ABSTRACT

Title of Dissertation: THE BACKHAUL PROBLEM AND RELATED TOPICS IN VEHICLE ROUTING

Daniel Orlando Casco, Doctor of Philosophy, 1991

Dissertation directed by: Dr. Bruce L. Golden, Professor, College of Business and Management

The problem studied in this dissertation is a variation of the classical vehicle routing problem (VRP) that has received limited research attention, and concerns the routing of vehicles over a set of mixed customers; that is, some customers are delivery or linehaul points but others are pickup or backhaul points. In contrast to deliveries, when a vehicle services a pickup point, product, bound for the distribution center, is loaded on the truck. Practical considerations usually dictate that the number of backhauls per route is small and they are serviced near the end of a route.

The vehicle routing problem with backhauls (VRPB) can be stated as follows: Find a set of vehicle routes that service the delivery and backhaul customers such that vehicle capacity is not violated and the total distance traveled is minimized.

In this dissertation, we examine three real-world routing applications with backhauls and several first-generation algorithms designed to solve VRPBs. The key dissertation research objective is to develop new heuristics to solve the VRPB that redress the shortcomings of existing solution methods in dealing with real-world considerations. The four new heuristics developed allow common carrier or supplier deliveries, dedicated backhaul routes, and mixed routes.

In order to evaluate their performance, the heuristics were coded in Pascal and a series of computational experiments were performed on a Macintosh platform.

The experiments consisted of generating twenty seven hundred random problems covering a range of possible combinations of critical problem parameters. These problems were solved by the heuristics and the main findings are as folllows: 1) on average, the new heuristics outperformed heuristics which allow only pure delivery and mixed routes, 2) the effectiveness of the new procedures was found to vary with changes in problem size, the percentage of backhaul nodes, and the delivery node concentration region, 3) the effectiveness of the new procedures was found not to vary with changes in the cost to insert backhaul locations in mixed routes, or with changes in common carrier costs, and 4) the execution times to solve 40-node and 100-node problems was found to be less than a minute.

THE BACKHAUL PROBLEM AND RELATED TOPICS IN VEHICLE ROUTING

by

Daniel Orlando Casco

Dissertation submitted to the Faculty of the Graduate School
of the University of Maryland in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
1991

C.1
MD
Vol. I-II

Advisory Committee:

Professor Bruce L. Golden, Chairman/Advisor
Professor Saul I. Gass
Associate Professor Paul Schonfeld
Associate Professor William B. Widhelm
Associate Professor Edward A. Wasil

Maryland
LD
3231
.M70d
Casco,
D.O.
vol.1-2

©Copyright by

Daniel Orlando Casco

1991

DEDICATION

To Mary, Tony, Callie, my mother, and my father.

ACKNOWLEDGMENT

I would like to express my appreciation to Dr. Bruce L. Golden and Dr. Edward A. Wasil for their guidance and encouragement, to the rest of my advisory committee for their insightful comments, to Dr. William Stewart for his assistance with the TSP heuristics, to Sue Rein and Laura Leigh Trimmer for helping me produce the final manuscript , and to my co-workers at GE Information Services for their support. Finally, I would like to thank my wife, Mary, for her love and understanding.

TABLE OF CONTENTS

Section		Page
	DEDICATION	ii
	ACKNOWLEGMENT	iii
	LIST OF TABLES	vi
	LIST OF FIGURES	vii
1.0	INTRODUCTION	1
	1.1 Problem Statement	1
	1.2 Real-world Routing Applications with Backhauls	4
	1.3 Modeling Issues	11
	1.4 Dissertation Overview	13
2.0	LITERATURE REVIEW	15
	2.1 Review of Solution Techniques for the VRPB	15
	2.2 Limitation of Solution Procedures for the VRPB	24
3.0	DISSERTATION RESEARCH OBJECTIVES	27
4.0	NEW SOLUTION PROCEDURES FOR THE VRPB	28
	4.1 Load-based Insertion Procedure (LBI)	28
	4.2 Load-based Insertion Procedure with Post-processor (LBI+P)	37
	4.3 The Generalized VRPB Procedure (GP)	43
	4.4 The Generalized VRPB Procedure with the ARCOPT Post-processor (GP+A)	50
5.0	COMPUTATIONAL TESTING AND EVALUATION	54
	5.1 Test Environment	54
	5.2 Experimental Parameters	55
	5.3 Computational Experiments	60
	5.4 Analysis and Evaluation of Results	61
	5.4.1 Basic Results	62
	5.4.2 Sensitivity Analysis	68
	5.4.2.1 Sensitivity to Percentage of Backhaul Nodes	68
	5.4.2.2 Interaction between Problem Size and Percentage of Backhauls Nodes	74
	5.4.2.3 Sensitivity to Delivery Node Concentration	82
	5.4.2.4 Interaction between Problem Size and Delivery Concentration Percentage	88
	5.4.2.5 Sensitivity to Backhaul Penalties	97

Section		Page
5.4.2.6	Interaction between Problem Size and Backhaul Penalties	102
5.4.2.7	Sensitivity to Radial Surcharges	109
5.4.2.8	Interaction between Problem Size and Radial Surcharges	115
5.4.3	Computational Effort Comparison	123
5.5	Summary of Computational Effort	124
6.0	REAL-WORLD TEST CASE: Ace Fire	127
7.0	SUMMARY AND CONCLUSIONS	137
8.0	AREAS FOR FUTURE RESEARCH	143
APPENDIX A	LBI High Quality Solutions	146
APPENDIX B	ARCOPT	154
APPENDIX C	Computational Results	169
APPENDIX D	LBI and LBI+P Code	218
APPENDIX E	GP and GP+A Code	265
REFERENCES		332

LIST OF TABLES

Number		Page
Table 1.1	On-Site Work Classification	7
Table 1.2	Work Order Distribution.	9
Table 1.3	Response to Modeling Questions	13
Table 2.1	Algorithms for Solving the VRPB	16
Table 2.2	Algorithms for Solving the VRPB.	26
Table 5.1	Dimensions of experimental design.	61
Table 5.2	Computational experiment summary . (problem size clustering)	64
Table 5.3	Computational experiment summary (backhaul percentage clustering).	70
Table 5.4	Computational experiment summary (problem size/backhaul percentage clustering).	77
Table 5.5	Computational experiment summary (delivery concentration clustering).	84
Table 5.6	Computational experiment summary (problem size/delivery concentration clustering).	91
Table 5.7	Computational experiment summary (backhaul penalty clustering).	98
Table 5.8	Computational experiment summary (problem size/backhaul penalty clustering).	104
Table 5.9	Computational experiment summary (radial surcharge clustering).	111
Table 5.10	Computational experiment summary (problem size/radial surcharge clustering).	118
Table 5.11	Computational Effort Comparison.	124
Table 6.1	Ace Fire Locations	130
Table 6.2	Ace Fire Modeling Assumptions.	131
Table 6.3	Ace Fire Modeling Results.	131
Table 6.4	Ace Fire Sensitivity Analysis	138

LIST OF FIGURES

Number		Page
Figure 1.1	Vehicle Routing Problem with Backhauls.	3
Figure 4.1	An example of the VRPB.	29
Figure 4.2	Initial solution using Clarke-Wright . heuristic and radial links	31
Figure 4.3	A VRPB solved using the load-based insertion procedure.	36
Figure 4.4	Dedicated delivery and radial routes.	39
Figure 4.5	Load-based insertion solution.	40
Figure 4.6	Load-based insertion with post-processor solution.	41
Figure 4.7	Load-based insertion with dedicated backhaul post-processor.	42
Figure 4.8	The generalized VRPB approach.	45
Figure 4.9	Dedicated delivery and radial routes	46
Figure 4.10	Load-based insertion solution.	47
Figure 4.11	Generalized VRPB solution.	48
Figure 4.12	Generalized VRPB procedure.	49
Figure 4.13	Generalized VRPB solution	52
Figure 4.14	Generalized VRPB solution with ARCOPT post-processor	53
Figure 5.1	Scale parameter.	56
Figure 5.2	Problem size parameter	57
Figure 5.3	Percentage of backhaul nodes parameter.	57
Figure 5.4	Delivery node concentration parameter.	58
Figure 5.5	Backhaul penalty parameter.	59
Figure 5.6	Radial surcharge parameter.	60
Figure 5.7	Number of solutions better than LBI . (problem size clustering)	65
Figure 5.8	Avg. percent improvement over LBI . (problem size clustering)	66
Figure 5.9	Avg. solution length (problem size clustering).	67
Figure 5.10	Number of solutions better than LBI (backhaul percentage clustering)	71
Figure 5.11	Avg. percent improvement over LBI (backhaul percentage clustering).	72
Figure 5.12	Avg. solution length (backhaul percentage clustering).	73
Figure 5.13	Average percent improvement over LBI (40-node problems and backhaul percentage clustering)	78
Figure 5.14	Avg. percent improvement over LBI (60-node problems and backhaul percentage clustering).	79
Figure 5.15	Avg. percent improvement over LBI (80-node problems and backhaul percentage clustering).	80
Figure 5.16	Avg. percent improvement over LBI (100-node problems and backhaul percentage clustering).	81
Figure 5.17	Number of solutions better than LBI (delivery concentration clustering).	85

Number		Page
Figure 5.18	Avg. percent improvement over LBI (delivery concentration clustering).	86
Figure 5.19	Avg. solution length (delivery concentration clustering).	87
Figure 5.20	Avg. percent improvement over LBI (40-node problems and delivery concentration clustering).	93
Figure 5.21	Avg. percent improvement over LBI (60-node problems and	94
Figure 5.22	Avg. percent improvement over LBI (80-node problems and delivery concentration clustering).	95
Figure 5.23	Avg. percent improvement over LBI (100-node problems and delivery concentration clustering).	96
Figure 5.24	Number of solutions better than LBI (backhaul penalty clustering).	99
Figure 5.25	Avg. percent improvement over LBI (backhaul penalty clustering).	100
Figure 5.26	Avg. solution length (backhaul penalty clustering).	101
Figure 5.27	Avg. percent improvement over LBI (40-node problems and backhaul penalty clustering).	105
Figure 5.28	Avg. percent improvement over LBI (60-node problems and backhaul penalty clustering).	106
Figure 5.29	Avg. percent improvement over LBI (80-node problems and backhaul penalty clustering).	107
Figure 5.30	Avg. percent improvement over LBI (100-node problems and backhaul penalty clustering).	108
Figure 5.31	Number of solutions better than LBI (radial surcharge clustering).	112
Figure 5.32	Avg. percent improvement over LBI (radial surcharge clustering).	113
Figure 5.33	Avg. solution length (radial surcharge clustering).	114
Figure 5.34	Avg. percent improvement over LBI (40-node problems and radial surcharge clustering).	119
Figure 5.35	Avg. percent improvement over LBI (60-node problems and radial surcharge clustering).	120
Figure 5.36	Avg. percent improvement over LBI (80-node problems and radial surcharge clustering).	121
Figure 5.37	Avg. percent improvement over LBI (100-node problems and radial surcharge clustering).	122
Figure 6.1	Ace Fire Sites.	132
Figure 6.2	LBI solution for Ace fire.	133
Figure 6.3	GP solution for Ace fire.	134
Figure 6.4	LBI solution for Ace fire.	135
Figure 6.5	GP solution for Ace fire.	136

1.0 INTRODUCTION

In this chapter, we define the vehicle routing problem with backhauls (VRPB), present three real-world routing applications with backhauls, review the associated VRPB modeling issues, and provide an overview of the chapters that follow.

1.1 PROBLEM STATEMENT

In the most basic version of the vehicle routing problem (VRP) a set of delivery customers with known demands is to be serviced by a homogeneous fleet of fixed-capacity vehicles from a single distribution center. The objective is to route the vehicles so that all customers are serviced and the total distance traveled by the fleet is minimized. Usually, vehicles leave with a full load and return empty to the depot. The basic VRP is an extremely complex combinatorial problem that has been studied extensively in the literature over the last twenty years. Golden et al. [9] provide us with a comprehensive study of the basic problem and its many dimensions and variations. One of the variations that has received limited research attention concerns the routing of vehicles over a set of mixed customers; that is, some customers are delivery or linehaul points and others are **pickup** or **backhaul** points. In contrast to deliveries, when a vehicle services a pickup point, a load of product is accepted that is bound for the distribution center. As an example, such a customer mix occurs in the grocery industry. Supermarkets are the delivery points and grocery suppliers (such as poultry processors or vegetable and fruit wholesalers) are the pickup points.

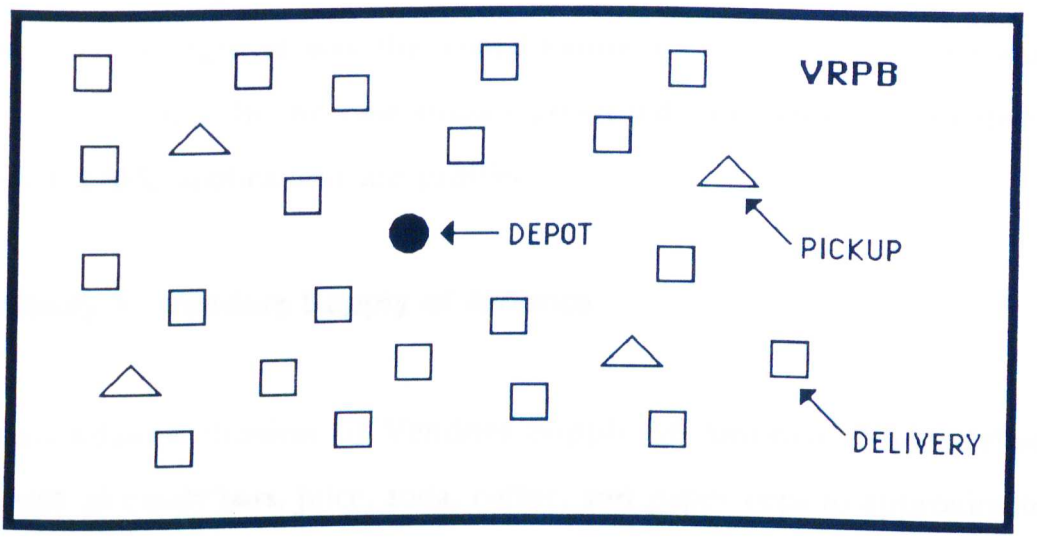
In recent years, the grocery industry has recognized the cost-cutting potential of servicing backhaul points as a part of all-delivery routes. The industry trimmed its distribution costs (saving upwards of $160 million in 1982) by allowing (empty) returning vehicles to pick up inbound products. Keep in mind that practical considerations (most vehicles are rear-loaded so that rearranging on-board delivery loads is difficult and deliveries are more important than pickups) usually dictate that the number of backhauls per route is **small** and that they are serviced **near the end of a route**.

Given this as background, the **vehicle routing problem with backhauls (VRPB)** can be stated as follows: Find a set of vehicle routes that service the delivery and backhaul customers such that vehicle capacity is not violated and the total distance traveled is minimized (see *Figure 1.1*).

