178, 142, 143, 107, 106, 105, 69, 70, 34, 33, 0)
9	1185.136	880	(0, 35, 71, 72, 37, 38, 74, 73, 108, 144, 109, 145, 180, 179, 215, 251, 287, 323, 359, 360, 324, 288, 252, 216, 181, 217, 253, 289, 325, 326, 290, 254, 218, 182, 146, 110, 111, 112, 76, 75, 39, 3, 2, 1, 36, 0)

Table B.26: Details of solution to problem Golden-7

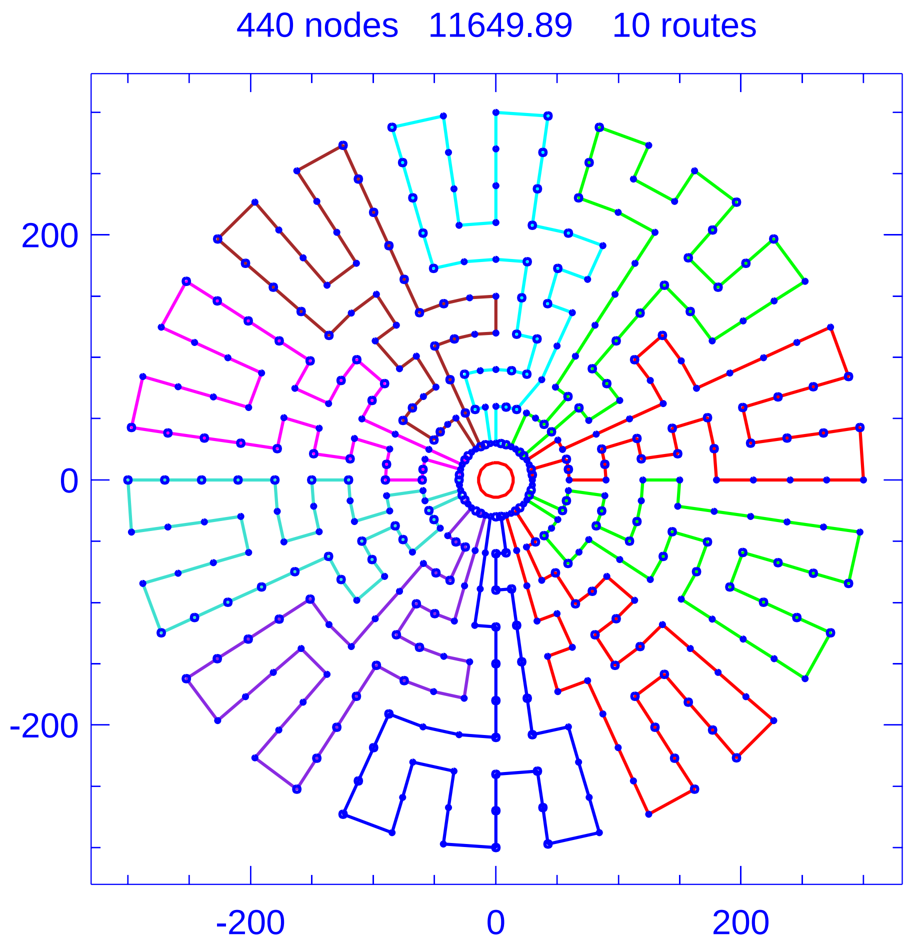


Figure B.22: Solution for problem Golden-8

Problem			Golden-8
Vehicle capacity			900
Maximum route length			1200.000
Number of nodes			440
Total route length			11649.889
Total number of routes			10
Route	Length	Load	Ordering
1	1148.110	890	(0, 4, 5, 49, 48, 92, 136, 180, 181, 182, 226, 225, 224, 268, 312, 356, 400, 399, 355, 311, 267, 266, 310, 354, 398, 397, 353, 309, 265, 221, 222, 223, 179, 178, 134, 135, 91, 90, 89, 45, 46, 47, 3, 2, 1, 44, 0)
2	1199.474	870	(0, 8, 7, 6, 50, 94, 93, 137, 138, 139, 183, 227, 271, 270, 269, 313, 357, 401, 402, 358, 314, 315, 359, 403, 404, 360, 361, 405, 406, 362, 318, 317, 316, 272, 228, 184, 140, 96, 95, 51, 52, 53, 9, 10, 0)
3	1179.852	900	(0, 11, 12, 56, 55, 54, 97, 141, 185, 186, 230, 229, 273, 274, 275, 319, 363, 407, 408, 364, 320, 276, 277, 321, 365, 409, 410, 366, 322, 278, 234, 233, 232, 231, 187, 143, 142, 98, 99, 100, 101, 102, 58, 57, 13, 0)

Route	Length	Load	Ordering
4	1173.792	900	(0, 14, 15, 59, 103, 147, 146, 145, 144, 188, 189, 190, 191, 235, 279, 323, 367, 411, 412, 368, 324, 280, 281, 325, 369, 413, 414, 370, 326, 282, 238, 237, 236, 192, 193, 149, 148, 104, 105, 106, 107, 63, 62, 61, 60, 16, 17, 0)
5	1118.147	890	(0, 18, 19, 20, 64, 108, 152, 151, 150, 194, 195, 196, 240, 239, 283, 327, 371, 415, 416, 372, 328, 284, 285, 329, 373, 417, 418, 374, 330, 286, 242, 241, 197, 198, 154, 153, 109, 110, 111, 67, 66, 65, 21, 22, 0)
6	1173.792	900	(0, 23, 24, 25, 69, 68, 112, 113, 157, 156, 155, 199, 200, 201, 245, 244, 243, 287, 331, 375, 419, 420, 376, 332, 288, 289, 333, 377, 421, 422, 378, 334, 290, 246, 202, 203, 204, 160, 159, 158, 114, 115, 116, 72, 71, 70, 26, 0)
7	1182.353	900	(0, 27, 28, 29, 73, 74, 75, 119, 118, 117, 161, 205, 249, 248, 247, 291, 335, 379, 423, 424, 380, 336, 292, 293, 337, 381, 425, 426, 382, 338, 294, 250, 251, 252, 253, 209, 208, 207, 206, 162, 163, 164, 120, 76, 32, 31, 30, 0)
8	1199.550	880	(0, 33, 77, 121, 165, 166, 210, 254, 298, 297, 296, 295, 339, 383, 427, 428, 384, 340, 341, 385, 429, 430, 386, 342, 343, 387, 431, 432, 388, 344, 300, 299, 255, 211, 167, 123, 122, 78, 79, 35, 34, 0)
9	1118.148	770	(0, 37, 36, 80, 124, 168, 169, 213, 212, 256, 257, 301, 345, 389, 433, 434, 390, 346, 302, 303, 347, 391, 435, 436, 392, 348, 304, 260, 259, 258, 214, 215, 216, 172, 171, 170, 126, 125, 81, 82, 38, 39, 0)
10	1156.671	900	(0, 40, 41, 85, 84, 83, 127, 128, 129, 173, 217, 218, 219, 263, 262, 261, 305, 349, 393, 437, 438, 394, 350, 306, 307, 351, 395, 439, 440, 396, 352, 308, 264, 220, 177, 133, 176, 175, 174, 130, 131, 132, 88, 87, 86, 42, 43, 0)

Table B.27: Details of solution to problem Golden-8

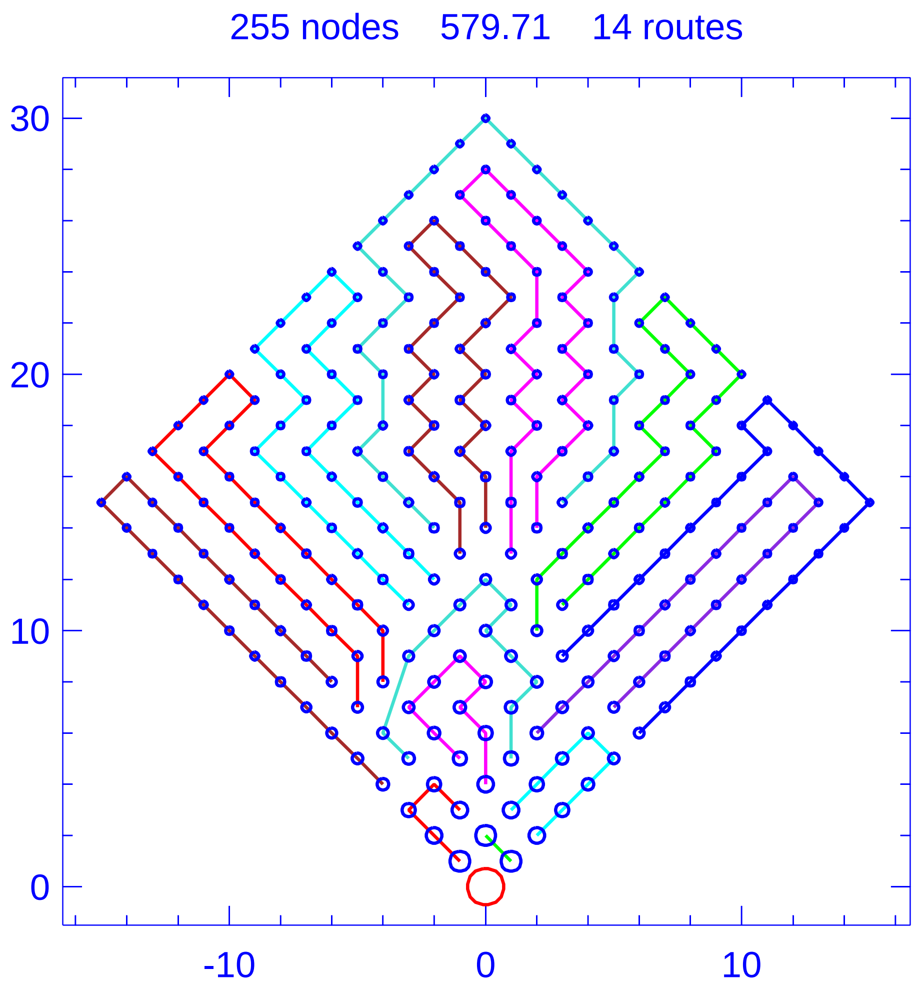


Figure B.23: Solution for problem Golden-9

Problem			Golden-9
Vehicle capacity			1000
Maximum route length			N/A
Number of nodes			255
Total route length			579.713
Total number of routes			14
Route	Length	Load	Ordering
1	10.233	916	(0, 1, 3, 6, 11, 7, 0)
2	4.828	600	(0, 2, 4, 0)
3	15.890	978	(0, 5, 9, 14, 20, 26, 19, 13, 8, 0)
4	43.941	988	(0, 10, 15, 21, 28, 36, 45, 55, 66, 78, 91, 105, 120, 136, 121, 106, 92, 79, 67, 56, 46, 37, 0)
5	20.999	979	(0, 12, 24, 31, 40, 49, 39, 30, 23, 17, 0)
6	28.820	997	(0, 16, 22, 48, 59, 71, 84, 72, 60, 50, 41, 32, 18, 0)
7	41.797	999	(0, 25, 33, 42, 52, 63, 75, 88, 102, 117, 133, 149, 134, 118, 103, 89, 76, 64, 53, 43, 34, 0)

Route	Length	Load	Ordering
8	50.499	994	(0, 27, 35, 44, 54, 65, 77, 90, 104, 119, 135, 150, 164, 177, 189, 176, 163, 148, 132, 116, 101, 87, 74, 62, 51, 0)
9	51.245	996	(0, 29, 47, 57, 68, 80, 93, 107, 122, 137, 151, 165, 178, 190, 179, 166, 152, 138, 123, 108, 94, 81, 69, 58, 38, 0)
10	56.127	996	(0, 61, 85, 99, 114, 130, 146, 161, 174, 187, 199, 209, 218, 227, 219, 210, 200, 188, 175, 162, 147, 131, 115, 100, 86, 73, 0)
11	58.923	996	(0, 70, 82, 95, 109, 124, 139, 153, 167, 180, 191, 201, 211, 220, 228, 221, 212, 202, 192, 181, 168, 154, 140, 125, 110, 96, 83, 0)
12	60.737	997	(0, 97, 127, 142, 156, 170, 182, 194, 204, 214, 223, 230, 236, 242, 237, 231, 224, 215, 205, 195, 183, 171, 157, 143, 112, 0)
13	64.879	999	(0, 98, 128, 158, 172, 184, 196, 206, 216, 232, 238, 243, 247, 251, 248, 244, 239, 233, 225, 217, 207, 197, 185, 173, 159, 144, 113, 0)
14	70.795	994	(0, 111, 126, 141, 155, 169, 193, 203, 213, 222, 229, 235, 241, 246, 250, 253, 255, 254, 252, 249, 245, 240, 234, 226, 208, 198, 186, 160, 145, 129, 0)

Table B.28: Details of solution to problem Golden-9

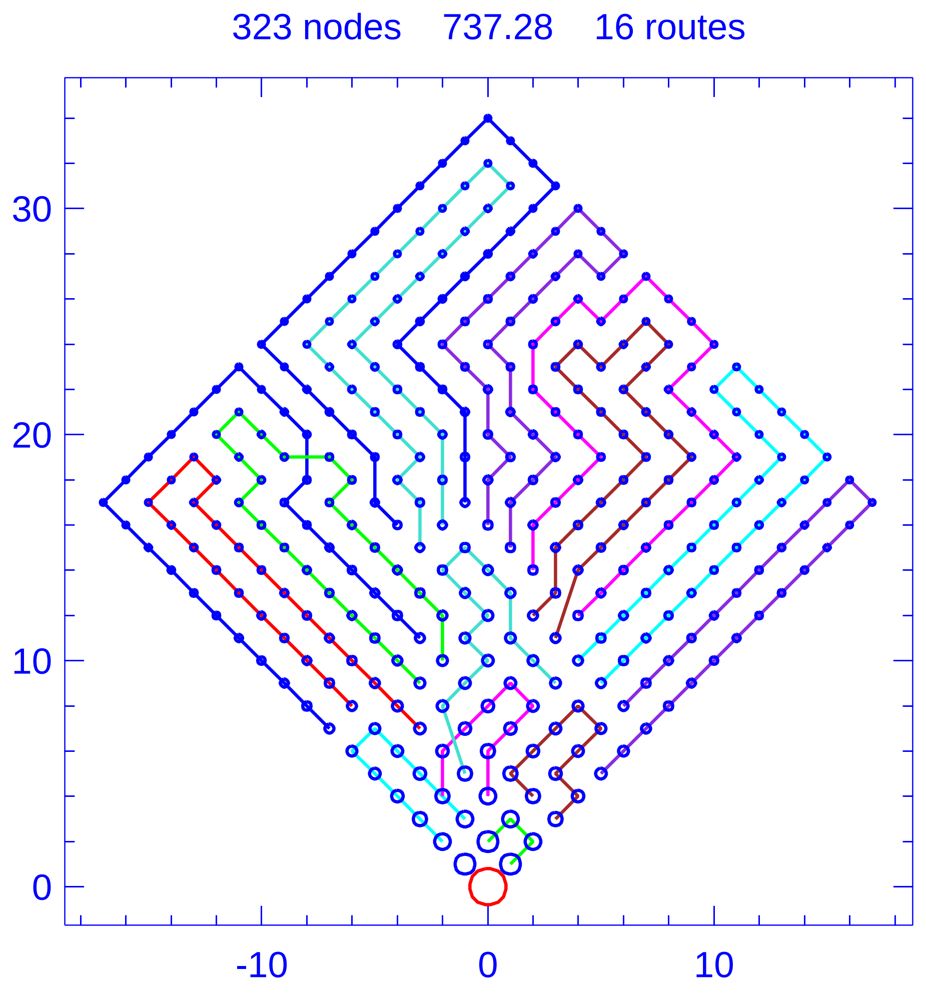


Figure B.24: Solution for problem Golden-10

Problem	Golden-10		
Vehicle capacity	1000		
Maximum route length	N/A		
Number of nodes	323		
Total route length	737.284		
Total number of routes	16		
Route	Length	Load	Ordering
1	2.828	300	(0, 1, 0)
2	7.657	960	(0, 2, 5, 8, 4, 0)
3	18.719	988	(0, 3, 6, 10, 15, 21, 29, 22, 16, 7, 0)
4	21.443	984	(0, 9, 14, 19, 26, 34, 42, 33, 25, 18, 13, 0)
5	20.957	998	(0, 11, 23, 31, 40, 50, 41, 32, 24, 12, 0)
6	35.304	1000	(0, 17, 39, 49, 60, 71, 84, 97, 111, 127, 112, 98, 72, 61, 51, 0)
7	49.598	992	(0, 20, 27, 35, 44, 54, 65, 77, 90, 104, 119, 135, 152, 170, 187, 169, 151, 134, 118, 103, 89, 76, 64, 53, 43, 0)

Route	Length	Load	Ordering
8	60.071	1000	(0, 28, 36, 45, 55, 66, 78, 91, 105, 120, 136, 153, 171, 188, 204, 219, 233, 246, 234, 221, 207, 175, 157, 140, 124, 109, 95, 82, 70, 0)
9	50.143	998	(0, 30, 38, 47, 57, 68, 80, 93, 107, 122, 138, 155, 173, 189, 172, 154, 137, 121, 106, 92, 79, 67, 56, 46, 37, 0)
10	53.383	997	(0, 48, 58, 69, 81, 94, 108, 123, 139, 156, 174, 190, 205, 220, 206, 191, 192, 176, 158, 141, 125, 110, 96, 83, 59, 0)
11	59.250	998	(0, 52, 63, 75, 88, 102, 117, 133, 150, 168, 186, 203, 218, 232, 245, 257, 244, 231, 217, 202, 185, 167, 149, 132, 116, 101, 87, 74, 62, 0)
12	62.671	996	(0, 73, 114, 130, 147, 165, 183, 200, 215, 229, 242, 255, 267, 277, 266, 254, 265, 253, 241, 228, 214, 199, 182, 164, 146, 129, 99, 85, 0)
13	67.561	996	(0, 86, 100, 115, 131, 148, 166, 184, 201, 216, 230, 243, 256, 268, 278, 287, 295, 286, 276, 285, 275, 264, 240, 227, 213, 198, 181, 163, 145, 113, 0)
14	75.605	997	(0, 126, 160, 177, 194, 209, 223, 236, 248, 259, 270, 280, 289, 297, 304, 310, 315, 319, 316, 311, 305, 298, 290, 281, 271, 260, 249, 237, 224, 210, 178, 143, 0)
15	71.560	996	(0, 128, 162, 180, 197, 212, 226, 252, 263, 274, 284, 293, 301, 294, 302, 308, 313, 307, 300, 292, 283, 273, 262, 251, 239, 211, 196, 179, 144, 0)
16	80.534	995	(0, 142, 159, 193, 208, 222, 235, 247, 258, 269, 279, 288, 296, 303, 309, 314, 318, 321, 323, 322, 320, 317, 312, 306, 299, 291, 282, 272, 261, 250, 238, 225, 195, 161, 0)

Table B.29: Details of solution to problem Golden-10

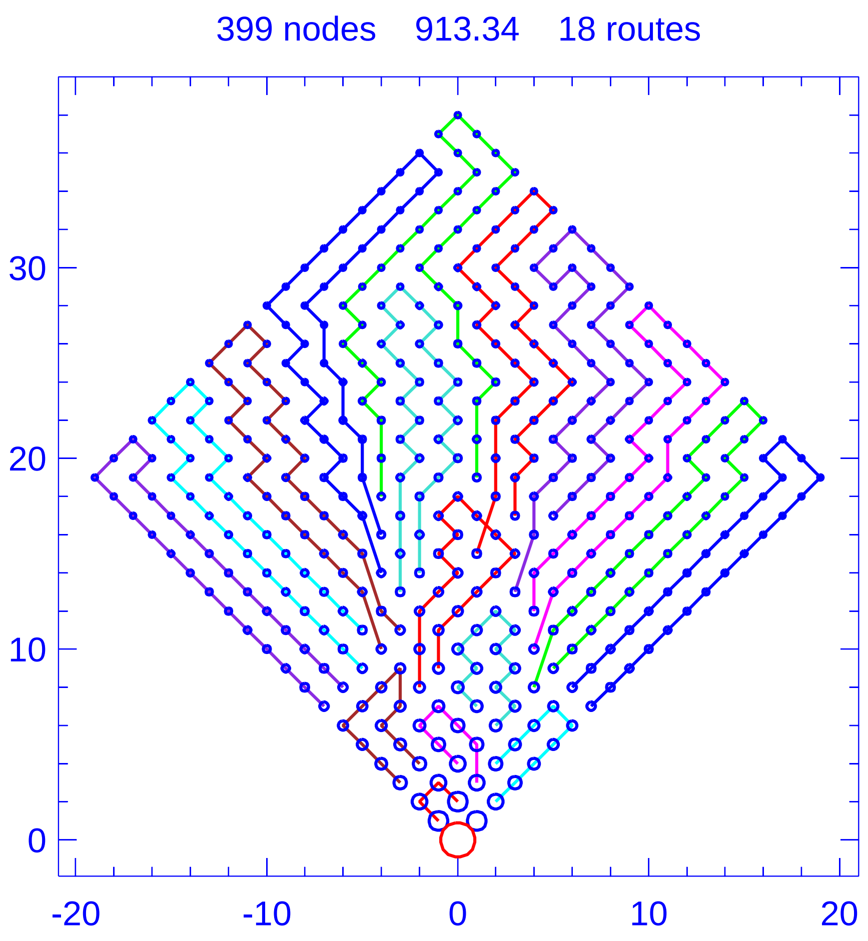


Figure B.25: Solution for problem Golden-11

Problem			Golden-11
Vehicle capacity			1000
Maximum route length			N/A
Number of nodes			399
Total route length			913.345
Total number of routes			18
Route	Length	Load	Ordering
1	7.657	960	(0, 1, 3, 7, 4, 0)
2	2.828	300	(0, 2, 0)
3	18.614	936	(0, 5, 9, 14, 20, 27, 34, 26, 19, 13, 0)
4	23.443	975	(0, 6, 10, 15, 21, 29, 38, 48, 30, 22, 16, 11, 0)
5	16.233	944	(0, 8, 18, 24, 31, 23, 17, 12, 0)
6	28.952	951	(0, 25, 33, 41, 51, 61, 73, 85, 72, 60, 50, 40, 32, 0)
7	58.083	983	(0, 28, 36, 45, 55, 66, 78, 91, 105, 120, 136, 153, 171, 190, 210, 229, 211, 191, 172, 154, 137, 121, 106, 92, 79, 67, 56, 46, 37, 0)
8	58.083	983	(0, 35, 44, 54, 65, 77, 90, 104, 119, 135, 152, 170, 189, 209, 228, 246, 227, 208, 188, 169, 151, 134, 118, 103, 89, 76, 64, 53, 43, 0)

Route	Length	Load	Ordering
9	41.686	1000	(0, 39, 59, 83, 97, 112, 127, 144, 161, 180, 162, 145, 129, 113, 98, 84, 71, 49, 0)
10	59.172	997	(0, 42, 74, 87, 101, 116, 132, 149, 167, 186, 206, 225, 244, 262, 279, 263, 245, 226, 207, 187, 168, 150, 133, 117, 102, 88, 75, 63, 52, 0)
11	61.977	975	(0, 47, 57, 68, 80, 93, 107, 122, 138, 155, 173, 192, 212, 230, 247, 264, 280, 265, 248, 231, 213, 193, 174, 156, 139, 123, 108, 94, 81, 69, 0)
12	66.680	996	(0, 58, 95, 109, 124, 140, 157, 175, 194, 214, 232, 249, 266, 281, 295, 309, 322, 310, 296, 282, 267, 250, 233, 215, 195, 176, 158, 141, 125, 82, 70, 0)
13	68.765	994	(0, 62, 100, 115, 131, 148, 166, 185, 205, 243, 261, 278, 294, 308, 321, 333, 344, 332, 320, 307, 293, 277, 260, 242, 224, 204, 184, 165, 147, 130, 114, 86, 0)
14	67.182	999	(0, 96, 126, 160, 198, 218, 236, 254, 270, 286, 300, 313, 326, 337, 348, 338, 327, 314, 301, 287, 271, 255, 237, 219, 199, 179, 143, 111, 0)
15	80.065	992	(0, 99, 146, 182, 202, 222, 240, 258, 275, 291, 305, 318, 330, 342, 353, 362, 352, 361, 370, 378, 371, 363, 354, 343, 331, 319, 306, 292, 276, 259, 241, 223, 203, 183, 164, 0)
16	85.804	1000	(0, 110, 159, 177, 196, 216, 234, 251, 268, 283, 297, 311, 323, 334, 345, 355, 364, 372, 379, 385, 390, 394, 391, 386, 380, 373, 365, 356, 346, 335, 324, 298, 284, 252, 235, 197, 142, 0)
17	79.642	998	(0, 128, 181, 220, 256, 273, 289, 303, 316, 328, 340, 350, 359, 368, 376, 383, 389, 384, 377, 369, 360, 351, 341, 329, 317, 304, 290, 274, 257, 239, 221, 201, 163, 0)
18	88.478	997	(0, 178, 217, 253, 269, 285, 299, 312, 325, 336, 347, 357, 366, 374, 381, 387, 392, 395, 397, 399, 398, 396, 393, 388, 382, 375, 367, 358, 349, 339, 315, 302, 288, 272, 238, 200, 0)

Table B.30: Details of solution to problem Golden-11

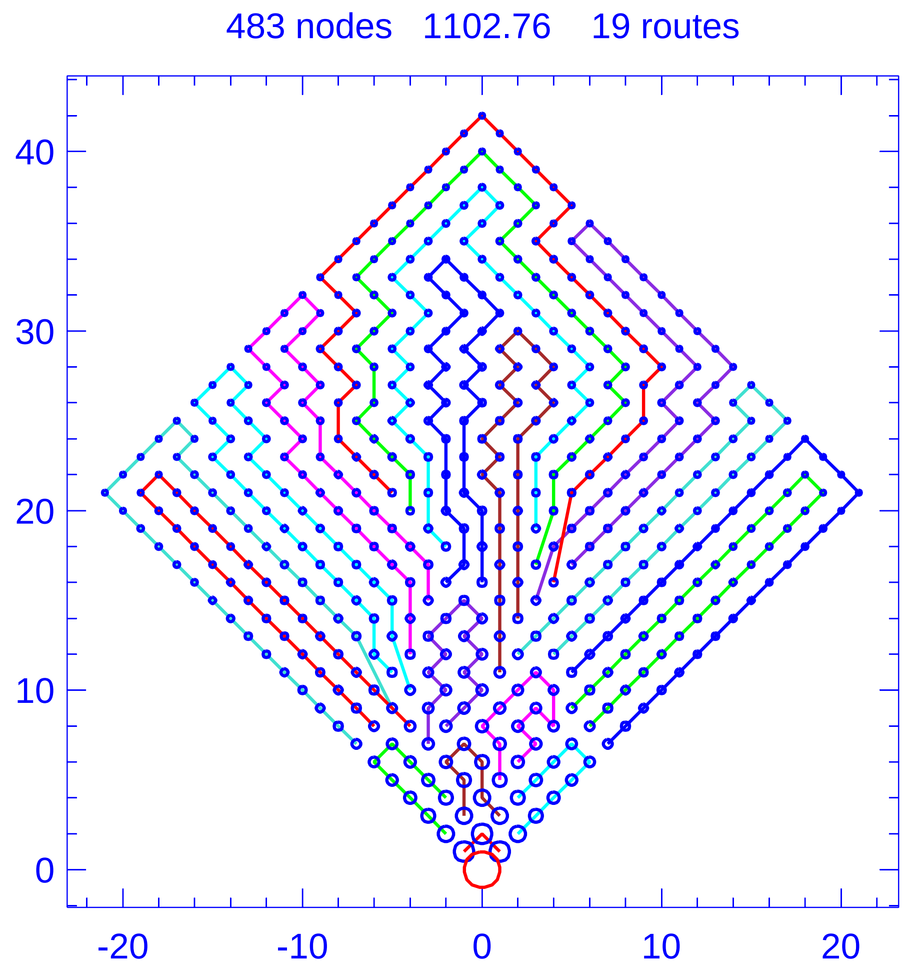


Figure B.26: Solution for problem Golden-12

Problem			Golden-12
Vehicle capacity			1000
Maximum route length			N/A
Number of nodes			483
Total route length			1102.755
Total number of routes			19
Route	Length	Load	Ordering
1	5.657	900	(0, 1, 4, 2, 0)
2	18.614	936	(0, 3, 6, 10, 15, 21, 29, 22, 16, 11, 0)
3	18.614	936	(0, 5, 9, 14, 20, 27, 34, 26, 19, 13, 0)
4	15.981	996	(0, 7, 17, 23, 31, 24, 12, 8, 0)
5	28.151	998	(0, 18, 32, 40, 50, 61, 73, 62, 42, 51, 41, 33, 25, 0)
6	67.094	1000	(0, 28, 36, 45, 55, 66, 78, 91, 105, 120, 136, 153, 171, 190, 210, 231, 253, 274, 294, 313, 295, 275, 255, 234, 213, 193, 174, 156, 139, 123, 108, 94, 47, 0)
7	36.247	996	(0, 30, 48, 59, 70, 83, 96, 111, 127, 112, 97, 84, 71, 60, 49, 39, 0)

Route	Length	Load	Ordering
8	64.409	998	(0, 35, 44, 54, 65, 77, 90, 104, 119, 135, 152, 170, 189, 209, 230, 252, 273, 293, 312, 292, 271, 249, 227, 206, 186, 167, 149, 132, 116, 101, 87, 74, 0)
9	58.542	987	(0, 37, 46, 56, 67, 79, 92, 106, 121, 137, 154, 172, 191, 211, 232, 254, 233, 212, 192, 173, 155, 138, 122, 107, 93, 80, 68, 57, 38, 0)
10	58.479	978	(0, 43, 53, 64, 76, 89, 103, 118, 134, 151, 169, 188, 208, 229, 251, 272, 250, 228, 207, 187, 168, 150, 133, 117, 102, 88, 75, 63, 52, 0)
11	69.614	994	(0, 58, 95, 125, 141, 158, 176, 195, 215, 236, 257, 277, 297, 315, 332, 349, 364, 348, 331, 314, 296, 276, 256, 235, 214, 194, 175, 157, 140, 124, 109, 81, 69, 0)
12	66.401	992	(0, 72, 98, 128, 162, 200, 242, 263, 284, 303, 322, 340, 356, 372, 386, 400, 387, 373, 357, 341, 323, 304, 264, 221, 181, 145, 113, 0)
13	76.958	1000	(0, 82, 110, 142, 159, 177, 196, 216, 237, 258, 278, 298, 316, 333, 350, 365, 379, 393, 406, 418, 407, 394, 380, 366, 351, 334, 317, 279, 259, 238, 217, 197, 178, 160, 126, 0)
14	67.241	999	(0, 85, 99, 114, 130, 147, 165, 184, 204, 225, 247, 269, 290, 310, 329, 346, 363, 347, 330, 311, 291, 270, 248, 226, 205, 185, 166, 148, 131, 115, 100, 86, 0)
15	88.505	997	(0, 129, 182, 202, 223, 245, 267, 288, 308, 327, 344, 361, 377, 391, 404, 416, 427, 437, 446, 454, 462, 455, 447, 438, 428, 417, 405, 392, 378, 362, 345, 328, 309, 289, 268, 246, 224, 203, 183, 164, 0)
16	77.237	999	(0, 143, 161, 199, 219, 262, 302, 320, 338, 354, 370, 384, 398, 411, 422, 432, 442, 433, 423, 412, 399, 385, 371, 355, 339, 321, 283, 241, 220, 180, 144, 0)
17	100.919	1000	(0, 146, 244, 266, 287, 307, 326, 360, 376, 390, 403, 415, 426, 436, 445, 453, 461, 468, 473, 477, 480, 482, 483, 481, 478, 474, 469, 463, 456, 448, 439, 429, 419, 408, 395, 381, 367, 352, 335, 299, 280, 260, 239, 0)
18	94.904	997	(0, 163, 222, 265, 286, 306, 325, 343, 359, 375, 389, 402, 414, 425, 435, 444, 452, 460, 467, 472, 476, 479, 475, 470, 464, 457, 449, 440, 430, 420, 409, 396, 382, 368, 336, 318, 300, 281, 261, 218, 0)
19	89.187	998	(0, 179, 198, 240, 282, 301, 319, 337, 353, 369, 383, 397, 410, 421, 431, 441, 450, 458, 465, 471, 466, 459, 451, 443, 434, 424, 413, 401, 388, 374, 358, 342, 324, 305, 285, 243, 201, 0)

Table B.31: Details of solution to problem Golden-12

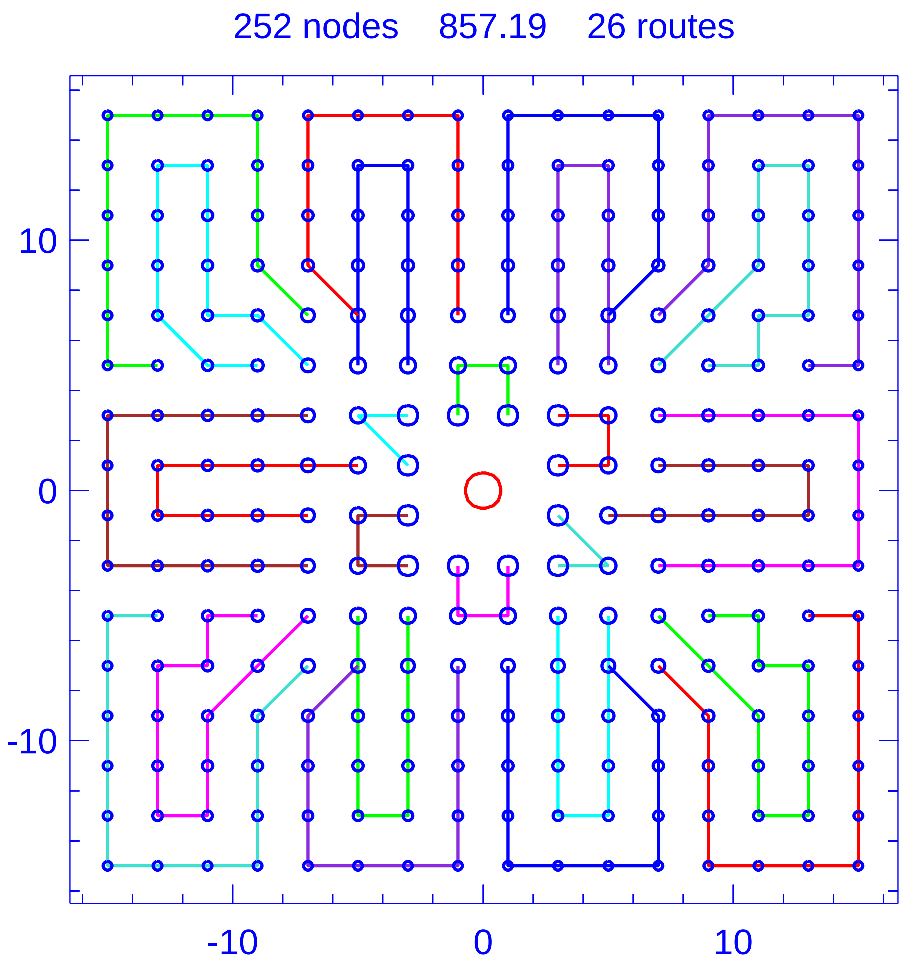


Figure B.27: Solution for problem Golden-13

Problem			Golden-13
Vehicle capacity			1000
Maximum route length			N/A
Number of nodes			252
Total route length			857.189
Total number of routes			26
Route	Length	Load	Ordering
1	13.405	960	(0, 1, 13, 14, 2, 0)
2	12.325	960	(0, 3, 17, 18, 4, 0)
3	12.233	780	(0, 5, 21, 6, 0)
4	13.405	960	(0, 7, 23, 24, 8, 0)
5	12.325	960	(0, 9, 27, 28, 10, 0)
6	12.233	780	(0, 11, 31, 12, 0)
7	30.902	988	(0, 15, 67, 105, 151, 152, 106, 68, 38, 16, 0)
8	30.902	988	(0, 19, 41, 71, 109, 155, 156, 110, 72, 20, 0)
9	28.170	936	(0, 22, 46, 78, 118, 166, 167, 119, 79, 47, 0)

Route	Length	Load	Ordering
10	30.902	988	(0, 25, 85, 127, 177, 178, 128, 86, 52, 26, 0)
11	30.902	988	(0, 29, 55, 89, 131, 181, 182, 132, 90, 30, 0)
12	28.170	936	(0, 32, 60, 96, 140, 192, 141, 97, 61, 33, 0)
13	37.232	996	(0, 34, 62, 98, 142, 194, 193, 252, 251, 191, 139, 95, 59, 0)
14	42.555	997	(0, 35, 64, 101, 102, 148, 147, 146, 145, 144, 100, 99, 63, 0)
15	50.656	987	(0, 36, 65, 103, 149, 203, 202, 201, 200, 199, 198, 197, 196, 195, 143, 0)
16	38.502	996	(0, 37, 66, 104, 150, 204, 205, 206, 207, 153, 107, 69, 39, 0)
17	38.502	996	(0, 40, 70, 108, 154, 208, 209, 210, 211, 157, 111, 73, 42, 0)
18	50.656	987	(0, 43, 74, 112, 158, 212, 213, 214, 215, 216, 217, 218, 219, 220, 164, 0)
19	42.555	997	(0, 44, 75, 115, 114, 113, 159, 160, 161, 162, 163, 116, 76, 0)
20	37.232	996	(0, 45, 77, 117, 165, 221, 222, 223, 224, 168, 120, 80, 48, 0)
21	42.555	997	(0, 49, 82, 123, 124, 174, 173, 172, 171, 170, 122, 121, 81, 0)
22	50.656	987	(0, 50, 83, 125, 175, 233, 232, 231, 230, 229, 228, 227, 226, 225, 169, 0)
23	38.502	996	(0, 51, 84, 126, 176, 234, 235, 236, 237, 179, 129, 87, 53, 0)
24	38.502	996	(0, 54, 88, 130, 180, 238, 239, 240, 241, 183, 133, 91, 56, 0)
25	50.656	987	(0, 57, 92, 134, 184, 242, 243, 244, 245, 246, 247, 248, 249, 250, 190, 0)
26	42.555	997	(0, 58, 93, 136, 135, 185, 186, 187, 188, 189, 137, 138, 94, 0)

Table B.32: Details of solution to problem Golden-13

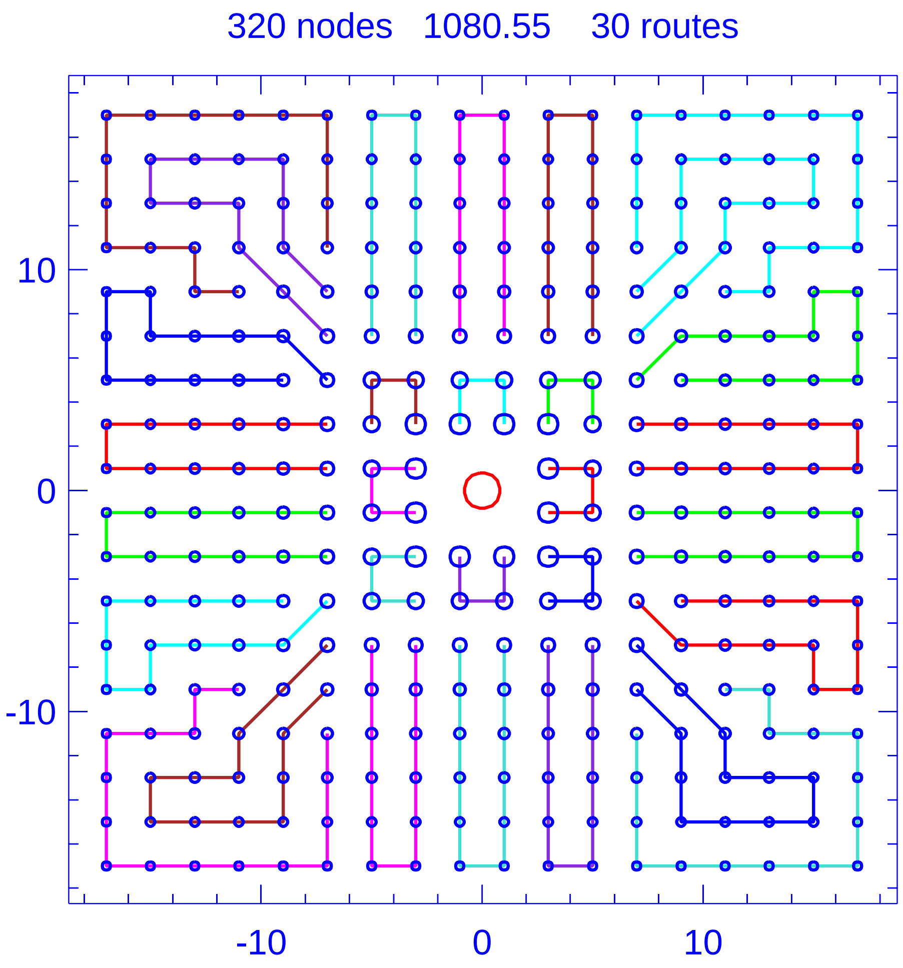


Figure B.28: Solution for problem Golden-14

Problem			Golden-14
Vehicle capacity			1000
Maximum route length			N/A
Number of nodes			320
Total route length			1080.553
Total number of routes			30
Route	Length	Load	Ordering
1	12.325	960	(0, 1, 13, 32, 12, 0)
2	16.074	840	(0, 2, 16, 15, 14, 0)
3	12.325	960	(0, 3, 17, 18, 4, 0)
4	16.074	840	(0, 5, 19, 20, 21, 0)
5	12.325	960	(0, 6, 22, 23, 7, 0)
6	16.074	840	(0, 8, 24, 25, 26, 0)
7	12.325	960	(0, 9, 27, 28, 10, 0)
8	16.074	840	(0, 11, 31, 30, 29, 0)
9	36.687	980	(0, 33, 61, 97, 141, 193, 253, 254, 194, 142, 98, 62, 34, 0)

Route	Length	Load	Ordering
10	43.726	964	(0, 35, 64, 100, 144, 196, 197, 257, 256, 255, 195, 143, 99, 63, 0)
11	47.787	997	(0, 36, 65, 102, 148, 147, 199, 200, 201, 202, 203, 149, 103, 66, 0)
12	38.218	980	(0, 37, 67, 105, 151, 205, 267, 268, 206, 152, 106, 68, 38, 0)
13	36.142	980	(0, 39, 69, 107, 153, 207, 269, 270, 208, 154, 108, 70, 40, 0)
14	38.218	980	(0, 41, 71, 109, 155, 209, 271, 272, 210, 156, 110, 72, 42, 0)
15	47.787	997	(0, 43, 74, 113, 159, 160, 216, 215, 214, 213, 212, 158, 112, 73, 0)
16	43.726	964	(0, 44, 75, 115, 163, 219, 218, 282, 283, 284, 220, 164, 116, 76, 0)
17	36.687	980	(0, 45, 77, 117, 165, 221, 285, 286, 222, 166, 118, 78, 46, 0)
18	36.687	980	(0, 47, 79, 119, 167, 223, 287, 288, 224, 168, 120, 80, 48, 0)
19	43.726	964	(0, 49, 82, 122, 170, 226, 227, 291, 290, 289, 225, 169, 121, 81, 0)
20	47.787	997	(0, 50, 83, 124, 174, 173, 229, 230, 231, 232, 233, 175, 125, 84, 0)
21	38.218	980	(0, 51, 85, 127, 177, 235, 301, 302, 236, 178, 128, 86, 52, 0)
22	36.142	980	(0, 53, 87, 129, 179, 237, 303, 304, 238, 180, 130, 88, 54, 0)
23	38.218	980	(0, 55, 89, 131, 181, 239, 305, 306, 240, 182, 132, 90, 56, 0)
24	47.787	997	(0, 57, 92, 135, 185, 186, 246, 245, 244, 243, 242, 184, 134, 91, 0)
25	43.726	964	(0, 58, 93, 137, 189, 249, 248, 316, 317, 318, 250, 190, 138, 94, 0)
26	36.687	980	(0, 59, 95, 139, 191, 251, 319, 320, 252, 192, 140, 96, 60, 0)
27	57.251	957	(0, 101, 145, 146, 198, 258, 259, 260, 261, 262, 263, 264, 265, 266, 204, 150, 104, 0)
28	57.251	957	(0, 111, 157, 211, 273, 274, 275, 276, 277, 278, 279, 280, 281, 217, 161, 162, 114, 0)
29	57.251	957	(0, 123, 171, 172, 228, 292, 293, 294, 295, 296, 297, 298, 299, 300, 234, 176, 126, 0)
30	57.251	957	(0, 133, 183, 241, 307, 308, 309, 310, 311, 312, 313, 314, 315, 247, 187, 188, 136, 0)

Table B.33: Details of solution to problem Golden-14

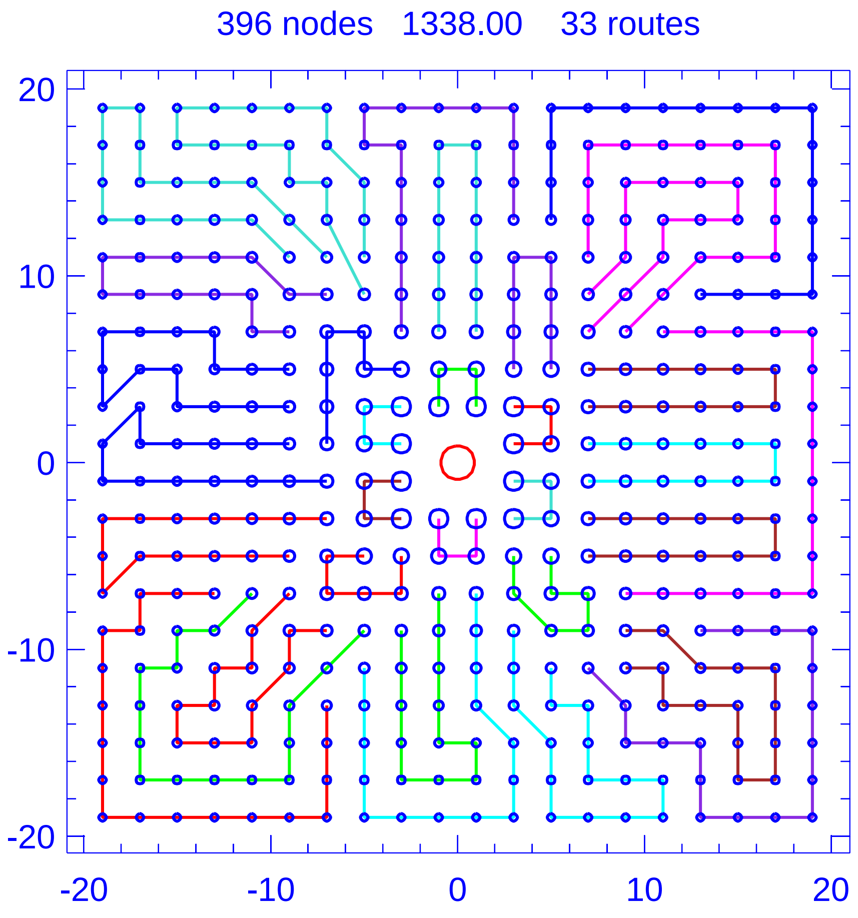


Figure B.29: Solution for problem Golden-15

Problem			Golden-15
Vehicle capacity			1000
Maximum route length			N/A
Number of nodes			396
Total route length			1337.995
Total number of routes			33
Route	Length	Load	Ordering
1	13.405	960	(0, 1, 13, 14, 2, 0)
2	12.325	960	(0, 3, 17, 18, 4, 0)
3	13.405	960	(0, 5, 21, 22, 6, 0)
4	13.405	960	(0, 7, 23, 24, 8, 0)
5	12.325	960	(0, 9, 27, 28, 10, 0)
6	13.405	960	(0, 11, 31, 32, 12, 0)
7	26.902	978	(0, 15, 37, 67, 105, 106, 68, 38, 16, 0)
8	24.902	1000	(0, 19, 20, 42, 43, 44, 45, 46, 0)
9	22.902	872	(0, 25, 49, 50, 51, 52, 26, 0)
10	25.730	944	(0, 29, 55, 90, 91, 57, 56, 30, 0)
11	36.142	980	(0, 33, 61, 97, 141, 193, 253, 320, 252, 192, 140, 96, 60, 0)

Route	Length	Load	Ordering
12	38.218	980	(0, 34, 62, 98, 142, 194, 254, 255, 195, 143, 99, 63, 35, 0)
13	47.787	997	(0, 36, 65, 102, 148, 147, 199, 200, 201, 202, 203, 149, 103, 66, 0)
14	36.142	980	(0, 39, 69, 107, 153, 207, 269, 270, 208, 154, 108, 70, 40, 0)
15	48.957	958	(0, 41, 71, 109, 155, 209, 271, 272, 342, 341, 340, 339, 338, 268, 206, 152, 0)
16	42.955	998	(0, 47, 79, 119, 167, 223, 287, 359, 358, 285, 286, 222, 166, 118, 78, 0)
17	44.740	993	(0, 48, 80, 120, 168, 224, 288, 360, 361, 362, 289, 225, 169, 121, 81, 0)
18	40.558	964	(0, 53, 87, 129, 179, 237, 238, 304, 303, 302, 236, 178, 128, 86, 0)
19	47.983	987	(0, 54, 88, 130, 180, 239, 305, 379, 378, 377, 376, 375, 301, 235, 177, 127, 0)
20	38.218	980	(0, 58, 94, 138, 190, 250, 318, 319, 251, 191, 139, 95, 59, 0)
21	56.097	988	(0, 64, 101, 146, 198, 258, 259, 260, 261, 262, 263, 264, 265, 266, 204, 150, 104, 0)
22	57.679	994	(0, 72, 157, 211, 212, 274, 275, 276, 277, 347, 346, 345, 344, 343, 273, 210, 156, 110, 0)
23	49.632	999	(0, 73, 74, 113, 161, 217, 281, 353, 354, 282, 218, 162, 114, 115, 75, 0)
24	48.611	994	(0, 76, 116, 164, 163, 219, 283, 355, 356, 357, 284, 220, 221, 165, 117, 77, 0)
25	48.460	990	(0, 82, 123, 124, 172, 173, 229, 230, 231, 232, 174, 125, 83, 84, 0)
26	55.819	996	(0, 85, 126, 175, 233, 299, 298, 297, 296, 295, 294, 293, 292, 228, 227, 171, 122, 0)
27	52.398	985	(0, 89, 131, 181, 240, 306, 380, 381, 382, 383, 309, 308, 307, 241, 183, 182, 132, 0)
28	55.769	990	(0, 92, 136, 187, 247, 315, 314, 313, 312, 311, 245, 246, 186, 185, 135, 134, 0)
29	56.440	1000	(0, 93, 137, 189, 249, 317, 393, 394, 395, 396, 321, 322, 323, 324, 256, 196, 144, 100, 0)
30	61.736	1000	(0, 111, 158, 213, 214, 215, 279, 278, 348, 349, 350, 351, 352, 280, 216, 160, 159, 112, 0)
31	63.678	986	(0, 133, 184, 242, 243, 244, 310, 384, 385, 386, 387, 388, 389, 390, 391, 392, 316, 248, 188, 0)
32	65.740	973	(0, 145, 197, 257, 325, 326, 327, 328, 329, 330, 331, 332, 333, 334, 335, 336, 337, 267, 205, 151, 0)
33	65.530	978	(0, 170, 226, 290, 291, 363, 364, 365, 366, 367, 368, 369, 370, 371, 372, 373, 374, 300, 234, 176, 0)

Table B.34: Details of solution to problem Golden-15

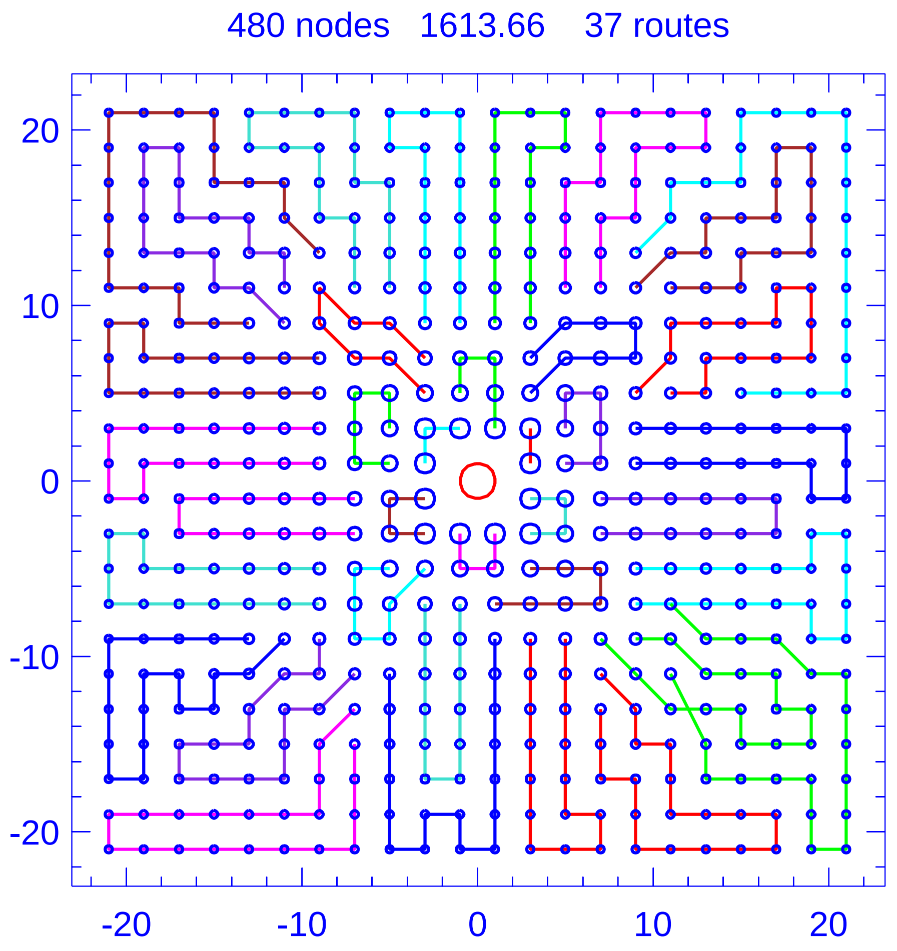


Figure B.30: Solution for problem Golden-16

Problem			Golden-16
Vehicle capacity			1000
Maximum route length			N/A
Number of nodes			480
Total route length			1613.664
Total number of routes			37
Route	Length	Load	Ordering
1	9.405	600	(0, 1, 2, 0)
2	16.261	916	(0, 3, 17, 39, 40, 18, 0)
3	10.325	900	(0, 4, 5, 6, 0)
4	13.405	960	(0, 7, 23, 24, 8, 0)
5	12.325	960	(0, 9, 27, 28, 10, 0)
6	13.405	960	(0, 11, 31, 32, 12, 0)
7	20.930	924	(0, 13, 33, 34, 35, 15, 14, 0)
8	29.104	964	(0, 16, 37, 36, 64, 65, 66, 67, 38, 0)
9	30.760	945	(0, 19, 42, 43, 74, 112, 73, 72, 41, 0)
10	20.930	924	(0, 21, 20, 44, 45, 46, 22, 0)

Route	Length	Load	Ordering
11	25.730	944	(0, 25, 49, 50, 84, 85, 51, 26, 0)
12	24.902	1000	(0, 29, 30, 58, 57, 56, 55, 54, 0)
13	36.687	980	(0, 47, 79, 119, 167, 223, 287, 288, 224, 168, 120, 80, 48, 0)
14	36.687	980	(0, 52, 86, 128, 178, 236, 302, 303, 237, 179, 129, 87, 53, 0)
15	36.687	980	(0, 59, 95, 139, 191, 251, 319, 320, 252, 192, 140, 96, 60, 0)
16	48.542	991	(0, 61, 97, 141, 193, 253, 321, 396, 480, 397, 398, 322, 254, 194, 142, 98, 62, 0)
17	51.207	967	(0, 63, 100, 101, 145, 197, 257, 258, 326, 325, 324, 256, 196, 144, 143, 99, 0)
18	48.542	991	(0, 68, 106, 152, 206, 268, 338, 337, 415, 416, 417, 339, 269, 207, 153, 107, 69, 0)
19	48.542	991	(0, 70, 108, 154, 208, 270, 340, 418, 419, 420, 342, 341, 271, 209, 155, 109, 71, 0)
20	51.697	991	(0, 75, 115, 163, 219, 283, 355, 354, 434, 435, 436, 356, 284, 220, 164, 116, 76, 0)
21	48.542	991	(0, 77, 117, 165, 221, 285, 357, 437, 438, 439, 359, 358, 286, 222, 166, 118, 78, 0)
22	51.697	991	(0, 81, 121, 169, 225, 289, 361, 360, 440, 441, 442, 362, 290, 226, 170, 122, 82, 0)
23	55.423	990	(0, 83, 125, 124, 173, 231, 230, 294, 295, 296, 297, 298, 232, 174, 175, 126, 0)
24	53.138	980	(0, 88, 130, 180, 238, 304, 378, 460, 459, 377, 376, 458, 457, 375, 301, 235, 177, 127, 0)
25	49.782	991	(0, 89, 131, 181, 239, 305, 379, 461, 462, 463, 381, 380, 306, 240, 182, 132, 90, 0)
26	54.615	999	(0, 91, 134, 185, 186, 246, 245, 313, 389, 390, 314, 315, 247, 187, 136, 92, 0)
27	55.697	999	(0, 93, 189, 249, 317, 393, 392, 476, 477, 478, 479, 395, 394, 318, 250, 190, 138, 94, 0)
28	62.597	1000	(0, 102, 146, 198, 199, 259, 327, 328, 329, 330, 331, 261, 260, 200, 201, 147, 148, 103, 0)
29	59.121	992	(0, 104, 150, 204, 203, 265, 335, 334, 333, 411, 412, 413, 414, 336, 266, 267, 205, 151, 105, 0)
30	59.121	992	(0, 110, 156, 210, 272, 273, 343, 421, 422, 423, 424, 346, 345, 344, 274, 212, 211, 157, 111, 0)
31	62.597	1000	(0, 113, 159, 160, 214, 215, 279, 278, 348, 349, 350, 351, 352, 280, 216, 217, 161, 114, 0)
32	66.852	1000	(0, 123, 172, 228, 229, 293, 292, 364, 365, 366, 367, 447, 446, 445, 444, 443, 363, 291, 227, 171, 0)
33	64.632	1000	(0, 133, 184, 242, 243, 309, 383, 384, 385, 386, 468, 467, 466, 465, 464, 382, 308, 307, 241, 183, 0)
34	68.724	994	(0, 135, 244, 310, 311, 312, 388, 387, 469, 470, 471, 472, 473, 474, 475, 391, 316, 248, 188, 137, 0)
35	72.451	995	(0, 149, 202, 264, 263, 262, 332, 410, 409, 408, 407, 406, 405, 404, 403, 402, 401, 400, 399, 323, 255, 195, 0)
36	70.451	990	(0, 158, 213, 275, 276, 277, 347, 425, 426, 427, 428, 429, 430, 431, 432, 433, 353, 281, 282, 218, 162, 0)
37	72.146	1000	(0, 176, 233, 299, 373, 372, 371, 370, 369, 368, 448, 449, 450, 451, 452, 453, 454, 455, 456, 374, 300, 234, 0)

Table B.35: Details of solution to problem Golden-16

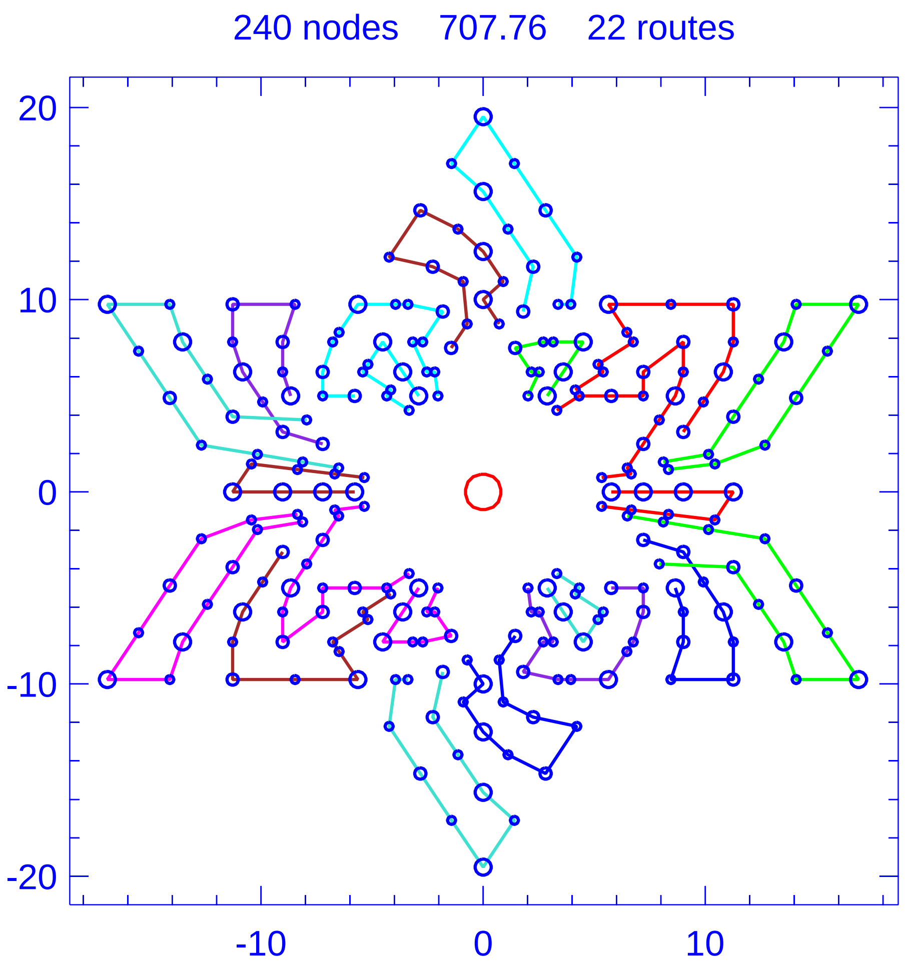


Figure B.31: Solution for problem Golden-17

Problem			Golden-17
Vehicle capacity			200
Maximum route length			N/A
Number of nodes			240
Total route length			707.756
Total number of routes			22
Route	Length	Load	Ordering
1	23.501	200	(0, 1, 61, 121, 181, 234, 174, 114, 54, 0)
2	20.672	190	(0, 2, 62, 122, 141, 86, 27, 26, 81, 21, 0)
3	19.410	170	(0, 3, 63, 123, 149, 94, 89, 34, 29, 0)
4	23.501	200	(0, 4, 64, 124, 184, 210, 150, 90, 30, 0)
5	20.672	190	(0, 5, 65, 125, 165, 110, 51, 50, 105, 45, 0)
6	19.410	170	(0, 6, 66, 126, 173, 118, 113, 58, 53, 0)

Route	Length	Load	Ordering
7	34.450	200	(0, 9, 96, 155, 214, 215, 156, 69, 101, 100, 40, 0)
8	34.450	200	(0, 12, 120, 179, 238, 239, 180, 72, 77, 76, 16, 0)
9	27.470	200	(0, 13, 73, 18, 19, 20, 7, 85, 144, 84, 25, 24, 23, 14, 0)
10	42.860	200	(0, 15, 75, 135, 195, 196, 197, 192, 240, 132, 137, 136, 17, 0)
11	29.047	200	(0, 22, 31, 82, 142, 91, 92, 151, 202, 183, 209, 154, 95, 36, 35, 0)
12	34.598	200	(0, 28, 8, 88, 68, 153, 212, 211, 152, 93, 33, 32, 0)
13	27.470	200	(0, 37, 97, 42, 43, 44, 10, 109, 168, 108, 49, 48, 47, 38, 0)
14	42.860	200	(0, 39, 99, 159, 219, 220, 221, 189, 216, 129, 161, 160, 41, 0)
15	29.047	200	(0, 46, 55, 106, 166, 115, 116, 175, 226, 186, 233, 178, 119, 60, 59, 0)
16	34.598	200	(0, 52, 11, 112, 71, 177, 236, 235, 176, 117, 57, 56, 0)
17	35.360	200	(0, 74, 83, 134, 143, 194, 182, 203, 204, 145, 67, 80, 79, 0)
18	43.401	200	(0, 78, 138, 139, 140, 127, 205, 187, 200, 199, 198, 193, 133, 0)
19	43.110	200	(0, 87, 147, 148, 128, 213, 188, 208, 207, 206, 201, 146, 0)
20	35.360	200	(0, 98, 107, 158, 167, 218, 185, 227, 228, 169, 70, 104, 103, 0)
21	43.401	200	(0, 102, 162, 163, 164, 130, 229, 190, 224, 223, 222, 217, 157, 0)
22	43.110	200	(0, 111, 171, 172, 131, 237, 191, 232, 231, 230, 225, 170, 0)

Table B.36: Details of solution to problem Golden-17

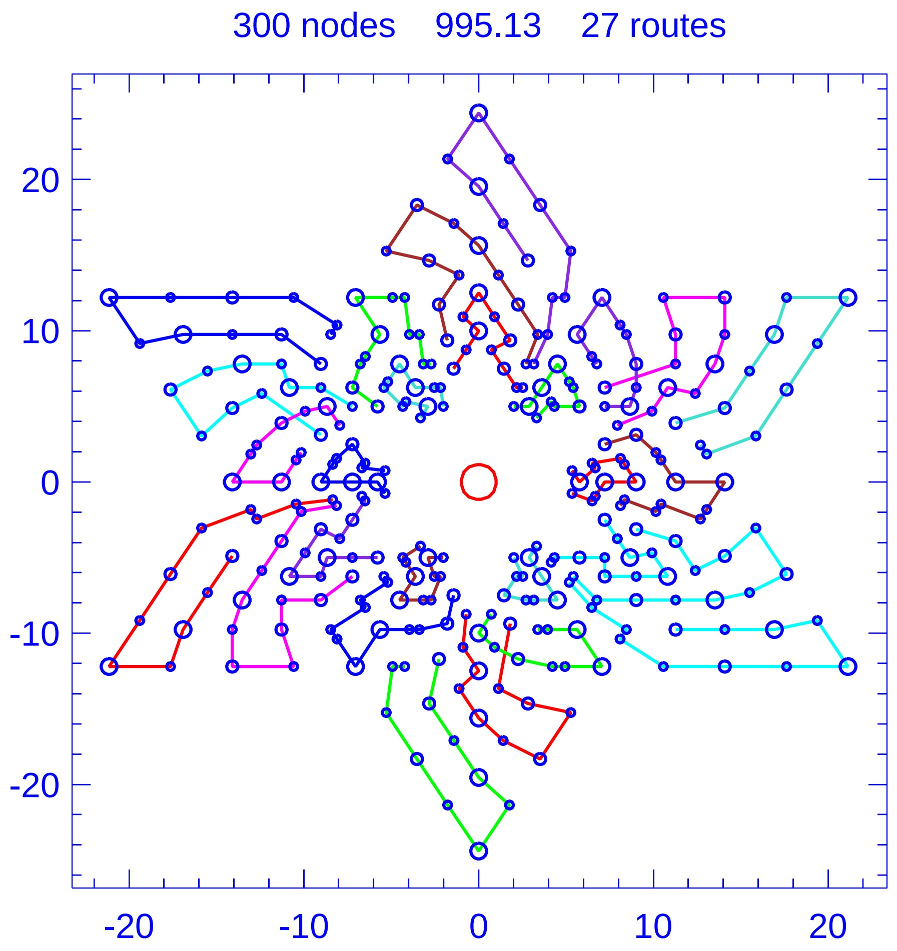


Figure B.32: Solution for problem Golden-18

Problem			Golden-18
Vehicle capacity			200
Maximum route length			N/A
Number of nodes			300
Total route length			995.133
Total number of routes			27
Route	Length	Load	Ordering
1	21.277	200	(0, 13, 1, 73, 18, 78, 133, 121, 61, 114, 15, 54, 0)
2	21.157	200	(0, 14, 74, 23, 24, 83, 134, 122, 62, 2, 21, 0)
3	28.477	200	(0, 16, 17, 12, 77, 72, 120, 119, 60, 59, 58, 113, 0)
4	32.773	200	(0, 19, 79, 138, 193, 181, 241, 294, 195, 234, 135, 174, 75, 0)
5	43.050	200	(0, 20, 80, 67, 140, 127, 205, 264, 263, 204, 145, 84, 0)
6	21.277	200	(0, 22, 31, 82, 63, 123, 149, 94, 34, 89, 3, 29, 0)
7	33.348	200	(0, 25, 7, 85, 144, 203, 254, 242, 182, 194, 143, 0)
8	21.157	200	(0, 30, 90, 39, 40, 99, 150, 124, 64, 4, 37, 0)
9	28.477	200	(0, 32, 33, 8, 93, 68, 88, 87, 28, 27, 26, 81, 0)
10	32.773	200	(0, 35, 95, 154, 209, 183, 243, 262, 211, 202, 151, 142, 91, 0)

Route	Length	Load	Ordering
11	43.050	200	(0, 36, 96, 69, 156, 129, 221, 280, 279, 220, 161, 100, 0)
12	21.277	200	(0, 38, 47, 98, 65, 125, 165, 110, 50, 105, 5, 45, 0)
13	33.348	200	(0, 41, 9, 101, 160, 219, 270, 244, 184, 210, 159, 0)
14	21.157	200	(0, 46, 106, 55, 56, 115, 166, 126, 66, 6, 53, 0)
15	28.477	200	(0, 48, 49, 10, 109, 70, 104, 103, 44, 43, 42, 97, 0)
16	32.773	200	(0, 51, 111, 170, 225, 185, 245, 278, 227, 218, 167, 158, 107, 0)
17	43.050	200	(0, 52, 112, 71, 172, 131, 237, 296, 295, 236, 177, 116, 0)
18	33.348	200	(0, 57, 11, 117, 176, 235, 286, 246, 186, 226, 175, 0)
19	43.782	200	(0, 76, 136, 137, 196, 255, 256, 197, 132, 180, 179, 178, 118, 0)
20	43.782	200	(0, 86, 146, 147, 148, 128, 213, 272, 271, 212, 153, 152, 92, 0)
21	43.782	200	(0, 102, 162, 163, 164, 130, 229, 288, 287, 228, 169, 168, 108, 0)
22	53.887	200	(0, 139, 199, 200, 187, 265, 247, 260, 259, 258, 253, 198, 0)
23	53.959	200	(0, 141, 201, 206, 261, 266, 267, 268, 248, 273, 188, 208, 207, 0)
24	53.887	200	(0, 155, 215, 216, 189, 281, 249, 276, 275, 274, 269, 214, 0)
25	53.959	200	(0, 157, 217, 222, 277, 282, 283, 284, 250, 289, 190, 224, 223, 0)
26	53.888	200	(0, 171, 231, 232, 191, 297, 251, 292, 291, 290, 285, 230, 0)
27	53.959	200	(0, 173, 233, 238, 293, 298, 299, 300, 252, 257, 192, 240, 239, 0)