There are two operations research problems similar to the VRPB - - the dial-a-ride problem[10] and the general pickup and delivery problem[13]. In the dial-a-ride problem, customers (such as the elderly or the handicapped) request to be picked up and transported to a specific destination. Thus, each pickup has its own destination and pickups can occur any place within the route. A natural extension to the VRPB is the general pickup and delivery problem. In this problem, a vehicle makes a pick up at one location and delivers the product to a second location that is not necessarily the depot. This type of operation occurs with messenger services.

VRPB Problem Statement

Find a set of vehicle routes that services the delivery **and** backhaul (pickup) customers such that vehicle capacity is not violated and total distance traveled is minimized



Backhauls serviced near the end of a route

Deliveries are high-priority stops
Small number of backhauls
Difficult to rearrange on-board load in rear-loaded vehicles

Figure 1.1 Vehicle Routing Problem with Backhauls.

1.2 REAL-WORLD ROUTING APPLICATIONS WITH BACKHAULS

In a study prepared for the National Council of Physical Distribution Management, Kearney[14] reports that a survey of companies revealed that the number one program (83% of survey respondents) for improving productivity in logistics was the coordination of inbound with outbound freight distribution. In the case studies presented next, three rather diverse real-world VRPB applications are profiled.

Case Study 1 : Vendors Supply of America

The mid-Atlantic division of Vendors Supply of America (VSA) delivers items such as candy bars, juice, soda, coffee, and paper cups to approximately 1,000 customers located in Virginia, Maryland, Delaware, New Jersey, and Pennsylvania from its Bridgeport, New Jersey location. About 600 customers are considered "regular" since they require deliveries every week, every other week, or perhaps once every six weeks. The remaining 400 customers are serviced infrequently, some only on a seasonal basis.

VSA currently operates a fleet of twelve tractor/trailer trucks to service its customers. Truck routes are manually constructed with each truck making several delivery stops per day. In addition, whenever possible, the dispatcher tries to incorporate backhaul points onto the routes. At these points, drivers will pick up items such as cranberry juice or potato chips destined for the depot.

VSA stocks about 1,200 items at the New Jersey depot for delivery to vendors, users of office coffee machines, and movie theaters. VSA also supplies items such as candy bars to various organizations for fund raising activities and has recently extended its assortment of delivery items to include fresh bakery pastry. All delivery items (except pastry) are pre-sorted and shrink wrapped into individualized pallets at the depot. This permits quick and easy delivery at the customer locations.

During the delivery portion of the route, fully-loaded trucks carry between 15 and 25 pallets. VSA prefers not to "cube out" its trucks (that is, load trucks with 44 pallets, double stacked), since this practice can result in damage to both delivery items and trucks. Once deliveries are completed, no more than 18 pallets are accepted for the inbound trip. The capacity slack is necessary to accommodate empty pallets or a refused load that might remain in the truck from the delivery activities. Once the merchandise is picked up from the supplier, VSA is responsible for safely transporting it back to the depot. Also, with fewer loaded pallets on board, product damage tends to be minimized.

The dispatcher manually constructs truck routes on a daily basis. Typically, a truck visits seven to twelve customer locations in a particular geographic area on a designated day of the week. For example, on a recent Monday, five trucks serviced customers in Norfolk and Richmond, two serviced Roanoke, and five serviced Delaware and Pennsylvania. Most routes are local trips requiring a total of eight to nine driving hours, while some routes (especially those involving pastry) are overnight trips requiring ten to fifteen driving

hours. On the long trips, drivers might leave on Sunday for example, stay overnight, service the delivery stops on Monday, and then return to the depot. The depot tracks trucks throughout the day by requiring the driver to periodically call in. Drivers are provided with a routing sheet that lists the order of the delivery stops and the expected arrival times. Customers know when to expect deliveries and deliveries are usually made within one hour of the expected time.

It is VSA's policy to schedule all backhauls at the end of a driver's route. In all cases, drivers have appointed hours at which to pick up the product at the backhaul location. Since VSA is a service company and its customers have come to expect timely deliveries, it does not want to delay any customer deliveries because of problems at a backhaul location (such as lack of a loading dock). In rare instances, if a truck is running late, another truck may be dispatched to the backhaul location.

VSA recognizes that significant cost savings can result by incorporating backhaul locations onto delivery routes and has encouraged its suppliers to provide allowances for product pick up. For example, one candy supplier offers an $800 credit per pick up; others offer a $1.00 per case allowance or $1.15 per hundred weight when VSA picks up the product. Over a three-month period, VSA estimates that it has saved over $41,000 by picking up products destined for the depot. In addition, VSA has some of its supplies delivered directly to the depot by vendors or common carriers when it makes economic or practical sense.

Case Study 2: Ace Fire Extinguisher Service

Ace Fire Extinguisher Service, Inc. services and sells fire extinguishers in the Washington, D.C. metropolitan area. Their customer base includes about 14,000 locations and extends from Baltimore, Maryland in the north to Manassas, Virginia in the south. Each customer is visited at least once annually for on-site service, delivery, or pick up of extinguishers. The various on-site activities are listed in *Table 1.1* below.

Service	Delivery	Pickup
inspect and tag extinguishers	serviced extinguishers	extinguishers to be serviced at the depot
recharge water-pressure extinguishers	new extinguishers and cabinets	old, unusable extinguishers
tear down or recharge dry-chemical units		trade-ins and loaners
install extinguishers and cabinets		
service fire alarm systems		

Table 1.1 On-site work classification.

Each day servicemen are given a stack of work orders to complete. These work orders fall into the following five categories listed in the order of importance (priority) :

- appointments for service,

- annual service repeat visit,
- VIP calls,
- delivery of new extinguishers, and
- annual service first visit.

This list has been ranked by management according to the priority of the work. The most important calls are the appointments that require the driver to visit a customer location at a specified time. Customers with service contracts are visited (usually without an appointment) so that annual service work (ASW) may be performed. ASW may require two (or more) visits to complete a job – – an initial visit to pick up extinguishers and a repeat visit to deliver or install serviced units. In order to ensure timely work completion, the repeat ASW visit ranks second on this list. Next on the list are the VIP calls. Customers in this category require service within 48 hours. Delivery of new extinguishers, cabinets, and other activities follow in importance. At the bottom of the list are first visit annual service calls.

In a typical week, servicemen receive close to 500 work orders and complete approximately 80% of them (about 400 calls are completed and 100 are missed). *Table 1.2* illustrates a mix of calls for a typical week. The first column gives the distribution of the 400 completed calls and the second column gives the distribution of the 100 missed calls among the five work-order categories.

Table 1.2 Work order distribution.

	Calls Completed	Calls Missed
Service Appointments	21.9%	8.4%
ASW(repeat visit)	25.7%	7.3%
VIP Calls	14.3%	0.0%
Delivery of Equipment	12.3%	5.3%
ASW(first visit)	25.8%	79.0%

The company does not employ a computer-assisted method to route its fleet of ten trucks. Servicemen are responsible for sequencing the customer visits. The clustering of work orders is based upon customer location, an eight-hour shift, and dispatcher experience. At present, there are ten customer clusters. Four encompass the immediate Washington, D.C. area, four are located in suburban Maryland, and the remaining two are in suburban Virginia. Note that trucks are seldom strained in terms of their extinguisher carrying capacity. It is customary for management to split up large work orders at the same location over several visits.

Although some of the traditional backhaul characteristics are not present in Ace's distribution environment (such as customers with inbound loads

constrained by vehicle capacity), several features of this problem lend themselves to a backhaul-like model. Recall that in a typical week servicemen are missing about 100 calls – – 79 ASW first visit calls and 21 higher priority calls. A detailed examination of the actual daily routes covered during a recent week revealed that servicemen often do not follow management's work-order ranking when sequencing their stops. Low-priority stops (such as ASW first visits) are frequently visited early in the route (especially when they are located near customers with appointments). In many cases, however, this precludes more important, higher-priority stops (such as ASW repeat visits) from being visited later in the route.

In order to serve its client base more effectively, management would like low-priority calls (ASW first visits and delivery of new extinguishers) to be visited near the end of a route instead of early-on. It may be desirable to model the low-priority calls as backhauls points – – that is, customers that are visited towards the end of a route in order that more important customers can be serviced first. Treating low-priority calls in this way should help to produce routes that serve customers in a more timely manner by decreasing the number of missed high-priority calls.

Case Study 3: Nassau Suffolk Express, Inc.

Nassau Suffolk Express is a small trucking company located in Bohemia (Suffolk County), New York. The company delivers business forms, computer paper, and related items from its warehouse to customer locations in and around New York City. In particular, deliveries are made to New Jersey, Connecticut, Westchester County, Nassau County, and New York City on a regular basis. There are also occasional deliveries to Boston and Pennsylvania.

On a typical day, 5 drivers each make about 18 deliveries. Each driver also makes 1 or 2 pickups per week on the way back to the depot in Suffolk County. Additional pickups are made by a sixth (dedicated pickup) vehicle. The driver of this vehicle spends half of his time working in the warehouse. The other half is spent driving primarily between the depot and various stops in New York City; each route consists of 10 to 12 pickups. Reserving vehicles for **pickups only** is a strategy that has been employed in other industries as well (such as distribution in the fast food and grocery industries). In other words, Nassau Suffolk Express seeks to handle backhauls effectively by employing two complementary strategies.

1.3 MODELING ISSUES

In thinking about designing heuristic methods to solve the VRPB, it is first necessary to specify the type of backhaul situation under consideration. A number of important questions, such as the following, come to mind.

1. What is the percentage of stops that are backhauls?
2. Must backhauls always come after all deliveries?
3. If the answer to the second question is no, then are there specific rules defining when a backhaul is allowed?
4. What is the size of a typical backhaul relative to vehicle capacity?
5. Is the number of trucks (drivers) fixed in advance?
6. Can vehicles run multiple trips per day?
7. Must all backhauls be serviced?
8. Can backhauls be split?
9. Will a vehicle ever service multiple backhauls only?

Answers to these questions will dictate the type of backhaul problem that needs to be addressed.

In *Table 1.3*, we attempt to answer these questions for the three case studies presented in this dissertation. Our answers are somewhat tentative since we don't always have all the information that we would like. Nonetheless, we have done our best to provide correct responses. Although we cannot say that the responses are indicative of the vast majority of backhaul or backhaul-like scenarios, we can observe that definite differences do exist among the three cases.

Table 1.3 Response to modeling questions.

Question Number	Case 1	Case 2	Case 3
1.	approx. 10-30%	approx. 50%	approx.10-20%
2.	Yes	No	Yes
3.	NA	No	NA
4.	NK	NA	NK
5.	Yes	Yes	Yes
6.	No	No	Rarely, but possible
7.	No	No	No
8.	No	No	No
9.	No	Rarely, but possible	Yes

NA— Not Applicable
NK— Not Known

1.4 DISSERTATION OVERVIEW

This dissertation will focus on solving the VRPB. In the chapters that follow :

- Existing heuristic procedures for solving the VRPB are described;
- Limitations of the existing heuristic procedures are discussed;
- New heuristic procedures are developed and coded to address the shortcomings of existing solution methods;
- A comprehensive set of computational experiments are conducted to evaluate these new procedures;
- A real-world routing problem with backhauls is analyzed to confirm our experimental findings.

In addition, the dissertation includes an empirical study of recently-developed heuristics which solve the traveling salesmen problem. These heuritics were used in our development of new and efficient heuristics for solving the VRPB.

2.0 LITERATURE REVIEW

In this chapter, we examine several the first-generation heuristics for solving the VRPB. For each heuristic, we outline its structure, describe the basic concepts used in constructing it, identify the key backhaul problem assumptions associated with each, and summarize its strengths and limitations in solving real-world VRPB problems.

2.1 REVIEW OF SOLUTION TECHNIQUES FOR THE VRPB

Several heuristics for solving the VRPB have been developed and tested. For the most part, these heuristics are modified versions of five procedures that have been used to solve the basic vehicle routing problem or the traveling salesmen problem: Clarke-Wright Savings [3], Cheapest Insertion [8], Spacefilling Curves [6], Branch and Bound [19], and Bender's Decomposition Approach [12]. The modified algorithms that have been recently developed are summarized in *Table 2.1* In developing these algorithms, different types of backhaul problem assumptions were considered.

For example, Goetschalckx and Horsley [6] assume that backhauls **always** come after all deliveries. Golden et al. [8] establish rules that permit pickups before all deliveries are completed. *Table 2.1* includes information about the type of backhaul problem that each algorithm can solve. It is important to keep this in mind when reading the discussion that follows about each algorithm, the key developmental issues, and the computational testing results.

Table 2.1 Algorithms for solving the VRPB.

Article	Routing Algorithm	Modifications	Constraints	Modeling Assumptions
Deif and Bodin [4]	CW	Delay connection of pickups to deliveries by altering savings definition to include a penalty multiplier	• Vehicle capacity • Route length	• No deliveries after pickups • 10%, 30%, 50% of points are backhauls
Golden et al. [8]	CW CI	CI inserts backhauls onto CW all-delivery routes. Insertion costs altered to penalize the number of deliveries after a pickup	• Vehicle capacity • Route length	• Deliveries after pickups are allowed • Many more deliveries than pickups • About 10% of points are backhauls
Goetschalckx and Horsley [6]	SFC	Use SFC and GM or KM to generate all-delivery and all-backhaul routes. Routes are then merged and improved using Two-Opt and Three-Opt procedures	• Vehicle capacity	• No deliveries after pickups • About 20%,33%, 50% of all points are backhauls
Yano et al [19]	SC	Generate initial feasible solution heuristically. Optimization is done using a set-covering routine.	• Vehicle weight and volume • Driving time • Total time on road • Open/close times at destination	• No deliveries after pickups • Maximum of 4 deliveries and 4 pickups per route
Jacobs-Blecha and Goetschalck [12]	BD GAPH TSPH	A heuristic based on Bender's decomposition was applied to a modified formulation of the VRPB. Three sub-problems are solved. Two correspond to the clustering decisions for delivery and backhaul points which are independent GAPs. The third consists of K independent TSPs with an additional constraint	• Vehicle capacity	• No deliveries after pickup • About 20%,33%, 50% of all points are backhauls

BD Bender's Decomposition
CI Cheapest Insertion
CW Clarke-Wright
GAPH Generalized Assignment Problem Heuristic
GM Greedy Method
KM K-Median Method
SC Set Covering
SFC Spacefilling Curves
TSPH TSP Heuristic

A straightforward attempt to handle backhauls using the Clarke-Wright procedure was first proposed by Deif and Bodin [4]. The Clarke-Wright procedure is one of the best known methods for generating solutions to the basic vehicle routing problem. In this procedure, if delivery stops i and j are joined on the same route, then a savings of:

$$S(i, j) = C(1, i) + C(1, j) - C(i, j) \qquad (2.1)$$

results, where C(i, j) is the cost of linking i and j. As a first step, Deif and Bodin computed the usual savings in (2.1) for all customer pairs but imposed the condition that pickups must occur after deliveries. Therefore, once a backhaul point is placed at the end of a route, no delivery points can be added after that backhaul. In general, this modification is not satisfactory since individual routes are terminated quite early, resulting in a large number of shortened routes.