Table B.37: Details of solution to problem Golden-18

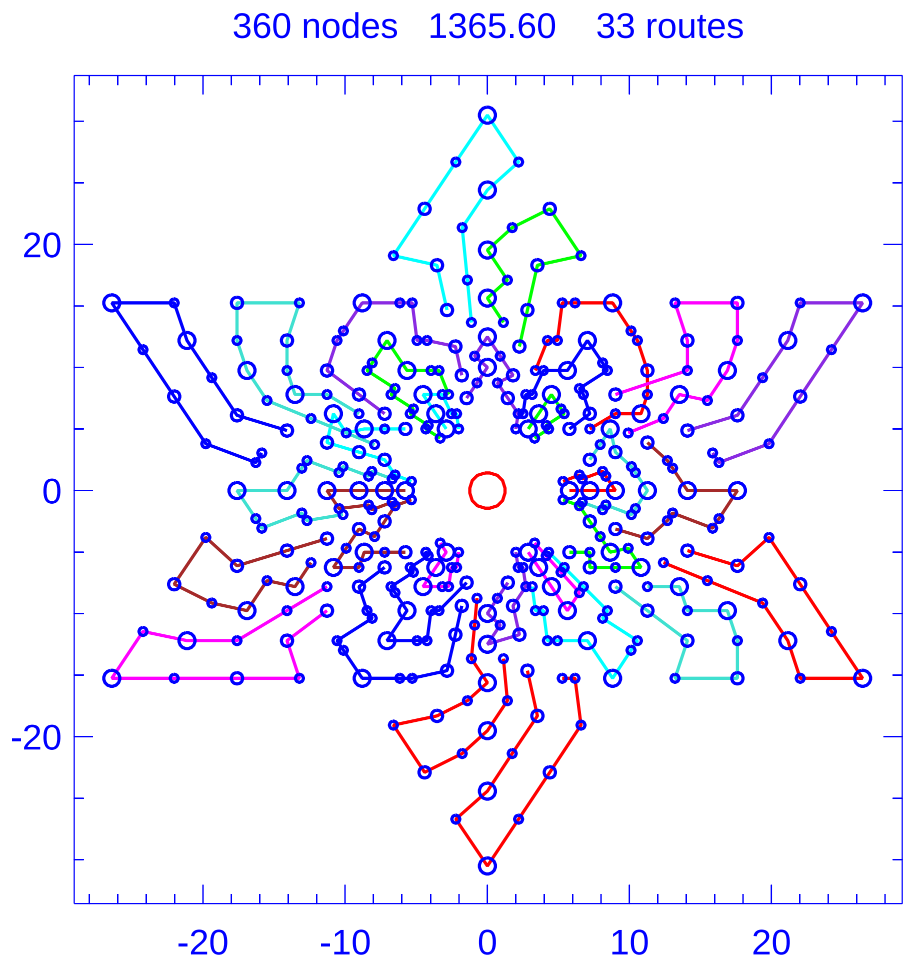


Figure B.33: Solution for problem Golden-19

Problem			Golden-19
Vehicle capacity			200
Maximum route length			N/A
Number of nodes			360
Total route length			1365.603
Total number of routes			33
Route	Length	Load	Ordering
1	19.410	170	(0, 1, 61, 121, 133, 78, 73, 18, 13, 0)
2	19.410	170	(0, 2, 62, 122, 134, 83, 74, 23, 14, 0)
3	19.071	160	(0, 3, 63, 123, 142, 82, 31, 22, 0)
4	23.839	200	(0, 4, 64, 124, 184, 217, 157, 102, 97, 0)
5	23.501	200	(0, 6, 66, 126, 186, 233, 173, 113, 53, 0)
6	29.302	200	(0, 19, 20, 7, 79, 138, 193, 181, 234, 135, 174, 75, 114, 0)
7	28.026	200	(0, 21, 26, 27, 28, 87, 88, 68, 93, 8, 33, 32, 0)
8	32.396	200	(0, 24, 84, 143, 194, 203, 254, 242, 182, 201, 141, 86, 81, 0)
9	42.643	200	(0, 25, 85, 67, 145, 204, 263, 314, 302, 321, 266, 261, 206, 146, 0)

Route	Length	Load	Ordering
10	32.252	200	(0, 29, 34, 89, 94, 149, 154, 209, 214, 269, 243, 183, 202, 151, 91, 0)
11	28.407	200	(0, 30, 39, 40, 100, 160, 69, 101, 9, 36, 35, 0)
12	28.026	200	(0, 37, 42, 43, 44, 103, 104, 70, 109, 10, 49, 48, 0)
13	19.896	180	(0, 38, 5, 65, 125, 165, 110, 105, 50, 45, 0)
14	53.589	200	(0, 41, 161, 221, 189, 276, 335, 334, 275, 216, 129, 156, 96, 0)
15	28.792	200	(0, 46, 55, 106, 115, 116, 176, 71, 117, 11, 57, 56, 0)
16	32.500	200	(0, 47, 98, 107, 158, 167, 218, 185, 245, 285, 230, 225, 170, 51, 0)
17	53.589	200	(0, 52, 112, 172, 131, 232, 291, 350, 351, 292, 191, 237, 177, 0)
18	28.026	200	(0, 54, 15, 16, 17, 12, 77, 72, 120, 119, 60, 59, 0)
19	39.976	200	(0, 58, 118, 178, 238, 293, 298, 353, 306, 246, 286, 235, 226, 175, 166, 0)
20	41.577	200	(0, 76, 136, 195, 294, 255, 354, 301, 241, 253, 198, 139, 0)
21	53.812	200	(0, 80, 140, 127, 200, 187, 265, 324, 323, 264, 205, 144, 0)
22	40.315	200	(0, 90, 99, 150, 159, 210, 219, 270, 244, 304, 337, 282, 277, 222, 162, 0)
23	42.250	200	(0, 92, 152, 211, 262, 271, 322, 303, 329, 274, 215, 155, 95, 0)
24	42.250	200	(0, 108, 168, 227, 278, 287, 338, 305, 345, 290, 231, 171, 111, 0)
25	67.577	200	(0, 137, 197, 257, 252, 360, 312, 317, 316, 315, 256, 196, 0)
26	52.646	200	(0, 147, 207, 267, 326, 327, 268, 188, 208, 128, 148, 0)
27	67.577	200	(0, 153, 213, 273, 248, 328, 308, 333, 332, 331, 272, 212, 0)
28	52.646	200	(0, 163, 223, 283, 342, 343, 284, 190, 224, 130, 164, 0)
29	67.577	200	(0, 169, 229, 289, 250, 344, 310, 349, 348, 347, 288, 228, 0)
30	52.646	200	(0, 179, 239, 299, 358, 359, 300, 192, 240, 132, 180, 0)
31	67.359	200	(0, 199, 259, 260, 247, 325, 307, 320, 319, 318, 313, 258, 0)
32	67.359	200	(0, 220, 280, 281, 249, 336, 309, 341, 340, 339, 330, 279, 0)
33	67.359	200	(0, 236, 296, 297, 251, 352, 311, 357, 356, 355, 346, 295, 0)

Table B.38: Details of solution to problem Golden-19

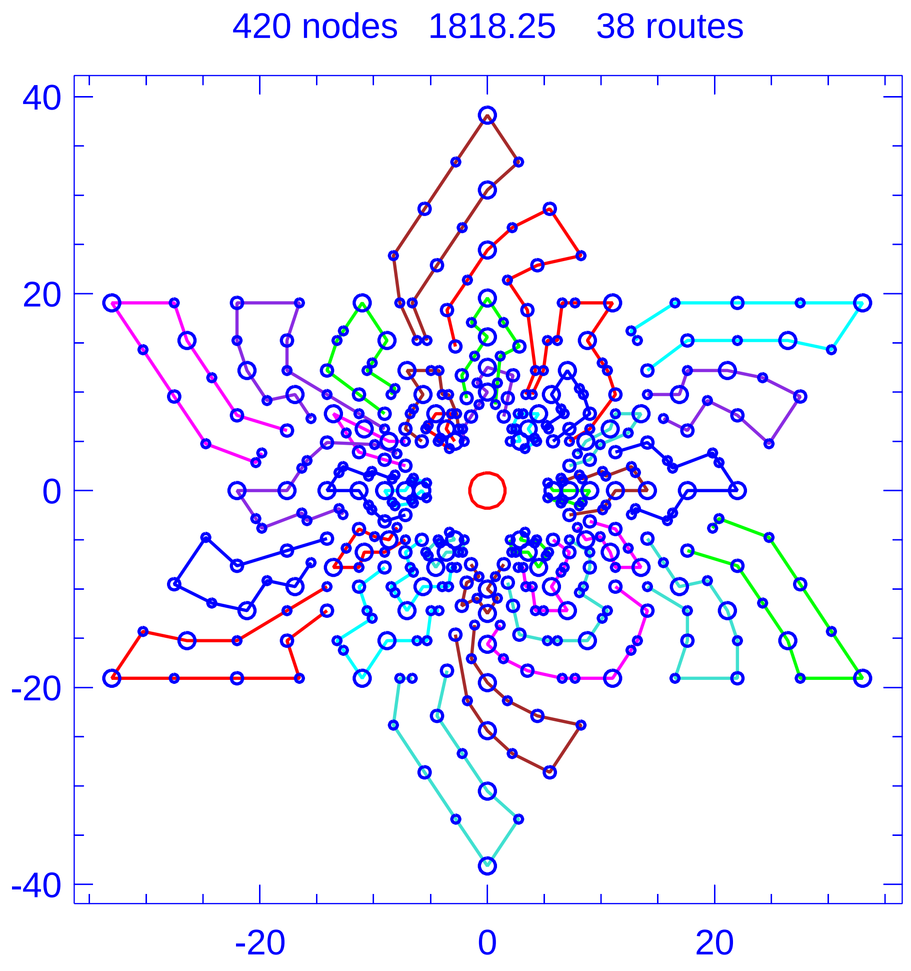


Figure B.34: Solution for problem Golden-20

Problem			Golden-20
Vehicle capacity			200
Maximum route length			N/A
Number of nodes			420
Total route length			1818.252
Total number of routes			38
Route	Length	Load	Ordering
1	20.585	180	(0, 3, 63, 142, 123, 149, 94, 89, 34, 29, 0)
2	19.896	180	(0, 13, 1, 61, 121, 174, 75, 114, 15, 54, 0)
3	21.277	200	(0, 14, 23, 74, 62, 122, 141, 86, 26, 81, 2, 21, 0)
4	32.500	200	(0, 16, 135, 234, 181, 241, 253, 198, 193, 138, 133, 78, 73, 18, 0)
5	33.654	200	(0, 17, 12, 77, 72, 180, 132, 137, 136, 76, 0)
6	32.955	200	(0, 19, 79, 80, 140, 127, 145, 67, 7, 20, 0)
7	28.407	200	(0, 22, 31, 32, 33, 8, 93, 68, 147, 87, 27, 0)
8	32.545	200	(0, 24, 84, 144, 203, 254, 242, 182, 194, 143, 134, 83, 0)
9	50.986	200	(0, 25, 85, 204, 263, 314, 302, 362, 381, 326, 321, 266, 261, 201, 0)

Route	Length	Load	Ordering
10	42.606	200	(0, 28, 88, 148, 207, 208, 188, 213, 128, 153, 152, 92, 0)
11	21.277	200	(0, 30, 39, 90, 64, 124, 157, 102, 42, 97, 4, 37, 0)
12	32.469	200	(0, 35, 95, 154, 209, 183, 243, 262, 211, 202, 151, 91, 82, 0)
13	32.292	200	(0, 36, 9, 69, 129, 161, 160, 100, 40, 0)
14	21.278	200	(0, 38, 5, 98, 47, 107, 158, 125, 65, 105, 50, 45, 0)
15	51.443	200	(0, 41, 101, 220, 279, 330, 304, 364, 397, 342, 337, 282, 277, 222, 0)
16	32.773	200	(0, 43, 103, 162, 217, 184, 244, 270, 219, 210, 159, 150, 99, 0)
17	34.257	200	(0, 44, 10, 104, 163, 164, 130, 169, 70, 109, 49, 0)
18	21.277	200	(0, 46, 55, 106, 66, 126, 173, 118, 58, 113, 6, 53, 0)
19	32.773	200	(0, 48, 108, 167, 218, 227, 278, 245, 185, 225, 170, 165, 110, 0)
20	29.714	200	(0, 51, 52, 111, 171, 112, 71, 117, 11, 57, 56, 0)
21	32.773	200	(0, 59, 119, 178, 233, 186, 246, 286, 235, 226, 175, 166, 115, 0)
22	42.853	200	(0, 60, 120, 179, 238, 293, 298, 353, 306, 346, 295, 236, 176, 116, 0)
23	66.986	200	(0, 96, 156, 216, 276, 335, 394, 395, 336, 249, 281, 189, 221, 0)
24	51.208	200	(0, 139, 199, 258, 313, 318, 373, 361, 301, 354, 255, 294, 195, 0)
25	67.800	200	(0, 146, 206, 267, 268, 327, 386, 387, 328, 248, 273, 272, 212, 0)
26	50.852	200	(0, 155, 215, 275, 334, 389, 363, 303, 329, 274, 269, 214, 0)
27	51.209	200	(0, 168, 228, 287, 338, 347, 398, 365, 305, 345, 290, 285, 230, 0)
28	67.265	200	(0, 172, 232, 191, 297, 356, 415, 416, 357, 251, 292, 231, 0)
29	54.257	200	(0, 177, 131, 237, 296, 355, 406, 366, 413, 358, 299, 239, 0)
30	67.575	200	(0, 196, 197, 192, 257, 252, 360, 419, 418, 359, 300, 240, 0)
31	68.664	200	(0, 200, 259, 260, 319, 378, 379, 320, 247, 265, 187, 205, 0)
32	65.808	200	(0, 223, 283, 343, 402, 403, 344, 250, 284, 190, 224, 0)
33	84.472	200	(0, 229, 289, 349, 310, 404, 370, 409, 408, 407, 348, 288, 0)
34	84.199	200	(0, 256, 316, 317, 312, 420, 372, 377, 376, 375, 414, 315, 0)
35	84.199	200	(0, 264, 324, 325, 307, 380, 367, 385, 384, 383, 374, 323, 0)
36	84.767	200	(0, 271, 331, 332, 333, 308, 388, 368, 393, 392, 391, 382, 322, 0)
37	84.199	200	(0, 280, 340, 341, 309, 396, 369, 401, 400, 399, 390, 339, 0)
38	84.199	200	(0, 291, 351, 352, 311, 417, 371, 412, 411, 410, 405, 350, 0)

Table B.39: Details of solution to problem Golden-20

B.5. Taillard Benchmarks

This section contains the best solutions we found for the 15 problem instances of Taillard [76].

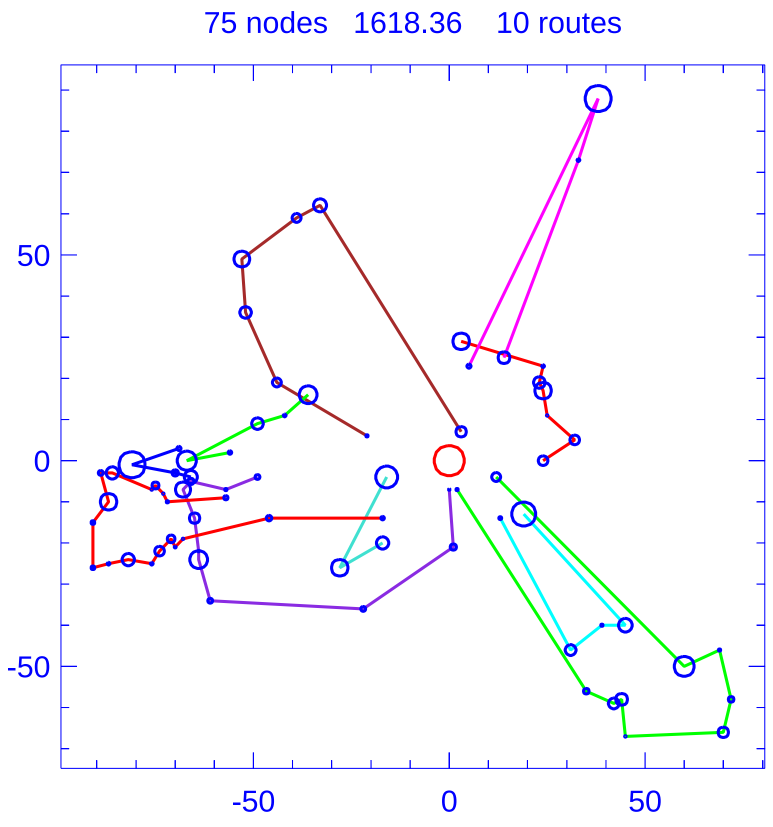


Figure B.35: Solution for problem Taillard-75A

Problem			Taillard-75A
Vehicle capacity			1445
Maximum route length			N/A
Number of nodes			75
Total route length			1618.357
Total number of routes			10
Route	Length	Load	Ordering
1	106.091	1382	(0, 12, 24, 18, 27, 21, 19, 15, 0)
2	219.880	1433	(0, 13, 6, 11, 2, 3, 10, 9, 7, 1, 22, 0)
3	132.325	1404	(0, 16, 4, 5, 8, 17, 0)
4	177.350	1308	(0, 20, 71, 73, 72, 70, 52, 53, 0)
5	192.523	1334	(0, 23, 75, 74, 25, 0)
6	80.331	1438	(0, 26, 67, 66, 0)

Route	Length	Load	Ordering
7	185.419	1287	(0, 28, 14, 51, 61, 30, 45, 33, 55, 46, 60, 54, 0)
8	163.139	1436	(0, 36, 44, 39, 57, 0)
9	219.473	1421	(0, 50, 47, 38, 43, 35, 49, 48, 37, 32, 34, 41, 40, 42, 29, 31, 68, 59, 69, 65, 0)
10	141.826	1313	(0, 62, 56, 58, 64, 63, 0)

Table B.40: Details of solution to problem Taillard-75A

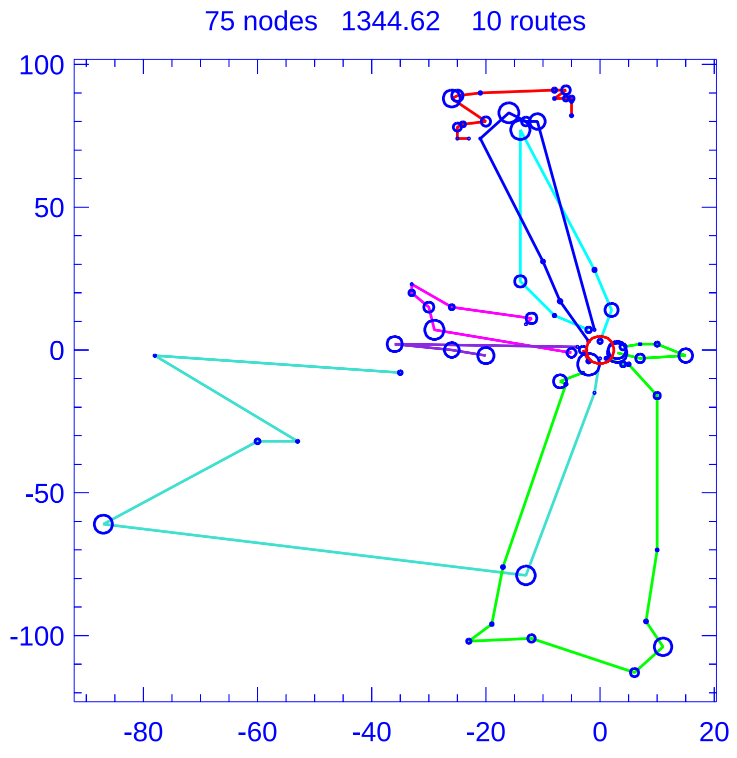


Figure B.36: Solution for problem Taillard-75B

Problem			Taillard-75B
Vehicle capacity			1679
Maximum route length			N/A
Number of nodes			75
Total route length			1344.619
Total number of routes			10
Route	Length	Load	Ordering
1	13.508	1219	(0, 1, 11, 7, 0)
2	32.385	1547	(0, 2, 5, 75, 72, 15, 9, 0)
3	160.700	1662	(0, 3, 52, 38, 19, 45, 41, 71, 0)
4	6.000	687	(0, 4, 0)
5	90.528	1617	(0, 6, 44, 37, 49, 54, 42, 39, 51, 0)
6	321.308	1594	(0, 8, 73, 63, 66, 67, 64, 65, 53, 0)
7	72.761	1590	(0, 13, 43, 50, 48, 0)
8	174.465	1659	(0, 14, 46, 47, 32, 18, 16, 23, 40, 0)
9	214.363	1677	(0, 26, 34, 24, 17, 31, 28, 35, 25, 29, 20, 27, 21, 22, 30, 33, 0)
10	258.599	1654	(0, 36, 70, 68, 59, 62, 60, 57, 58, 61, 56, 55, 69, 12, 10, 74, 0)

Table B.41: Details of solution to problem Taillard-75B

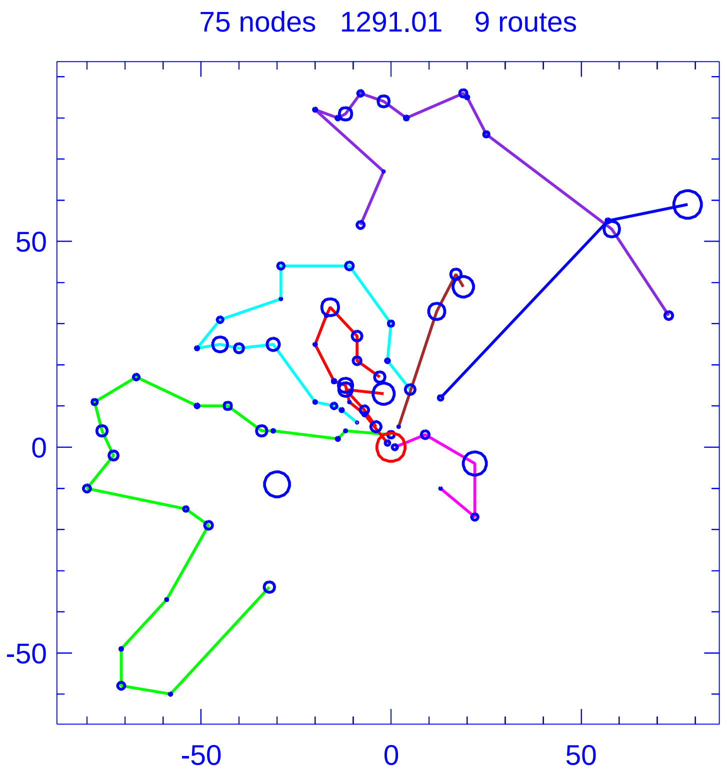


Figure B.37: Solution for problem Taillard-75C

Problem			Taillard-75C
Vehicle capacity			1122
Maximum route length			N/A
Number of nodes			75
Total route length			1291.007
Total number of routes			9
Route	Length	Load	Ordering
1	41.677	1115	(0, 4, 53, 1, 51, 0)
2	288.783	1095	(0, 5, 55, 10, 22, 27, 14, 23, 21, 19, 16, 15, 26, 13, 70, 69, 71, 67, 66, 68, 0)
3	164.121	1089	(0, 6, 8, 9, 62, 20, 12, 28, 25, 17, 18, 41, 38, 63, 56, 60, 0)
4	92.401	1109	(0, 7, 36, 50, 48, 0)
5	65.112	916	(0, 11, 59, 72, 74, 73, 0)
6	62.642	856	(0, 24, 0)
7	294.440	1115	(0, 29, 31, 40, 33, 37, 42, 43, 35, 34, 46, 49, 39, 44, 0)
8	198.393	1109	(0, 30, 32, 75, 0)
9	83.440	1116	(0, 57, 2, 54, 64, 3, 52, 47, 45, 61, 58, 65, 0)

Table B.42: Details of solution to problem Taillard-75C

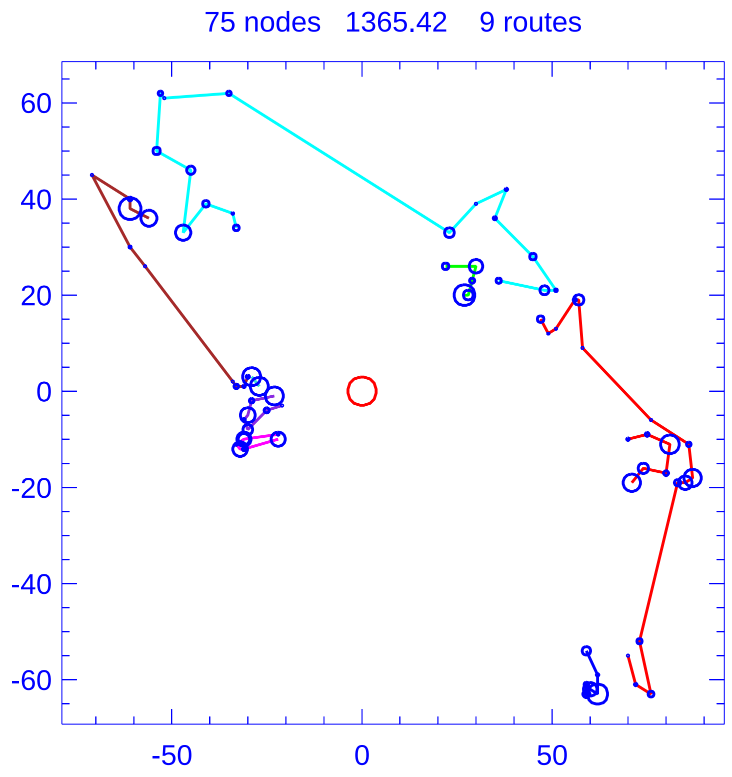


Figure B.38: Solution for problem Taillard-75D

Problem			Taillard-75D
Vehicle capacity			1699
Maximum route length			N/A
Number of nodes			75
Total route length			1365.419
Total number of routes			9
Route	Length	Load	Ordering
1	172.041	1678	(0, 2, 5, 4, 3, 6, 11, 0)
2	82.984	1644	(0, 16, 19, 20, 23, 21, 0)
3	289.025	1583	(0, 22, 12, 17, 24, 14, 15, 18, 13, 38, 36, 33, 26, 32, 35, 30, 34, 37, 0)
4	177.865	1657	(0, 27, 25, 29, 39, 31, 28, 51, 59, 45, 43, 0)
5	70.861	1576	(0, 40, 52, 49, 44, 53, 46, 47, 0)
6	59.002	1349	(0, 42, 54, 0)
7	67.656	1452	(0, 57, 41, 50, 56, 55, 48, 58, 0)
8	180.327	1559	(0, 60, 64, 69, 65, 68, 66, 0)
9	265.658	1677	(0, 67, 62, 63, 61, 10, 9, 1, 8, 7, 73, 75, 70, 74, 72, 71, 0)

Table B.43: Details of solution to problem Taillard-75D

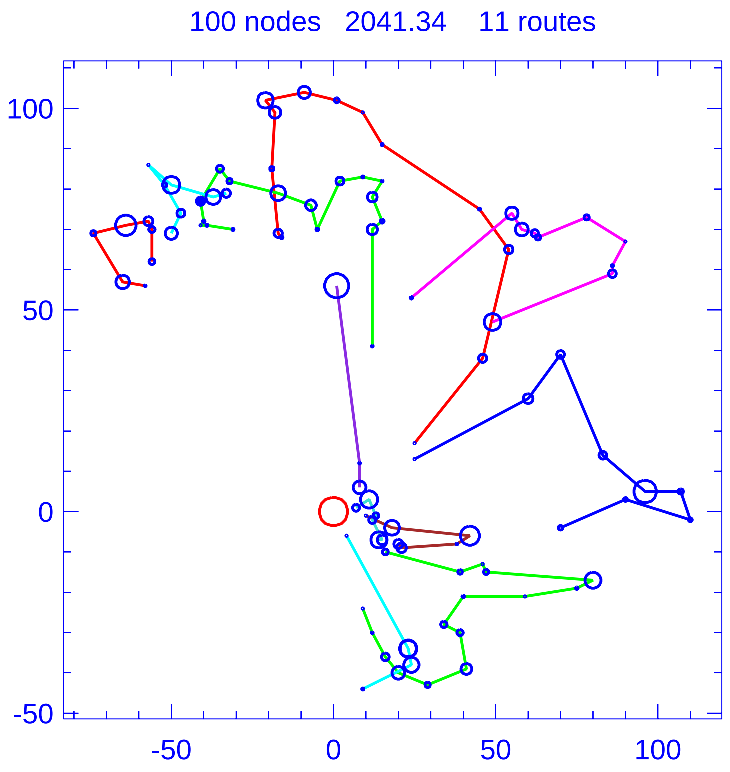


Figure B.39: Solution for problem Taillard-100A

Problem			Taillard-100A
Vehicle capacity			1409
Maximum route length			N/A
Number of nodes			100
Total route length			2041.336
Total number of routes			11
Route	Length	Load	Ordering
1	282.126	1398	(0, 7, 2, 3, 17, 6, 18, 10, 4, 11, 43, 46, 47, 52, 0)
2	216.660	1394	(0, 19, 22, 23, 36, 27, 32, 33, 31, 24, 74, 68, 73, 30, 38, 25, 65, 0)
3	106.239	1409	(0, 21, 29, 34, 20, 62, 0)
4	87.133	1347	(0, 28, 37, 35, 26, 63, 64, 0)
5	251.395	1344	(0, 42, 45, 44, 39, 41, 48, 50, 51, 49, 100, 0)
6	39.913	1392	(0, 53, 60, 57, 55, 61, 56, 0)
7	116.562	1314	(0, 54, 58, 98, 0)
8	255.647	1408	(0, 59, 67, 40, 75, 66, 69, 71, 72, 70, 0)
9	213.745	1385	(0, 76, 89, 87, 94, 96, 97, 95, 0)
10	253.517	1404	(0, 81, 77, 86, 78, 79, 84, 85, 90, 15, 14, 1, 13, 12, 5, 9, 8, 16, 99, 0)
11	218.398	1408	(0, 91, 88, 82, 92, 83, 80, 93, 0)

Table B.44: Details of solution to problem Taillard-100A

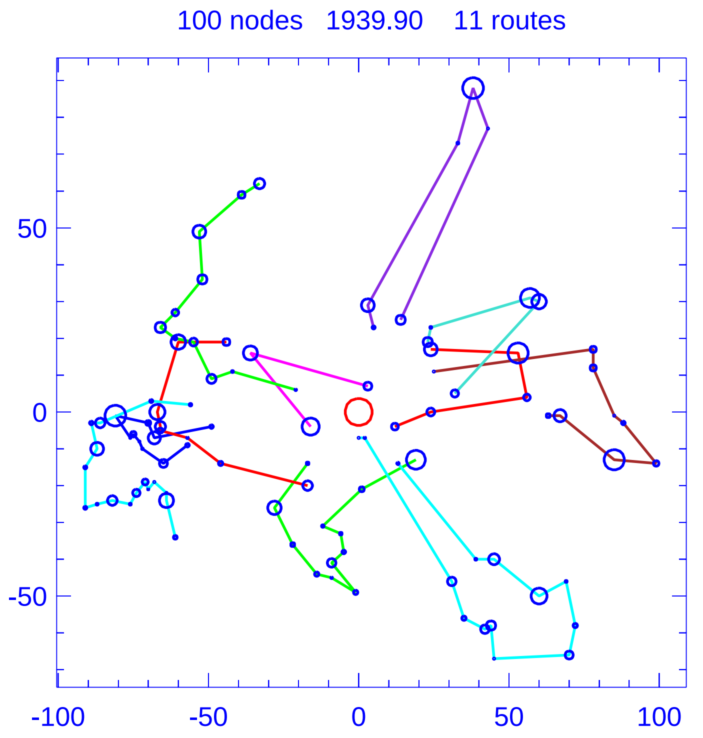


Figure B.40: Solution for problem Taillard-100B

Problem			Taillard-100B
Vehicle capacity			1842
Maximum route length			N/A
Number of nodes			100
Total route length			1939.904
Total number of routes			11
Route	Length	Load	Ordering
1	128.345	1834	(0, 13, 12, 98, 100, 27, 0)
2	162.178	1823	(0, 16, 14, 84, 86, 82, 83, 87, 85, 88, 51, 67, 65, 0)
3	222.665	1820	(0, 17, 5, 4, 6, 11, 2, 3, 10, 9, 7, 1, 8, 22, 28, 0)
4	222.255	1748	(0, 18, 97, 92, 89, 95, 99, 96, 90, 94, 0)
5	92.417	1428	(0, 20, 62, 26, 0)
6	140.999	1783	(0, 21, 19, 93, 91, 24, 0)
7	199.203	1775	(0, 23, 15, 74, 75, 76, 25, 0)
8	173.542	1835	(0, 50, 33, 47, 38, 43, 35, 39, 44, 55, 54, 0)
9	167.420	1842	(0, 52, 77, 64, 36, 46, 60, 69, 66, 0)
10	200.601	1798	(0, 53, 56, 58, 81, 80, 79, 78, 70, 72, 73, 71, 0)
11	230.279	1817	(0, 61, 30, 45, 59, 68, 31, 29, 42, 40, 41, 34, 32, 37, 48, 49, 57, 63, 0)

Table B.45: Details of solution to problem Taillard-100B

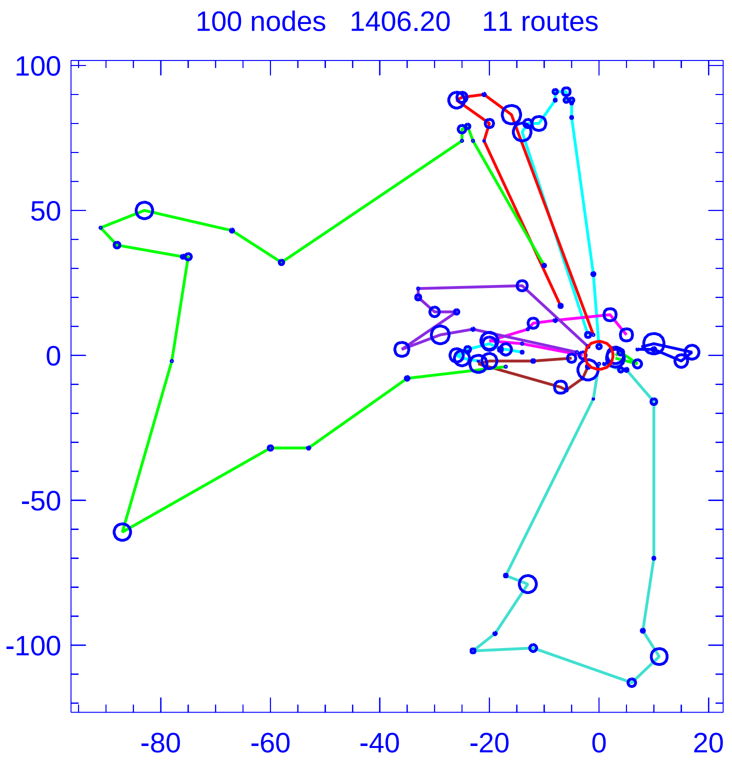


Figure B.41: Solution for problem Taillard-100C

Problem			Taillard-100C
Vehicle capacity			2043
Maximum route length			N/A
Number of nodes			100
Total route length			1406.202
Total number of routes			11
Route	Length	Load	Ordering
1	10.770	1066	(0, 1, 0)
2	17.049	1784	(0, 2, 5, 9, 4, 0)
3	189.175	2029	(0, 3, 19, 16, 23, 31, 35, 28, 17, 24, 34, 26, 45, 71, 0)
4	54.416	2017	(0, 6, 91, 48, 97, 70, 68, 36, 11, 0)
5	61.625	1965	(0, 7, 95, 94, 51, 39, 52, 41, 100, 0)
6	259.736	2012	(0, 8, 73, 59, 63, 62, 60, 57, 58, 61, 56, 55, 69, 12, 10, 74, 0)
7	115.716	1998	(0, 13, 96, 44, 43, 42, 37, 49, 54, 38, 14, 0)
8	38.685	2039	(0, 15, 72, 75, 99, 76, 77, 0)
9	191.838	2042	(0, 40, 18, 25, 29, 20, 27, 32, 46, 0)
10	411.790	2036	(0, 47, 33, 21, 22, 30, 78, 82, 81, 84, 79, 80, 83, 65, 66, 67, 64, 53, 87, 0)
11	55.402	2011	(0, 92, 89, 93, 86, 90, 85, 50, 88, 98, 0)

Table B.46: Details of solution to problem Taillard-100C

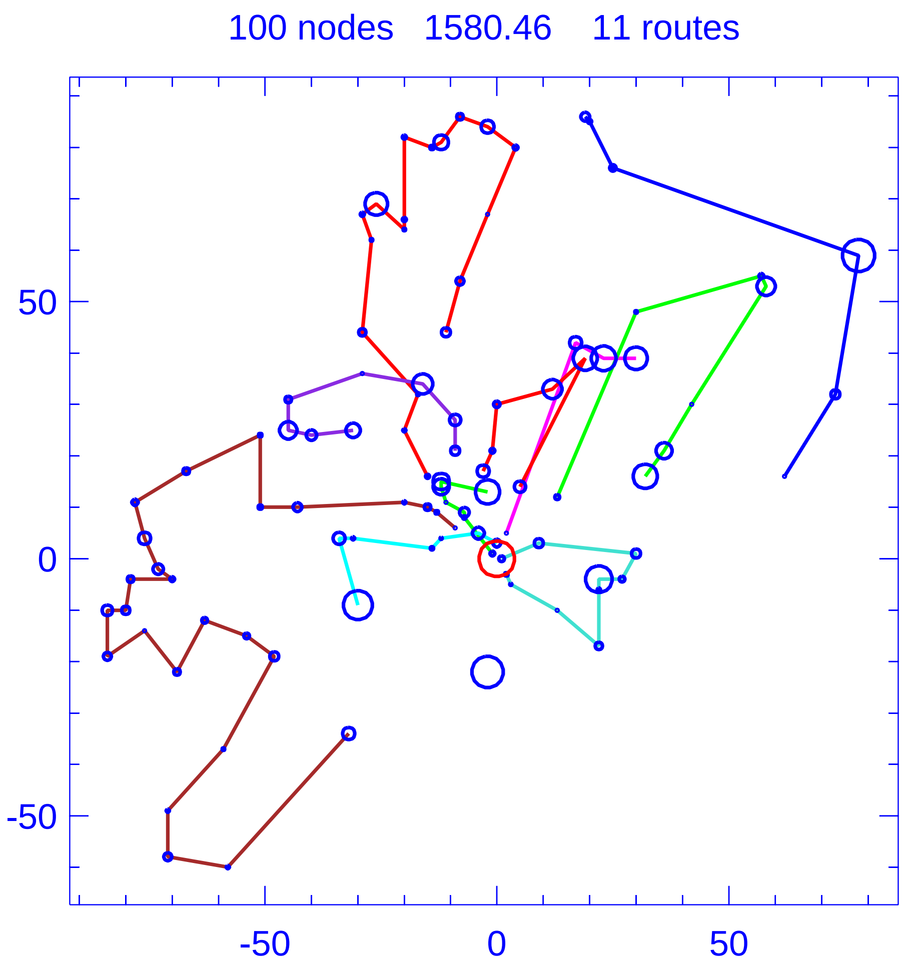


Figure B.42: Solution for problem Taillard-100D

Problem			Taillard-100D
Vehicle capacity			1297
Maximum route length			N/A
Number of nodes			100
Total route length			1580.458
Total number of routes			11
Route	Length	Load	Ordering
1	222.655	1295	(0, 3, 52, 47, 41, 82, 86, 84, 83, 85, 49, 46, 34, 35, 43, 42, 39, 44, 38, 0)
2	43.619	1294	(0, 4, 64, 53, 54, 1, 2, 57, 0)
3	83.402	1229	(0, 5, 51, 55, 10, 22, 27, 24, 0)
4	335.947	1273	(0, 6, 8, 9, 62, 14, 23, 25, 21, 19, 16, 15, 93, 89, 26, 87, 90, 92, 88, 91, 13, 70, 69, 71, 67, 66, 68, 0)

Route	Length	Load	Ordering
5	108.222	1292	(0, 7, 50, 95, 94, 0)
6	82.894	1063	(0, 11, 59, 80, 77, 72, 81, 74, 73, 76, 78, 0)
7	128.642	1297	(0, 20, 12, 28, 17, 18, 45, 61, 58, 0)
8	266.350	1278	(0, 37, 33, 40, 30, 29, 100, 0)
9	95.898	1291	(0, 60, 48, 36, 63, 56, 65, 0)
10	168.647	1263	(0, 75, 96, 32, 31, 99, 97, 98, 0)
11	44.181	1017	(0, 79, 0)

Table B.47: Details of solution to problem Taillard-100D

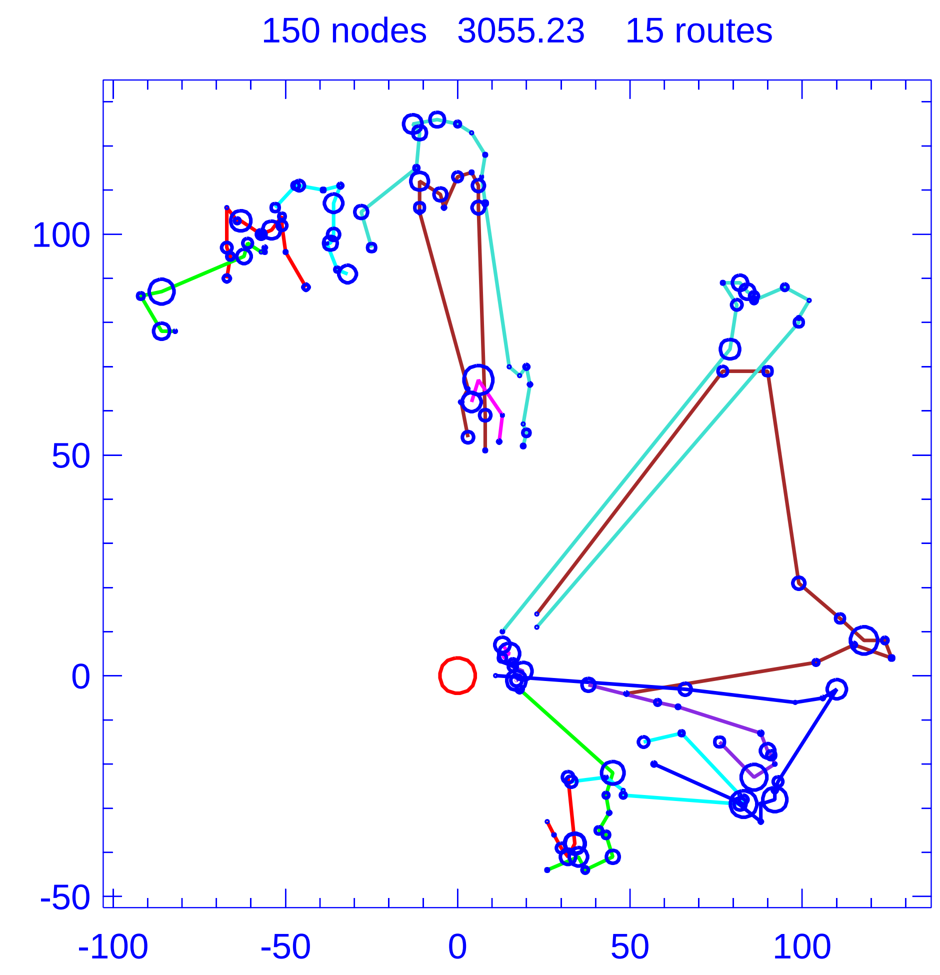


Figure B.43: Solution for problem Taillard-150A

Problem			Taillard-150A
Vehicle capacity			1544
Maximum route length			N/A
Number of nodes			150
Total route length			3055.232
Total number of routes			15
Route	Length	Load	Ordering
1	110.198	1540	(0, 19, 22, 23, 36, 20, 34, 28, 0)
2	146.137	1541	(0, 21, 29, 27, 32, 33, 31, 24, 25, 26, 65, 0)
3	184.406	1518	(0, 37, 35, 38, 30, 124, 128, 121, 126, 135, 133, 0)
4	327.050	1537	(0, 52, 47, 40, 67, 75, 66, 69, 71, 72, 70, 130, 0)
5	33.431	895	(0, 53, 60, 54, 0)
6	287.249	1477	(0, 58, 42, 46, 43, 49, 51, 50, 48, 41, 39, 44, 45, 59, 0)
7	40.338	1179	(0, 61, 56, 63, 55, 57, 64, 0)
8	244.327	1544	(0, 62, 134, 74, 68, 73, 118, 115, 114, 125, 120, 116, 136, 0)
9	264.361	1537	(0, 76, 89, 87, 92, 82, 83, 79, 84, 80, 90, 93, 81, 138, 0)

Route	Length	Load	Ordering
10	279.742	1509	(0, 77, 78, 86, 88, 91, 94, 96, 97, 95, 0)
11	260.533	1542	(0, 85, 143, 148, 141, 142, 149, 139, 140, 145, 150, 146, 0)
12	252.642	1540	(0, 107, 101, 16, 9, 12, 13, 1, 14, 15, 2, 7, 111, 103, 108, 0)
13	138.568	1477	(0, 110, 99, 98, 112, 0)
14	291.877	1451	(0, 113, 106, 104, 100, 102, 109, 105, 8, 5, 11, 4, 10, 18, 6, 17, 3, 144, 147, 0)
15	194.373	1544	(0, 119, 122, 123, 117, 129, 127, 132, 131, 137, 0)

Table B.48: Details of solution to problem Taillard-150A

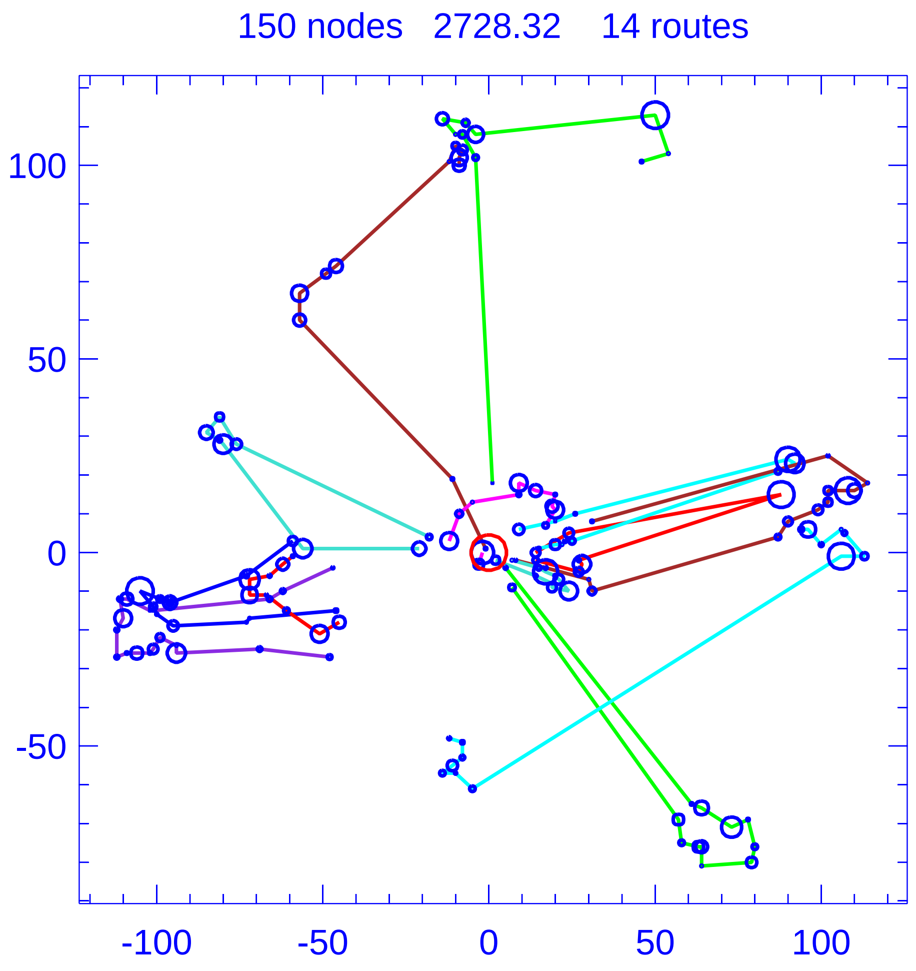


Figure B.44: Solution for problem Taillard-150B

Problem			Taillard-150B
Vehicle capacity			1918
Maximum route length			N/A
Number of nodes			150
Total route length			2728.318
Total number of routes			14
Route	Length	Load	Ordering
1	184.416	1918	(0, 13, 24, 100, 135, 139, 133, 127, 0)
2	241.176	1869	(0, 14, 8, 1, 7, 9, 10, 3, 2, 11, 6, 4, 5, 144, 0)
3	191.563	1903	(0, 20, 18, 130, 93, 91, 101, 134, 132, 138, 12, 123, 0)
4	244.173	1916	(0, 22, 136, 121, 137, 106, 98, 105, 92, 97, 103, 107, 104, 108, 122, 0)
5	9.405	908	(0, 26, 148, 0)
6	53.543	1792	(0, 28, 120, 16, 131, 128, 129, 126, 125, 17, 142, 0)
7	253.001	1884	(0, 51, 61, 30, 45, 31, 29, 42, 40, 41, 34, 32, 37, 48, 49, 35, 38, 50, 54, 53, 0)

Route	Length	Load	Ordering
8	220.869	1902	(0, 52, 57, 46, 36, 44, 39, 43, 47, 33, 68, 59, 65, 0)
9	156.744	1909	(0, 56, 58, 63, 64, 55, 60, 69, 67, 66, 0)
10	312.588	1915	(0, 74, 76, 75, 112, 115, 118, 111, 109, 116, 141, 0)
11	327.352	1914	(0, 84, 86, 82, 83, 88, 85, 87, 96, 99, 95, 89, 102, 90, 94, 0)
12	258.430	1877	(0, 114, 110, 113, 119, 117, 71, 73, 72, 70, 146, 145, 0)
13	81.556	1865	(0, 124, 27, 21, 19, 25, 15, 23, 140, 143, 147, 0)
14	193.503	1913	(0, 149, 81, 78, 79, 80, 77, 62, 150, 0)

Table B.49: Details of solution to problem Taillard-150B

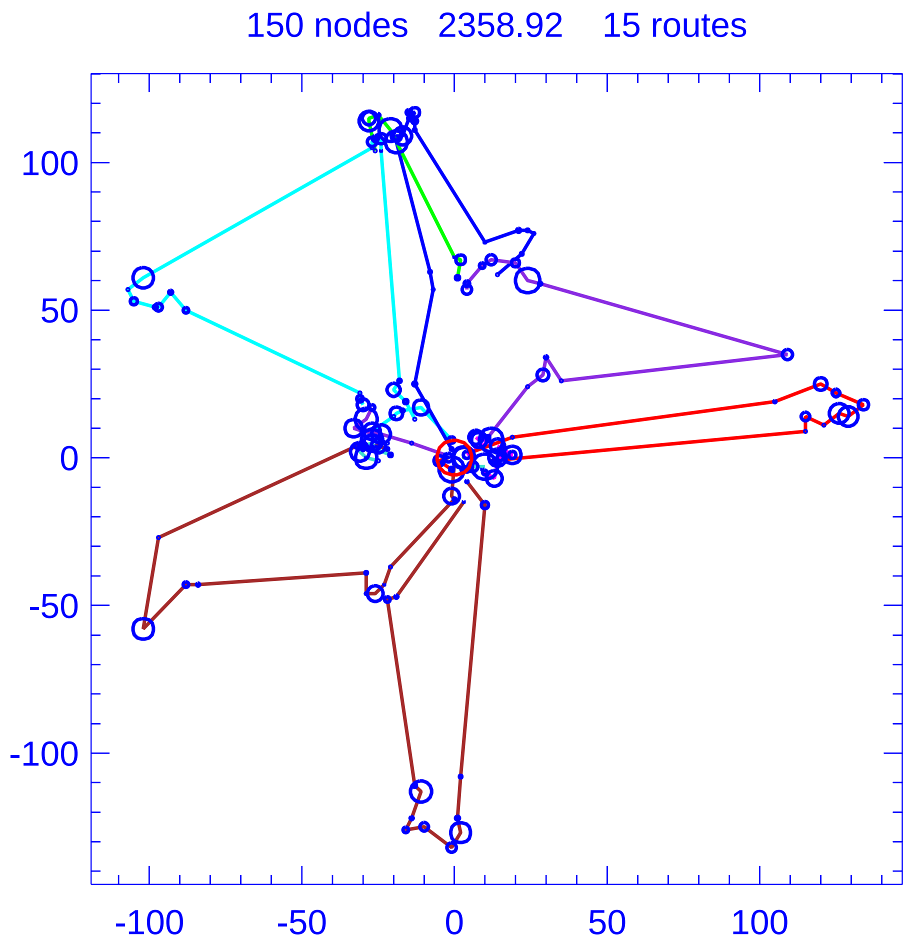


Figure B.45: Solution for problem Taillard-150C

Problem			Taillard-150C
Vehicle capacity			2021
Maximum route length			N/A
Number of nodes			150
Total route length			2358.920
Total number of routes			15
Route	Length	Load	Ordering
1	14.285	1402	(0, 1, 11, 6, 7, 0)
2	6.000	1528	(0, 2, 4, 0)
3	69.944	2005	(0, 3, 41, 52, 39, 51, 50, 94, 95, 0)
4	281.616	2006	(0, 8, 70, 68, 149, 145, 147, 150, 144, 64, 67, 66, 65, 53, 89, 92, 0)
5	28.871	1746	(0, 9, 104, 106, 100, 107, 0)
6	22.676	1388	(0, 10, 71, 12, 5, 76, 0)
7	75.931	1997	(0, 13, 36, 48, 96, 43, 44, 42, 0)
8	276.804	1991	(0, 14, 45, 142, 141, 19, 16, 23, 31, 35, 28, 17, 34, 24, 26, 135, 133, 127, 129, 128, 126, 0)
9	282.124	1971	(0, 15, 111, 116, 125, 118, 124, 120, 123, 121, 122, 119, 103, 0)

Route	Length	Load	Ordering
10	245.145	2019	(0, 33, 20, 29, 25, 18, 140, 139, 138, 0)
11	345.050	2010	(0, 37, 49, 54, 78, 82, 83, 80, 79, 84, 81, 30, 22, 21, 27, 32, 47, 38, 46, 40, 0)
12	291.461	1992	(0, 73, 146, 148, 59, 63, 62, 60, 57, 58, 61, 56, 55, 69, 74, 0)
13	47.482	1990	(0, 77, 72, 75, 108, 105, 101, 102, 99, 110, 0)
14	66.193	2000	(0, 87, 98, 97, 88, 85, 90, 86, 93, 91, 0)
15	305.338	2003	(0, 109, 115, 113, 112, 114, 117, 130, 132, 134, 131, 136, 143, 137, 0)

Table B.50: Details of solution to problem Taillard-150C

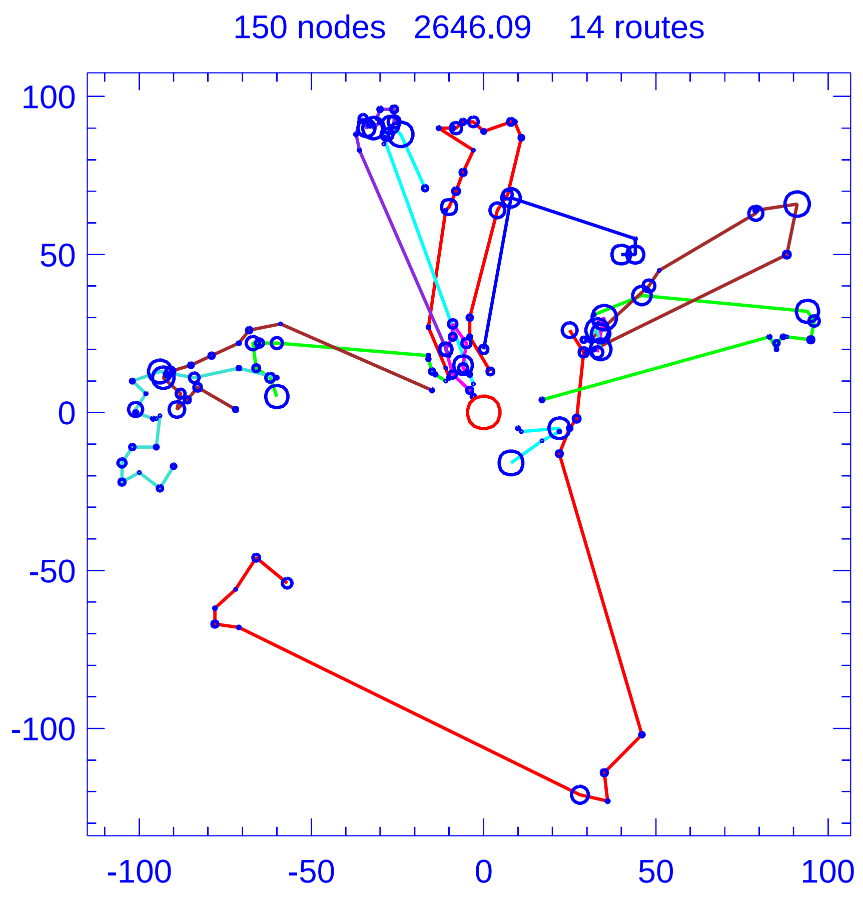


Figure B.46: Solution for problem Taillard-150D

Problem			Taillard-150D
Vehicle capacity			1874
Maximum route length			N/A
Number of nodes			150
Total route length			2646.095
Total number of routes			14
Route	Length	Load	Ordering
1	223.121	1873	(0, 2, 55, 52, 47, 45, 38, 44, 39, 49, 46, 34, 35, 43, 42, 37, 33, 40, 50, 36, 63, 56, 59, 0)
2	155.785	1800	(0, 6, 8, 9, 3, 62, 20, 12, 28, 14, 27, 24, 0)
3	191.917	1874	(0, 7, 85, 131, 126, 122, 41, 0)
4	210.791	1872	(0, 10, 18, 17, 25, 21, 19, 111, 116, 115, 118, 113, 26, 15, 13, 0)
5	62.616	1873	(0, 11, 5, 1, 53, 64, 58, 61, 65, 4, 51, 57, 0)
6	266.948	1836	(0, 22, 23, 16, 114, 119, 117, 112, 120, 109, 110, 121, 93, 89, 87, 90, 92, 88, 91, 0)
7	220.057	1874	(0, 54, 82, 86, 125, 123, 84, 124, 128, 130, 133, 132, 127, 129, 83, 0)
8	179.969	1856	(0, 60, 48, 96, 94, 95, 0)

Route	Length	Load	Ordering
9	438.207	1650	(0, 68, 70, 69, 71, 67, 66, 136, 137, 134, 135, 74, 77, 80, 149, 140, 0)
10	221.533	1868	(0, 75, 107, 103, 102, 101, 100, 106, 105, 108, 104, 98, 142, 0)
11	59.761	1784	(0, 78, 76, 72, 81, 73, 79, 0)
12	232.480	1845	(0, 141, 97, 99, 31, 32, 30, 29, 145, 146, 0)
13	95.367	1842	(0, 143, 138, 144, 147, 0)
14	87.542	1731	(0, 148, 139, 150, 0)

Table B.51: Details of solution to problem Taillard-150D

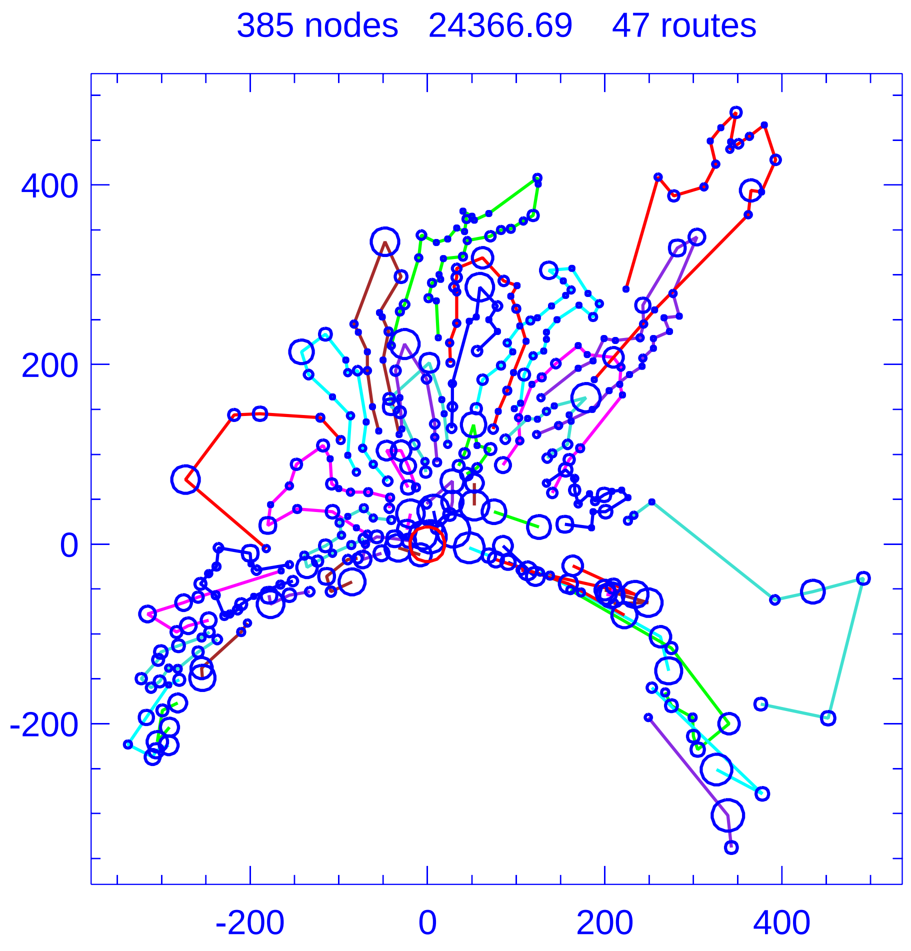


Figure B.47: Solution for problem Taillard-385

Problem			Taillard-385
Vehicle capacity			65
Maximum route length			N/A
Number of nodes			385
Total route length			24366.686
Total number of routes			47
Route	Length	Load	Ordering
1	674.838	65	(0, 56, 57, 50, 49, 385, 258, 0)
2	1034.159	65	(0, 71, 64, 65, 15, 17, 16, 29, 13, 24, 22, 23, 20, 21, 30, 31, 12, 11, 10, 9, 8, 7, 25, 26, 14, 370, 377, 68, 66, 67, 110, 0)
3	583.527	65	(0, 84, 83, 80, 79, 78, 58, 59, 76, 77, 75, 81, 82, 88, 87, 0)
4	716.226	65	(0, 90, 103, 74, 73, 61, 60, 19, 18, 63, 62, 69, 72, 188, 0)
5	238.986	65	(0, 91, 89, 100, 99, 92, 0)
6	442.219	65	(0, 93, 96, 101, 102, 104, 105, 108, 371, 129, 97, 0)
7	460.289	65	(0, 95, 98, 125, 107, 70, 106, 124, 376, 0)
8	613.490	65	(0, 121, 146, 114, 115, 118, 113, 112, 123, 130, 126, 0)

Route	Length	Load	Ordering
9	713.376	65	(0, 122, 109, 111, 119, 120, 28, 27, 6, 369, 117, 116, 147, 154, 152, 144, 135, 134, 0)
10	311.078	65	(0, 132, 128, 127, 131, 2, 136, 137, 0)
11	820.255	65	(0, 133, 145, 151, 150, 149, 148, 155, 156, 157, 160, 159, 158, 4, 5, 198, 201, 197, 366, 161, 162, 163, 164, 166, 153, 143, 141, 0)
12	179.464	64	(0, 138, 318, 0)
13	665.575	65	(0, 139, 173, 142, 168, 167, 165, 192, 196, 204, 206, 205, 209, 190, 183, 184, 0)
14	506.437	65	(0, 140, 171, 170, 172, 169, 194, 176, 177, 178, 179, 180, 0)
15	985.245	65	(0, 174, 175, 193, 203, 210, 211, 212, 215, 365, 218, 217, 220, 219, 33, 32, 216, 214, 213, 207, 200, 195, 191, 381, 0)
16	538.932	63	(0, 181, 182, 186, 185, 345, 189, 344, 202, 367, 208, 343, 340, 339, 338, 378, 0)
17	1377.670	65	(0, 199, 373, 37, 39, 38, 40, 45, 384, 43, 42, 44, 48, 47, 46, 41, 36, 34, 35, 368, 0)
18	763.888	65	(0, 222, 221, 352, 351, 0)
19	834.607	64	(0, 226, 225, 350, 355, 356, 354, 353, 0)
20	599.191	64	(0, 227, 228, 242, 246, 0)
21	664.515	65	(0, 239, 238, 235, 233, 249, 250, 379, 0)
22	732.813	65	(0, 240, 237, 234, 232, 231, 224, 223, 3, 230, 229, 236, 241, 0)
23	387.830	65	(0, 261, 260, 259, 374, 0)
24	591.920	65	(0, 265, 254, 255, 257, 380, 256, 251, 252, 248, 243, 244, 245, 247, 253, 1, 266, 0)
25	483.437	65	(0, 268, 332, 270, 274, 0)
26	264.376	58	(0, 269, 299, 0)
27	630.058	65	(0, 271, 337, 328, 326, 0)
28	261.191	64	(0, 275, 262, 263, 276, 287, 0)
29	492.326	65	(0, 292, 291, 283, 281, 280, 282, 51, 187, 55, 52, 53, 54, 85, 86, 306, 304, 0)
30	335.669	64	(0, 293, 286, 320, 277, 264, 279, 278, 285, 284, 319, 289, 290, 305, 0)
31	165.218	61	(0, 294, 288, 295, 321, 303, 309, 0)
32	263.428	65	(0, 296, 273, 272, 298, 0)
33	472.892	64	(0, 297, 267, 329, 0)
34	847.182	65	(0, 300, 327, 359, 361, 362, 363, 323, 324, 0)
35	141.154	59	(0, 301, 322, 0)
36	73.913	64	(0, 302, 311, 0)
37	85.406	62	(0, 307, 308, 0)
38	22.361	42	(0, 310, 0)
39	162.018	65	(0, 312, 94, 316, 372, 0)
40	78.168	61	(0, 313, 315, 0)
41	17.088	56	(0, 314, 0)
42	65.299	60	(0, 317, 0)
43	941.271	65	(0, 325, 357, 360, 0)
44	516.501	65	(0, 330, 331, 335, 0)
45	441.095	64	(0, 334, 333, 336, 0)
46	1195.941	65	(0, 341, 342, 349, 346, 347, 348, 383, 375, 0)
47	974.168	64	(0, 364, 358, 382, 0)

Table B.52: Details of solution to problem Taillard-385

B.6. Li Benchmarks

This section contains the best solutions we found for the 12 problem instances of Li, et al [51].