To form longer routes that satisfy the pickups-only-after-deliveries condition, Deif and Bodin proposed to delay the connection of pickups with deliveries. This was accomplished by altering the savings definition to include a term that penalizes (and, hence, delays) the connection of a pickup and delivery. The backhaul savings definition is given by:

$$BS(i, j) = C(1, i) + C(1, j) - C(i, j) - \pi s \qquad (2.2)$$

where s is an estimate of the maximum savings,

π is a penalty multiplier, and

i is a backhaul and j is not or vice versa.

The S(i,j) and the BS(i,j) terms (as defined in (2.1) and (2.2), respectively) are computed for all possible i and j combinations, then sorted, and inserted as in the Clarke-Wright method. Note that the πs term assures that links joining delivery and pickup routes are introduced late in route construction.

Finally, Deif and Bodin performed a computational experiment to test the effectiveness of the savings modification in solving the backhaul problem with various choices of the penalty multiplier (π). Test problems of 100, 200, and 300 customers for which 10, 30, or 50 per cent were specified as backhauls were generated. Ten values of π from .05 to .50 were tried. Four values of π (.05, .10, .15, and .20) produced the best (least total distance) routes.

A different approach that solves the backhaul problem using an insertion-based procedure was proposed by Golden et al. [8]. Routes are first developed for all of the deliveries and then backhaul points are considered for insertion between stops on these routes. The cost of inserting a backhaul point k on a route between stops i and j is specified by:

$$\mathbf{C(k) = C(i, k) + C(k, j) - C(i, j) + Pd} \qquad (2.3)$$

where P is a penalty multiplier, and

d is the number of deliveries following k.

The penalty multiplier serves to delay the insertion of the backhauls until the end or nearly the end of a route. For each backhaul k, the insertion cost is specified by the minimum C(k) for all links (i,j). We point out that once a backhaul is inserted the direction of that route is **fixed** in the direction that

generated the minimum C(k). Since the decision to insert a backhaul is based on the number of delivery stops remaining after the backhaul is serviced, we refer to this as a **stop-based backhaul insertion procedure**.

Golden et al. [8] evaluated the performance of the Deif and Bodin method [4] and the stop-based backhaul insertion procedure. Note that in this analysis the Deif and Bodin method was modified to allow some deliveries after a pickup. Golden et al. [8] generated a set of three problems, each consisting of 50 deliveries and 5 backhauls. In both procedures, as the penalty term is increased (π is increased over the range .05 to .50; P over the range .50 to 15.50), the number of deliveries after pickups decreases to zero while the total route length increases.

Deif and Bodin's method produced lower cost routes when backhauls were required to be at the very end of the route. The stop-based backhaul insertion procedure generally produced shorter routes when a backhaul was allowed to be inserted as soon as vehicle capacity permits.

The heuristic procedure that is used by Goetschalckx and Horsley [6] to solve the backhaul problem is based on the notion of spacefilling curves developed by Bartholdi and Platzman [1]. Delivery and backhaul points are first transformed from the plane to the line using the spacefilling curve transformation and then sorted in non-decreasing order according to their position along the line. That is, the transformation assigns a number between 0 and 1 to each point. Next, each set of points is clustered independently to

form all-delivery and all-backhaul routes. (We point out that clustering and routing are essentially accomplished simultaneously -- – once points are clustered they are then visited in the order in which they were sorted.) Two different methods, one a greedy method and the other an approach based on the solution to the K-median problem, are used to cluster the points. The clustering step using the greedy method proceeds as follows. A new route is formed starting with the point not yet on a route having the smallest assigned number. Larger and larger numbered points are added to the route until the truck is full. A new route is then started and the method is repeated until all points are placed on a route. To form clusters using the K-median method first requires specifying K, the number of routes to be formed. The line is then divided into K identical intervals and the medians are selected to be the points closest to the midpoints of the intervals. Clustering begins by selecting the point not yet on a route having the smallest assigned number. If truck capacity remains, this point is assigned to route R, where R is the closest median point. If the truck is overloaded, this point is assigned to route R + 1. The method is repeated until all points are placed on a route. If all points cannot be placed on a route, K is increased by one and the clustering is repeated. With this method, points which violate vehicle capacity are "pushed" to the next cluster. If there are an insufficient number of clusters, the line is then further divided until enough clusters are formed so that all points may be placed on a route. After applying the clustering step to obtain all-delivery and all-backhaul routes, each delivery route is merged with a corresponding (nearby) backhaul route to form one route in which backhauls take place after all deliveries have been made.

To examine the performance of the two spacefilling curve heuristics, Goetschalckx and Horsley generated 15 random test problems, ranging in size from 25 to 200 points, with 20% to 50% specified as backhauls. For each problem, truck capacity was varied over several values to produce a total of 57 test problems. Routes produced by the greedy method and the K-median method for this set of problems were improved using a Two-opt procedure followed by a Three-opt procedure. The authors concluded that both methods generated a set of routes with comparable total lengths, but the greedy method used fewer trucks and utilized their capacities more effectively.

Yano et al. [19] report on the development of a microcomputer-based routing system that incorporates backhaul locations into truck routes for a retail chain consisting of 40 stores with a fleet of 11 vehicles. The route generation procedure is a two-step operation that consists of finding an initial feasible solution using a route-construction heuristic followed by a set-covering routine that uses a branch-and-bound framework to find the optimal solution. They report finding solutions to problems with 20 to 40 pickup and delivery locations in 10 to 30 minutes on a microcomputer. All pickups followed deliveries and the authors point out that because of truck volume and route distance constraints routes with as many as 4 stops were rare. The authors also discuss a model that incorporates common carrier trucks when pickups and deliveries exceed the capacity of the eleven-vehicle fleet.

Jacobs-Blecha and Goetschalckx [12] develop a Bender's Decomposition approach to solving the VRPB. There are two subproblems that correspond to

the clustering decisions for the linehaul and backhaul points which are independent Generalized Assignment Problems (GAPs). The third subproblem consists of K independent, single-route TSPs. These problems have the additional side constraints required to form all-delivery and all-backhaul routes.

Their basic solution algorithm is described below.

1. **Initialization**. Define initial location and shape of clusters based on the location and demands of the customers and suppliers.
2. **GAP Coefficients**. Estimate cost coefficients for the clustering subproblems (GAPs) for both delivery and pickup locations.
3. **GAP Solution**. Solve the GAPs using a Lagrangian relaxation technique.
4. **Route Solution**. Solve the constrained TSP for each cluster formed in Step 3. Select an initial "artificial" interface link (that is, a link connecting a backhaul point with a delivery point) for each cluster. Find independent Hamiltonian paths from interface link delivery point to depot, and from interface link backhaul point to depot. Use Two-opt and Three-opt procedures to improve the solution. Repeat this process for different interface links until best solution is found.
5. **Acceptance Testing**. If the current solution is acceptable, then stop; otherwise, return to Step 2.

Since no benchmark problems are available in the literature, the authors perform an empirical comparison of their algorithm and the modified Clarke-Wright algorithm proposed by Deif and Bodin [4]. The parallel version of the Clarke-Wright algorithm is used. In this implementation, the set of partial routes grows as the savings list is scanned. If a link on the savings list cannot be inserted due to capacity constraints or because there are no common endpoints, then a new partial route is started immediately. If a link is encountered which has nodes on the ends of two existing partial routes, those partial routes are concatenated into a larger partial route by the current link from the savings list, as long as route constraints are not violated.

The computational test environment consisted of the following.

- Fifteen test problems were generated with supplies and demands uniformly distributed with a mean of 500 and a standard deviation of 200.
- A total of 62 problems resulted, varying in size from 25 to 150 total customers.
- Linehaul problem sizes were 20, 30, 45, 75,100 points.
- Pickup points made up 25%, 50%, and 100% of the deliveries.
- Vehicle capacity was varied depending on average number of points per route desired.
- All procedures were coded in Microsoft Pascal with similar data structures and executed on an 8 MHZ IBM-AT.

An analysis of the computational results showed the following.

- The decomposition procedure yields on average a 2.4% improvement in total distance over the modified Clarke-Wright algorithm suggested by Deif and Bodin.
- As the problem size increases, the revised decomposition procedure consistently yields better solutions than the modified Clarke-Wright algorithm suggested by Deif and Bodin.
- The modified Clarke-Wright algorithm needed only one third of the time required by the decomposition procedure to yield a solution which on average was only 2.4% longer.
- On average, the execution times for 50-node and 100-node problems using the modified Clarke-Wright algorithm were 20 and 100 seconds, respectively.
- On average, the execution times for 50-node and 100-node problems using the decomposition procedure developed by Jacobs-Blecha and Goetschalckx [12]were 110 and 500 seconds, respectively.

2.2 LIMITATIONS OF SOLUTION PROCEDURES FOR THE VRPB

Our review of the first-generation algorithms reveals that existing solution procedures have limitations in solving real-world VRPB problems.
In particular, the algorithms are limited:

- With respect to the ability to address critical real-world constraints,
- In that most cases they are too slow and cumbersome for microcomputer implementation, and

- In that they only solve the mixed route backhaul option (that is, the mixed delivery and backhaul route). These algorithms do not take into account the economic trade-offs that become available when common carrier services, supplier delivery services, or dedicated backhaul fleets are used.

A summary of the advantages and disadvantages of the first-generation algorithms reviewed is presented in *Table 2.2.*

It is important to note that optimal procedures for solving the VRPB are not available in the literature today. Furthermore, benchmark VRPBs with known optimal solutions are also not available in the literature.

Table 2.2 Algorithms for solving the VRPB.

Article	Routing Algorithm	Advantages	Disadvantages
Deif and Bodin [4]	CW	• It relies on a simple savings criterion which outperforms more rigorous approaches	• Procedure never looks more than one step ahead. • Often produces routes with low utilization • Number of routes not known in advance • High ranking of outlying points creates routes in a circumferential manner vis-a-vis the depot
Golden et al. [8]	CW CI	• Relies on simple insertion procedure • Initial VRP tour not tied to any specific solution procedure • Backhaul insertion eases transition from established delivery routes to mixed routes	• Does not take into account load considerations
Goetschalckx and Horsley [6]	SFC GM KM	• Computationally very fast • Does not require a distance matrix • User specifies number of vehicles	• Not accurate enough • Does not handle route time or distance constraints
Yano et al. [19]	SC	• Exact solution approach	• Only applicable to special VRPB cases • Difficult to extend to large, more general problems
Jacobs-Blecha and Goetschalckx [12]	BD GAPH TSPH	• Theoretically elegant • Yields balanced looking routes • User specifies number of vehicles • Yields good solution	• Does not handle route time or distance constraints • Too slow computationally

BD Bender's Decomposition
CI Cheapest Insertion
CW Clarke-Wright
GAPH Generalized Assignment Problem Heuristic
GM Greedy Method
KM K-Median Method
SC Set Covering
SFC Spacefilling Curves
TSPH TSP Heuristic

3.0 DISSERTATION RESEARCH OBJECTIVES

Keeping in mind the limitations of existing VRPB procedures, the dissertation objectives are as follows:

- To develop new heuristics to solve the VRPB that address the shortcomings of existing solution methods in dealing with real-world VRPB applications,
- To enhance these new heuristics in order to produce integrated solutions that consist of complementary backhaul servicing options such as common carrier or supplier deliveries, dedicated backhaul fleets, and the most commonly modeled **mixed route** option, and
- To code the new heuristics on a microcomputer and then to evaluate the accuracy, the performance, and the sensitivity of these new heuristics on a set of test problems.

4.0 NEW SOLUTION PROCEDURES FOR THE VRPB

In this chapter, we describe and illustrate four new solution procedures for the VRPB. These procedures produce integrated solutions that consist of complementary backhaul servicing options.

4.1 LOAD-BASED INSERTION PROCEDURE (LBI)

An alternative to the stop-based backhaul insertion procedure of Golden et al. [8] is a load-based backhaul insertion procedure for the VRPB. This heuristic features four basic improvements to the stop-based procedure.

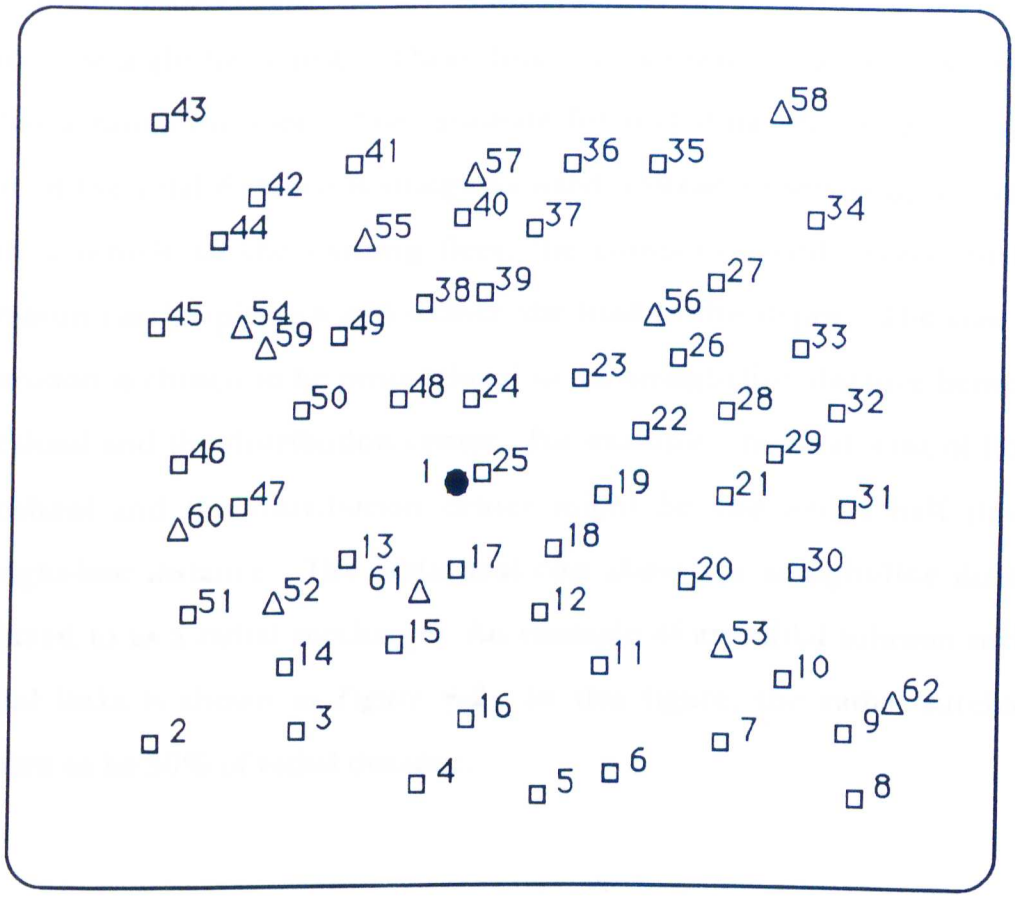
1. An initial feasible solution to the VRPB is generated by using a Clarke-Wright solution for the delivery points and a straight-line (backhaul to distribution center) route for each pickup point.