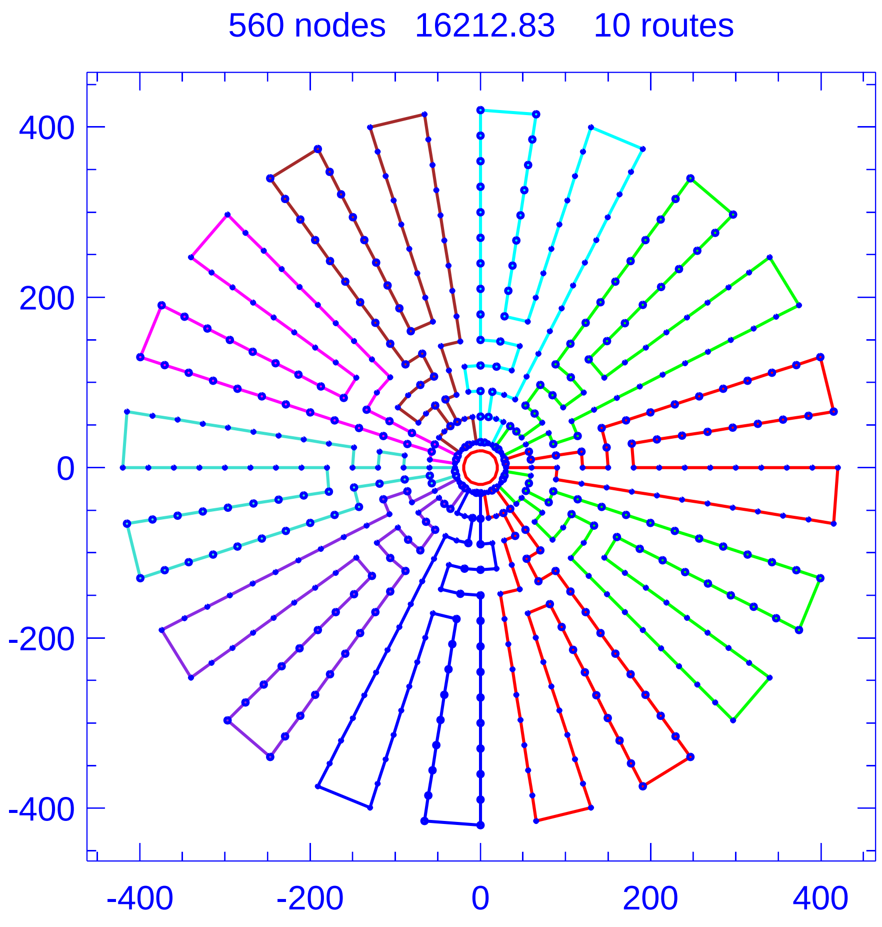


Figure B.48: Solution for problem Li-21

Problem			Li-21
Vehicle capacity			1200
Maximum route length			1800.000
Number of nodes			560
Total route length			16212.825
Total number of routes			10
Route	Length	Load	Ordering
1	1574.207	1040	(0, 2, 1, 41, 81, 120, 160, 200, 240, 280, 320, 360, 400, 440, 480, 520, 560, 521, 481, 441, 401, 361, 321, 281, 241, 201, 202, 242, 282, 322, 362, 402, 442, 482, 522, 523, 483, 443, 403, 363, 323, 283, 243, 203, 163, 162, 161, 121, 122, 82, 42, 43, 3, 0)

Route	Length	Load	Ordering
2	1654.235	1180	(0, 4, 44, 84, 83, 123, 124, 164, 204, 244, 284, 324, 364, 404, 444, 484, 524, 525, 485, 445, 405, 365, 325, 285, 245, 205, 206, 246, 286, 326, 366, 406, 446, 486, 526, 527, 487, 447, 407, 367, 327, 287, 247, 207, 167, 166, 165, 125, 126, 127, 87, 86, 85, 45, 46, 47, 7, 6, 5, 0)
3	1654.235	1140	(0, 8, 48, 49, 50, 90, 89, 88, 128, 168, 208, 248, 288, 328, 368, 408, 448, 488, 528, 529, 489, 449, 409, 369, 329, 289, 249, 209, 210, 250, 290, 330, 370, 410, 450, 490, 530, 531, 491, 451, 411, 371, 331, 291, 251, 211, 171, 170, 169, 129, 130, 131, 132, 92, 91, 51, 11, 10, 9, 0)
4	1682.480	1180	(0, 14, 13, 12, 52, 53, 54, 94, 93, 133, 173, 172, 212, 252, 292, 332, 372, 412, 452, 492, 532, 533, 493, 453, 413, 373, 333, 293, 253, 213, 214, 254, 294, 334, 374, 414, 454, 494, 534, 535, 495, 455, 415, 375, 335, 295, 255, 215, 175, 174, 134, 135, 136, 137, 97, 96, 95, 55, 56, 57, 17, 16, 15, 0)
5	1541.254	1060	(0, 18, 58, 98, 138, 178, 177, 176, 216, 256, 296, 336, 376, 416, 456, 496, 536, 537, 497, 457, 417, 377, 337, 297, 257, 217, 218, 258, 298, 338, 378, 418, 458, 498, 538, 539, 499, 459, 419, 379, 339, 299, 259, 219, 179, 139, 99, 59, 60, 20, 19, 0)
6	1574.207	1040	(0, 22, 21, 61, 101, 100, 140, 141, 181, 180, 220, 260, 300, 340, 380, 420, 460, 500, 540, 541, 501, 461, 421, 381, 341, 301, 261, 221, 222, 262, 302, 342, 382, 422, 462, 502, 542, 543, 503, 463, 423, 383, 343, 303, 263, 223, 183, 182, 142, 102, 62, 63, 23, 0)
7	1654.235	1180	(0, 24, 64, 104, 103, 143, 144, 184, 224, 264, 304, 344, 384, 424, 464, 504, 544, 545, 505, 465, 425, 385, 345, 305, 265, 225, 226, 266, 306, 346, 386, 426, 466, 506, 546, 547, 507, 467, 427, 387, 347, 307, 267, 227, 187, 186, 185, 145, 146, 147, 107, 106, 105, 65, 66, 67, 27, 26, 25, 0)
8	1654.235	1140	(0, 28, 68, 69, 70, 110, 109, 108, 148, 188, 228, 268, 308, 348, 388, 428, 468, 508, 548, 549, 509, 469, 429, 389, 349, 309, 269, 229, 230, 270, 310, 350, 390, 430, 470, 510, 550, 551, 511, 471, 431, 391, 351, 311, 271, 231, 191, 190, 189, 149, 150, 151, 152, 112, 111, 71, 31, 30, 29, 0)
9	1588.330	1100	(0, 34, 33, 32, 72, 73, 74, 114, 113, 153, 193, 192, 232, 272, 312, 352, 392, 432, 472, 512, 552, 553, 513, 473, 433, 393, 353, 313, 273, 233, 234, 274, 314, 354, 394, 434, 474, 514, 554, 555, 515, 475, 435, 395, 355, 315, 275, 235, 195, 194, 154, 155, 115, 75, 35, 0)
10	1635.405	1140	(0, 37, 36, 76, 77, 117, 116, 156, 157, 158, 198, 197, 196, 236, 276, 316, 356, 396, 436, 476, 516, 556, 557, 517, 477, 437, 397, 357, 317, 277, 237, 238, 278, 318, 358, 398, 438, 478, 518, 558, 559, 519, 479, 439, 399, 359, 319, 279, 239, 199, 159, 119, 118, 78, 79, 80, 40, 39, 38, 0)

Table B.53: Details of solution to problem Li-21

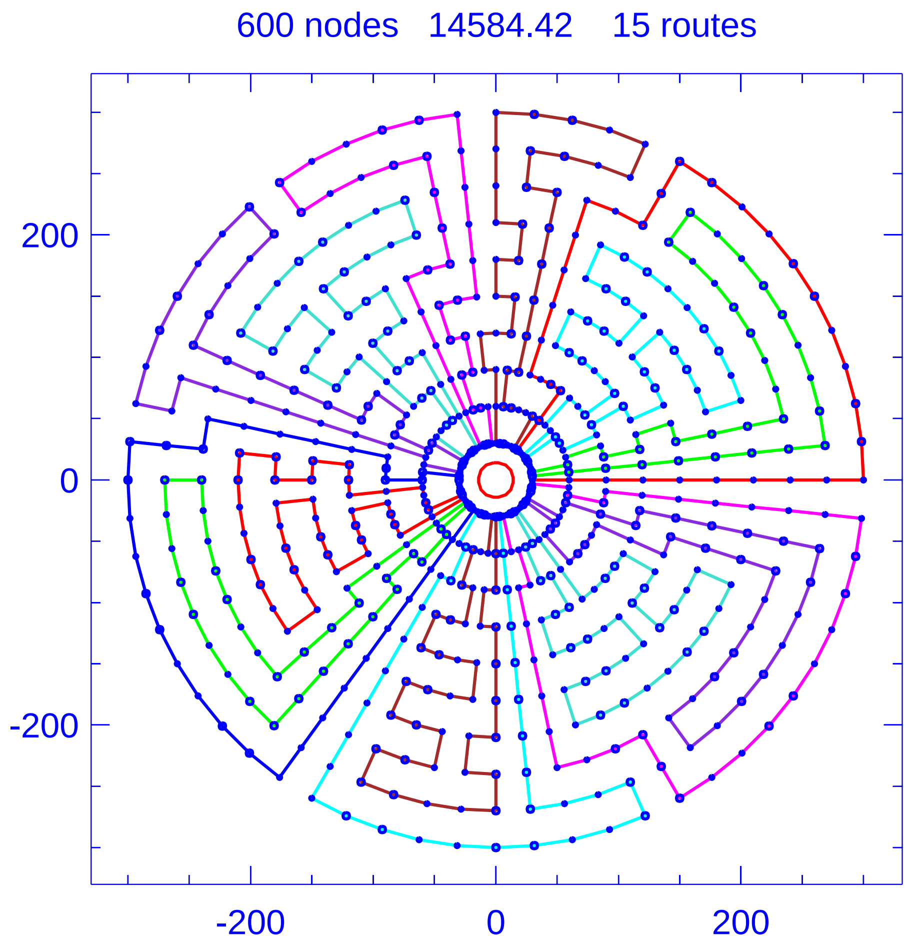


Figure B.49: Solution for problem Li-22

Problem			Li-22
Vehicle capacity			900
Maximum route length			1000.000
Number of nodes			600
Total route length			14584.422
Total number of routes			15
Route	Length	Load	Ordering
1	995.660	590	(0, 1, 61, 121, 181, 241, 301, 361, 421, 481, 541, 542, 543, 544, 545, 546, 547, 548, 549, 550, 551, 491, 431, 432, 433, 373, 313, 253, 193, 133, 132, 131, 130, 70, 10, 9, 0)
2	995.323	890	(0, 2, 62, 122, 182, 242, 302, 362, 422, 482, 483, 484, 485, 486, 487, 488, 489, 490, 430, 429, 428, 427, 426, 425, 424, 423, 363, 303, 243, 244, 184, 183, 123, 124, 64, 63, 3, 4, 5, 6, 0)
3	944.406	880	(0, 7, 67, 66, 65, 125, 126, 186, 185, 245, 246, 247, 248, 308, 307, 306, 305, 304, 364, 365, 366, 367, 368, 369, 370, 371, 372, 312, 311, 310, 309, 249, 250, 251, 252, 192, 191, 190, 189, 188, 187, 127, 128, 129, 69, 68, 8, 0)

Route	Length	Load	Ordering
4	973.679	860	(0, 15, 14, 13, 12, 11, 71, 72, 73, 74, 75, 135, 134, 194, 254, 314, 374, 434, 435, 495, 494, 493, 492, 552, 553, 554, 555, 556, 496, 436, 376, 375, 315, 316, 256, 255, 195, 196, 197, 137, 136, 76, 16, 0)
5	979.959	780	(0, 17, 77, 78, 79, 139, 138, 198, 199, 259, 258, 257, 317, 377, 437, 497, 557, 558, 559, 560, 561, 562, 502, 501, 500, 499, 498, 438, 378, 318, 319, 320, 260, 200, 140, 80, 20, 19, 18, 0)
6	998.126	900	(0, 23, 22, 21, 81, 141, 201, 202, 203, 263, 262, 261, 321, 322, 323, 383, 382, 381, 380, 379, 439, 440, 441, 442, 443, 444, 445, 446, 386, 385, 384, 324, 325, 326, 266, 265, 264, 204, 144, 143, 142, 82, 83, 84, 85, 25, 24, 0)
7	992.520	740	(0, 27, 26, 86, 87, 147, 146, 145, 205, 206, 207, 267, 327, 387, 447, 507, 506, 505, 504, 503, 563, 564, 565, 566, 567, 568, 569, 509, 508, 448, 388, 328, 268, 208, 148, 88, 89, 29, 28, 0)
8	970.538	650	(0, 31, 30, 90, 91, 151, 150, 149, 209, 269, 329, 389, 449, 450, 510, 570, 571, 572, 573, 574, 575, 576, 577, 578, 579, 580, 520, 460, 400, 340, 280, 220, 160, 100, 40, 39, 0)
9	884.743	890	(0, 32, 33, 34, 35, 95, 94, 93, 92, 152, 212, 211, 210, 270, 271, 331, 330, 390, 391, 392, 393, 394, 395, 396, 397, 337, 336, 335, 334, 333, 332, 272, 273, 274, 275, 276, 216, 215, 214, 213, 153, 154, 155, 156, 96, 36, 0)
10	985.902	820	(0, 37, 97, 157, 217, 277, 278, 338, 398, 458, 457, 456, 455, 454, 453, 452, 451, 511, 512, 513, 514, 515, 516, 517, 518, 519, 459, 399, 339, 279, 219, 218, 158, 159, 99, 98, 38, 0)
11	992.519	740	(0, 44, 43, 42, 41, 101, 102, 162, 161, 221, 281, 341, 401, 461, 521, 581, 582, 583, 584, 585, 586, 587, 588, 589, 590, 530, 529, 528, 527, 467, 407, 347, 287, 227, 167, 107, 47, 0)
12	967.061	860	(0, 45, 105, 104, 103, 163, 164, 224, 223, 222, 282, 283, 284, 285, 345, 344, 343, 342, 402, 403, 404, 464, 463, 462, 522, 523, 524, 525, 526, 466, 465, 405, 406, 346, 286, 226, 225, 165, 166, 106, 46, 0)
13	998.800	620	(0, 49, 48, 108, 109, 169, 168, 228, 288, 348, 408, 468, 469, 470, 471, 531, 591, 592, 593, 594, 595, 596, 597, 598, 599, 600, 540, 480, 420, 360, 300, 240, 180, 179, 119, 120, 60, 59, 58, 0)
14	919.285	880	(0, 50, 51, 111, 110, 170, 171, 231, 230, 229, 289, 290, 291, 292, 293, 353, 352, 351, 350, 349, 409, 410, 411, 412, 413, 414, 415, 416, 417, 357, 356, 355, 354, 294, 295, 296, 236, 235, 234, 233, 232, 172, 112, 52, 53, 0)
15	985.902	900	(0, 54, 55, 115, 114, 113, 173, 174, 175, 176, 177, 237, 297, 298, 358, 418, 478, 477, 476, 475, 474, 473, 472, 532, 533, 534, 535, 536, 537, 538, 539, 479, 419, 359, 299, 239, 238, 178, 118, 117, 116, 56, 57, 0)

Table B.54: Details of solution to problem Li-22

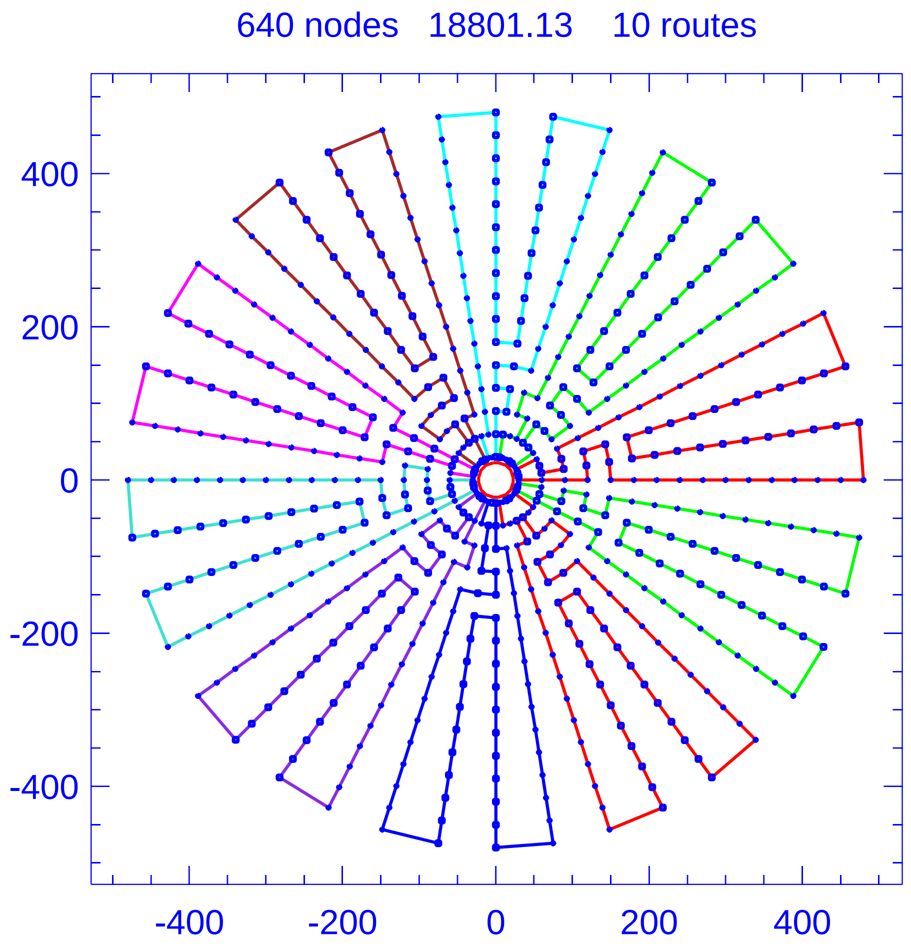


Figure B.50: Solution for problem Li-23

Problem			Li-23
Vehicle capacity			1400
Maximum route length			2200.000
Number of nodes			640
Total route length			18801.129
Total number of routes			10
Route	Length	Load	Ordering
1	1880.113	1280	(0, 3, 2, 1, 41, 81, 121, 122, 123, 163, 162, 161, 201, 241, 281, 321, 361, 401, 441, 481, 521, 561, 601, 602, 562, 522, 482, 442, 402, 362, 322, 282, 242, 202, 203, 243, 283, 323, 363, 403, 443, 483, 523, 563, 603, 604, 564, 524, 484, 444, 404, 364, 324, 284, 244, 204, 164, 124, 84, 83, 82, 42, 43, 44, 4, 0)
2	1941.311	1380	(0, 7, 6, 5, 45, 46, 47, 87, 86, 85, 125, 126, 127, 167, 166, 165, 205, 245, 285, 325, 365, 405, 445, 485, 525, 565, 605, 606, 566, 526, 486, 446, 406, 366, 326, 286, 246, 206, 207, 247, 287, 327, 367, 407, 447, 487, 527, 567, 607, 608, 568, 528, 488, 448, 408, 368, 328, 288, 248, 208, 168, 128, 129, 89, 88, 48, 49, 50, 10, 9, 8, 0)

Route	Length	Load	Ordering
3	1833.037	1200	(0, 12, 11, 51, 91, 90, 130, 131, 171, 170, 169, 209, 249, 289, 329, 369, 409, 449, 489, 529, 569, 609, 610, 570, 530, 490, 450, 410, 370, 330, 290, 250, 210, 211, 251, 291, 331, 371, 411, 451, 491, 531, 571, 611, 612, 572, 532, 492, 452, 412, 372, 332, 292, 252, 212, 172, 132, 92, 52, 53, 13, 0)
4	1913.066	1300	(0, 14, 54, 94, 93, 133, 173, 213, 253, 293, 333, 373, 413, 453, 493, 533, 573, 613, 614, 574, 534, 494, 454, 414, 374, 334, 294, 254, 214, 215, 255, 295, 335, 375, 415, 455, 495, 535, 575, 615, 616, 576, 536, 496, 456, 416, 376, 336, 296, 256, 216, 176, 175, 174, 134, 135, 136, 137, 97, 96, 95, 55, 56, 57, 17, 16, 15, 0)
5	1800.085	1220	(0, 18, 58, 98, 138, 178, 177, 217, 257, 297, 337, 377, 417, 457, 497, 537, 577, 617, 618, 578, 538, 498, 458, 418, 378, 338, 298, 258, 218, 219, 259, 299, 339, 379, 419, 459, 499, 539, 579, 619, 620, 580, 540, 500, 460, 420, 380, 340, 300, 260, 220, 180, 179, 139, 99, 59, 60, 20, 19, 0)
6	1913.066	1300	(0, 23, 22, 21, 61, 62, 63, 103, 102, 101, 100, 140, 141, 142, 143, 183, 182, 181, 221, 261, 301, 341, 381, 421, 461, 501, 541, 581, 621, 622, 582, 542, 502, 462, 422, 382, 342, 302, 262, 222, 223, 263, 303, 343, 383, 423, 463, 503, 543, 583, 623, 624, 584, 544, 504, 464, 424, 384, 344, 304, 264, 224, 184, 144, 104, 64, 24, 0)
7	1913.066	1300	(0, 27, 26, 25, 65, 66, 67, 107, 106, 105, 145, 146, 147, 187, 186, 185, 225, 265, 305, 345, 385, 425, 465, 505, 545, 585, 625, 626, 586, 546, 506, 466, 426, 386, 346, 306, 266, 226, 227, 267, 307, 347, 387, 427, 467, 507, 547, 587, 627, 628, 588, 548, 508, 468, 428, 388, 348, 308, 268, 228, 188, 148, 149, 109, 108, 68, 28, 0)
8	1833.037	1240	(0, 29, 69, 70, 110, 150, 151, 191, 190, 189, 229, 269, 309, 349, 389, 429, 469, 509, 549, 589, 629, 630, 590, 550, 510, 470, 430, 390, 350, 310, 270, 230, 231, 271, 311, 351, 391, 431, 471, 511, 551, 591, 631, 632, 592, 552, 512, 472, 432, 392, 352, 312, 272, 232, 192, 152, 112, 111, 71, 31, 30, 0)
9	1941.311	1340	(0, 34, 33, 32, 72, 73, 74, 114, 113, 153, 193, 233, 273, 313, 353, 393, 433, 473, 513, 553, 593, 633, 634, 594, 554, 514, 474, 434, 394, 354, 314, 274, 234, 235, 275, 315, 355, 395, 435, 475, 515, 555, 595, 635, 636, 596, 556, 516, 476, 436, 396, 356, 316, 276, 236, 196, 195, 194, 154, 155, 156, 157, 117, 116, 115, 75, 76, 77, 37, 36, 35, 0)
10	1833.037	1240	(0, 38, 78, 118, 158, 198, 197, 237, 277, 317, 357, 397, 437, 477, 517, 557, 597, 637, 638, 598, 558, 518, 478, 438, 398, 358, 318, 278, 238, 239, 279, 319, 359, 399, 439, 479, 519, 559, 599, 639, 640, 600, 560, 520, 480, 440, 400, 360, 320, 280, 240, 200, 199, 159, 160, 120, 119, 79, 80, 40, 39, 0)

Table B.55: Details of solution to problem Li-23

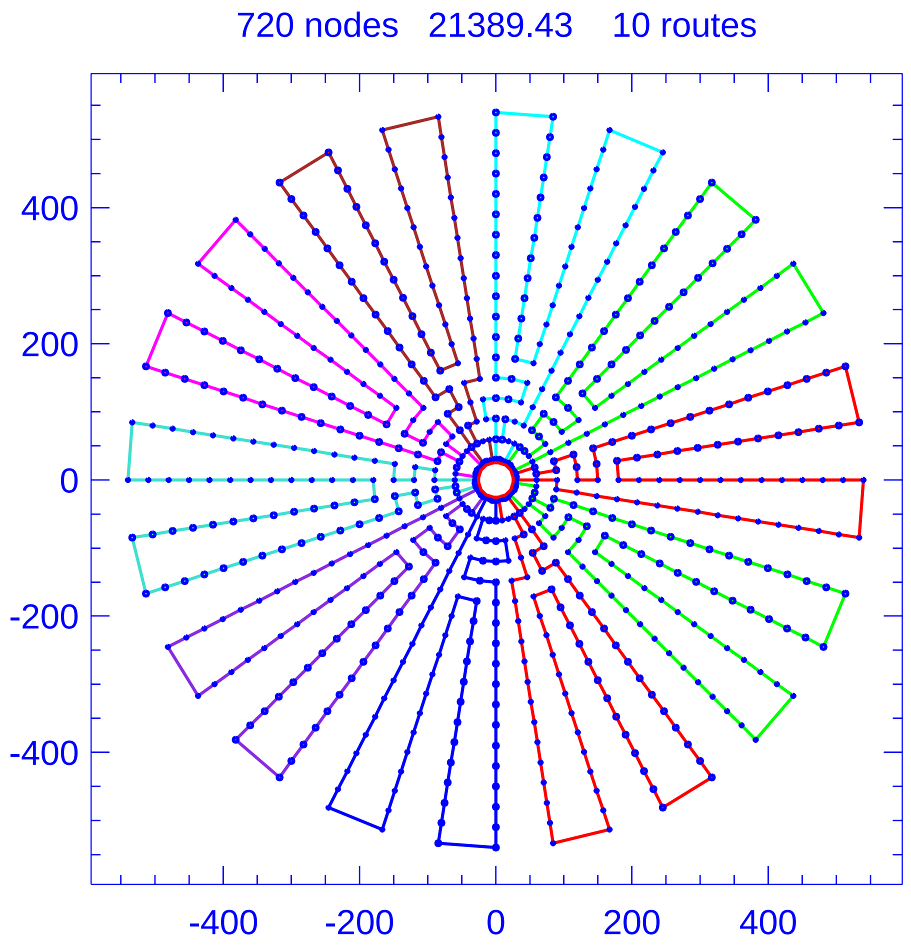


Figure B.51: Solution for problem Li-24

Problem			Li-24
Vehicle capacity Maximum route length Number of nodes Total route length Total number of routes			1500 2400.000 720 21389.430 10
Route	Length	Load	Ordering
1	2124.820	1420	(0, 2, 1, 41, 81, 120, 160, 200, 240, 280, 320, 360, 400, 440, 480, 520, 560, 600, 640, 680, 720, 681, 641, 601, 561, 521, 481, 441, 401, 361, 321, 281, 241, 201, 202, 242, 282, 322, 362, 402, 442, 482, 522, 562, 602, 642, 682, 683, 643, 603, 563, 523, 483, 443, 403, 363, 323, 283, 243, 203, 163, 162, 161, 121, 122, 123, 83, 82, 42, 43, 3, 0)
2	2138.943	1440	(0, 4, 44, 84, 124, 164, 204, 244, 284, 324, 364, 404, 444, 484, 524, 564, 604, 644, 684, 685, 645, 605, 565, 525, 485, 445, 405, 365, 325, 285, 245, 205, 206, 246, 286, 326, 366, 406, 446, 486, 526, 566, 606, 646, 686, 687, 647, 607, 567, 527, 487, 447, 407, 367, 327, 287, 247, 207, 167, 166, 165, 125, 126, 127, 87, 86, 85, 45, 46, 47, 7, 6, 5, 0)

Route	Length	Load	Ordering
3	2171.896	1460	(0, 8, 48, 49, 50, 90, 89, 88, 128, 168, 208, 248, 288, 328, 368, 408, 448, 488, 528, 568, 608, 648, 688, 689, 649, 609, 569, 529, 489, 449, 409, 369, 329, 289, 249, 209, 210, 250, 290, 330, 370, 410, 450, 490, 530, 570, 610, 650, 690, 691, 651, 611, 571, 531, 491, 451, 411, 371, 331, 291, 251, 211, 171, 170, 169, 129, 130, 131, 132, 92, 91, 51, 11, 10, 9, 0)
4	2105.990	1420	(0, 14, 13, 12, 52, 53, 54, 94, 93, 133, 173, 172, 212, 252, 292, 332, 372, 412, 452, 492, 532, 572, 612, 652, 692, 693, 653, 613, 573, 533, 493, 453, 413, 373, 333, 293, 253, 213, 214, 254, 294, 334, 374, 414, 454, 494, 534, 574, 614, 654, 694, 695, 655, 615, 575, 535, 495, 455, 415, 375, 335, 295, 255, 215, 175, 174, 134, 135, 95, 55, 15, 0)
5	2153.066	1460	(0, 17, 16, 56, 57, 97, 96, 136, 137, 138, 178, 177, 176, 216, 256, 296, 336, 376, 416, 456, 496, 536, 576, 616, 656, 696, 697, 657, 617, 577, 537, 497, 457, 417, 377, 337, 297, 257, 217, 218, 258, 298, 338, 378, 418, 458, 498, 538, 578, 618, 658, 698, 699, 659, 619, 579, 539, 499, 459, 419, 379, 339, 299, 259, 219, 179, 139, 99, 98, 58, 59, 60, 20, 19, 18, 0)
6	2124.820	1420	(0, 22, 21, 61, 101, 100, 140, 141, 181, 180, 220, 260, 300, 340, 380, 420, 460, 500, 540, 580, 620, 660, 700, 701, 661, 621, 581, 541, 501, 461, 421, 381, 341, 301, 261, 221, 222, 262, 302, 342, 382, 422, 462, 502, 542, 582, 622, 662, 702, 703, 663, 623, 583, 543, 503, 463, 423, 383, 343, 303, 263, 223, 183, 182, 142, 143, 103, 102, 62, 63, 23, 0)
7	2138.943	1440	(0, 24, 64, 104, 144, 184, 224, 264, 304, 344, 384, 424, 464, 504, 544, 584, 624, 664, 704, 705, 665, 625, 585, 545, 505, 465, 425, 385, 345, 305, 265, 225, 226, 266, 306, 346, 386, 426, 466, 506, 546, 586, 626, 666, 706, 707, 667, 627, 587, 547, 507, 467, 427, 387, 347, 307, 267, 227, 187, 186, 185, 145, 146, 147, 107, 106, 105, 65, 66, 67, 27, 26, 25, 0)
8	2171.896	1460	(0, 28, 68, 108, 148, 188, 228, 268, 308, 348, 388, 428, 468, 508, 548, 588, 628, 668, 708, 709, 669, 629, 589, 549, 509, 469, 429, 389, 349, 309, 269, 229, 230, 270, 310, 350, 390, 430, 470, 510, 550, 590, 630, 670, 710, 711, 671, 631, 591, 551, 511, 471, 431, 391, 351, 311, 271, 231, 191, 190, 189, 149, 150, 151, 152, 112, 111, 110, 109, 69, 70, 71, 31, 30, 29, 0)
9	2105.990	1420	(0, 34, 33, 32, 72, 73, 74, 114, 113, 153, 193, 192, 232, 272, 312, 352, 392, 432, 472, 512, 552, 592, 632, 672, 712, 713, 673, 633, 593, 553, 513, 473, 433, 393, 353, 313, 273, 233, 234, 274, 314, 354, 394, 434, 474, 514, 554, 594, 634, 674, 714, 715, 675, 635, 595, 555, 515, 475, 435, 395, 355, 315, 275, 235, 195, 194, 154, 155, 115, 75, 35, 0)
10	2153.066	1460	(0, 37, 36, 76, 77, 117, 116, 156, 157, 158, 198, 197, 196, 236, 276, 316, 356, 396, 436, 476, 516, 556, 596, 636, 676, 716, 717, 677, 637, 597, 557, 517, 477, 437, 397, 357, 317, 277, 237, 238, 278, 318, 358, 398, 438, 478, 518, 558, 598, 638, 678, 718, 719, 679, 639, 599, 559, 519, 479, 439, 399, 359, 319, 279, 239, 199, 159, 119, 118, 78, 79, 80, 40, 39, 38, 0)

Table B.56: Details of solution to problem Li-24

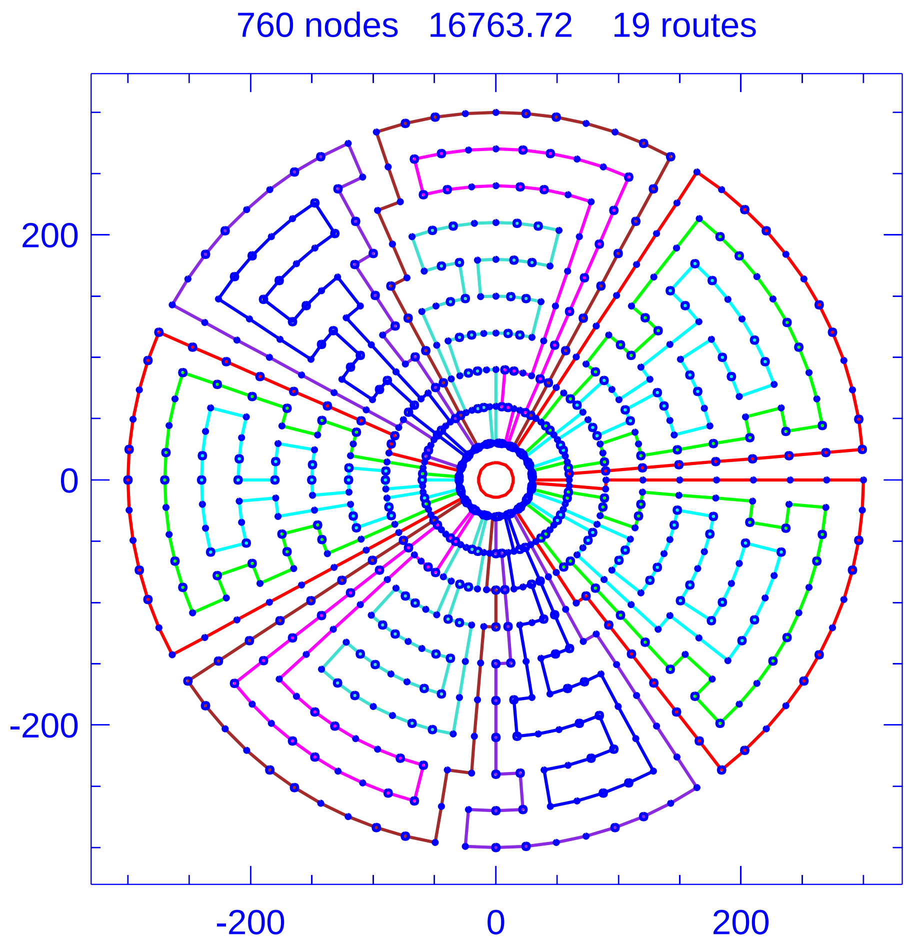


Figure B.52: Solution for problem Li-25

Problem			Li-25
Vehicle capacity			900
Maximum route length			900.000
Number of nodes			760
Total route length			16763.719
Total number of routes			19
Route	Length	Load	Ordering
1	887.622	690	(0, 3, 2, 1, 77, 78, 154, 230, 306, 382, 458, 534, 610, 686, 687, 688, 689, 690, 691, 692, 693, 694, 695, 696, 697, 621, 545, 469, 393, 317, 241, 165, 89, 13, 12, 11, 0)
2	899.527	830	(0, 4, 80, 79, 155, 156, 157, 233, 232, 231, 307, 383, 459, 460, 536, 535, 611, 612, 613, 614, 615, 616, 617, 618, 619, 620, 544, 468, 392, 391, 390, 314, 315, 316, 240, 239, 163, 164, 88, 87, 86, 10, 0)
3	869.281	900	(0, 8, 7, 6, 5, 81, 82, 83, 84, 160, 159, 158, 234, 235, 311, 310, 309, 308, 384, 385, 386, 387, 388, 464, 463, 462, 461, 537, 538, 539, 540, 541, 542, 543, 467, 466, 465, 389, 313, 312, 236, 237, 238, 162, 161, 85, 9, 0)

Route	Length	Load	Ordering
4	895.060	840	(0, 15, 14, 90, 166, 242, 318, 394, 470, 546, 622, 698, 699, 700, 701, 702, 703, 704, 705, 706, 707, 708, 632, 556, 557, 481, 405, 406, 330, 254, 178, 102, 26, 25, 24, 23, 22, 0)
5	899.527	840	(0, 16, 92, 91, 167, 243, 319, 395, 471, 547, 623, 624, 625, 626, 627, 628, 629, 630, 631, 555, 554, 553, 552, 551, 550, 549, 548, 472, 396, 320, 244, 168, 169, 170, 171, 95, 94, 93, 17, 18, 19, 0)
6	836.556	880	(0, 20, 96, 172, 173, 174, 175, 176, 252, 251, 250, 249, 248, 247, 246, 245, 321, 322, 323, 324, 325, 401, 400, 399, 398, 397, 473, 474, 475, 476, 477, 478, 479, 480, 404, 403, 402, 326, 327, 328, 329, 253, 177, 101, 100, 99, 98, 97, 21, 0)
7	895.060	760	(0, 29, 28, 27, 103, 179, 255, 256, 332, 331, 407, 483, 482, 558, 634, 633, 709, 710, 711, 712, 713, 714, 715, 716, 717, 641, 565, 489, 413, 337, 261, 185, 184, 108, 109, 110, 111, 35, 34, 33, 32, 0)
8	897.047	800	(0, 30, 106, 105, 104, 180, 181, 257, 333, 409, 408, 484, 485, 486, 487, 563, 562, 561, 560, 559, 635, 636, 637, 638, 639, 640, 564, 488, 412, 411, 410, 334, 335, 336, 260, 259, 258, 182, 183, 107, 31, 0)
9	890.101	610	(0, 36, 112, 188, 187, 186, 262, 338, 414, 490, 566, 642, 718, 719, 720, 721, 722, 723, 724, 725, 726, 727, 728, 729, 653, 577, 501, 425, 349, 273, 197, 121, 45, 44, 0)
10	892.089	790	(0, 37, 38, 114, 113, 189, 265, 264, 263, 339, 340, 416, 415, 491, 567, 643, 644, 645, 646, 647, 648, 649, 650, 651, 652, 576, 575, 499, 500, 424, 423, 422, 346, 347, 348, 272, 196, 120, 119, 43, 0)
11	851.924	880	(0, 41, 40, 39, 115, 116, 192, 191, 190, 266, 267, 268, 344, 343, 342, 341, 417, 418, 419, 495, 494, 493, 492, 568, 569, 570, 571, 572, 573, 574, 498, 497, 496, 420, 421, 345, 269, 270, 271, 195, 194, 193, 117, 118, 42, 0)
12	892.581	710	(0, 48, 47, 46, 122, 198, 274, 350, 426, 502, 578, 654, 730, 731, 732, 733, 734, 735, 736, 737, 738, 739, 740, 664, 588, 589, 513, 437, 361, 285, 286, 210, 209, 133, 57, 56, 0)
13	897.048	830	(0, 49, 50, 126, 125, 124, 123, 199, 275, 351, 427, 503, 579, 655, 656, 657, 658, 659, 660, 661, 662, 663, 587, 586, 585, 584, 583, 582, 581, 580, 504, 428, 352, 276, 200, 201, 202, 203, 127, 51, 0)
14	806.801	900	(0, 52, 53, 54, 130, 129, 128, 204, 205, 281, 280, 279, 278, 277, 353, 354, 355, 356, 357, 358, 359, 435, 434, 433, 432, 431, 430, 429, 505, 506, 507, 508, 509, 510, 511, 512, 436, 360, 284, 283, 282, 206, 207, 208, 132, 131, 55, 0)
15	895.060	700	(0, 58, 134, 135, 211, 287, 363, 362, 438, 514, 590, 591, 667, 666, 665, 741, 742, 743, 744, 745, 746, 747, 748, 749, 673, 597, 521, 445, 369, 368, 292, 216, 140, 64, 63, 0)
16	892.088	820	(0, 59, 60, 61, 137, 136, 212, 213, 214, 290, 289, 288, 364, 440, 439, 515, 516, 517, 518, 519, 595, 594, 593, 592, 668, 669, 670, 671, 672, 596, 520, 444, 443, 442, 441, 365, 366, 367, 291, 215, 139, 138, 62, 0)
17	897.540	650	(0, 67, 66, 65, 141, 217, 293, 294, 370, 446, 522, 598, 674, 750, 751, 752, 753, 754, 755, 756, 757, 758, 759, 760, 685, 609, 533, 457, 381, 305, 229, 153, 228, 152, 76, 75, 0)
18	889.609	870	(0, 68, 69, 145, 144, 143, 142, 218, 219, 295, 371, 447, 523, 524, 600, 599, 675, 676, 677, 678, 679, 680, 681, 682, 683, 684, 608, 607, 531, 532, 456, 380, 304, 303, 302, 301, 225, 226, 227, 151, 150, 74, 0)

Route	Length	Load	Ordering
19	879.199	900	(0, 70, 146, 147, 223, 222, 221, 220, 296, 372, 448, 449, 525, 601, 602, 603, 604, 605, 606, 530, 529, 528, 527, 526, 450, 451, 452, 453, 454, 455, 379, 378, 377, 376, 375, 374, 373, 297, 298, 299, 300, 224, 148, 149, 73, 72, 71, 0)

Table B.57: Details of solution to problem Li-25

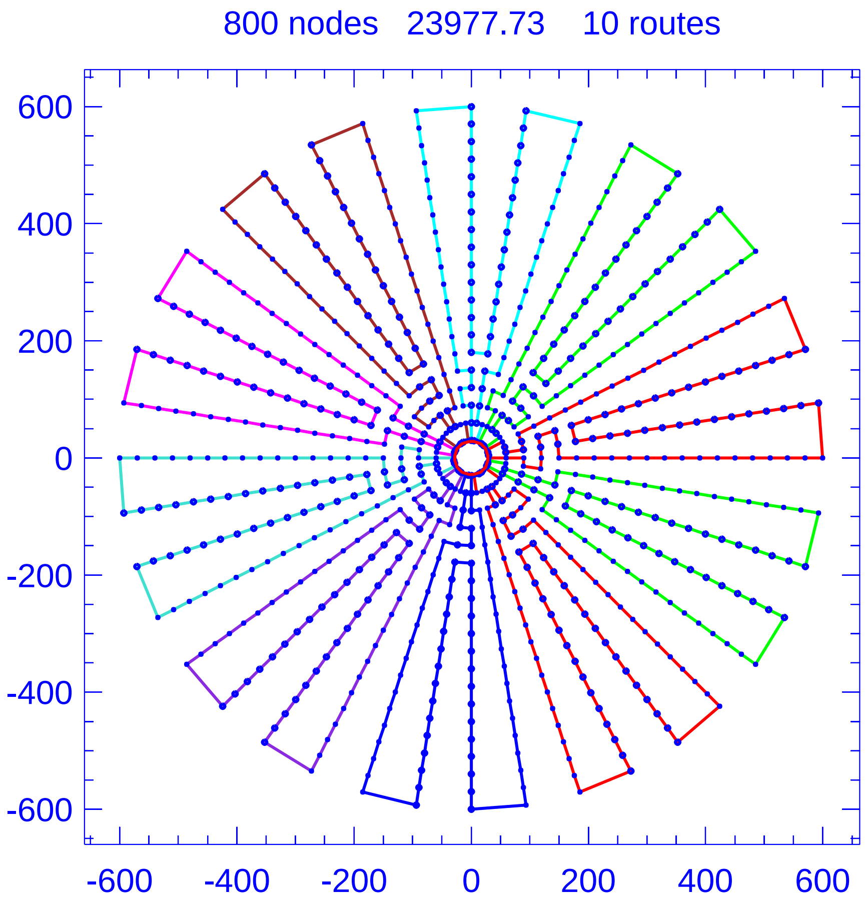


Figure B.53: Solution for problem Li-26

Problem			Li-26
Vehicle capacity Maximum route length Number of nodes Total route length Total number of routes			1700 2500.000 800 23977.732 10
Route	Length	Load	Ordering
1	2430.726	1620	(0, 3, 2, 1, 41, 81, 120, 160, 121, 122, 123, 163, 162, 161, 201, 241, 281, 321, 361, 401, 441, 481, 521, 561, 601, 641, 681, 721, 761, 762, 722, 682, 642, 602, 562, 522, 482, 442, 402, 362, 322, 282, 242, 202, 203, 243, 283, 323, 363, 403, 443, 483, 523, 563, 603, 643, 683, 723, 763, 764, 724, 684, 644, 604, 564, 524, 484, 444, 404, 364, 324, 284, 244, 204, 164, 124, 84, 83, 82, 42, 43, 44, 4, 0)
2	2430.726	1620	(0, 7, 6, 5, 45, 46, 47, 87, 86, 85, 125, 126, 127, 167, 166, 165, 205, 245, 285, 325, 365, 405, 445, 485, 525, 565, 605, 645, 685, 725, 765, 766, 726, 686, 646, 606, 566, 526, 486, 446, 406, 366, 326, 286, 246, 206, 207, 247, 287, 327, 367, 407, 447, 487, 527, 567, 607, 647, 687, 727, 767, 768, 728, 688, 648, 608, 568, 528, 488, 448, 408, 368, 328, 288, 248, 208, 168, 128, 129, 89, 88, 48, 8, 0)

Route	Length	Load	Ordering
3	2350.698	1560	(0, 9, 49, 50, 90, 130, 170, 169, 209, 249, 289, 329, 369, 409, 449, 489, 529, 569, 609, 649, 689, 729, 769, 770, 730, 690, 650, 610, 570, 530, 490, 450, 410, 370, 330, 290, 250, 210, 211, 251, 291, 331, 371, 411, 451, 491, 531, 571, 611, 651, 691, 731, 771, 772, 732, 692, 652, 612, 572, 532, 492, 452, 412, 372, 332, 292, 252, 212, 172, 171, 131, 132, 92, 91, 51, 11, 10, 0)
4	2458.971	1660	(0, 14, 13, 12, 52, 53, 54, 94, 93, 133, 173, 213, 253, 293, 333, 373, 413, 453, 493, 533, 573, 613, 653, 693, 733, 773, 774, 734, 694, 654, 614, 574, 534, 494, 454, 414, 374, 334, 294, 254, 214, 215, 255, 295, 335, 375, 415, 455, 495, 535, 575, 615, 655, 695, 735, 775, 776, 736, 696, 656, 616, 576, 536, 496, 456, 416, 376, 336, 296, 256, 216, 176, 175, 174, 134, 135, 136, 137, 97, 96, 95, 55, 56, 57, 17, 16, 15, 0)
5	2317.745	1540	(0, 18, 58, 98, 138, 178, 177, 217, 257, 297, 337, 377, 417, 457, 497, 537, 577, 617, 657, 697, 737, 777, 778, 738, 698, 658, 618, 578, 538, 498, 458, 418, 378, 338, 298, 258, 218, 219, 259, 299, 339, 379, 419, 459, 499, 539, 579, 619, 659, 699, 739, 779, 780, 740, 700, 660, 620, 580, 540, 500, 460, 420, 380, 340, 300, 260, 220, 180, 179, 139, 99, 59, 60, 20, 19, 0)
6	2430.726	1620	(0, 23, 22, 21, 61, 101, 100, 140, 141, 142, 143, 183, 182, 181, 221, 261, 301, 341, 381, 421, 461, 501, 541, 581, 621, 661, 701, 741, 781, 782, 742, 702, 662, 622, 582, 542, 502, 462, 422, 382, 342, 302, 262, 222, 223, 263, 303, 343, 383, 423, 463, 503, 543, 583, 623, 663, 703, 743, 783, 784, 744, 704, 664, 624, 584, 544, 504, 464, 424, 384, 344, 304, 264, 224, 184, 144, 104, 103, 102, 62, 63, 64, 24, 0)
7	2430.726	1620	(0, 27, 26, 25, 65, 66, 67, 107, 106, 105, 145, 146, 147, 187, 186, 185, 225, 265, 305, 345, 385, 425, 465, 505, 545, 585, 625, 665, 705, 745, 785, 786, 746, 706, 666, 626, 586, 546, 506, 466, 426, 386, 346, 306, 266, 226, 227, 267, 307, 347, 387, 427, 467, 507, 547, 587, 627, 667, 707, 747, 787, 788, 748, 708, 668, 628, 588, 548, 508, 468, 428, 388, 348, 308, 268, 228, 188, 148, 149, 109, 108, 68, 28, 0)
8	2350.698	1560	(0, 29, 69, 70, 110, 150, 151, 191, 190, 189, 229, 269, 309, 349, 389, 429, 469, 509, 549, 589, 629, 669, 709, 749, 789, 790, 750, 710, 670, 630, 590, 550, 510, 470, 430, 390, 350, 310, 270, 230, 231, 271, 311, 351, 391, 431, 471, 511, 551, 591, 631, 671, 711, 751, 791, 792, 752, 712, 672, 632, 592, 552, 512, 472, 432, 392, 352, 312, 272, 232, 192, 152, 112, 111, 71, 31, 30, 0)
9	2458.971	1660	(0, 34, 33, 32, 72, 73, 74, 114, 113, 153, 193, 233, 273, 313, 353, 393, 433, 473, 513, 553, 593, 633, 673, 713, 753, 793, 794, 754, 714, 674, 634, 594, 554, 514, 474, 434, 394, 354, 314, 274, 234, 235, 275, 315, 355, 395, 435, 475, 515, 555, 595, 635, 675, 715, 755, 795, 796, 756, 716, 676, 636, 596, 556, 516, 476, 436, 396, 356, 316, 276, 236, 196, 195, 194, 154, 155, 156, 157, 117, 116, 115, 75, 76, 77, 37, 36, 35, 0)
10	2317.745	1540	(0, 38, 78, 118, 158, 198, 197, 237, 277, 317, 357, 397, 437, 477, 517, 557, 597, 637, 677, 717, 757, 797, 798, 758, 718, 678, 638, 598, 558, 518, 478, 438, 398, 358, 318, 278, 238, 239, 279, 319, 359, 399, 439, 479, 519, 559, 599, 639, 679, 719, 759, 799, 800, 760, 720, 680, 640, 600, 560, 520, 480, 440, 400, 360, 320, 280, 240, 200, 199, 159, 119, 79, 80, 40, 39, 0)

Table B.58: Details of solution to problem Li-26

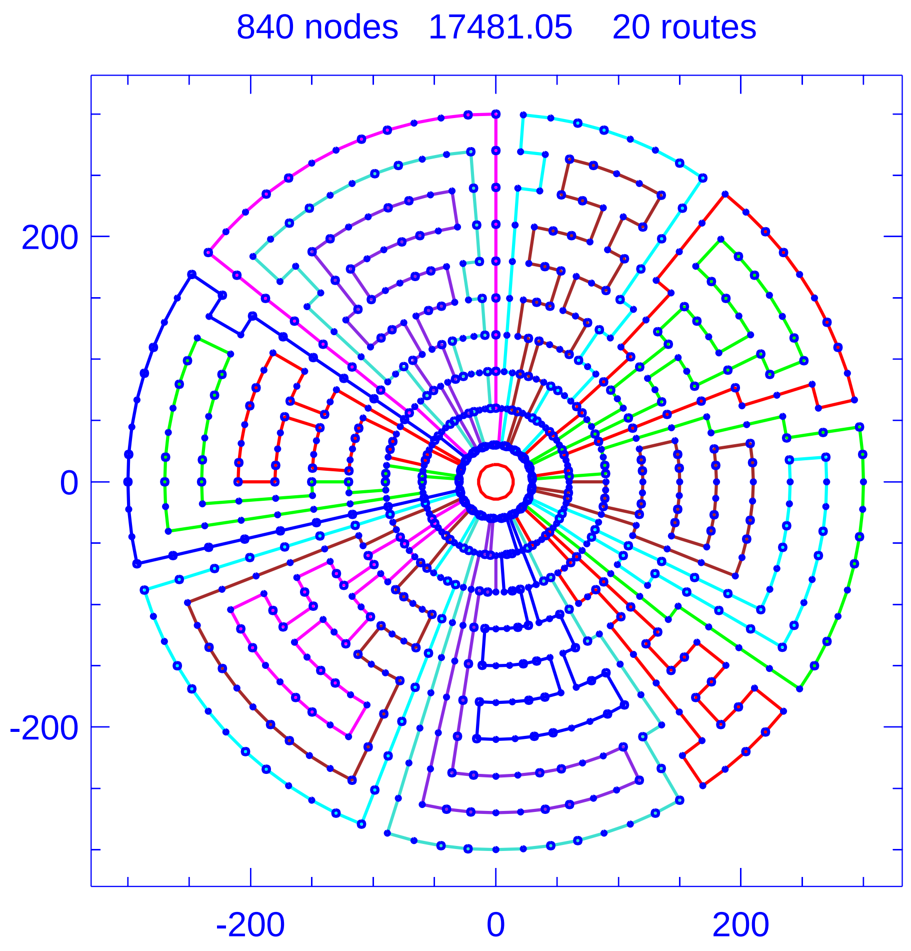


Figure B.54: Solution for problem Li-27

Problem			Li-27
Vehicle capacity			900
Maximum route length			900.000
Number of nodes			840
Total route length			17481.053
Total number of routes			20
Route	Length	Load	Ordering
1	893.894	790	(0, 4, 3, 87, 88, 89, 90, 174, 258, 342, 426, 510, 509, 593, 677, 676, 760, 761, 762, 763, 764, 765, 766, 767, 768, 769, 685, 601, 517, 516, 432, 348, 347, 263, 179, 178, 94, 95, 11, 10, 0)
2	896.712	870	(0, 5, 6, 7, 91, 175, 259, 343, 344, 345, 429, 428, 427, 511, 595, 594, 678, 679, 680, 681, 682, 683, 684, 600, 599, 598, 597, 596, 512, 513, 514, 515, 431, 430, 346, 262, 261, 260, 176, 177, 93, 92, 8, 9, 0)

Route	Length	Load	Ordering
3	889.408	780	(0, 16, 15, 14, 13, 12, 96, 97, 98, 99, 183, 182, 181, 180, 264, 265, 266, 350, 349, 433, 434, 518, 602, 686, 770, 771, 772, 773, 774, 775, 776, 777, 693, 692, 608, 609, 525, 441, 357, 273, 189, 188, 104, 20, 19, 0)
4	894.468	900	(0, 17, 101, 100, 184, 185, 186, 270, 269, 268, 267, 351, 352, 353, 437, 436, 435, 519, 520, 604, 603, 687, 688, 689, 690, 691, 607, 606, 605, 521, 522, 523, 524, 440, 439, 438, 354, 355, 356, 272, 271, 187, 103, 102, 18, 0)
5	889.408	870	(0, 23, 22, 21, 105, 106, 190, 274, 358, 442, 526, 610, 694, 778, 779, 780, 781, 782, 783, 784, 785, 786, 787, 788, 789, 790, 706, 622, 538, 454, 370, 286, 202, 201, 117, 33, 32, 31, 0)
6	898.955	890	(0, 24, 25, 26, 110, 109, 108, 107, 191, 192, 193, 194, 278, 277, 276, 275, 359, 360, 444, 443, 527, 611, 695, 696, 697, 698, 699, 700, 701, 702, 703, 704, 705, 621, 620, 536, 537, 453, 369, 285, 284, 283, 199, 200, 116, 115, 114, 30, 0)
7	886.068	900	(0, 27, 111, 112, 196, 195, 279, 280, 364, 363, 362, 361, 445, 446, 447, 448, 449, 450, 534, 533, 532, 531, 530, 529, 528, 612, 613, 614, 615, 616, 617, 618, 619, 535, 451, 452, 368, 367, 366, 365, 281, 282, 198, 197, 113, 29, 28, 0)
8	891.651	840	(0, 34, 118, 119, 203, 287, 371, 455, 539, 623, 624, 708, 707, 791, 792, 793, 794, 795, 796, 797, 798, 799, 800, 801, 802, 718, 634, 550, 466, 382, 298, 214, 130, 46, 45, 0)
9	752.033	870	(0, 35, 36, 120, 204, 288, 372, 373, 374, 458, 457, 456, 540, 541, 542, 543, 544, 545, 546, 547, 463, 462, 461, 460, 459, 375, 376, 377, 378, 294, 293, 292, 291, 290, 289, 205, 206, 207, 208, 124, 123, 122, 121, 37, 38, 39, 0)
10	887.738	750	(0, 40, 41, 125, 209, 210, 211, 212, 296, 295, 379, 380, 464, 548, 632, 631, 630, 629, 628, 627, 626, 625, 709, 710, 711, 712, 713, 714, 715, 716, 717, 633, 549, 465, 381, 297, 213, 129, 128, 127, 126, 42, 43, 44, 0)
11	898.381	850	(0, 48, 47, 131, 215, 299, 383, 467, 551, 635, 719, 803, 804, 805, 806, 807, 808, 809, 810, 811, 812, 813, 814, 815, 731, 647, 563, 479, 395, 311, 227, 226, 225, 224, 140, 141, 57, 56, 0)
12	885.494	830	(0, 49, 133, 132, 216, 217, 301, 300, 384, 468, 552, 636, 720, 721, 722, 723, 724, 725, 726, 727, 728, 729, 730, 646, 562, 478, 477, 476, 475, 391, 392, 393, 394, 310, 309, 308, 307, 306, 222, 223, 139, 138, 54, 55, 0)
13	877.094	900	(0, 51, 50, 134, 135, 219, 218, 302, 303, 387, 386, 385, 469, 470, 471, 555, 554, 553, 637, 638, 639, 640, 641, 642, 643, 644, 645, 561, 560, 559, 558, 557, 556, 472, 473, 474, 390, 389, 388, 304, 305, 221, 220, 136, 137, 53, 52, 0)
14	896.138	730	(0, 60, 59, 58, 142, 143, 144, 228, 312, 396, 480, 564, 648, 732, 816, 817, 818, 819, 820, 821, 822, 823, 824, 825, 826, 827, 743, 659, 660, 576, 492, 408, 407, 323, 239, 238, 154, 70, 0)
15	898.955	870	(0, 61, 62, 146, 145, 229, 313, 397, 481, 565, 649, 733, 734, 735, 736, 737, 738, 739, 740, 741, 742, 658, 657, 656, 655, 654, 653, 652, 651, 650, 566, 482, 398, 314, 230, 231, 232, 148, 147, 63, 64, 65, 66, 67, 0)
16	794.659	900	(0, 68, 152, 151, 150, 149, 233, 234, 235, 319, 318, 317, 316, 315, 399, 400, 401, 402, 403, 404, 488, 487, 486, 485, 484, 483, 567, 568, 569, 570, 571, 572, 573, 574, 575, 491, 490, 489, 405, 406, 322, 321, 320, 236, 237, 153, 69, 0)

Route	Length	Load	Ordering
17	893.894	860	(0, 72, 71, 155, 156, 157, 158, 242, 241, 240, 324, 325, 326, 410, 409, 493, 577, 661, 745, 744, 828, 829, 830, 831, 832, 748, 747, 746, 662, 663, 664, 580, 579, 578, 494, 495, 411, 327, 243, 159, 75, 74, 73, 0)
18	893.895	610	(0, 76, 160, 244, 328, 412, 496, 497, 581, 665, 749, 833, 834, 835, 836, 837, 838, 839, 840, 757, 758, 759, 675, 591, 592, 508, 424, 425, 341, 257, 173, 172, 171, 170, 86, 2, 1, 84, 0)
19	876.520	890	(0, 77, 78, 162, 161, 245, 246, 330, 329, 413, 414, 498, 582, 666, 750, 751, 752, 753, 754, 755, 756, 673, 674, 590, 589, 672, 671, 670, 669, 668, 667, 583, 499, 415, 331, 247, 248, 164, 163, 79, 80, 81, 0)
20	785.686	900	(0, 82, 166, 165, 249, 333, 332, 416, 500, 584, 585, 586, 587, 588, 505, 506, 507, 423, 422, 421, 504, 503, 502, 501, 417, 418, 419, 420, 337, 338, 339, 340, 256, 255, 254, 253, 336, 335, 334, 250, 251, 252, 169, 85, 168, 167, 83, 0)

Table B.59: Details of solution to problem Li-27

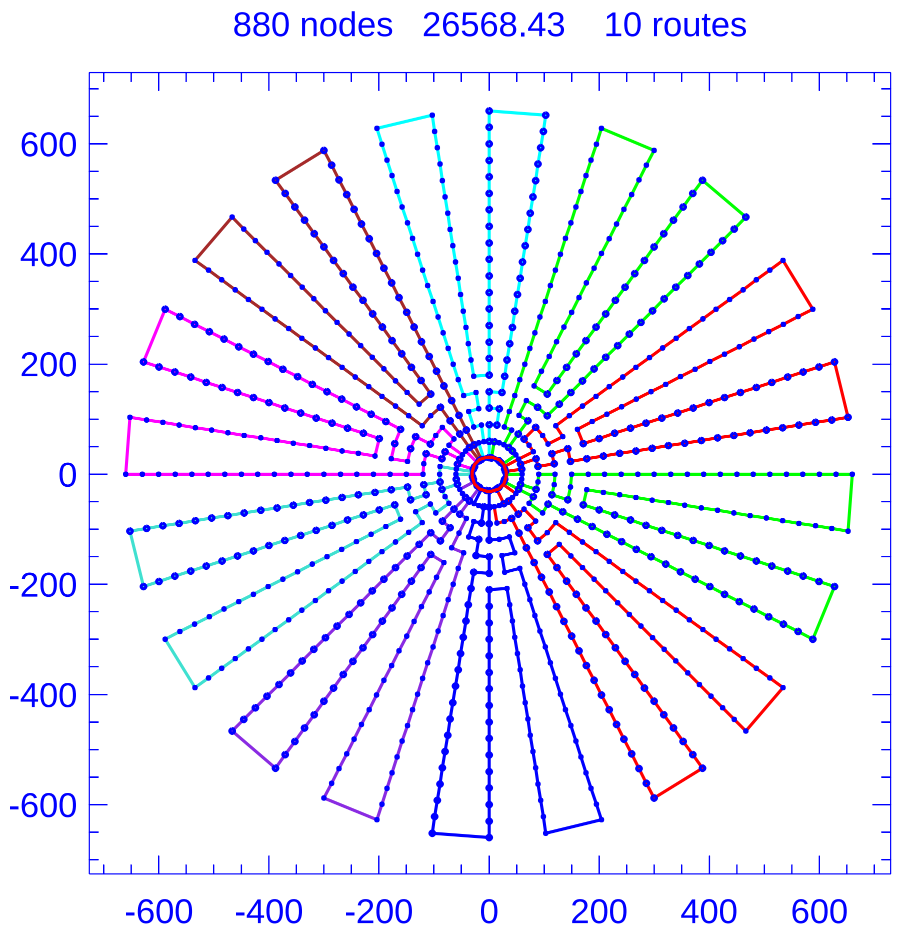


Figure B.55: Solution for problem Li-28

Problem			Li-28
Vehicle capacity			1800
Maximum route length			2800.000
Number of nodes			880
Total route length			26568.431
Total number of routes			10
Route	Length	Load	Ordering
1	2675.434	1800	(0, 3, 2, 42, 43, 83, 82, 122, 123, 163, 162, 202, 242, 282, 322, 362, 402, 442, 482, 522, 562, 602, 642, 682, 722, 762, 802, 842, 843, 803, 763, 723, 683, 643, 603, 563, 523, 483, 443, 403, 363, 323, 283, 243, 203, 204, 244, 284, 324, 364, 404, 444, 484, 524, 564, 604, 644, 684, 724, 764, 804, 844, 845, 805, 765, 725, 685, 645, 605, 565, 525, 485, 445, 405, 365, 325, 285, 245, 205, 165, 164, 124, 125, 126, 86, 85, 84, 44, 4, 0)

Route	Length	Load	Ordering
2	2651.896	1780	(0, 7, 6, 5, 45, 46, 47, 87, 127, 128, 168, 167, 166, 206, 246, 286, 326, 366, 406, 446, 486, 526, 566, 606, 646, 686, 726, 766, 806, 846, 847, 807, 767, 727, 687, 647, 607, 567, 527, 487, 447, 407, 367, 327, 287, 247, 207, 208, 248, 288, 328, 368, 408, 448, 488, 528, 568, 608, 648, 688, 728, 768, 808, 848, 849, 809, 769, 729, 689, 649, 609, 569, 529, 489, 449, 409, 369, 329, 289, 249, 209, 169, 129, 89, 88, 48, 49, 50, 10, 9, 8, 0)
3	2642.481	1700	(0, 12, 11, 51, 91, 90, 130, 131, 171, 170, 210, 250, 290, 330, 370, 410, 450, 490, 530, 570, 610, 650, 690, 730, 770, 810, 850, 851, 811, 771, 731, 691, 651, 611, 571, 531, 491, 451, 411, 371, 331, 291, 251, 211, 212, 252, 292, 332, 372, 412, 452, 492, 532, 572, 612, 652, 692, 732, 772, 812, 852, 853, 813, 773, 733, 693, 653, 613, 573, 533, 493, 453, 413, 373, 333, 293, 253, 213, 173, 172, 132, 133, 93, 92, 52, 53, 13, 0)
4	2562.453	1680	(0, 14, 54, 94, 134, 174, 214, 254, 294, 334, 374, 414, 454, 494, 534, 574, 614, 654, 694, 734, 774, 814, 854, 855, 815, 775, 735, 695, 655, 615, 575, 535, 495, 455, 415, 375, 335, 295, 255, 215, 216, 256, 296, 336, 376, 416, 456, 496, 536, 576, 616, 656, 696, 736, 776, 816, 856, 857, 817, 777, 737, 697, 657, 617, 577, 537, 497, 457, 417, 377, 337, 297, 257, 217, 177, 176, 175, 135, 95, 55, 15, 0)
5	2695.462	1780	(0, 17, 16, 56, 57, 97, 96, 136, 137, 138, 178, 179, 180, 220, 219, 218, 258, 298, 338, 378, 418, 458, 498, 538, 578, 618, 658, 698, 738, 778, 818, 858, 859, 819, 779, 739, 699, 659, 619, 579, 539, 499, 459, 419, 379, 339, 299, 259, 260, 300, 340, 380, 420, 460, 500, 540, 580, 620, 660, 700, 740, 780, 820, 860, 861, 821, 781, 741, 701, 661, 621, 581, 541, 501, 461, 421, 381, 341, 301, 261, 221, 181, 141, 140, 139, 99, 98, 58, 59, 19, 18, 0)
6	2684.849	1780	(0, 20, 60, 100, 101, 102, 142, 143, 183, 182, 222, 262, 302, 342, 382, 422, 462, 502, 542, 582, 622, 662, 702, 742, 782, 822, 862, 863, 823, 783, 743, 703, 663, 623, 583, 543, 503, 463, 423, 383, 343, 303, 263, 223, 224, 264, 304, 344, 384, 424, 464, 504, 544, 584, 624, 664, 704, 744, 784, 824, 864, 865, 825, 785, 745, 705, 665, 625, 585, 545, 505, 465, 425, 385, 345, 305, 265, 225, 185, 184, 144, 145, 105, 104, 103, 63, 62, 61, 21, 22, 23, 0)
7	2623.651	1740	(0, 24, 64, 65, 66, 106, 146, 147, 187, 186, 226, 266, 306, 346, 386, 426, 466, 506, 546, 586, 626, 666, 706, 746, 786, 826, 866, 867, 827, 787, 747, 707, 667, 627, 587, 547, 507, 467, 427, 387, 347, 307, 267, 227, 228, 268, 308, 348, 388, 428, 468, 508, 548, 588, 628, 668, 708, 748, 788, 828, 868, 869, 829, 789, 749, 709, 669, 629, 589, 549, 509, 469, 429, 389, 349, 309, 269, 229, 189, 188, 148, 108, 107, 67, 27, 26, 25, 0)
8	2667.217	1770	(0, 30, 29, 28, 68, 69, 70, 110, 109, 149, 150, 190, 191, 231, 230, 270, 310, 350, 390, 430, 470, 510, 550, 590, 630, 670, 710, 750, 790, 830, 870, 871, 831, 791, 751, 711, 671, 631, 591, 551, 511, 471, 431, 391, 351, 311, 271, 272, 312, 352, 392, 432, 472, 512, 552, 592, 632, 672, 712, 752, 792, 832, 872, 873, 833, 793, 753, 713, 673, 633, 593, 553, 513, 473, 433, 393, 353, 313, 273, 233, 232, 192, 193, 153, 152, 151, 111, 71, 31, 32, 0)

Route	Length	Load	Ordering
9	2675.434	1790	(0, 33, 34, 74, 73, 72, 112, 113, 114, 154, 194, 234, 274, 314, 354, 394, 434, 474, 514, 554, 594, 634, 674, 714, 754, 794, 834, 874, 875, 835, 795, 755, 715, 675, 635, 595, 555, 515, 475, 435, 395, 355, 315, 275, 235, 236, 276, 316, 356, 396, 436, 476, 516, 556, 596, 636, 676, 716, 756, 796, 836, 876, 877, 837, 797, 757, 717, 677, 637, 597, 557, 517, 477, 437, 397, 357, 317, 277, 237, 197, 196, 195, 155, 156, 116, 115, 75, 76, 77, 37, 36, 35, 0)
10	2689.556	1780	(0, 38, 78, 118, 117, 157, 158, 198, 238, 278, 318, 358, 398, 438, 478, 518, 558, 598, 638, 678, 718, 758, 798, 838, 878, 879, 839, 799, 759, 719, 679, 639, 599, 559, 519, 479, 439, 399, 359, 319, 279, 239, 240, 280, 320, 360, 400, 440, 480, 520, 560, 600, 640, 680, 720, 760, 800, 840, 880, 841, 801, 761, 721, 681, 641, 601, 561, 521, 481, 441, 401, 361, 321, 281, 241, 201, 161, 200, 199, 159, 160, 121, 81, 120, 119, 79, 80, 41, 1, 40, 39, 0)

Table B.60: Details of solution to problem Li-28

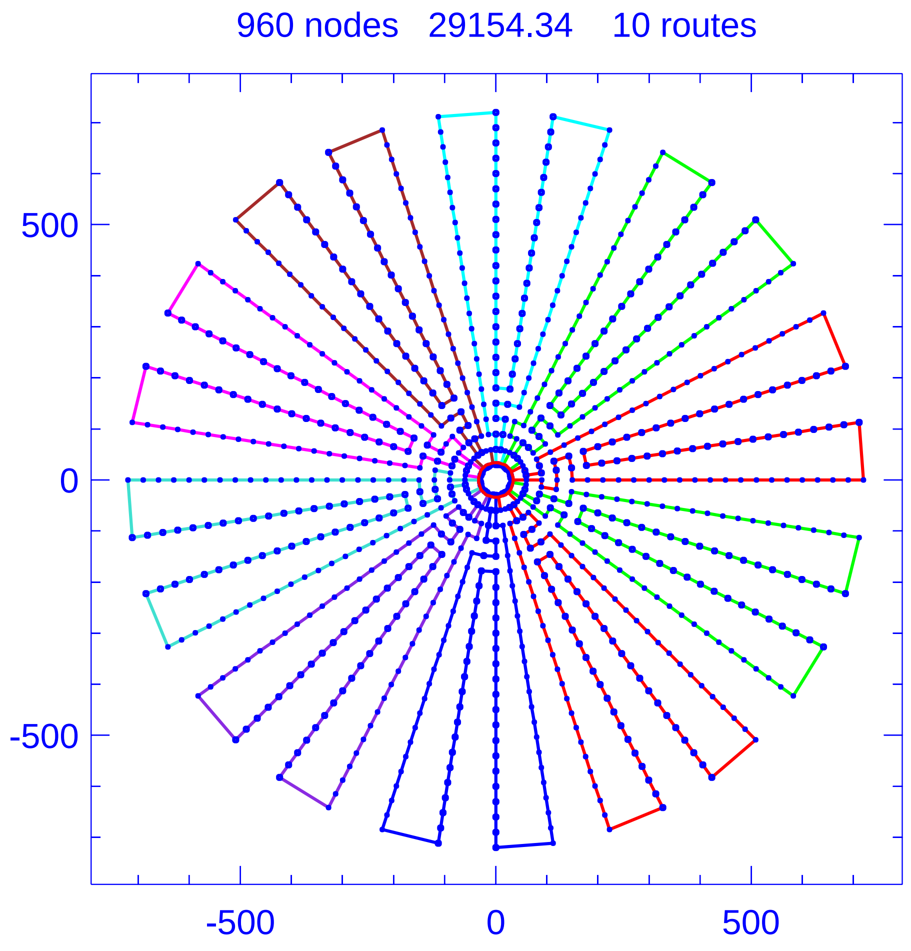


Figure B.56: Solution for problem Li-29

Problem			Li-29
Vehicle capacity			2000
Maximum route length			3000.000
Number of nodes			960
Total route length			29154.336
Total number of routes			10
Route	Length	Load	Ordering
1	2948.386	1940	(0, 3, 2, 1, 41, 81, 120, 160, 121, 122, 123, 163, 162, 161, 201, 241, 281, 321, 361, 401, 441, 481, 521, 561, 601, 641, 681, 721, 761, 801, 841, 881, 921, 922, 882, 842, 802, 762, 722, 682, 642, 602, 562, 522, 482, 442, 402, 362, 322, 282, 242, 202, 203, 243, 283, 323, 363, 403, 443, 483, 523, 563, 603, 643, 683, 723, 763, 803, 843, 883, 923, 924, 884, 844, 804, 764, 724, 684, 644, 604, 564, 524, 484, 444, 404, 364, 324, 284, 244, 204, 164, 124, 84, 83, 82, 42, 43, 44, 4, 0)