2. The decision to insert a backhaul considers the delivery load that remains on the vehicle after the backhaul is serviced.

3. Solutions to the VRPB are filtered through a set of multi-criteria rules that can help to identify high-quality solutions.

4. Initial Clarke-Wright solutions are generated with route capacities smaller than the actual vehicle capacities in order to increase the possibilities for profitable backhaul insertions.

In the remainder of this section, the load-based procedure is examined in detail and the effects that the four improvements have on generating solutions to the VRPB are discussed. *Figure 4.1* illustrates a typical VRPB.



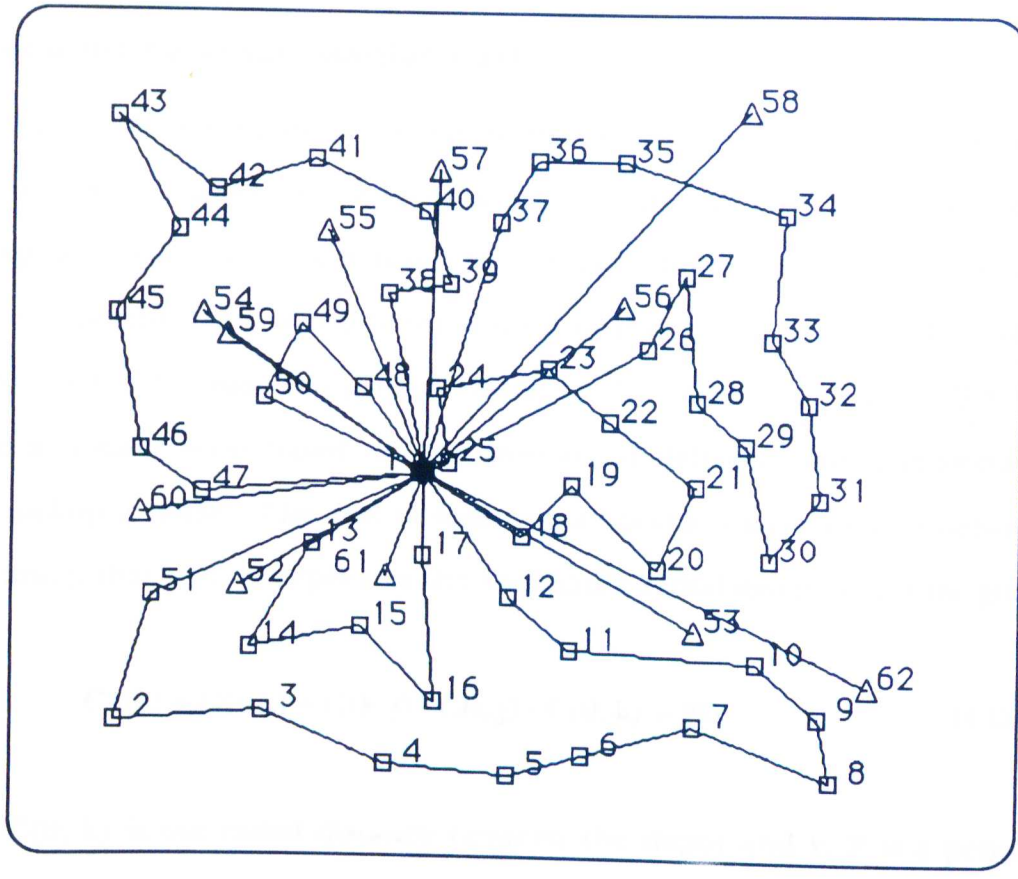
Locations	Number	Symbols
Distribution Center	1	●
Delivery	50	□
Backhaul	11	▲

Parameters	Settings
Vehicle Capacity	160
Vehicle Speed (mph)	20
Pickup Load	40

Figure 4.1 An example of the VRPB.

Generating an Initial Solution

As in the stop-based procedure, the Clarke-Wright heuristic is used to generate a set of routes for the delivery locations. To incorporate the backhaul locations into the initial solution, each backhaul is joined to the distribution center by a direct (straight-line) link. These links are known as radial links and they entail a radial distance. The rationale for including the radial distances as part of the total distance is straightforward. Instead of servicing the backhaul with a vehicle in the existing fleet, the company could choose to have a common carrier pick up and deliver the load to the depot. The cost of this operation is chosen to be proportional to the straight-line distance between the backhaul and the distribution center. For example, the total cost of linking a backhaul and the distribution center might be one and a half times the straight-line distance. The additional cost above the straight-line distance is referred to as a radial surcharge. An example of an initial solution including radial links is shown in *Figure 4.2*. In this figure, the radial surcharge is chosen to be 50% of radial distance.



Clark-Wright distance = 585

Radial distance = 284

Radial surcharge = 142
(.5 x 284)

Total distance = 1011

Figure 4.2. Initial solution using Clarke-Wright heuristic and radial links.

Modifying the Backhaul Insertion Cost

In the the stop-based insertion procedure the decision to insert a backhaul was based on the number of delivery stops remaining after the backhaul. Applying this procedure could result in a large delivery following a backhaul since the specific load at each remaining delivery stop is not taken into account. The load-based approach modifies the insertion cost in (2.3) to include a penalty term based on the amount of delivery load that remains after a pickup is made. The cost of inserting a backhaul k on a route between stops i and j, that also incorporates the backhaul's radial distance, is now given by:

$$C^*(k) = C(i, k) + C(k, j) - C(i, j) - C(0, k) + PD \qquad (4.1)$$

Where C(0, k) is the radial distance between the depot and k, P is a penalty multiplier, and D is the number of delivery units after servicing backhaul k.

This load-based insertion cost should allow more flexibility in servicing backhauls than does the stop-based procedure. The stop-based procedure treats all delivery stops equally without regard to the delivery load. The new load-based procedure recognizes that not all deliveries are of the same size and that insertions prior to the end of a route might be considered if the delivery load that remains is small enough. With the cost shown in *(4.1)*, inserting a backhaul before several delivery stops with small loads might now look appealing.

The driver can make the pickup and is not inconvenienced by having to shuffle large on-board delivery loads to accommodate the backhaul load. This type of information is not incorporated into the stop-based procedure where only remaining stops (and not remaining load) are counted.

Identifying High-Quality Solutions

By varying the penalty multiplier in the load-based insertion procedure, many candidate solutions to the VRPB can be generated. If P is small, the resulting solutions will contain backhauls too early on a route, nearly filling the truck with delivery and pickup loads and, therefore, inconveniencing the driver by forcing him to rearrange the on-board load. Typically, this also increases the amount of service time at each customer location. In many distribution operations, deliveries are more important than pickups. Management prefers that most of the units on a route be delivered before pickups are considered. Keeping these observations in mind, we propose a set of rules that can help identify high-quality, feasible solutions to the VRPB. In particular, these rules require the following.

Rule 1. A vehicle servicing a particular set of customers should deliver a pre-determined percentage of its load before accepting any backhauls.

Rule 2. If there is a delivery after a backhaul, then accept the pickup as long as no more than a pre-determined percentage of the vehicle is full after accepting the pickup.

Rule 3. If no deliveries remain, then accept all backhauls as long as vehicle capacity permits.

The first rule ensures that a substantial portion of the units are delivered before backhauls are accepted. The second rule ensures that some vehicle capacity for load shuffling remains after accepting a backhaul. The final rule allows the entire capacity of the vehicle to be used for pickups once all deliveries have been completed.

An illustration of the application of these high-quality solution rules is shown in *Appendix A*.

Increasing the Flexibility of Making Backhaul Insertions

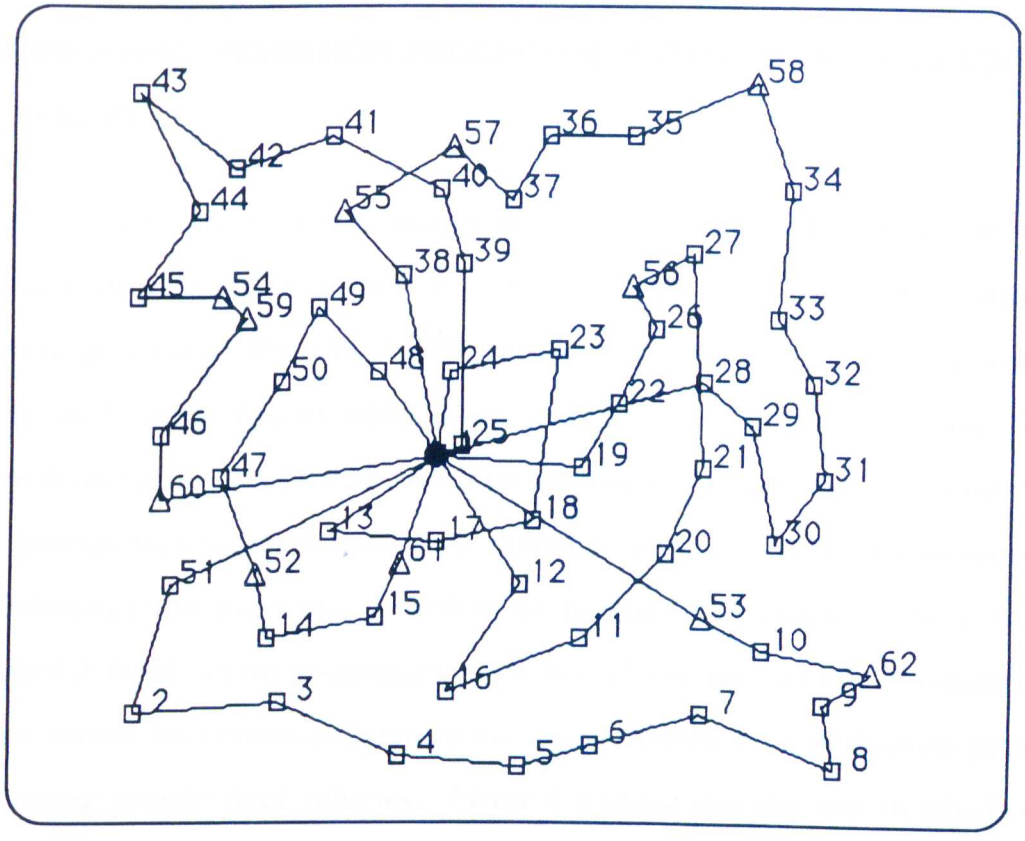
In the initial solution phase, it appears that when the Clarke-Wright procedure generates individual routes that service a large number of delivery locations little flexibility remains to insert backhauls. With a few long-distance routes composed of many deliveries, there are small number of low-cost places to insert backhauls. The backhauls are usually "squeezed" onto the end of a route. To remedy this type of "squeeze play", the Clarke-Wright procedure is initiated with an artificial vehicle capacity that is less than the actual vehicle capacity. For example, vehicles might have an actual capacity of 160 units. However, to produce an initial solution that is composed of more routes with fewer deliveries per route, the capacity of the vehicles might be set at 140 units during the Clarke-Wright solution phase. More routes with fewer deliveries usually mean unused vehicle capacity so that backhauls can sometimes be

inserted in locations that are more appealing than near the end of a route. *Figure 4.3* illustrates the results of applying the load-based insertion procedure to our sample configuration.

Note that the initial Clarke-Wright solution was developed using a capacity constraint of 140 units so as to provide unused vehicle capacity for the insertion of backhaul locations.

The final solution shown in *Figure 4.3,* is composed of six routes with five drivers completing the routes within a eight-haul work day. Note that driver three serves routes three and four, and that the average number of hours on the road per driver is 5.78 hours.

This procedure was implemented in Pascal and the code used to evaluate the procedure is presented in *Appendix D.*



Route	Driver	Hrs/Driver
1	1	7.2
2	2	5.8
3	3	6.7
4	3	-
5	4	6.5
6	5	7.7
	Average	**5.78 Hrs./ Driver**

Figure 4.3 A VRPB solved using the load-based insertion procedure.

4.2 LOAD-BASED INSERTION PROCEDURE WITH POST-PROCESSOR (LBI+P)

This procedure improves the routes developed by the load-based insertion procedure described in *Section 4.1*. For example, in *Figure 4.4* we show a randomly generated 40-node VRPB problem for which a set of dedicated delivery and radial routes have been generated. These routes are then improved using the load-based insertion procedure to produce a solution in which pickup locations are served on mixed routes(that is, routes containing both deliveries and pickups), as well as on pickup-only routes. This is shown in *Figure 4.5*. Next, a post-processor is used to evaluate the benefit of combining pickups served by common carriers on radial routes into dedicated pickup routes using private fleet vehicles. *Figure 4.6* illustrates the way in which the configuration shown in *Figure 4.5* has been improved through the use of the private fleet alternative for pickups.

The new procedure can be summarized as follows.

Step 0. Identify delivery, pickup, and depot locations.

Step 1. Apply the Clarke-Wright procedure to delivery locations only.

Step 2. Calculate radial distances for pickup locations.

Step 3. Apply load-based insertion procedure to radials and the Clarke-Wright solution.

Step 4. Apply the Clarke-Wright procedure to rejected pickups.

Step 5. Retain the dedicated pickup routes if they produce a savings over the radial distances associated with the pickup locations in the radial routes.

Figure 4.7 illustrates the six steps in this procedure. It is important to note that this procedure will yield solutions at least as accurate as the load-based insertion procedure by itself, since dedicated pickup routes will only be accepted if they yield shorter distances than the individual radial links.

This procedure was implemented in Pascal and the code used to evaluate the procedure is presented in *Appendix D.*

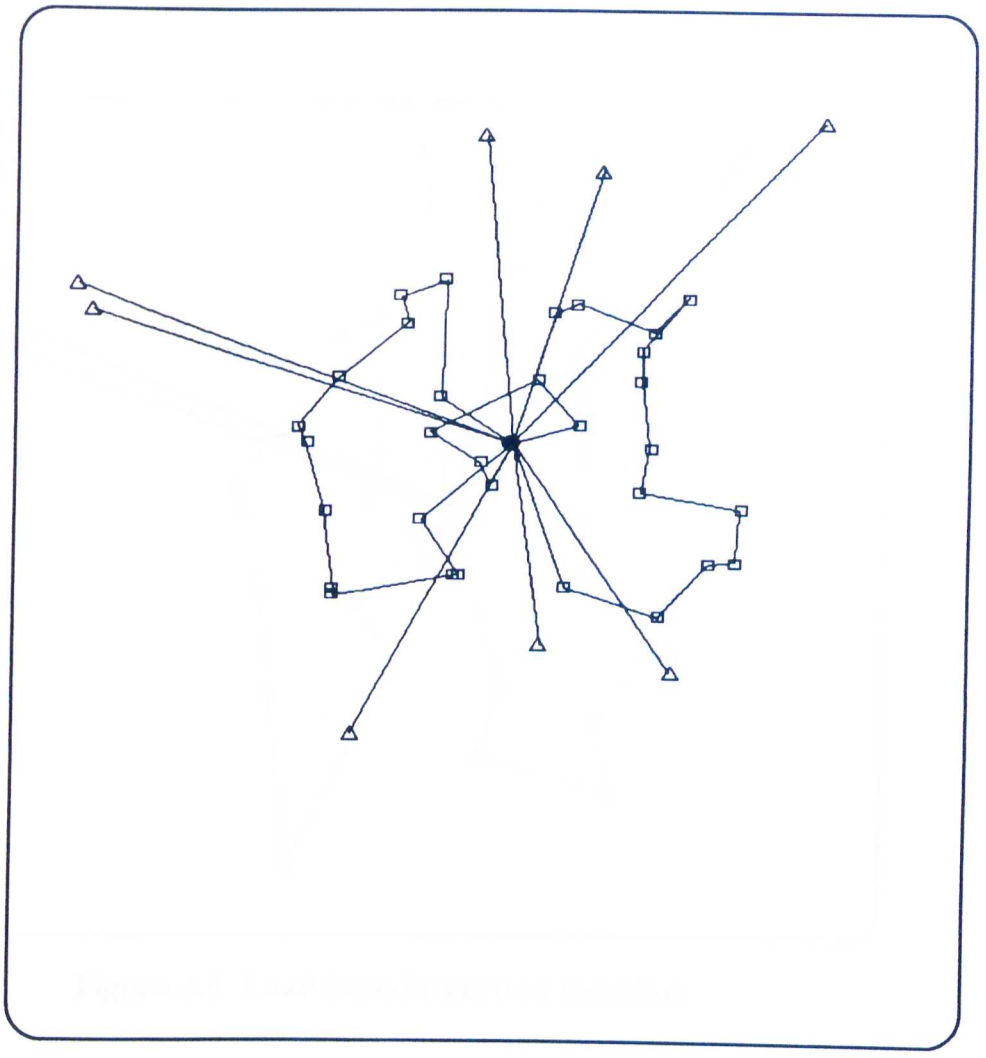


Figure 4.4 Dedicated delivery and radial routes.

This figure shows a randomly generated 40-node VRPB and the corresponding dedicated delivery routes that were generated using the Clarke-Wright procedure. It also shows the radial routes for the pickup locations. The total distance of this solution is 1,520 units.

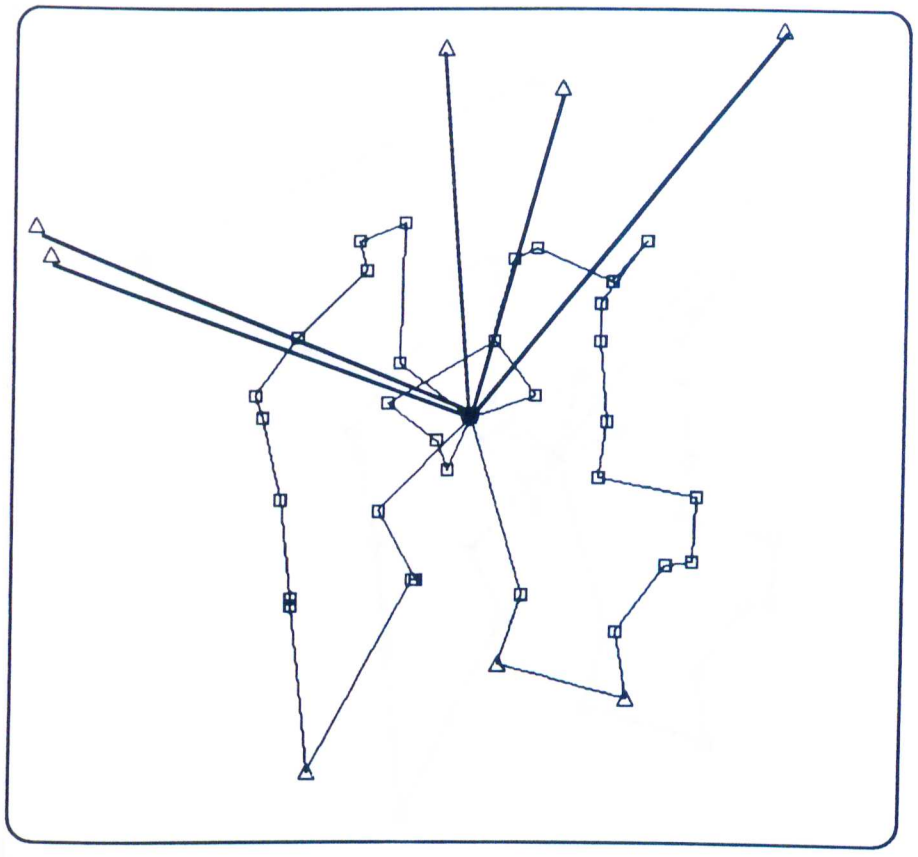


Figure 4.5 Load-based insertion solution.

This figure shows the solution that results from using load-based insertion procedure. Three pickup locations have been inserted onto two delivery routes while the remaining five pickup locations are served by dedicated radial routes. The total distance of this solution is 1,398 units.

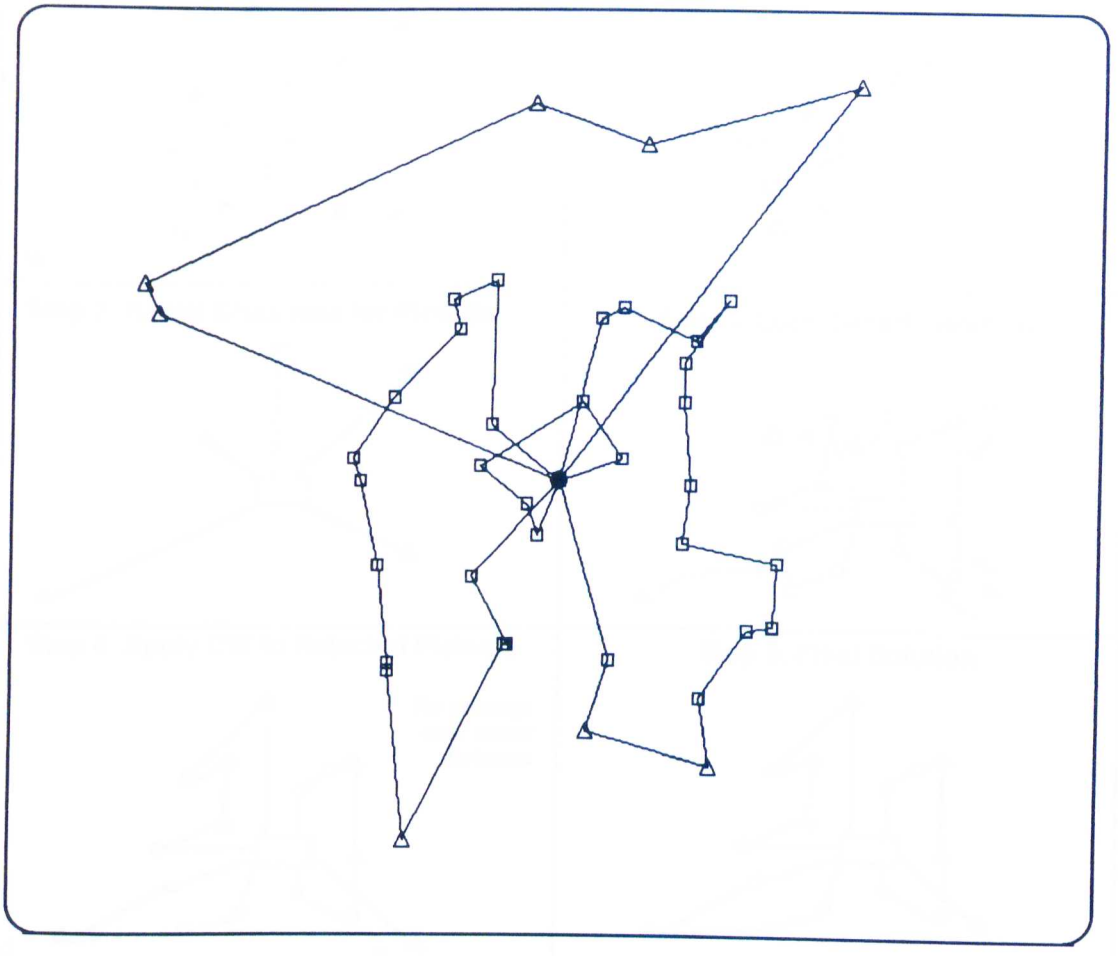


Figure 4.6 Load-based insertion with post-processor solution.

Five pickup locations are combined into a dedicated pickup route to reduce the total distance to 1,299 units.

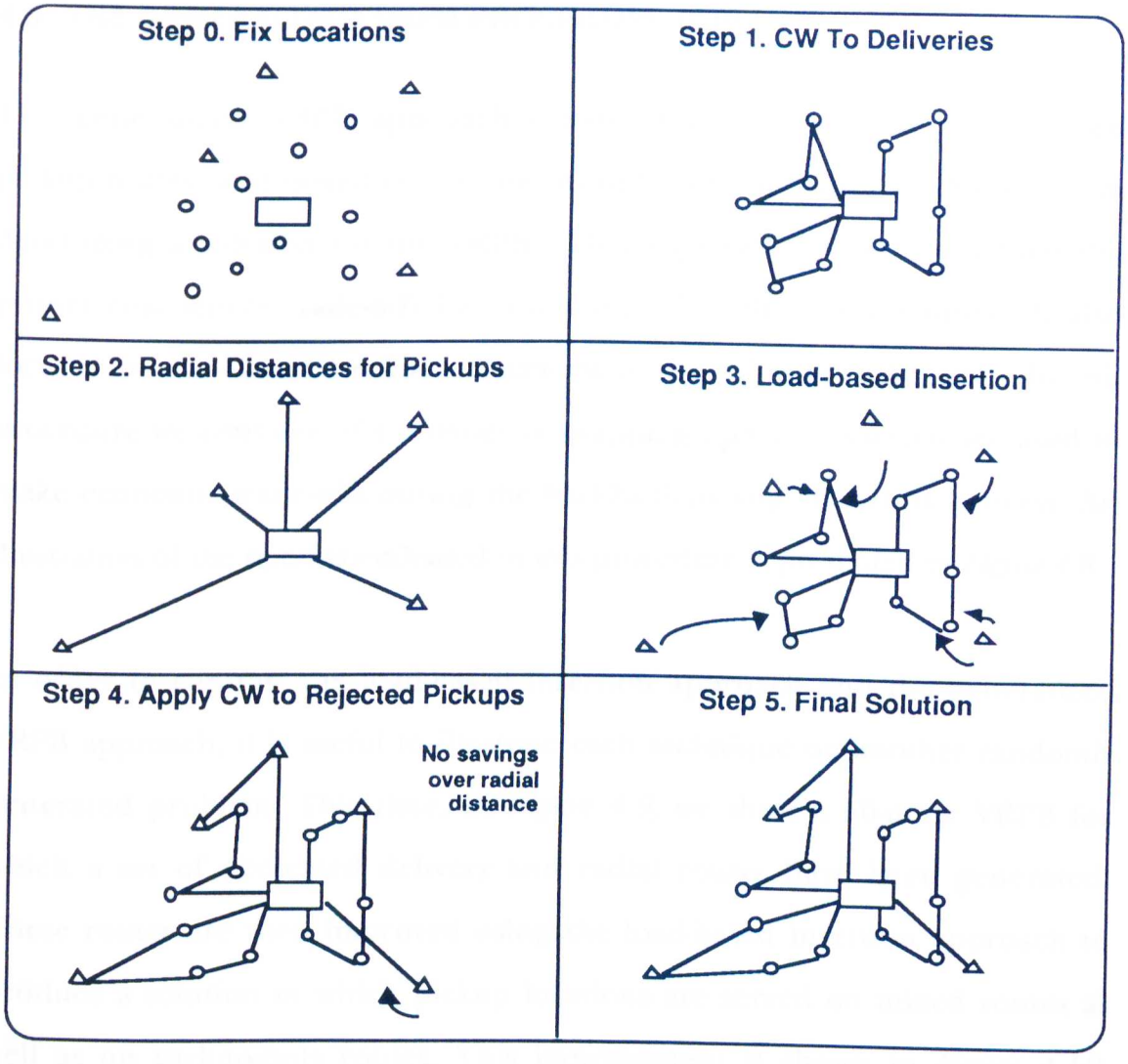


Figure 4.7 Load-based insertion with dedicated backhaul post-processor.

4.3 THE GENERALIZED VRPB PROCEDURE (GP)

The generalized VRPB approach considers common carriers, dedicated pickup routes, and mixed routes (routes with both pickups and deliveries) in developing a solution for the VRPB. This approach focuses on setting the proper cost-benefit trade-offs for evaluating all of these alternatives. It also focuses on the distance reduction aspects of the solution technique. In this procedure we consider of a number of swapping operations which are used to make economic trade-offs during the backhaul(pickup) insertion process. An illustration of the options evaluated in this procedure is presented in *Figure 4.8.*

In order to contrast the load-based insertion approach and the generalized VRPB approach, it is useful to illustrate each technique on another randomly generated problem. This time, in *Figure 4.9,* we show a 50-node VRPB for which a set of dedicated delivery and radial routes have been generated. These routes are then improved using the load-based insertion approach to produce a solution in which pickup locations are served on mixed routes as well as on pickup-only routes. This improvement is shown in *Figure 4.10.* Next, we solve the same problem using the GVRPB technique to produce the solution shown in *Figure 4.11.* This example illustrates a case where the generalized VRPB approach yields a more attractive solution than the load-based insertion approach.

The generalized VRPB procedure requires the following seven steps.

Step 0. Identify delivery, pickup, and depot locations.

Step 1. Apply the Clarke-Wright procedure to delivery locations only.

Step 2. Calculate radial distances for pickup locations.

Step 3. Apply the Clarke-Wright procedure to pickup locations only.

Step 4. Compare the distances of dedicated pickup routes against the distances of radial routes.

Step 5. Accept dedicated pickup routes if they yield a savings over the radial routes and if they also meet a route acceptance criteria based on the number of backhauls per route.

Step 6. Take the remaining Pickups(that is, those pickups not included in the dedicated pickup routes), and determine whether they would be most cost-effectively served by inserting them in a mixed route, or in a dedicated pickup route, or by a common carrier route(radial route).

Note that route acceptance criteria referred to in **Step 5** of the generalized VRPB procedure is designed to discourage the acceptance of dedicated backhaul routes with a small number of nodes even through they may provide a lower-cost alternative to individual radial routes. During our experimentation, these type of routes tended to restrict the cost saving opportunities of the pickup insertion process which followed in **Step 6**.