Route	Length	Load	Ordering
2	2948.387	1940	(0, 7, 6, 5, 45, 46, 47, 87, 86, 85, 125, 126, 127, 167, 166, 165, 205, 245, 285, 325, 365, 405, 445, 485, 525, 565, 605, 645, 685, 725, 765, 805, 845, 885, 925, 926, 886, 846, 806, 766, 726, 686, 646, 606, 566, 526, 486, 446, 406, 366, 326, 286, 246, 206, 207, 247, 287, 327, 367, 407, 447, 487, 527, 567, 607, 647, 687, 727, 767, 807, 847, 887, 927, 928, 888, 848, 808, 768, 728, 688, 648, 608, 568, 528, 488, 448, 408, 368, 328, 288, 248, 208, 168, 128, 129, 89, 88, 48, 8, 0)
3	2868.358	1880	(0, 9, 49, 50, 90, 130, 131, 171, 170, 169, 209, 249, 289, 329, 369, 409, 449, 489, 529, 569, 609, 649, 689, 729, 769, 809, 849, 889, 929, 930, 890, 850, 810, 770, 730, 690, 650, 610, 570, 530, 490, 450, 410, 370, 330, 290, 250, 210, 211, 251, 291, 331, 371, 411, 451, 491, 531, 571, 611, 651, 691, 731, 771, 811, 851, 891, 931, 932, 892, 852, 812, 772, 732, 692, 652, 612, 572, 532, 492, 452, 412, 372, 332, 292, 252, 212, 172, 132, 92, 91, 51, 11, 10, 0)
4	2896.603	1920	(0, 14, 13, 12, 52, 53, 54, 94, 93, 133, 173, 213, 253, 293, 333, 373, 413, 453, 493, 533, 573, 613, 653, 693, 733, 773, 813, 853, 893, 933, 934, 894, 854, 814, 774, 734, 694, 654, 614, 574, 534, 494, 454, 414, 374, 334, 294, 254, 214, 215, 255, 295, 335, 375, 415, 455, 495, 535, 575, 615, 655, 695, 735, 775, 815, 855, 895, 935, 936, 896, 856, 816, 776, 736, 696, 656, 616, 576, 536, 496, 456, 416, 376, 336, 296, 256, 216, 176, 175, 174, 134, 135, 95, 55, 56, 16, 15, 0)
5	2915.434	1920	(0, 17, 57, 97, 96, 136, 137, 138, 178, 177, 217, 257, 297, 337, 377, 417, 457, 497, 537, 577, 617, 657, 697, 737, 777, 817, 857, 897, 937, 938, 898, 858, 818, 778, 738, 698, 658, 618, 578, 538, 498, 458, 418, 378, 338, 298, 258, 218, 219, 259, 299, 339, 379, 419, 459, 499, 539, 579, 619, 659, 699, 739, 779, 819, 859, 899, 939, 940, 900, 860, 820, 780, 740, 700, 660, 620, 580, 540, 500, 460, 420, 380, 340, 300, 260, 220, 180, 179, 139, 99, 98, 58, 59, 60, 20, 19, 18, 0)
6	2948.386	1940	(0, 23, 22, 21, 61, 101, 100, 140, 141, 142, 143, 183, 182, 181, 221, 261, 301, 341, 381, 421, 461, 501, 541, 581, 621, 661, 701, 741, 781, 821, 861, 901, 941, 942, 902, 862, 822, 782, 742, 702, 662, 622, 582, 542, 502, 462, 422, 382, 342, 302, 262, 222, 223, 263, 303, 343, 383, 423, 463, 503, 543, 583, 623, 663, 703, 743, 783, 823, 863, 903, 943, 944, 904, 864, 824, 784, 744, 704, 664, 624, 584, 544, 504, 464, 424, 384, 344, 304, 264, 224, 184, 144, 104, 103, 102, 62, 63, 64, 24, 0)
7	2948.387	1940	(0, 27, 26, 25, 65, 66, 67, 107, 106, 105, 145, 146, 147, 187, 186, 185, 225, 265, 305, 345, 385, 425, 465, 505, 545, 585, 625, 665, 705, 745, 785, 825, 865, 905, 945, 946, 906, 866, 826, 786, 746, 706, 666, 626, 586, 546, 506, 466, 426, 386, 346, 306, 266, 226, 227, 267, 307, 347, 387, 427, 467, 507, 547, 587, 627, 667, 707, 747, 787, 827, 867, 907, 947, 948, 908, 868, 828, 788, 748, 708, 668, 628, 588, 548, 508, 468, 428, 388, 348, 308, 268, 228, 188, 148, 149, 109, 108, 68, 28, 0)
8	2868.358	1880	(0, 29, 69, 70, 110, 150, 151, 191, 190, 189, 229, 269, 309, 349, 389, 429, 469, 509, 549, 589, 629, 669, 709, 749, 789, 829, 869, 909, 949, 950, 910, 870, 830, 790, 750, 710, 670, 630, 590, 550, 510, 470, 430, 390, 350, 310, 270, 230, 231, 271, 311, 351, 391, 431, 471, 511, 551, 591, 631, 671, 711, 751, 791, 831, 871, 911, 951, 952, 912, 872, 832, 792, 752, 712, 672, 632, 592, 552, 512, 472, 432, 392, 352, 312, 272, 232, 192, 152, 112, 111, 71, 31, 30, 0)

Route	Length	Load	Ordering
9	2929.556	1940	(0, 34, 33, 32, 72, 73, 74, 114, 113, 153, 193, 233, 273, 313, 353, 393, 433, 473, 513, 553, 593, 633, 673, 713, 753, 793, 833, 873, 913, 953, 954, 914, 874, 834, 794, 754, 714, 674, 634, 594, 554, 514, 474, 434, 394, 354, 314, 274, 234, 235, 275, 315, 355, 395, 435, 475, 515, 555, 595, 635, 675, 715, 755, 795, 835, 875, 915, 955, 956, 916, 876, 836, 796, 756, 716, 676, 636, 596, 556, 516, 476, 436, 396, 356, 316, 276, 236, 196, 195, 194, 154, 155, 156, 116, 115, 75, 76, 36, 35, 0)
10	2882.481	1900	(0, 37, 77, 117, 157, 158, 198, 197, 237, 277, 317, 357, 397, 437, 477, 517, 557, 597, 637, 677, 717, 757, 797, 837, 877, 917, 957, 958, 918, 878, 838, 798, 758, 718, 678, 638, 598, 558, 518, 478, 438, 398, 358, 318, 278, 238, 239, 279, 319, 359, 399, 439, 479, 519, 559, 599, 639, 679, 719, 759, 799, 839, 879, 919, 959, 960, 920, 880, 840, 800, 760, 720, 680, 640, 600, 560, 520, 480, 440, 400, 360, 320, 280, 240, 200, 199, 159, 119, 118, 78, 79, 80, 40, 39, 38, 0)

Table B.61: Details of solution to problem Li-29

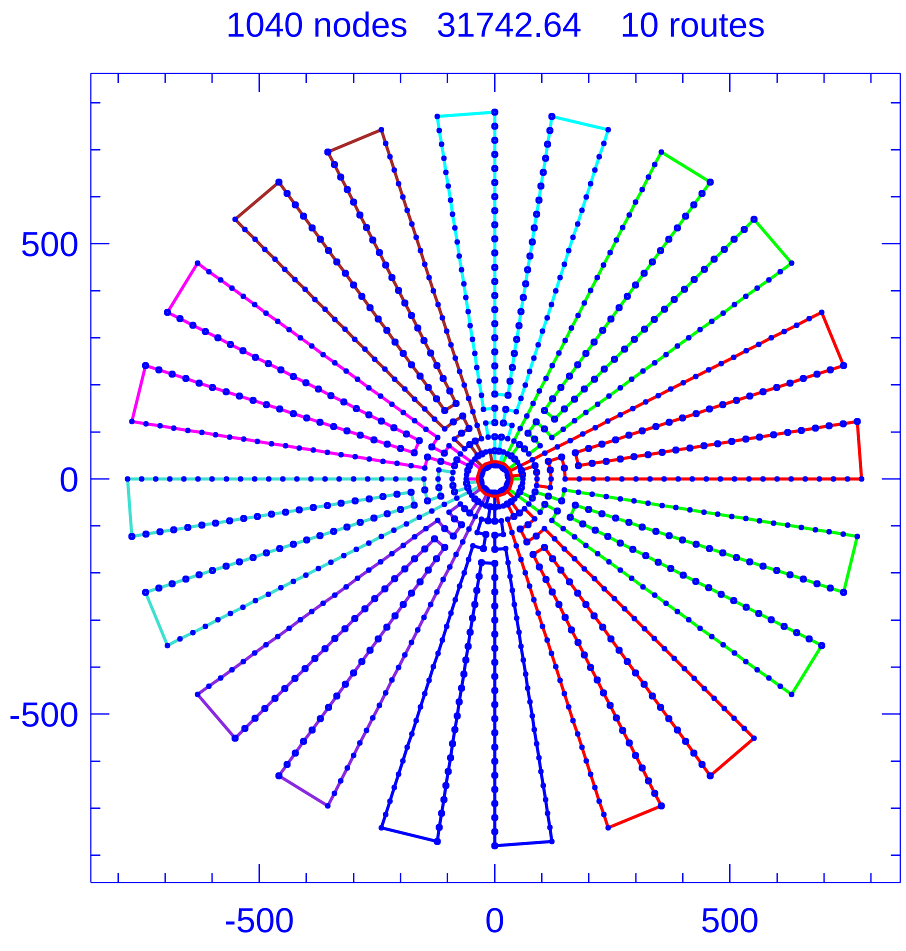


Figure B.57: Solution for problem Li-30

Problem			Li-30
Vehicle capacity			2100
Maximum route length			3200.000
Number of nodes			1040
Total route length			31742.639
Total number of routes			10
Route	Length	Load	Ordering
1	3193.094	2080	(0, 3, 2, 42, 43, 83, 82, 81, 120, 160, 121, 122, 123, 163, 162, 161, 201, 241, 281, 321, 361, 401, 441, 481, 521, 561, 601, 641, 681, 721, 761, 801, 841, 881, 921, 961, 1001, 1002, 962, 922, 882, 842, 802, 762, 722, 682, 642, 602, 562, 522, 482, 442, 402, 362, 322, 282, 242, 202, 203, 243, 283, 323, 363, 403, 443, 483, 523, 563, 603, 643, 683, 723, 763, 803, 843, 883, 923, 963, 1003, 1004, 964, 924, 884, 844, 804, 764, 724, 684, 644, 604, 564, 524, 484, 444, 404, 364, 324, 284, 244, 204, 164, 124, 84, 44, 4, 0)

Route	Length	Load	Ordering
2	3174.264	2080	(0, 7, 6, 5, 45, 46, 47, 87, 86, 85, 125, 126, 127, 167, 166, 165, 205, 245, 285, 325, 365, 405, 445, 485, 525, 565, 605, 645, 685, 725, 765, 805, 845, 885, 925, 965, 1005, 1006, 966, 926, 886, 846, 806, 766, 726, 686, 646, 606, 566, 526, 486, 446, 406, 366, 326, 286, 246, 206, 207, 247, 287, 327, 367, 407, 447, 487, 527, 567, 607, 647, 687, 727, 767, 807, 847, 887, 927, 967, 1007, 1008, 968, 928, 888, 848, 808, 768, 728, 688, 648, 608, 568, 528, 488, 448, 408, 368, 328, 288, 248, 208, 168, 128, 88, 48, 8, 0)
3	3160.141	2060	(0, 9, 49, 50, 90, 89, 129, 130, 170, 169, 209, 249, 289, 329, 369, 409, 449, 489, 529, 569, 609, 649, 689, 729, 769, 809, 849, 889, 929, 969, 1009, 1010, 970, 930, 890, 850, 810, 770, 730, 690, 650, 610, 570, 530, 490, 450, 410, 370, 330, 290, 250, 210, 211, 251, 291, 331, 371, 411, 451, 491, 531, 571, 611, 651, 691, 731, 771, 811, 851, 891, 931, 971, 1011, 1012, 972, 932, 892, 852, 812, 772, 732, 692, 652, 612, 572, 532, 492, 452, 412, 372, 332, 292, 252, 212, 172, 171, 131, 132, 92, 91, 51, 11, 10, 0)
4	3188.387	2100	(0, 14, 13, 12, 52, 53, 54, 94, 93, 133, 173, 213, 253, 293, 333, 373, 413, 453, 493, 533, 573, 613, 653, 693, 733, 773, 813, 853, 893, 933, 973, 1013, 1014, 974, 934, 894, 854, 814, 774, 734, 694, 654, 614, 574, 534, 494, 454, 414, 374, 334, 294, 254, 214, 215, 255, 295, 335, 375, 415, 455, 495, 535, 575, 615, 655, 695, 735, 775, 815, 855, 895, 935, 975, 1015, 1016, 976, 936, 896, 856, 816, 776, 736, 696, 656, 616, 576, 536, 496, 456, 416, 376, 336, 296, 256, 216, 176, 175, 174, 134, 135, 136, 96, 95, 55, 56, 16, 15, 0)
5	3155.434	2080	(0, 17, 57, 97, 137, 138, 178, 177, 217, 257, 297, 337, 377, 417, 457, 497, 537, 577, 617, 657, 697, 737, 777, 817, 857, 897, 937, 977, 1017, 1018, 978, 938, 898, 858, 818, 778, 738, 698, 658, 618, 578, 538, 498, 458, 418, 378, 338, 298, 258, 218, 219, 259, 299, 339, 379, 419, 459, 499, 539, 579, 619, 659, 699, 739, 779, 819, 859, 899, 939, 979, 1019, 1020, 980, 940, 900, 860, 820, 780, 740, 700, 660, 620, 580, 540, 500, 460, 420, 380, 340, 300, 260, 220, 180, 179, 139, 99, 98, 58, 59, 60, 61, 21, 20, 19, 18, 0)
6	3193.094	2080	(0, 23, 22, 62, 63, 103, 102, 101, 100, 140, 141, 142, 143, 183, 182, 181, 221, 261, 301, 341, 381, 421, 461, 501, 541, 581, 621, 661, 701, 741, 781, 821, 861, 901, 941, 981, 1021, 1022, 982, 942, 902, 862, 822, 782, 742, 702, 662, 622, 582, 542, 502, 462, 422, 382, 342, 302, 262, 222, 223, 263, 303, 343, 383, 423, 463, 503, 543, 583, 623, 663, 703, 743, 783, 823, 863, 903, 943, 983, 1023, 1024, 984, 944, 904, 864, 824, 784, 744, 704, 664, 624, 584, 544, 504, 464, 424, 384, 344, 304, 264, 224, 184, 144, 104, 64, 24, 0)
7	3174.264	2080	(0, 27, 26, 25, 65, 66, 67, 107, 106, 105, 145, 146, 147, 187, 186, 185, 225, 265, 305, 345, 385, 425, 465, 505, 545, 585, 625, 665, 705, 745, 785, 825, 865, 905, 945, 985, 1025, 1026, 986, 946, 906, 866, 826, 786, 746, 706, 666, 626, 586, 546, 506, 466, 426, 386, 346, 306, 266, 226, 227, 267, 307, 347, 387, 427, 467, 507, 547, 587, 627, 667, 707, 747, 787, 827, 867, 907, 947, 987, 1027, 1028, 988, 948, 908, 868, 828, 788, 748, 708, 668, 628, 588, 548, 508, 468, 428, 388, 348, 308, 268, 228, 188, 148, 108, 68, 28, 0)

Route	Length	Load	Ordering
8	3160.141	2060	(0, 29, 69, 70, 110, 109, 149, 150, 190, 189, 229, 269, 309, 349, 389, 429, 469, 509, 549, 589, 629, 669, 709, 749, 789, 829, 869, 909, 949, 989, 1029, 1030, 990, 950, 910, 870, 830, 790, 750, 710, 670, 630, 590, 550, 510, 470, 430, 390, 350, 310, 270, 230, 231, 271, 311, 351, 391, 431, 471, 511, 551, 591, 631, 671, 711, 751, 791, 831, 871, 911, 951, 991, 1031, 1032, 992, 952, 912, 872, 832, 792, 752, 712, 672, 632, 592, 552, 512, 472, 432, 392, 352, 312, 272, 232, 192, 191, 151, 152, 112, 111, 71, 31, 30, 0)
9	3188.387	2100	(0, 34, 33, 32, 72, 73, 74, 114, 113, 153, 193, 233, 273, 313, 353, 393, 433, 473, 513, 553, 593, 633, 673, 713, 753, 793, 833, 873, 913, 953, 993, 1033, 1034, 994, 954, 914, 874, 834, 794, 754, 714, 674, 634, 594, 554, 514, 474, 434, 394, 354, 314, 274, 234, 235, 275, 315, 355, 395, 435, 475, 515, 555, 595, 635, 675, 715, 755, 795, 835, 875, 915, 955, 995, 1035, 1036, 996, 956, 916, 876, 836, 796, 756, 716, 676, 636, 596, 556, 516, 476, 436, 396, 356, 316, 276, 236, 196, 195, 194, 154, 155, 156, 116, 115, 75, 76, 36, 35, 0)
10	3155.434	2080	(0, 37, 77, 117, 157, 158, 198, 197, 237, 277, 317, 357, 397, 437, 477, 517, 557, 597, 637, 677, 717, 757, 797, 837, 877, 917, 957, 997, 1037, 1038, 998, 958, 918, 878, 838, 798, 758, 718, 678, 638, 598, 558, 518, 478, 438, 398, 358, 318, 278, 238, 239, 279, 319, 359, 399, 439, 479, 519, 559, 599, 639, 679, 719, 759, 799, 839, 879, 919, 959, 999, 1039, 1040, 1000, 960, 920, 880, 840, 800, 760, 720, 680, 640, 600, 560, 520, 480, 440, 400, 360, 320, 280, 240, 200, 199, 159, 119, 118, 78, 79, 80, 41, 1, 40, 39, 38, 0)

Table B.62: Details of solution to problem Li-30

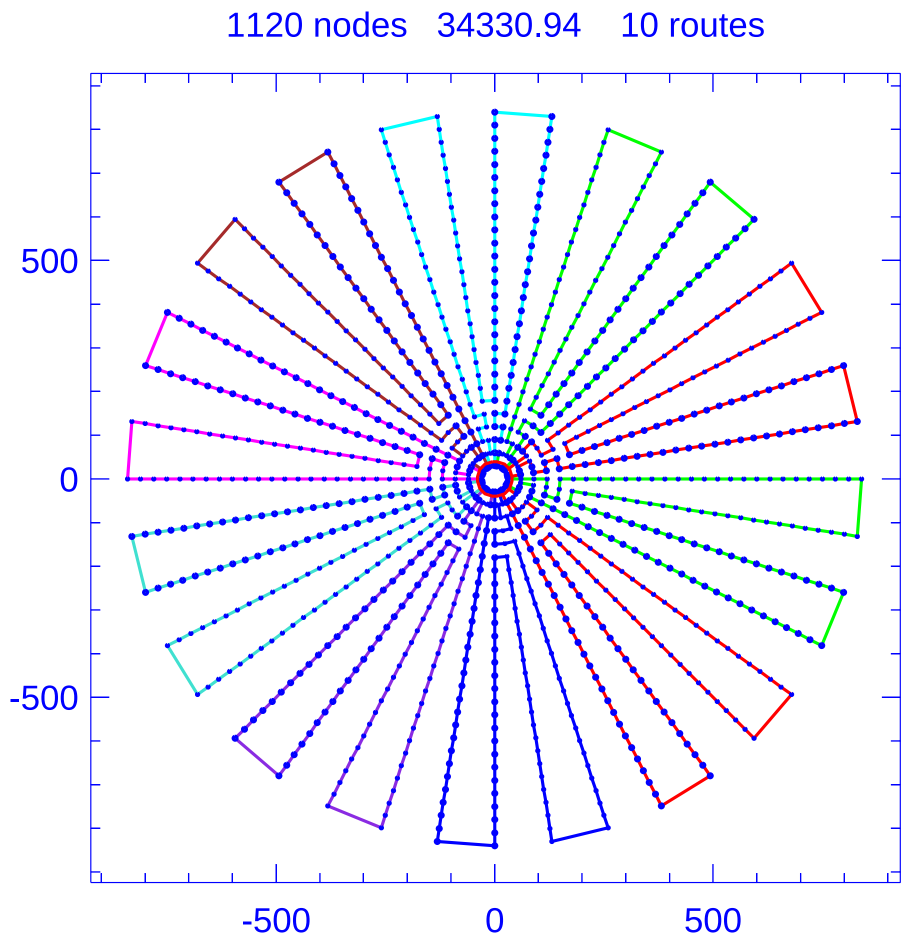


Figure B.58: Solution for problem Li-31

Problem			Li-31
Vehicle capacity Maximum route length Number of nodes Total route length Total number of routes			2300 3500.000 1120 34330.940 10
Route	Length	Load	Ordering
1	3466.047	2300	(0, 4, 3, 2, 42, 43, 44, 84, 83, 82, 122, 123, 163, 162, 202, 242, 282, 322, 362, 402, 442, 482, 522, 562, 602, 642, 682, 722, 762, 802, 842, 882, 922, 962, 1002, 1042, 1082, 1083, 1043, 1003, 963, 923, 883, 843, 803, 763, 723, 683, 643, 603, 563, 523, 483, 443, 403, 363, 323, 283, 243, 203, 204, 244, 284, 324, 364, 404, 444, 484, 524, 564, 604, 644, 684, 724, 764, 804, 844, 884, 924, 964, 1004, 1044, 1084, 1085, 1045, 1005, 965, 925, 885, 845, 805, 765, 725, 685, 645, 605, 565, 525, 485, 445, 405, 365, 325, 285, 245, 205, 165, 164, 124, 125, 126, 86, 85, 45, 5, 0)

Route	Length	Load	Ordering
2	3414.264	2240	(0, 7, 6, 46, 47, 87, 127, 128, 168, 167, 166, 206, 246, 286, 326, 366, 406, 446, 486, 526, 566, 606, 646, 686, 726, 766, 806, 846, 886, 926, 966, 1006, 1046, 1086, 1087, 1047, 1007, 967, 927, 887, 847, 807, 767, 727, 687, 647, 607, 567, 527, 487, 447, 407, 367, 327, 287, 247, 207, 208, 248, 288, 328, 368, 408, 448, 488, 528, 568, 608, 648, 688, 728, 768, 808, 848, 888, 928, 968, 1008, 1048, 1088, 1089, 1049, 1009, 969, 929, 889, 849, 809, 769, 729, 689, 649, 609, 569, 529, 489, 449, 409, 369, 329, 289, 249, 209, 169, 129, 89, 88, 48, 49, 50, 10, 9, 8, 0)
3	3418.971	2180	(0, 12, 11, 51, 91, 90, 130, 131, 171, 170, 210, 250, 290, 330, 370, 410, 450, 490, 530, 570, 610, 650, 690, 730, 770, 810, 850, 890, 930, 970, 1010, 1050, 1090, 1091, 1051, 1011, 971, 931, 891, 851, 811, 771, 731, 691, 651, 611, 571, 531, 491, 451, 411, 371, 331, 291, 251, 211, 212, 252, 292, 332, 372, 412, 452, 492, 532, 572, 612, 652, 692, 732, 772, 812, 852, 892, 932, 972, 1012, 1052, 1092, 1093, 1053, 1013, 973, 933, 893, 853, 813, 773, 733, 693, 653, 613, 573, 533, 493, 453, 413, 373, 333, 293, 253, 213, 173, 172, 132, 133, 93, 92, 52, 53, 13, 0)
4	3433.094	2240	(0, 14, 54, 94, 134, 174, 214, 254, 294, 334, 374, 414, 454, 494, 534, 574, 614, 654, 694, 734, 774, 814, 854, 894, 934, 974, 1014, 1054, 1094, 1095, 1055, 1015, 975, 935, 895, 855, 815, 775, 735, 695, 655, 615, 575, 535, 495, 455, 415, 375, 335, 295, 255, 215, 216, 256, 296, 336, 376, 416, 456, 496, 536, 576, 616, 656, 696, 736, 776, 816, 856, 896, 936, 976, 1016, 1056, 1096, 1097, 1057, 1017, 977, 937, 897, 857, 817, 777, 737, 697, 657, 617, 577, 537, 497, 457, 417, 377, 337, 297, 257, 217, 177, 176, 175, 135, 136, 137, 97, 96, 95, 55, 56, 57, 17, 16, 15, 0)
5	3433.094	2240	(0, 19, 18, 58, 59, 60, 100, 99, 98, 138, 178, 218, 258, 298, 338, 378, 418, 458, 498, 538, 578, 618, 658, 698, 738, 778, 818, 858, 898, 938, 978, 1018, 1058, 1098, 1099, 1059, 1019, 979, 939, 899, 859, 819, 779, 739, 699, 659, 619, 579, 539, 499, 459, 419, 379, 339, 299, 259, 219, 220, 260, 300, 340, 380, 420, 460, 500, 540, 580, 620, 660, 700, 740, 780, 820, 860, 900, 940, 980, 1020, 1060, 1100, 1101, 1061, 1021, 981, 941, 901, 861, 821, 781, 741, 701, 661, 621, 581, 541, 501, 461, 421, 381, 341, 301, 261, 221, 181, 180, 179, 139, 140, 141, 101, 61, 21, 20, 0)
6	3466.047	2300	(0, 24, 23, 22, 62, 63, 64, 104, 103, 102, 142, 143, 183, 182, 222, 262, 302, 342, 382, 422, 462, 502, 542, 582, 622, 662, 702, 742, 782, 822, 862, 902, 942, 982, 1022, 1062, 1102, 1103, 1063, 1023, 983, 943, 903, 863, 823, 783, 743, 703, 663, 623, 583, 543, 503, 463, 423, 383, 343, 303, 263, 223, 224, 264, 304, 344, 384, 424, 464, 504, 544, 584, 624, 664, 704, 744, 784, 824, 864, 904, 944, 984, 1024, 1064, 1104, 1105, 1065, 1025, 985, 945, 905, 865, 825, 785, 745, 705, 665, 625, 585, 545, 505, 465, 425, 385, 345, 305, 265, 225, 185, 184, 144, 145, 146, 106, 105, 65, 25, 0)

Route	Length	Load	Ordering
7	3414.264	2240	(0, 27, 26, 66, 67, 107, 147, 148, 188, 187, 186, 226, 266, 306, 346, 386, 426, 466, 506, 546, 586, 626, 666, 706, 746, 786, 826, 866, 906, 946, 986, 1026, 1066, 1106, 1107, 1067, 1027, 987, 947, 907, 867, 827, 787, 747, 707, 667, 627, 587, 547, 507, 467, 427, 387, 347, 307, 267, 227, 228, 268, 308, 348, 388, 428, 468, 508, 548, 588, 628, 668, 708, 748, 788, 828, 868, 908, 948, 988, 1028, 1068, 1108, 1109, 1069, 1029, 989, 949, 909, 869, 829, 789, 749, 709, 669, 629, 589, 549, 509, 469, 429, 389, 349, 309, 269, 229, 189, 149, 109, 108, 68, 69, 70, 30, 29, 28, 0)
8	3418.971	2180	(0, 32, 31, 71, 111, 110, 150, 190, 230, 270, 310, 350, 390, 430, 470, 510, 550, 590, 630, 670, 710, 750, 790, 830, 870, 910, 950, 990, 1030, 1070, 1110, 1111, 1071, 1031, 991, 951, 911, 871, 831, 791, 751, 711, 671, 631, 591, 551, 511, 471, 431, 391, 351, 311, 271, 231, 232, 272, 312, 352, 392, 432, 472, 512, 552, 592, 632, 672, 712, 752, 792, 832, 872, 912, 952, 992, 1032, 1072, 1112, 1113, 1073, 1033, 993, 953, 913, 873, 833, 793, 753, 713, 673, 633, 593, 553, 513, 473, 433, 393, 353, 313, 273, 233, 193, 192, 191, 151, 152, 153, 113, 112, 72, 73, 33, 0)
9	3433.094	2240	(0, 34, 74, 114, 154, 194, 234, 274, 314, 354, 394, 434, 474, 514, 554, 594, 634, 674, 714, 754, 794, 834, 874, 914, 954, 994, 1034, 1074, 1114, 1115, 1075, 1035, 995, 955, 915, 875, 835, 795, 755, 715, 675, 635, 595, 555, 515, 475, 435, 395, 355, 315, 275, 235, 236, 276, 316, 356, 396, 436, 476, 516, 556, 596, 636, 676, 716, 756, 796, 836, 876, 916, 956, 996, 1036, 1076, 1116, 1117, 1077, 1037, 997, 957, 917, 877, 837, 797, 757, 717, 677, 637, 597, 557, 517, 477, 437, 397, 357, 317, 277, 237, 197, 196, 195, 155, 156, 157, 117, 116, 115, 75, 76, 77, 37, 36, 35, 0)
10	3433.094	2240	(0, 38, 78, 79, 80, 120, 119, 118, 158, 198, 238, 278, 318, 358, 398, 438, 478, 518, 558, 598, 638, 678, 718, 758, 798, 838, 878, 918, 958, 998, 1038, 1078, 1118, 1119, 1079, 1039, 999, 959, 919, 879, 839, 799, 759, 719, 679, 639, 599, 559, 519, 479, 439, 399, 359, 319, 279, 239, 240, 280, 320, 360, 400, 440, 480, 520, 560, 600, 640, 680, 720, 760, 800, 840, 880, 920, 960, 1000, 1040, 1080, 1120, 1081, 1041, 1001, 961, 921, 881, 841, 801, 761, 721, 681, 641, 601, 561, 521, 481, 441, 401, 361, 321, 281, 241, 201, 161, 200, 199, 159, 160, 121, 81, 41, 1, 40, 39, 0)

Table B.63: Details of solution to problem Li-31

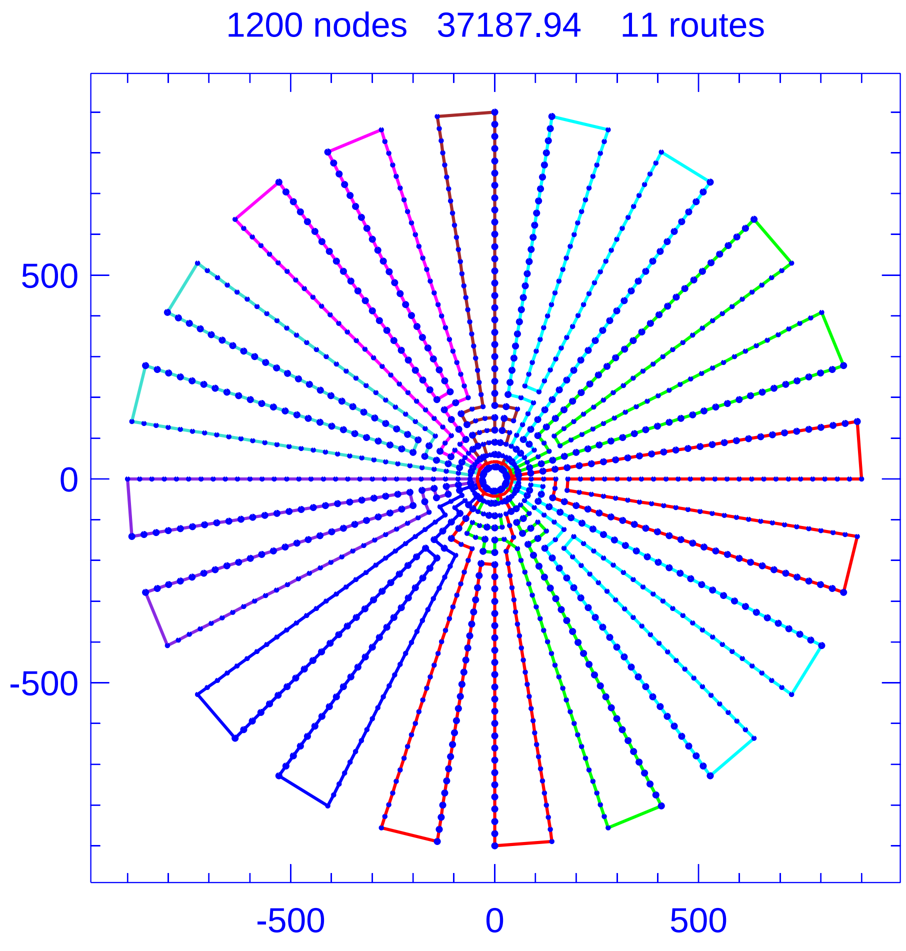


Figure B.59: Solution for problem Li-32

Problem			Li-32
Vehicle capacity Maximum route length Number of nodes Total route length Total number of routes			2500 3600.000 1200 37187.943 11
Route	Length	Load	Ordering
1	3597.773	2240	(0, 1, 41, 81, 121, 161, 200, 199, 239, 279, 319, 359, 399, 439, 479, 519, 559, 599, 639, 679, 719, 759, 799, 839, 879, 919, 959, 999, 1039, 1079, 1119, 1159, 1199, 1200, 1160, 1120, 1080, 1040, 1000, 960, 920, 880, 840, 800, 760, 720, 680, 640, 600, 560, 520, 480, 440, 400, 360, 320, 280, 240, 201, 241, 281, 321, 361, 401, 441, 481, 521, 561, 601, 641, 681, 721, 761, 801, 841, 881, 921, 961, 1001, 1041, 1081, 1121, 1161, 1162, 1122, 1082, 1042, 1002, 962, 922, 882, 842, 802, 762, 722, 682, 642, 602, 562, 522, 482, 442, 402, 362, 322, 282, 242, 202, 162, 122, 82, 42, 2, 0)

Route	Length	Load	Ordering
2	3597.773	2240	(0, 3, 43, 83, 123, 163, 203, 243, 283, 323, 363, 403, 443, 483, 523, 563, 603, 643, 683, 723, 763, 803, 843, 883, 923, 963, 1003, 1043, 1083, 1123, 1163, 1164, 1124, 1084, 1044, 1004, 964, 924, 884, 844, 804, 764, 724, 684, 644, 604, 564, 524, 484, 444, 404, 364, 324, 284, 244, 204, 205, 245, 285, 325, 365, 405, 445, 485, 525, 565, 605, 645, 685, 725, 765, 805, 845, 885, 925, 965, 1005, 1045, 1085, 1125, 1165, 1166, 1126, 1086, 1046, 1006, 966, 926, 886, 846, 806, 766, 726, 686, 646, 606, 566, 526, 486, 446, 406, 366, 326, 286, 246, 206, 166, 165, 164, 124, 84, 44, 4, 0)
3	3586.047	2300	(0, 5, 45, 85, 125, 126, 127, 167, 207, 247, 287, 327, 367, 407, 447, 487, 527, 567, 607, 647, 687, 727, 767, 807, 847, 887, 927, 967, 1007, 1047, 1087, 1127, 1167, 1168, 1128, 1088, 1048, 1008, 968, 928, 888, 848, 808, 768, 728, 688, 648, 608, 568, 528, 488, 448, 408, 368, 328, 288, 289, 329, 369, 409, 449, 489, 529, 569, 609, 649, 689, 729, 769, 809, 849, 889, 929, 969, 1009, 1049, 1089, 1129, 1169, 1170, 1130, 1090, 1050, 1010, 970, 930, 890, 850, 810, 770, 730, 690, 650, 610, 570, 530, 490, 450, 410, 370, 330, 290, 250, 249, 248, 208, 168, 128, 88, 87, 86, 46, 47, 7, 6, 0)
4	2374.321	1760	(0, 8, 48, 49, 50, 51, 52, 53, 93, 92, 91, 90, 89, 129, 130, 170, 169, 209, 210, 211, 251, 291, 331, 371, 411, 451, 491, 531, 571, 611, 651, 691, 731, 771, 811, 851, 891, 931, 971, 1011, 1051, 1091, 1131, 1171, 1172, 1132, 1092, 1052, 1012, 972, 932, 892, 852, 812, 772, 732, 692, 652, 612, 572, 532, 492, 452, 412, 372, 332, 292, 252, 212, 213, 214, 174, 173, 172, 171, 131, 132, 133, 134, 94, 95, 55, 54, 14, 13, 12, 11, 10, 9, 0)
5	3595.462	2210	(0, 15, 16, 56, 96, 136, 135, 175, 215, 255, 254, 253, 293, 333, 373, 413, 453, 493, 533, 573, 613, 653, 693, 733, 773, 813, 853, 893, 933, 973, 1013, 1053, 1093, 1133, 1173, 1174, 1134, 1094, 1054, 1014, 974, 934, 894, 854, 814, 774, 734, 694, 654, 614, 574, 534, 494, 454, 414, 374, 334, 294, 295, 335, 375, 415, 455, 495, 535, 575, 615, 655, 695, 735, 775, 815, 855, 895, 935, 975, 1015, 1055, 1095, 1135, 1175, 1176, 1136, 1096, 1056, 1016, 976, 936, 896, 856, 816, 776, 736, 696, 656, 616, 576, 536, 496, 456, 416, 376, 336, 296, 256, 216, 176, 177, 178, 138, 137, 97, 57, 17, 0)
6	3575.434	2280	(0, 18, 58, 98, 99, 139, 179, 219, 218, 217, 257, 297, 337, 377, 417, 457, 497, 537, 577, 617, 657, 697, 737, 777, 817, 857, 897, 937, 977, 1017, 1057, 1097, 1137, 1177, 1178, 1138, 1098, 1058, 1018, 978, 938, 898, 858, 818, 778, 738, 698, 658, 618, 578, 538, 498, 458, 418, 378, 338, 298, 258, 259, 299, 339, 379, 419, 459, 499, 539, 579, 619, 659, 699, 739, 779, 819, 859, 899, 939, 979, 1019, 1059, 1099, 1139, 1179, 1180, 1140, 1100, 1060, 1020, 980, 940, 900, 860, 820, 780, 740, 700, 660, 620, 580, 540, 500, 460, 420, 380, 340, 300, 260, 220, 180, 140, 100, 60, 59, 19, 0)
7	3598.971	2270	(0, 20, 21, 61, 101, 141, 181, 221, 261, 301, 341, 381, 421, 461, 501, 541, 581, 621, 661, 701, 741, 781, 821, 861, 901, 941, 981, 1021, 1061, 1101, 1141, 1181, 1182, 1142, 1102, 1062, 1022, 982, 942, 902, 862, 822, 782, 742, 702, 662, 622, 582, 542, 502, 462, 422, 382, 342, 302, 262, 263, 303, 343, 383, 423, 463, 503, 543, 583, 623, 663, 703, 743, 783, 823, 863, 903, 943, 983, 1023, 1063, 1103, 1143, 1183, 1184, 1144, 1104, 1064, 1024, 984, 944, 904, 864, 824, 784, 744, 704, 664, 624, 584, 544, 504, 464, 424, 384, 344, 304, 264, 224, 223, 222, 182, 183, 143, 142, 102, 62, 22, 0)