Figure 4.12 illustrates these seven steps. This procedure was implemented in Pascal and the code used to evaluate the procedure is presented in *Appendix E.*

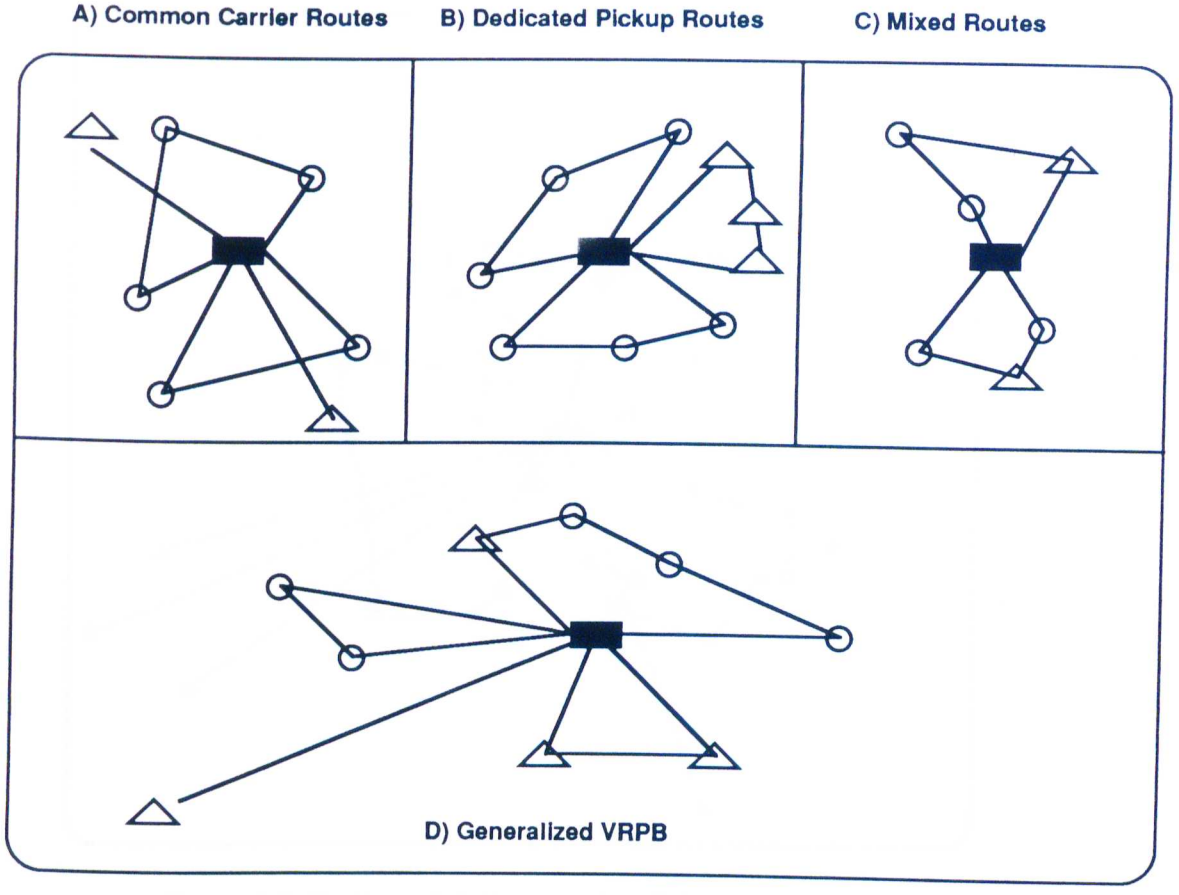


Figure 4.8 The generalized VRPB approach.

In panel A, we show an in-house fleet servicing deliveries and common carriers servicing pickups. In panel B, we show an in-house fleet servicing deliveries and pickups in separate routes. In panel C, we show an in-house fleet servicing deliveries and pickups in mixed routes. Finally, in Panel D a combination of all possible delivery and pickup routes is shown.

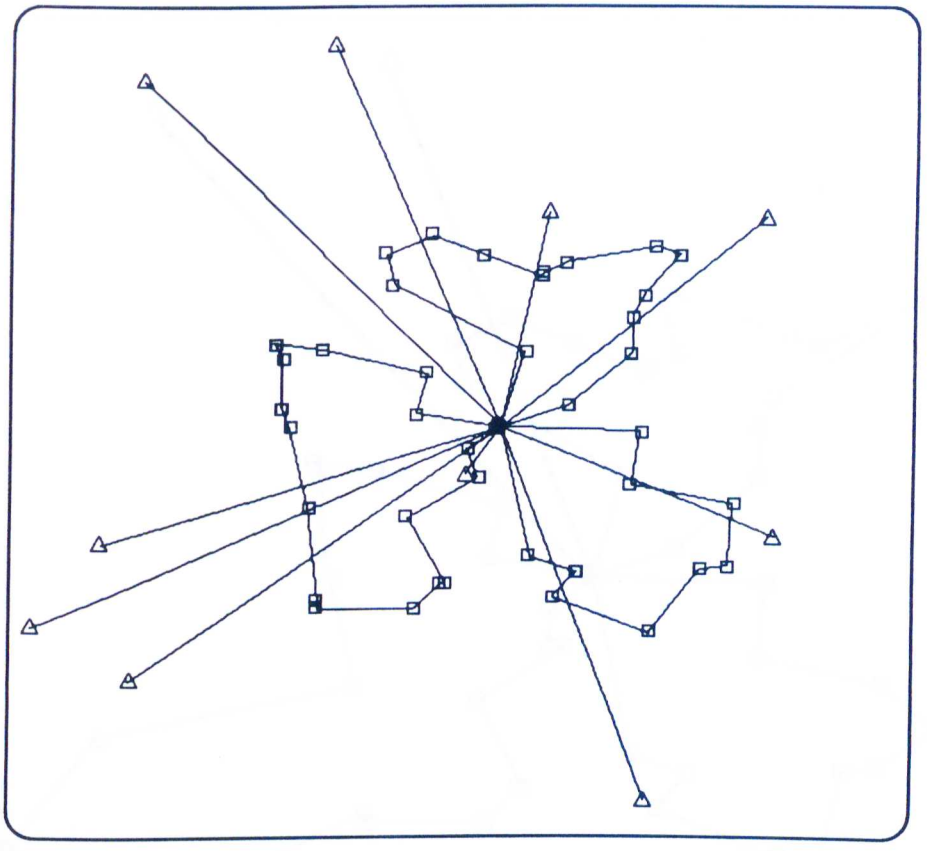


Figure 4.9 Dedicated delivery and radial routes.

This figure shows a randomly generated 50-node VRPB. The dedicated delivery routes were generated using the Clarke-Wright procedure. There are radial routes for the pickup locations. The total distance of this solution is 1,643 units.

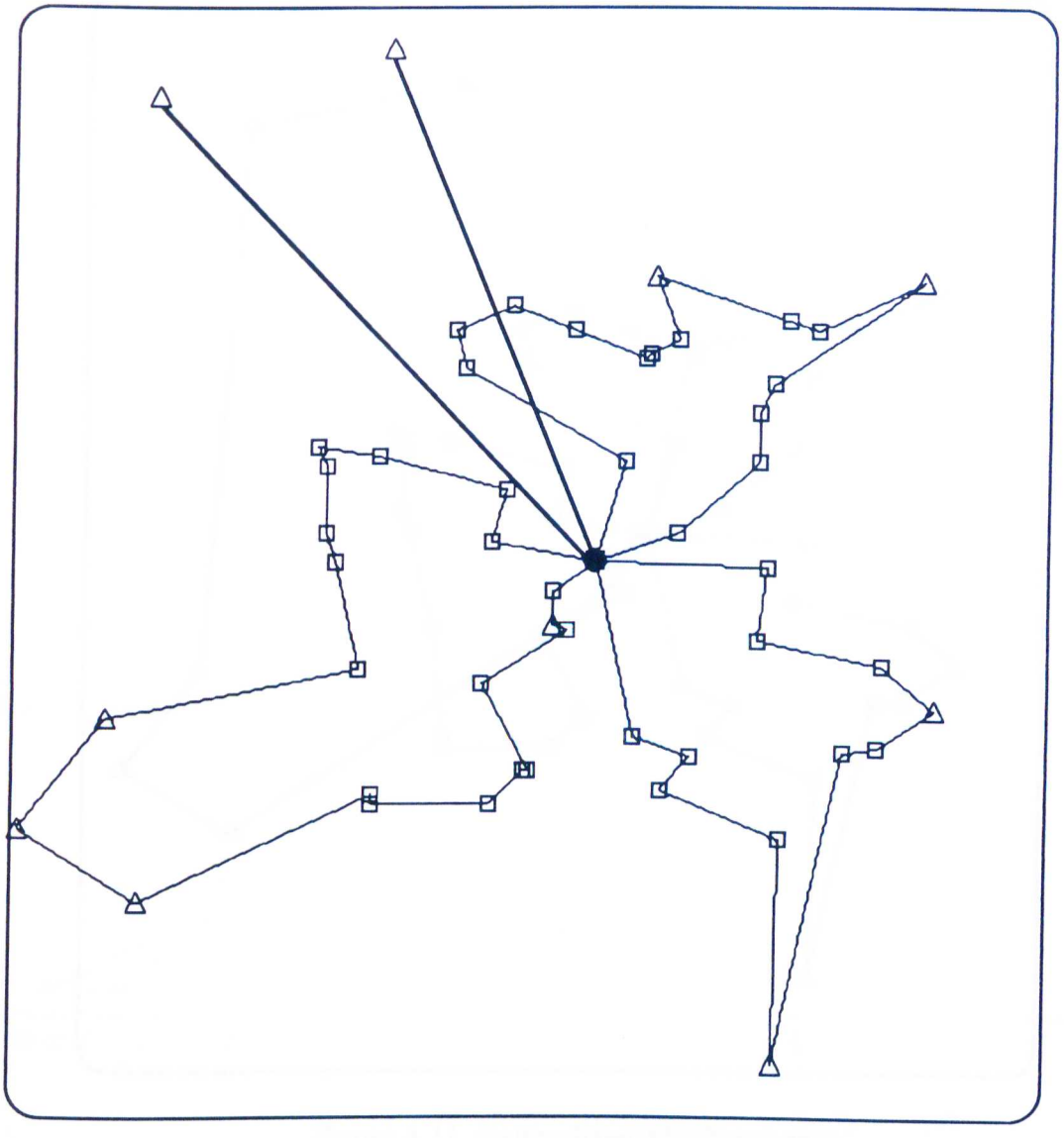


Figure 4.10 Load-based insertion solution.

This figure shows the resulting load-based insertion solution. Eight pickup locations have been inserted onto three delivery routes while the remaining two pickup locations are served by dedicated radial routes. The total distance of this solution is 1,373 units.

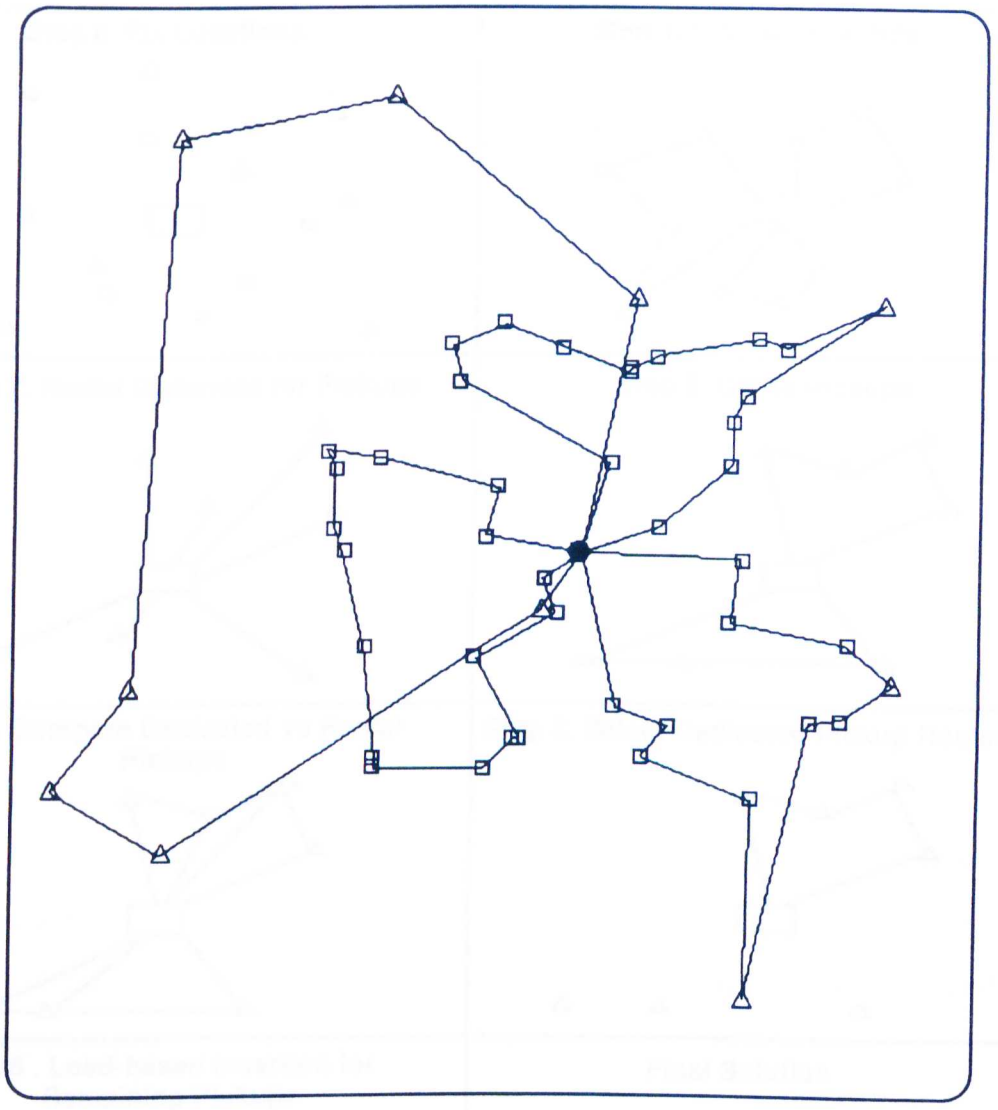


Figure 4.11 Generalized VRPB solution.

This figure shows the resulting generalized VRPB solution. Three pickup locations have been inserted onto two delivery routes while the remaining seven pickup locations are served by a dedicated pickup route. The total distance of this solution is 1,355 units.

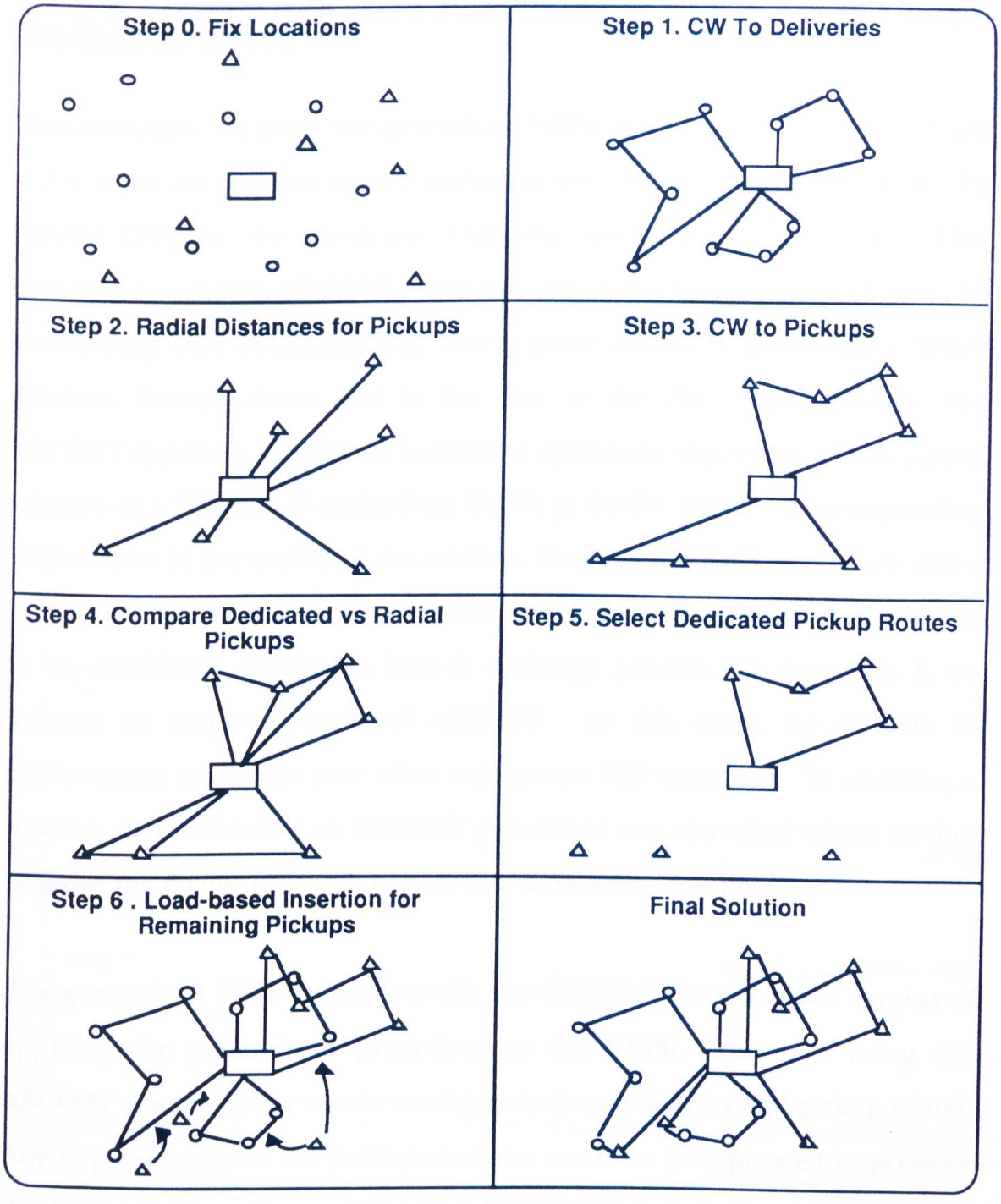


Figure 4.12 Generalized VRPB procedure.

4.4 THE GENERALIZED VRPB PROCEDURE WITH THE ARCOPT POST-PROCESSOR (GP+A)

This approach combines the generalized VRPB procedure described in *Section 4.3* with an accelerated branch exchange heuristic procedure developed by Stewart [16] for the symmetric Traveling Salesman Problem (TSP). This procedure - called ARCOPT - derives improvements in performance by considering only exchanges that have a good chance of producing a better solution. Stewart shows that in the case of the Three-Opt heuristic, the ARCOPT approach reduces the number of operations required to obtain a good solution to a TSP with N nodes from $O(N^3)$ to $O(N^2)$, without a corresponding degradation in the quality of the solution. Stewart's ARCOPT procedure uses a number of mutually exclusive minimal spanning trees to determine the arcs to be considered during the branch exchange process. In Appendix B, we present an empirical study of ARCOPT. In this study, we confirm its performance advantage over other well-known TSP heuristics. In addition, a number of refinements to ARCOPT procedure are identified which further improve the quality of its solution, as well as its execution time.

The generalized VRPB procedure with the ARCOPT post-processor consists of applying the generalized procedure to the VRPB, and then using the ARCOPT post-processor on the resulting dedicated delivery and pickup routes. Any improvements to the dedicated routes are then incorporated to produce the final solution. *Figures 4.13* and *4.14* illustrate a case where the ARCOPT procedure yields an improved final solution.

The generalized VRPB procedure with the ARCOPT Post-processor is as follows.

Step 0. Identify delivery, pickup, and depot locations.

Step 1. Apply the Clarke-Wright procedure to delivery locations only.

Step 2. Calculate radial distances for pickup locations.

Step 3. Apply the Clarke-Wright procedure to pickup locations only.

Step 4. Compare the distances of dedicated pickup routes against. radial routes.

Step 5. Accept dedicated pickup routes if they yield a savings over the radial routes and if they also meet a route acceptance criteria based on the number of backhauls per route.

Step 6. Take the remaining pickups (that is, those pickups not included in the dedicated pickup routes) and determine whether they would be most cost-effectively served by inserting them in a mixed route, or in a dedicated pickup route, or by a common carrier route(radial route).

Step 7. Use the ARCOPT procedure to improve dedicated delivery and pickup routes.

This procedure was implemented in Pascal and the code used to evaluate the procedure is presented in *Appendix E.*

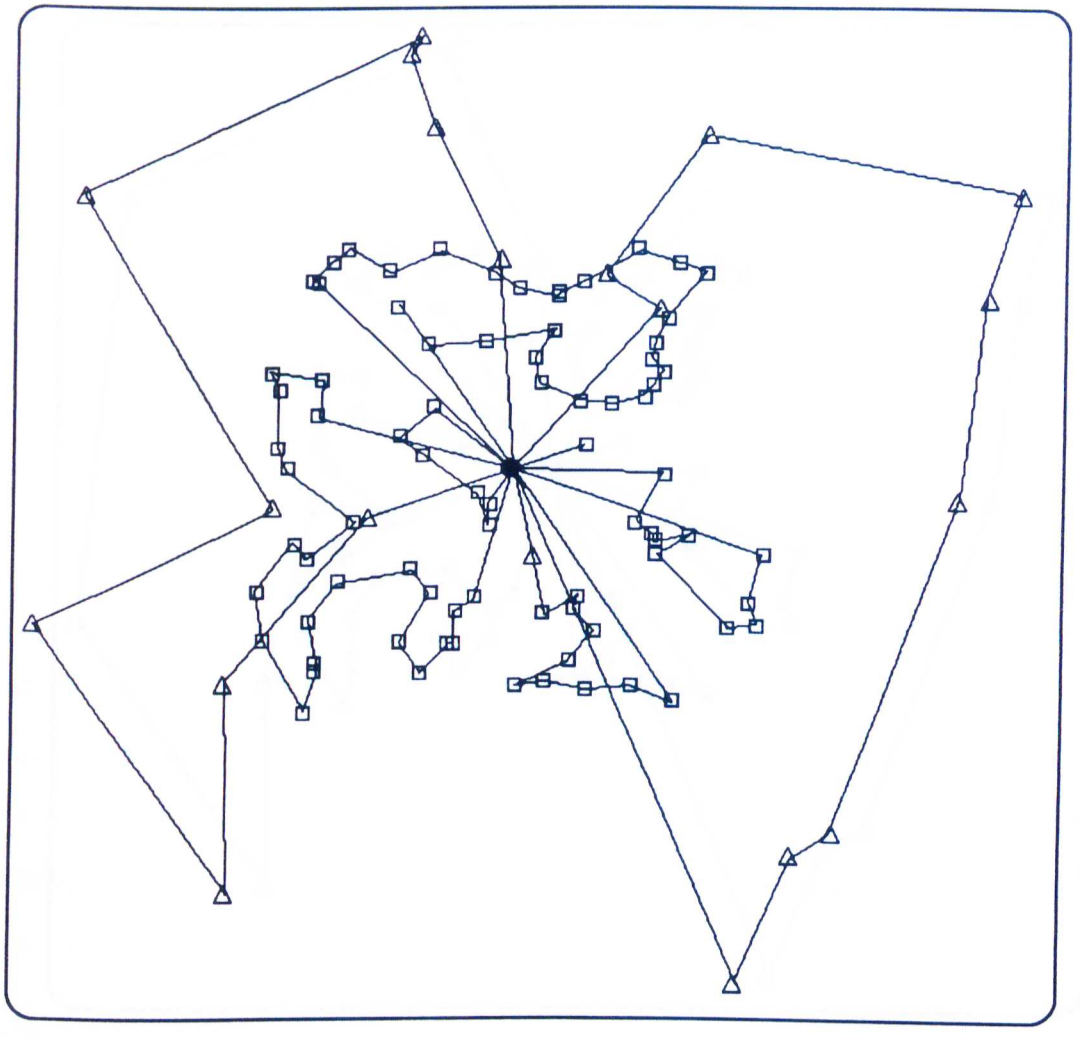


Figure 4.13 Generalized VRPB solution.

This figure shows the resulting generalized VRPB solution for a randomly generated 100-node problem. The total distance of this solution is 2,150 units.

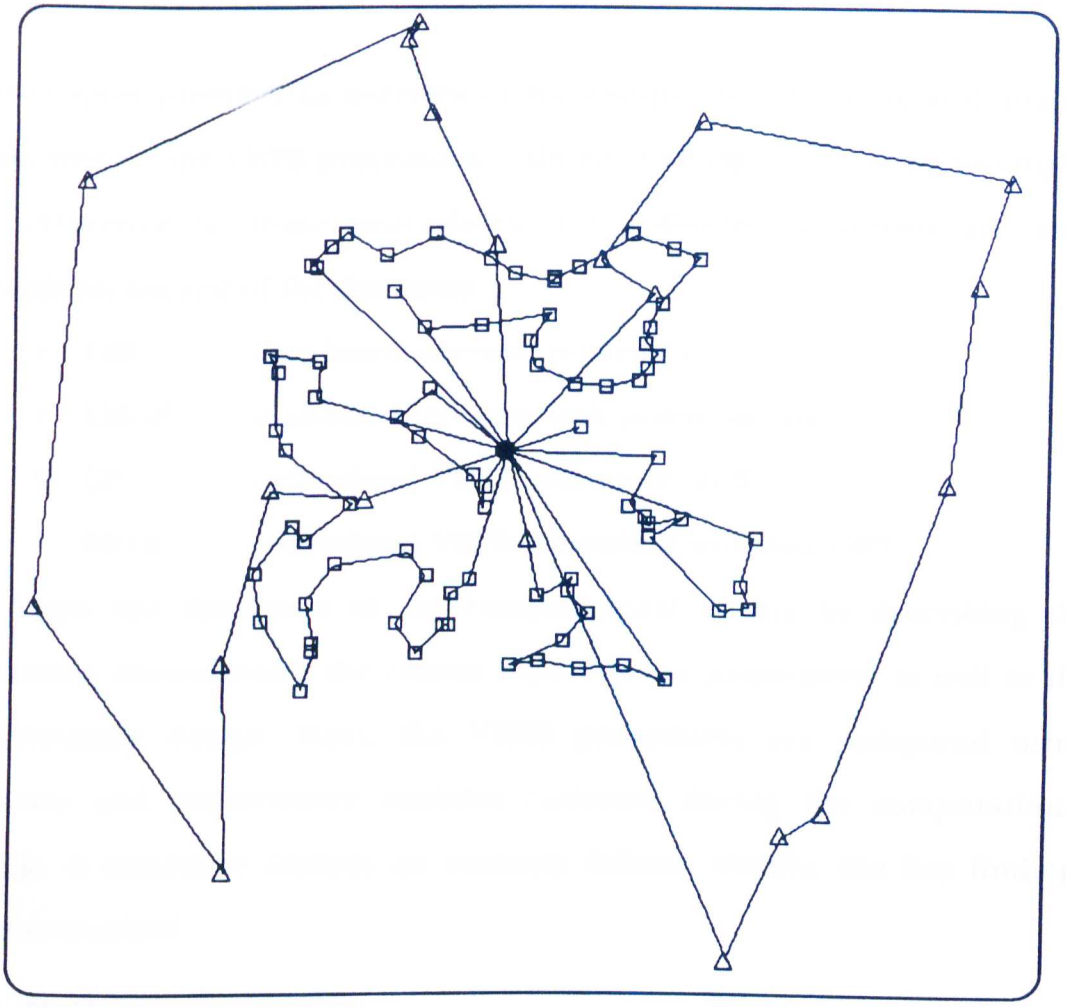


Figure 4.14 Generalized VRPB solution with ARCOPT post-processor.

This figure shows that the generalized VRPB solution in *Figure 4.13* can be improved by applying the ARCOPT procedure to the dedicated routes. Note how the large dedicated pickup route on left hand side has been improved. The total distance of this solution is 2,145 units.

5.0 COMPUTATIONAL TESTING AND EVALUATION

This chapter provides an overview of the computational testing and analysis performed on the VRPB procedures outlined in *Chapter 4*. In order to simplify the reference to these procedures, the following acronyms are used throughout the rest of the discussion :

- LBI - load-based insertion procedure,
- LBI+P - load-based insertion with post-processor,
- GP - generalized VRPB procedure, and
- GP+A - generalized VRPB procedure with ARCOPT.

We begin the discussion of the computational testing by describing the processing environment, the critical experimental parameters, as well as the experimental design. Next, the VRPB procedures are compared using accuracy and performance statistics collected during the computational testing. A sensitivity analysis on accuracy follows. Finally, the key findings are summarized.

5.1 TEST ENVIRONMENT

Computational testing of the various VRPB heuristics was done on several Macintosh platforms ranging from a MacPlus with 1 Megabyte of RAM supplemented with a 16 MHZ Processor Card (RADIUS Accelerator 16) and a math co-processor to a standard Macintosh IIcx. All computer codes were written in Pascal using LightSpeed Pascal (compiler Version 2.0 by Think Technologies).

Some of the routines required the use of storage compression techniques to solve the larger problems. These techniques included converting large two-dimensional symmetric arrays into much smaller one-dimension arrays, as well as scaling array elements from two-byte integers to single-byte integers.

5.2 EXPERIMENTAL PARAMETERS

The critical parameters used to generate the experimental design are explained below:

Parameter	Explanation	Experimental Settings
Scale	Dimension of Euclidean space where nodes are generated.	250 by 250 unit square.
Problem Size	Number of nodes generated for a particular experiment	40, 60, 80, and 100 nodes.
Percent of Backhaul Nodes	Percentage of total nodes which will be considered backhauls during experiment.	20%, 30% and 40%.
Delivery Node Concentration	Portion of Euclidean space in which delivery locations will be randomly generated.	4%, 16%, and 40%.
Backhaul Penalty	Distance penalty attributed to the insertion of a backhaul in a route where deliveries remain to be served.	100%, 200%, 300%, 400%, and 500%.
Radial Surcharges	Distance surcharge (if any) to applied common carrier routes.	0%, 25%, and 50%.
Number of Random Problems	Number of randomly generated problems per parameter value combination	5 problems.