Route	Length	Load	Ordering
8	3595.462	2260	(0, 23, 63, 103, 104, 144, 184, 185, 225, 265, 305, 345, 385, 425, 465, 505, 545, 585, 625, 665, 705, 745, 785, 825, 865, 905, 945, 985, 1025, 1065, 1105, 1145, 1185, 1186, 1146, 1106, 1066, 1026, 986, 946, 906, 866, 826, 786, 746, 706, 666, 626, 586, 546, 506, 466, 426, 386, 346, 306, 307, 347, 387, 427, 467, 507, 547, 587, 627, 667, 707, 747, 787, 827, 867, 907, 947, 987, 1027, 1067, 1107, 1147, 1187, 1188, 1148, 1108, 1068, 1028, 988, 948, 908, 868, 828, 788, 748, 708, 668, 628, 588, 548, 508, 468, 428, 388, 348, 308, 268, 267, 266, 226, 186, 146, 145, 105, 106, 66, 65, 64, 24, 0)
9	3599.114	2310	(0, 25, 26, 27, 67, 107, 147, 187, 227, 228, 229, 269, 309, 349, 389, 429, 469, 509, 549, 589, 629, 669, 709, 749, 789, 829, 869, 909, 949, 989, 1029, 1069, 1109, 1149, 1189, 1190, 1150, 1110, 1070, 1030, 990, 950, 910, 870, 830, 790, 750, 710, 670, 630, 590, 550, 510, 470, 430, 390, 350, 310, 270, 271, 311, 351, 391, 431, 471, 511, 551, 591, 631, 671, 711, 751, 791, 831, 871, 911, 951, 991, 1031, 1071, 1111, 1151, 1191, 1192, 1152, 1112, 1072, 1032, 992, 952, 912, 872, 832, 792, 752, 712, 672, 632, 592, 552, 512, 472, 432, 392, 352, 312, 272, 232, 193, 153, 113, 114, 74, 73, 33, 34, 0)
10	2481.538	1830	(0, 28, 29, 30, 31, 32, 72, 71, 70, 69, 68, 108, 109, 110, 111, 112, 152, 151, 150, 149, 148, 188, 189, 190, 230, 231, 191, 192, 233, 273, 313, 353, 393, 433, 473, 513, 553, 593, 633, 673, 713, 753, 793, 833, 873, 913, 953, 993, 1033, 1073, 1113, 1153, 1193, 1194, 1154, 1114, 1074, 1034, 994, 954, 914, 874, 834, 794, 754, 714, 674, 634, 594, 554, 514, 474, 434, 394, 354, 314, 274, 234, 235, 236, 196, 195, 194, 154, 155, 156, 116, 115, 75, 76, 77, 37, 36, 35, 0)
11	3586.047	2300	(0, 38, 78, 118, 117, 157, 197, 237, 277, 276, 275, 315, 355, 395, 435, 475, 515, 555, 595, 635, 675, 715, 755, 795, 835, 875, 915, 955, 995, 1035, 1075, 1115, 1155, 1195, 1196, 1156, 1116, 1076, 1036, 996, 956, 916, 876, 836, 796, 756, 716, 676, 636, 596, 556, 516, 476, 436, 396, 356, 316, 317, 357, 397, 437, 477, 517, 557, 597, 637, 677, 717, 757, 797, 837, 877, 917, 957, 997, 1037, 1077, 1117, 1157, 1197, 1198, 1158, 1118, 1078, 1038, 998, 958, 918, 878, 838, 798, 758, 718, 678, 638, 598, 558, 518, 478, 438, 398, 358, 318, 278, 238, 198, 158, 159, 160, 120, 119, 79, 80, 40, 39, 0)

Table B.64: Details of solution to problem Li-32

B.7. Additional Data

In the final section of this Appendix, we provide the complete results of several experiments using our parallel algorithm. We selected problem 15 from [36], problem 385 from [76], and problem 25 from [51] and ran our procedure many times while varying several parameters. An "R" in the fourth column indicates that the master selects solutions to send out by using the randomized procedure described in Section 4.3, and a "B" indicates that the solution was obtained by having the master simply send the best solution out each time. For each problem, we ran the algorithm 10 times and present the average solution value and the overall best solution value in the two right-most columns.

Processors	Number of IP Solvers	Seconds Allowed	Distribution Strategy	Average Solution	Best Solution
8	0	200	R	1348.69	1345.61
8	0	200	B	1348.70	1346.95
8	0	400	R	1346.34	1344.37
8	0	400	B	1346.09	1343.34
8	1	200	R	1343.43	1342.10
8	1	200	B	1344.59	1341.49
8	1	400	R	1343.08	1341.26
8	1	400	B	1343.39	1341.02
8	2	200	R	1343.77	1341.00
8	2	200	B	1344.18	1342.55
8	2	400	R	1343.36	1341.72
8	2	400	B	1342.20	1340.46
16	0	200	R	1346.50	1345.06
16	0	200	B	1347.15	1344.97
16	0	400	R	1345.61	1344.29
16	0	400	B	1344.79	1342.27
16	1	200	R	1343.86	1341.61
16	1	200	B	1342.82	1341.44
16	1	400	R	1343.57	1340.66
16	1	400	B	1342.49	1340.28
16	2	200	R	1342.90	1341.13
16	2	200	B	1343.00	1340.16
16	2	400	R	1342.61	1339.81
16	2	400	B	1342.11	1339.36
32	0	200	R	1345.06	1343.70
32	0	200	B	1344.49	1341.94
32	0	400	R	1344.02	1341.50
32	0	400	B	1344.50	1341.62
32	1	200	R	1343.09	1341.12
32	1	200	B	1342.62	1341.11
32	1	400	R	1342.40	1340.10
32	1	400	B	1342.05	1340.25
32	2	200	R	1342.77	1339.68
32	2	200	B	1343.28	1341.34
32	2	400	R	1342.06	1340.52
32	2	400	B	1341.44	1340.11
32	4	200	R	1342.17	1339.83
32	4	200	B	1341.98	1339.33
32	4	400	R	1341.82	1340.34
32	4	400	B	1341.38	1339.10
64	0	200	R	1343.75	1341.43
64	0	200	B	1342.81	1340.45
64	0	400	R	1344.08	1341.22
64	0	400	B	1342.83	1340.15

(continued on next page)

Processors	Number of IP Solvers	Seconds Allowed	Distribution Strategy	Average Solution	Best Solution
64	1	200	R	1343.12	1341.45
64	1	200	B	1343.58	1342.73
64	1	400	R	1343.04	1340.79
64	1	400	B	1342.16	1340.48
64	2	200	R	1342.40	1340.18
64	2	400	R	1342.65	1339.26
64	2	400	B	1341.52	1340.36
64	4	200	R	1342.22	1340.06
64	4	200	B	1341.80	1339.84
64	4	400	R	1341.82	1339.31
64	4	400	B	1340.78	1338.93
64	8	100	B	1342.39	1338.81
64	8	200	R	1341.83	1337.99
64	8	200	B	1341.59	1340.30
64	8	400	R	1341.27	1340.39
64	8	400	B	1341.53	1340.30

Table B.65: Performance on problem 15 from Golden et al. [36]

Processors	Number of IP Solvers	Seconds Allowed	Distribution Strategy	Average Solution	Best Solution
8	0	200	R	24470.4	24426.9
8	0	200	B	24465.8	24436.4
8	0	400	R	24477.0	24461.4
8	0	400	B	24471.8	24434.4
8	1	200	R	24435.9	24388.9
8	1	200	B	24445.7	24408.4
8	1	400	R	24429.4	24388.0
8	1	400	B	24404.6	24378.9
8	2	200	R	24438.0	24409.0
8	2	200	B	24428.0	24388.5
8	2	400	R	24431.9	24386.3
8	2	400	B	24420.8	24372.2
16	0	200	R	24466.5	24419.3
16	0	200	B	24468.6	24421.2
16	0	400	R	24484.6	24463.7
16	0	400	B	24450.7	24424.3
16	1	200	R	24438.3	24398.8
16	1	200	B	24430.4	24413.6
16	1	400	R	24429.2	24372.0
16	1	400	B	24433.5	24370.3
16	2	200	R	24438.8	24394.4
16	2	200	B	24430.1	24381.9
16	2	400	R	24402.9	24372.0
16	2	400	B	24408.6	24371.2
32	0	200	R	24444.9	24401.6
32	0	200	B	24474.3	24427.4
32	0	400	R	24445.6	24409.6
32	0	400	B	24440.0	24412.2
32	1	200	R	24423.8	24385.6
32	1	200	B	24418.7	24380.5
32	1	400	R	24423.6	24381.5
32	1	400	B	24418.5	24389.9
32	2	200	R	24427.9	24392.1
32	2	200	B	24422.0	24385.3
32	2	400	R	24415.7	24388.9
32	2	400	B	24421.9	24392.3
32	4	200	R	24424.4	24377.6
32	4	200	B	24410.8	24371.2
32	4	400	R	24402.5	24379.2
32	4	400	B	24394.0	24369.6
64	0	200	R	24420.2	24385.3
64	0	200	B	24429.1	24389.6
64	0	400	R	24416.5	24382.1
64	0	400	B	24431.8	24390.3

(continued on next page)

Processors	Number of IP Solvers	Seconds Allowed	Distribution Strategy	Average Solution	Best Solution
64	1	200	R	24434.0	24378.9
64	1	200	B	24412.0	24386.1
64	1	400	R	24416.3	24382.1
64	1	400	B	24400.9	24379.3
64	2	200	R	24418.2	24383.4
64	2	200	B	24422.5	24397.6
64	2	400	R	24407.6	24379.9
64	2	400	B	24398.3	24381.7
64	4	200	R	24416.3	24388.0
64	4	200	B	24404.7	24378.9
64	4	400	R	24405.2	24378.0
64	4	400	B	24406.0	24369.1
64	8	200	R	24418.7	24379.9
64	8	200	B	24390.3	24371.1
64	8	400	R	24406.6	24368.2
64	8	400	B	24390.5	24370.5

Table B.66: Performance on problem 385 from Taillard [76]

Processors	Number of IP Solvers	Seconds Allowed	Distribution Strategy	Average Solution	Best Solution
8	0	200	R	17014.3	16889.0
8	0	200	B	16984.4	16864.0
8	0	400	R	16942.0	16837.1
8	0	400	B	17029.5	16989.1
8	1	200	R	16944.7	16835.1
8	1	200	B	16888.6	16845.2
8	1	400	R	16989.2	16818.5
8	1	400	B	16937.0	16840.1
16	0	200	R	16945.3	16794.1
16	0	200	B	17017.8	16921.6
16	0	400	R	16989.5	16792.5
16	0	400	B	16912.9	16814.8
16	1	200	R	16926.3	16804.3
16	1	200	B	16932.0	16801.5
16	1	400	R	16919.2	16791.5
16	1	400	B	16913.1	16793.2
16	2	200	R	17024.9	16932.5
16	2	200	B	16972.8	16845.5
16	2	400	R	16930.0	16789.7
16	2	400	B	16916.4	16782.0
32	0	200	R	16995.9	16809.7
32	0	200	B	16950.6	16841.3
32	0	400	R	16879.2	16786.5
32	0	400	B	16873.9	16811.4
32	1	200	R	16907.8	16809.3
32	1	200	B	16871.3	16806.0
32	1	400	R	16856.6	16809.3
32	1	400	B	16835.7	16791.5
32	2	200	R	16905.9	16809.3
32	2	200	B	16892.8	16809.3
32	2	400	R	16844.8	16786.5
32	2	400	B	16841.5	16769.5
32	4	200	R	16925.5	16811.6
32	4	200	B	16913.6	16797.2
32	4	400	R	16879.2	16788.4
32	4	400	B	16859.2	16768.7
64	0	200	R	16877.6	16809.3
64	0	200	B	16863.8	16792.5
64	0	400	R	16833.5	16787.3
64	0	400	B	16816.3	16787.3
64	1	200	R	16855.1	16809.3
64	1	200	B	16855.4	16808.5
64	1	400	R	16826.0	16794.9
64	1	400	B	16807.3	16778.4

(continued on next page)

Processors	Number of IP Solvers	Seconds Allowed	Distribution Strategy	Average Solution	Best Solution
64	2	200	R	16834.9	16789.6
64	2	200	B	16854.1	16791.5
64	2	400	R	16819.9	16791.5
64	2	400	B	16804.3	16781.6
64	4	200	R	16867.9	16763.7
64	4	200	B	16900.3	16791.5
64	4	400	R	16814.9	16786.5
64	4	400	B	16805.4	16778.2
64	8	200	R	16823.2	16768.7
64	8	200	B	16833.9	16776.0
64	8	400	R	16803.5	16763.7
64	8	400	B	16840.8	16786.5

Table B.67: Performance on problem 25 from Li et al. [51]

Appendix C

Some Technical Details of the VRPH Library

This Appendix contains technical details regarding the VRPH library and provides instructions for compiling the VRPH library using both Windows and Unix or Linux operating systems.

C.1. Installing VRPH

In this section, we describe how the files in VRPH are organized and provide instructions for compiling VRPH on different platforms.

C.1.1 Directory Structure

The first step in using VRPH is to move the files onto the system. VRPH is distributed as a single zipped file, *VRPH.zip*. Copying the file *VRPH.zip* into a directory and extracting the file will create a root directory *VRPH* with the subdirectories given in Table C.1.

C.1.2 Compiling VRPH

In this section, we provide instructions for compiling the VRPH library on different platforms. We have successfully compiled VRPH on Windows (XP and Vista) using Visual Studio 2005 and 2008, Linux using g++, and Cygwin running

Directory Name	Purpose
bin	Binary executable files generated during the compile process
data	Subdirectories containing the well-known benchmark problems of Chrisofides et al. [19, 20], Golden et al. [36], Taillard [76], and Li et al. [51]
pics	Default location for images of the solutions
inc	Header files required by VRPH
lib	Library file vrph.lib (or vrph.a if Linux or Cygwin)
sols	Default location for solution files produced by VRPH
src	Source files
temp	Directory for temporary files generated by VRPH

Table C.1: Directory structure of VRPH

under Windows, also with the g++ compiler. The source code for VRPH is written in standard C/C++ and is completely self-contained. No additional dependencies are required to produce a functional installation. We developed an optional interface between VRPH and the PLPlot library in order to produce high-quality postscript images of the solutions. We also developed an optional interface with the commercial mathematical programming solver CPLEX.

Compiling VRPH results in a static library, named *VRPH/lib/vrph.lib* on Windows and *VRPH/lib/vrph.a* on Linux or Cygwin. The binary executable is then built by linking in this library, producing *vrph/bin/RTR.exe* on Windows, or *vrph/bin/RTR* on Linux or Cygwin.

C.1.2.1 Configuration

VRPH is essentially platform independent, and we have been able to compile the code without modification on several different platforms. There are only two options that need to be set. First, if the user has access to CPLEX and wishes

ClarkeWright.cpp	Concatenate.cpp	CrossExchange.cpp
Flip.cpp	MoveString.cpp	OnePointMove.cpp
OrOpt.cpp	Postsert.cpp	Presert.cpp
RNG.cpp	Swap.cpp	SwapEnds.cpp
Sweep.cpp	ThreeOpt.cpp	ThreePointMove.cpp
TwoOpt.cpp	TwoPointMove.cpp	VRP.cpp
VRPCPX.cpp	VRPDebug.cpp	VRPGenerator.cpp
VRPGraphics.cpp	VRPIO.cpp	VRPMove.cpp
VRPNode.cpp	VRPRoute.cpp	VRPSolution.cpp
VRPSolvers.cpp	VRPTabuList.cpp	VRPTSPLib.cpp
	VRPUtils.cpp	

Table C.2: List of source files required by the VRPH library

to solve the VRP as a set partitioning problem, then the *HAS_CPLEX* flag should be set to 1 in the file *VRPH/inc/VRPConfig.h*. Second, if the open source PLPlot library is installed, then the value of *HAS_PLPLOT* should be set to 1. The static library *VRPH/lib/vrph.lib* requires the source files from the *VRPH/src* directory given in Table C.2. After compiling the *VRPH* library, the binary executable *RTR.exe* requires the source file *VRPH/src/RTR.cpp*.

C.1.2.2 Building on Windows

If either Visual Studio 2005 or 2008 is available, then the simplest method of building *VRPH* on Windows is to load the solution file *VRPH.sln* from either the *VRPH/vs2005* or the *VRPH/vs2008* directory and then build the solution. The solution should compile with no errors and no warnings. This produces the library file *VRPH/lib/VRPH.lib* in the and the *VRPH_rtr.exe* executable in the *VRPH/bin* directory.

If these compilers are not available, then VRPH can be built by creating a project in another development environment. The first step is to create a project to build the static library *VRPH.lib*. This can be done by adding the list of source files given in Table C.2 and ensuring that the *VRPH/inc* directory is listed as an additional directory for the necessary header files. After building this project using the static library configuration, the binary solver can be built by creating additional projects that link in the previously created library.

C.1.2.3 Building on Cygwin and Linux

The procedure for building VRPH on either Linux or Cygwin on Windows is identical. After suitably modifying the *VRPConfig.h* file, VRPH can be built using the *Makefile* that resides in the root *VRPH* directory. The default compiler in the *makefile* must be available on the system (we used the freely available GNU *g++* compiler by setting *CC=g++* in the very first line of the *Makefile*). From the root *VRPH* directory, run *make* in order to create the *VRPH/lib/vrph.a* and *VRPH/bin/RTR* files.

C.2. Plotting Solutions

We developed an interface between VRPH and the open source PLPlot library [63] for generating high-quality postscript (.ps) images of solutions to VRP instances. PLPlot is a platform independent library and we have used it in conjunction with

VRPH on both Windows- and Linux-based systems. The *VRP* class contains a *plot* method that allows the user to plot a VRP solution using several options.

C.3. Using CPLEX with VRPH

The VRP can be formulated as a set partitioning or set covering problem (see [9]). In [37], we present a parallel algorithm that combines a metaheuristic algorithm with a set covering formulation. We implemented this algorithm using VRPH by developing an interface with the commercial mathematical programming software package CPLEX. The file *VRPH/src/demos/set_partitioning.cpp* demonstrates the usage of this interface.

Bibliography

[1] Y. Agarwal, K. Mathur, and H.M. Salkin. A set-partitioning-based exact algorithm for the vehicle routing problem. *Networks*, 19:731–749, 1989.

[2] E. Alba and B. Dorronsoro. Solving the vehicle routing problem by using cellular genetic algorithms. In J. Gottlieb and G. Raidl, editors, *EvoCOP*, volume 3004 of *Lecture Notes in Computer Science*, pages 11–20, Berlin, 2004. Springer.

[3] E. Alba and B. Dorronsoro. *Cellular Genetic Algorithms*. Springer, New York, 2008.

[4] G.B. Alvarenga, G.R. Mateus, and G. de Tomi. A genetic and set partitioning two-phase approach for the vehicle routing problem with time windows. *Computers & Operations Research*, 34:1561–1584, 2007.

[5] D. Applegate, R. Bixby, V. Chvtal, W. Cook, D. Espinoza, M. Goycoolea, and K. Helsgaun. The Concorde source code. `http://www.tsp.gatech.edu/concorde.html`, 2008.

[6] U.M. Apte and F. Mason. Analysis and improvement of delivery operations at the San Francisco public library. *Journal of Operations Management*, 24:325–346, 2006.

[7] A. Bixby. *Polyhedral analysis and effective algorithms for the capacitated vehicle routing problem*. PhD thesis, Northwestern University, Evanston, IL, 1998.

[8] D. Applegate R. Bixby, V. Chvtal, and W. Cook. *The Traveling Salesman Problem: A Computational Study*. Princeton University Press, 2006.

[9] J. Bramel and D. Simchi-Levi. Set-covering-based algorithms for the capacitated VRP. In P. Toth and D. Vigo, editors, *The Vehicle Routing Problem*, pages 85–108. Society for Industrial and Applied Mathematics, Philadelphia, PA, 2001.

[10] O. Bräysy and M. Gendreau. VRPTW, Part I: Route construction and local search algorithms. *Transportation Science*, 39:104–118, 2005.

[11] O. Bräysy and M. Gendreau. VRPTW, Part II: Metaheuristics. *Transportation Science*, 39:119–139, 2005.

[12] California Public Utility Commission. Advice letter number 1352, 1994.

[13] A. Campbell, L. Clarke, A. Kleywegt, and M. Savelsbergh. The inventory routing problem. In T.G. Crainic and G. Laporte, editors, *Fleet Management and Logistics*, pages 95–112. Kluwer, 1998.

[14] A. Campbell, L. Clarke, and M. Savelsbergh. Inventory routing in practice. In P. Toth and D. Vigo, editors, *The Vehicle Routing Problem*, pages 309–330. SIAM, 2002.

[15] A. Campbell and B. Thomas. Probabilistic traveling salesman problem with deadlines. *Transportation Science*, 42:1–21, 2008.

[16] I-M. Chao, B. Golden, and E. Wasil. An improved heuristic for the period vehicle routing problem. *Networks*, 26:22–44, 1995.

[17] S. Chen, B. Golden, and E. Wasil. Split delivery vehicle routing problem: Applications, algorithms, test problems, and computational results. *Networks*, 4:318–329, 2007.

[18] N. Christofides and J. Beasley. The period routing problem. *Networks*, 14:237–256, 1984.

[19] N. Christofides and S. Eilon. An algorithm for the vehicle dispatching problem. *Operational Research Quarterly*, 20:309–318, 1969.

[20] N. Christofides, A. Mingozzi, and P. Toth. The vehicle routing problem. In N. Christofides, A. Mingozzi, P. Toth, and C. Sandi, editors, *Combinatorial Optimization*, pages 315–338. John Wiley, 1979.

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

[22] J.-F. Cordeau, M. Gendreau, A. Hertz, G. Laporte, and J.-S. Sormany. New heuristics for the vehicle routing problem. In A. Langevin and D. Riopel, editors, *Logistics Systems: Design and Optimization*, pages 270–297. Springer, 2005.

[23] T. Crainic, , and H. Nourredine. Parallel meta-heuristics applications. In E. Alba et al., editor, *Parallel Metaheuristics*, pages 447–494. John Wiley and Sons, Hoboken, NJ, 2005.

[24] T. Crainic. Parallel solution methods for vehicle routing problems. In B. Golden, S. Raghavan, and E. Wasil, editors, *The Vehicle Routing Problem: Latest Advances and New Challenges*, pages 171–198. Springer, New York, 2008.

[25] G. Dantzig and J. Ramser. The truck dispatching problem. *Management Science*, 6:80–91, 1959.

[26] M. Desrochers, J. Desrosiers, and M.M. Solomon. A new optimization algorithm for the vehicle routing problem with time windows. *Operations Research*, 40:342–354, 1992.

[27] K. Doerner, R. Hartl, G. Kiechle, M. Lucka, and M. Reimann. Parallel ant systems for the capacitated vehicle routing problem. In *Evolutionary Computation in Combinatorial Optimization*, pages 72–83, 2004.

[28] B. Dorronsoro, D. Arias, F. Luna, A.J. Nebro, and E. Alba. A grid-based hybrid cellular genetic algorithm for very large scale instances of the CVRP. In W. Smari, editor, *2007 High Performance Computing & Simulation Conference (HPCS 2007)*, pages 759–765, Czech Republic, 2007.

[29] G. Dueck. New optimization heuristics: The great-deluge algorithm and the record-to-record travel. *Journal of Computational Physics*, 104:86–92, 1993.

[30] R. De Franceschi, M. Fischetti, and P. Toth. A new ILP-based refinement heuristic for vehicle routing problems. *Mathematical Programming*, 105(2–3):471–499, 2006.

[31] P. Francis, K. Smilowitz, and M. Tzur. Flexibility and complexity in periodic distribution problems. *Naval Research Logistics*, 54:136–150, 2007.

[32] M. Gendreau, A. Hertz, and G. Laporte. A tabu search heuristic for the vehicle routing problem. *Management Science*, 40(10):1276–1290, Oct. 1994.

[33] M. Gendreau, Y. Potvin, O. Bräysy, G. Hasle, and A. Løkketangen. Metaheuristics for the vehicle routing problem and its extensions: A categorized bibliography. In B. Golden, S. Raghavan, and E. Wasil, editors, *The Vehicle Routing Problem: Latest Advances and New Challenges*, pages 143–169. Springer, New York, 2008.

[34] GLPK. The GNU Linear Programming Kit. `http://www.gnu.org/software/glpk/`, 2008.

[35] B. Golden, S. Raghavan, and E. Wasil. *The Vehicle Routing Problem: Latest Advances and New Challenges.* Springer, New York, 2008.

[36] B. Golden, E. Wasil, J. Kelly, and I-M. Chao. The impact of metaheuristics on solving the vehicle routing problem: Algorithms, problem sets, and computational results. In T. Crainic and G. Laporte, editors, *Fleet Management and Logistics*, pages 33–56. Kluwer, Boston, 1998.

[37] C. Groër, B. Golden, and E. Wasil. A cooperative parallel algorithm for solving the vehicle routing problem. 2008. (submitted).

[38] E. Hadjiconstantinou, N. Christofides, and A. Mingozzi. A new exact algorithm for the vehicle routing problem based on q-paths and k-shortest paths relaxation. *Annals of Operations Research*, 61:21–43, 1995.

[39] K. Helsgaun. An effective implementation of the Lin-Kernighan traveling salesman heuristic. *European Journal of Operational Research*, 126(1):106–130, 2000.

[40] Itron Inc. `http://www.itron.com`.

[41] N. Jozefowiez, F. Semet, and E-G. Talbi. Parallel and hybrid models for multi-objective optimization: Application to the vehicle routing problem. In J. Merelo et al., editor, *Proceedings of the 7th International Conference on Parallel Problem Solving from Nature*, volume 2439, pages 271–280. Springer, 2002.

[42] N. Jozefowiez, F. Semet, and E-G. Talbi. Enhancements of NSGA II and its application to the vehicle routing problem with route balancing. In E.G. Talbi et al., editor, *Artificial Evolution*, volume 3871, pages 131–142. Springer, 2006.

[43] N. Jozefowiez, F. Semet, and E-G. Talbi. Target aiming pareto search and its application to the vehicle routing problem with route balancing. *Journal of Heuristics*, 13:455–469, 2007.

[44] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, 1983.

[45] J. Kytöjoki, T. Nuortio, O. Bräysy, and M. Gendreau. An efficient variable neighborhood search heuristic for very large scale vehicle routing problems. *Computers & Operations Research*, 47(2):329–336, 2005.

[46] Larry Levy, RouteSmart Technologies Inc. Personal communication, 2007.

[47] A. Le Bouthillier and T. Crainic. A cooperative parallel meta-heuristic for the vehicle routing problem with time windows. *Computers & Operations Research*, 32:1685–1708, 2005.

[48] A. Le Bouthillier, T. Crainic, and P. Kropf. Towards a guided cooperative search. Technical Report CRT-05-09, Centre de recherche sur les transports, Université de Montréal, Montréal, Canada, 2005.

[49] L. Levy and L. Bodin. Scheduling the postal carriers for the united states postal service: An application of arc partitioning and routing. In B. Golden and A. Assad, editors, *Vehicle Routing: Methods and Studies*, pages 359–394. North-Holland, 1988.

[50] L. Levy and L. Bodin. The arc oriented location routing problem. *INFOR*, 27:74–93, 1989.

[51] F. Li, B. Golden, and E. Wasil. Very large-scale vehicle routing: New test problems, algorithms, and results. *Computers & Operations Research*, 32:1165–1179, 2005.

[52] S. Lin and B.W. Kernighan. An effective heuristic algorithm for the traveling salesman problem. *Operations Research*, 21:2245–2269, 1973.

[53] D. Mester and O. Bräysy. Active guided evolution strategies for the large scale vehicle routing problem with time windows. *Computers & Operations Research*, 32:1593–1614, 2005.

[54] D. Mester and O. Bräysy. Active-guided evolution strategies for large-scale vehicle routing problems. *Computers & Operations Research*, 34:2964–2975, 2007.

[55] N. Moin and S. Salhi. Inventory routing problems: A logistical overview. *Journal of the Operational Research Society*, 58:1185–1194, 2007.

[56] Y. Nagata. Fast EAX algorithm considering population diversity for traveling salesman problems. In J. Gottlieb and G. Raidl, editors, *EvoCOP*, volume 3906 of *Lecture Notes in Computer Science*, pages 171–182, Berlin, 2006. Springer.

[57] Y. Nagata. Edge assembly crossover for the capacitated vehicle routing problem. In C. Cotta and J. Hemert, editors, *EvoCOP*, volume 4446 of *Lecture Notes in Computer Science*, pages 142–153, Berlin, 2007. Springer.

[58] Y. Nagata and O. Bräysy. Efficient local search limitation strategies for vehicle routing problems. In J. Hemert and C. Cotta, editors, *EvoCOP*, volume 4972 of *Lecture Notes in Computer Science*, pages 48–60, Berlin, 2008. Springer.

[59] R. Nuggehalli. personal communication, 2006.

[60] I. Or. *Traveling salesman-type combinatorial problems and their relation to the logistics of blood banking*. PhD thesis, Northwestern University, Evanston, IL, 1976.

[61] I. Osman. Metastrategy simulated annealing and tabu search algorithms for the vehicle routing problem. *Annals of Operations Research*, 41:421–451, 1993.

[62] D. Pisinger and S. Røpke. A general heuristic for vehicle routing problems. *Computers & Operations Research*, 34:2403–2435, 2007.

[63] PLPlot. The PLPlot software package. `http://plplot.sourceforge.net/`, 2008.

[64] C. Prins. A simple and effective evolutionary algorithm for the vehicle routing problem. *Computers & Operations Research*, 31:1985–2002, 2004.

[65] C. Prins. A GRASP evolutionary local search hybrid for the vehicle routing problem. In F. Pereira and J. Tavares, editors, *Bio-inspired algorithms for the Vehicle Routing Problem*, pages 35–53. Springer, 2009.

[66] T. Ralphs. Parallel branch and cut for capacitated vehicle routing. *Parallel Computing*, 29:607–620, 2003.

[67] T. Ralphs, L. Kopman, W. Pulleyblank, and L. Trotter. On the capacitated vehicle routing problem. *Mathematical Programming*, 94:343–359, 2003.

[68] C. Rego. Node-ejection chains for the vehicle routing problem: Sequential and parallel algorithms. *Parallel Computing*, 27:201–222, 2001.

[69] M. Reimann, K. Doerner, and R. Hartl. D-ants: Savings based ants divide and conquer the vehicle routing problem. *Computers & Operations Research*, 31:563–591, 2004.

[70] G. Reinelt. TSPLIB - A Traveling Salesman Problem Library. *ORSA Journal on Computing*, 3:376–384, 1991.

[71] J. Renaud, F. Boctor, and G. Laporte. An improved petal heuristic for the vehicle routing problem. *The Journal of the Operational Research Society*, 47(2):329–336, 1996.

[72] Y. Rochat and E. Taillard. Probabilistic diversification and intensification in local search for vehicle routing. *Journal of Heuristics*, 1:147–167, 1995.

[73] J. Schulze and T. Fahle. A parallel algorithm for the vehicle routing problem with time window constraints. *Annals of Operations Research*, 86:585–607, 1999.

[74] J. Sniezek and L. Bodin. Using mixed integer programming for solving the capacitated arc routing problem with vehicle/site dependencies with an application to the routing of residential sanitation vehicles. *Annals of Operations Research*, 144:33–58, 2006.

[75] E. Taillard. Parallel iterative search methods for vehicle routing problems. *Networks*, 23:661–676, 1993.

[76] E. Taillard. VRP benchmarks. `http://mistic.heig-vd.ch/taillard/problemes.dir/vrp.dir/vrp.html`, 1993.

[77] E. Taillard and B. Boffey. A heuristic column generation method for the heterogeneous fleet VRP. *RAIRO*, 33(1):1–14, 1999.

[78] C.D. Tarantilis. Solving the vehicle routing problem with adaptive memory programming methodology. *Computers & Operations Research*, 32:2309–2327, 2005.

[79] P. Toth and A. Tramontani. An integer linear programming local search for capacitated vehicle routing problems. In B. Golden, S. Raghavan, and E. Wasil, editors, *The Vehicle Routing Problem: Latest Advances and New Challenges*, pages 275–295. Springer, New York, 2008.

[80] P. Toth and D. Vigo. An overview of vehicle routing problems. In P. Toth and D. Vigo, editors, *The Vehicle Routing Problem*, pages 1–26. SIAM, 2002.

[81] UPS. Driving success: Why the UPS model for managing 103,500 drivers is a competitive advantage. `http://pressroom.ups.com/mediakits/popups/factsheet/0,1889,1201,00.html`, 2007.

[82] R. Wong. personal communication, 2006.

[83] R. Wong. Vehicle routing for small package delivery and pickup services. In B. Golden, S. Raghavan, and E. Wasil, editors, *The Vehicle Routing Problem: Latest Advances and New Challenges*, pages 475–485. Springer, New York, 2008.

[84] P.C. Yellow. A computational modification to the savings method of vehicle scheduling. *Operational Research Quarterly*, 21:281–293, 1970.