Test problems were randomly generated in Euclidean space within a square of 250 by 250 units. Problem sizes ranged from 40 to 100 nodes. Backhaul percentages ranged from 20% to 40%. Delivery concentration as a percentage of the total Euclidean space varied from 4% to 40%. Backhaul insertion penalties ranged between 100 and 500. A total of five network problems were randomly generated for each parameter combination.

Figures 5.1 through *5.6* illustrate the impact of these parameters.

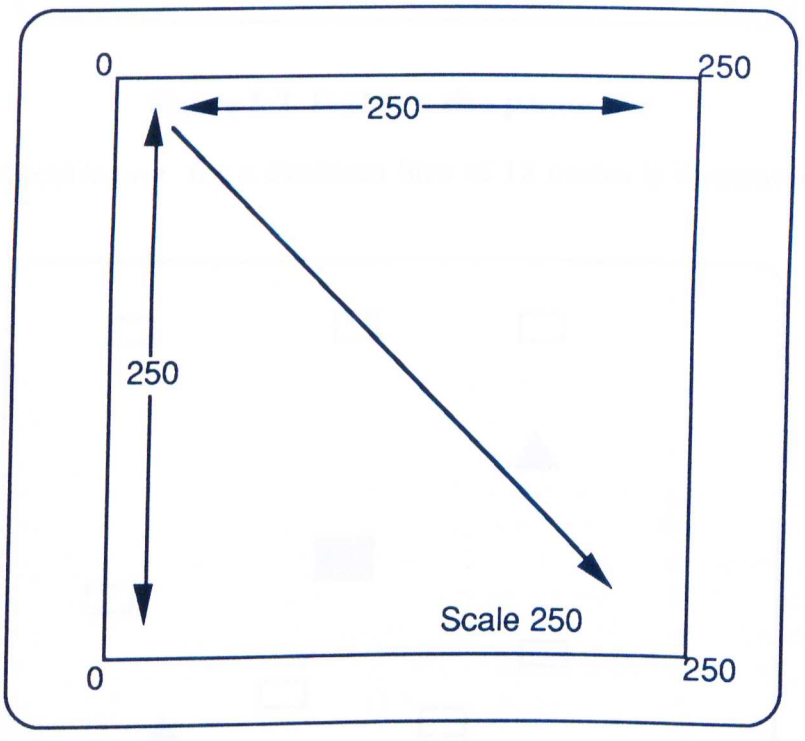


Figure 5.1 Scale parameter.

This figure illustrates the Euclidean space for a Scale value of 250 (that is 250 units by 250 units). This Scale value was used for all the problems in the experiment .

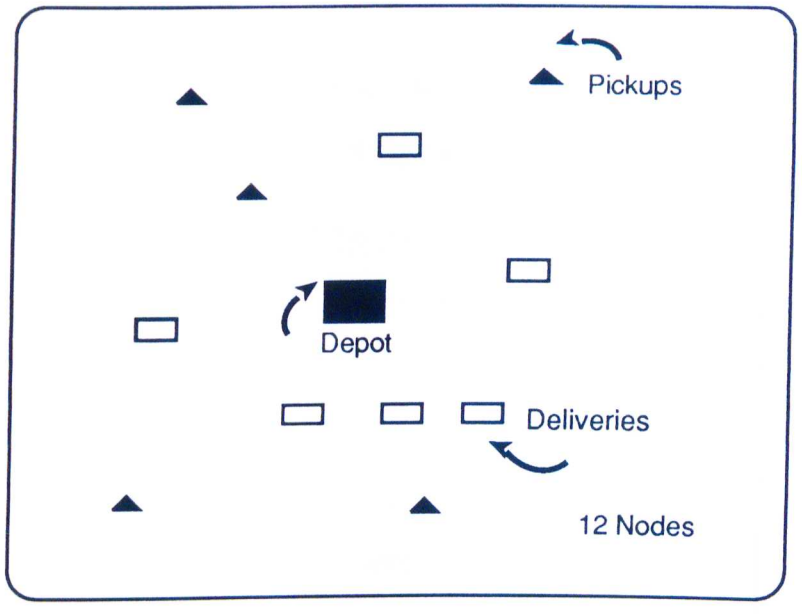


Figure 5.2 Problem size parameter

A typical layout for a Problem Size of 12 nodes is illustrated.

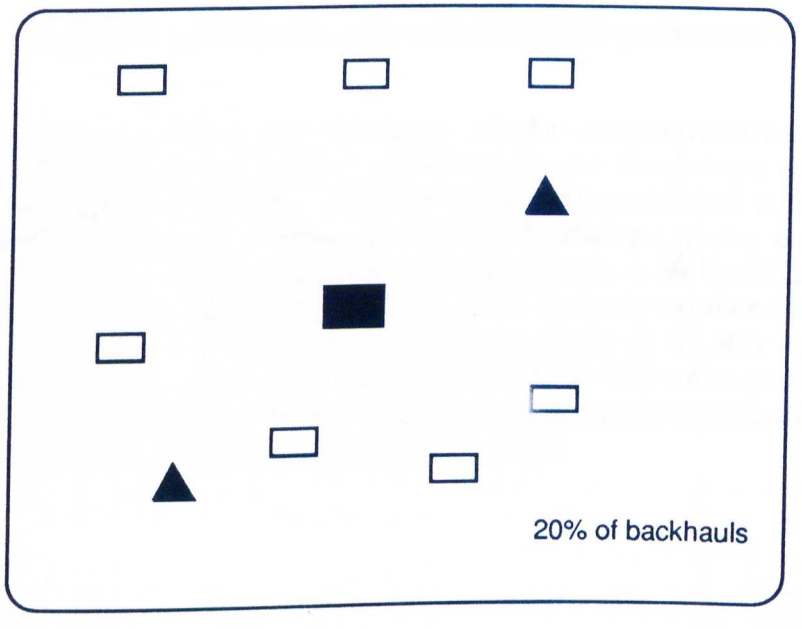


Figure 5.3 Percentage of backhaul nodes parameter.

The **Percentage of Backhaul Nodes** value is set at 20%. For this typical 10-node problem, we have 2 backhauls, 7 deliveries, and 1 depot.

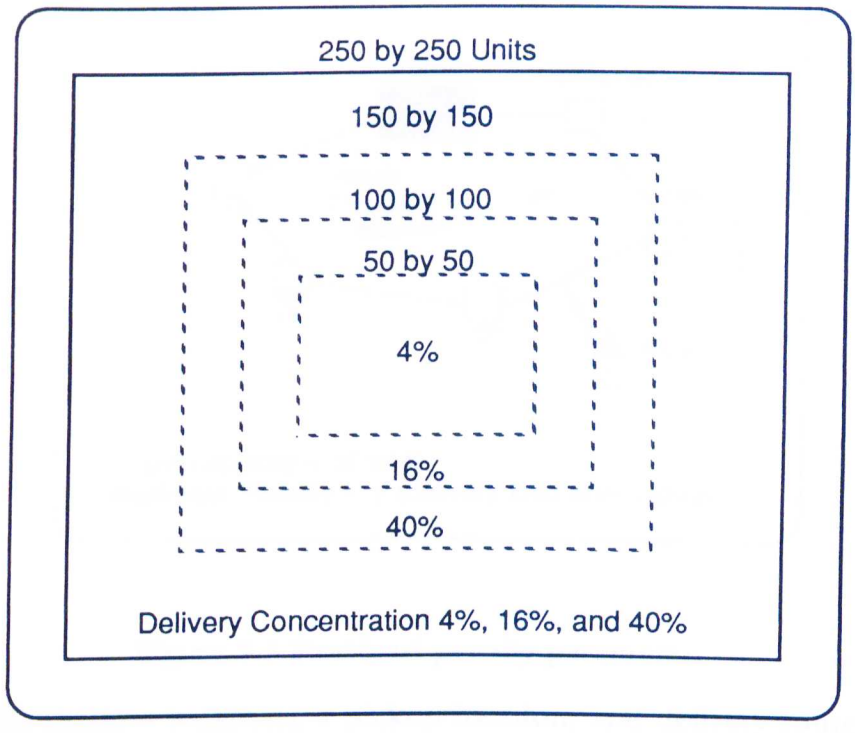


Figure 5.4 Delivery node concentration parameter.

This figure illustrates how the **Delivery Node Concentration** parameter impacts the distribution of delivery locations in the Euclidean space. In our computational experimentation, delivery nodes are restricted to a subarea of the entire Euclidean plane. Earlier, the entire Euclidean plane was defined as 250 by 250 unit square. In the most concentrated case, a 50 by 50 unit square is expected to contain all the delivery nodes. This delivery concentration area of 2,500 square units is 4% of the entire Euclidean space of 62,500 squared units. Furthermore, a delivery concentration area of 100 by 100 units produces a 16% delivery concentration value, and a delivery concentration area of 150 by 150 units produces a 40% delivery concentration value.

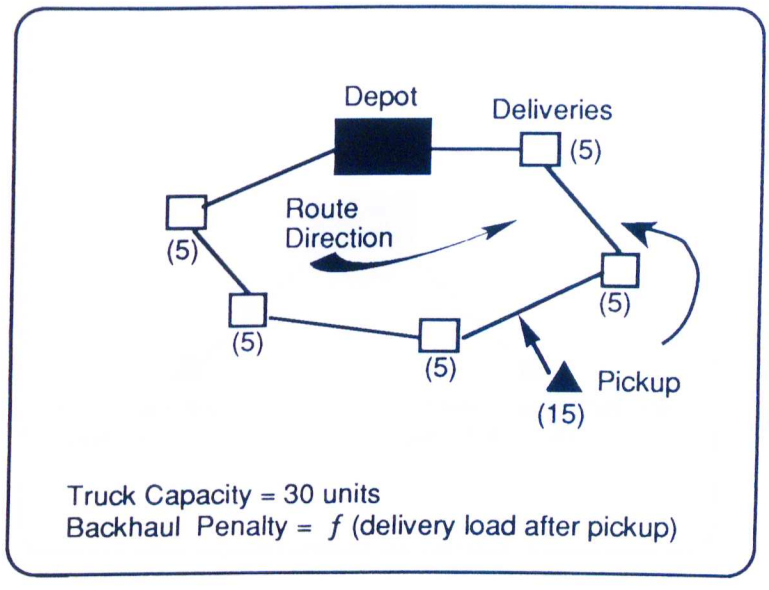


Figure 5.5 Backhaul penalty parameter.

A depot, five delivery locations, a pickup location, and delivery route are used in this illustration to show how the **Backhaul Penalty** parameter will impact backhaul insertion selections. In order to serve the pickup location near the end of the route, the truck is expected to serve the deliveries on a counter-clockwise direction . The closest route insertion option for the pickup location is between the third and fourth delivery locations. Further analysis reveals that even though this insertion will add the least amount of distance to the route, the time spent rearranging the truck after the pickup will be substantial. Note that the truck, with capacity for 30 units, will need to deliver 10 more units while taking 15 units back to the depot. Given this scenario, the backhaul penalty is introduced to discourage pickups while large amounts of delivery load remain in the truck.

During our experimentation we used a range of values for the backhaul penalty. To set this values in practice, we need to analyze the amount of time a driver would spend rearranging a truck. The amount of time required to rearrange a load could be spent driving, if backhauls were not permitted on a delivery route. Thus by combining factors such as the average speed of the truck and the average time required to rearrange a truck after a backhaul is accepted, we could determine an appropriate distance-based penalty for inserting backhauls on delivery routes. A further refinement to this penalty might include the addition of a capacity surcharge to reflect the impact of truck congestion on the time required to rearrange the truck load. The surcharge would decrease as the truck unloads; conversely it would increase as the truck takes on backhaul loads.

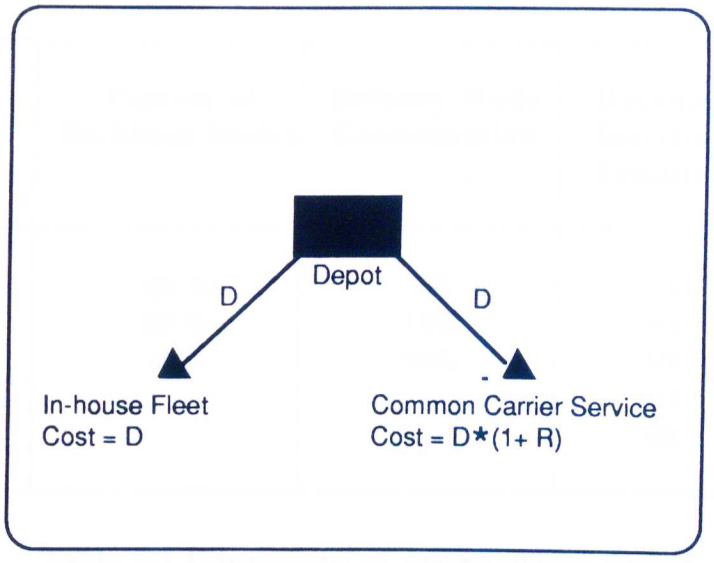


Figure 5.6 Radial surcharge parameter.

If we serve a backhaul location using the in-house truck fleet, we incur a cost of D. However, this cost is inflated by 1+ R, where 0<= R <=1, when we use a common carrier. The term R is the Radial Surcharge parameter

5.3 COMPUTATIONAL EXPERIMENTS

Test problems were randomly generated in Euclidean space within a square of 250 by 250 units. Problem sizes ranged from 40 to 100 nodes. Backhaul percentages ranged from 20% to 40%. Delivery concentration as a percentage of the total Euclidean space varied from 4% to 40%. Backhaul insertion penalties ranged between 100 and 500. A total of five network problems were randomly generated for each parameter combination. Figure 5.1 illustrates the dimensions of the experimental design. This design yielded a total of 2,700 problems. These problems were solved by each of the four solution procedures outlined in Section 4 to produce a total of 10,800 solutions. The entire output from this experiment has been included in *Appendix C.*

Problem Size	Percent of Backhaul Nodes	Delivery Node Concentration	Backhaul Insertion Penalty	Radial Surcharge Penalty
40 Nodes 60 Nodes 80 Nodes 100 Nodes	20 % 30 % 40%	4% 16% 40%	100 200 300 400 500	0% 25% 50%

Table 5.1 Dimensions of experimental design.

The product of all the possible parameter values for the problem size, the backhaul percentage, the delivery concentration, the backhaul insertion penalty, and the radial surcharge penalty yields 540 parameter combinations (4x3x3x5x3=540). Five random problems were generated for each of the possible parameter combinations. The resulting 2,700 problems (540x5=2,700) were then solved by each of the four VRPB solution procedures defined in Section 4.0. This experiment produced 10,800 solutions (2,700x4=10,800).

5.4 ANALYSIS AND EVALUATION OF RESULTS

The four procedures (LBI, LBI+P, GP, and GP+A) were coded, and a series of computational experiments were performed.

A total of 2,700 problems were solved.

The solutions generated by each procedure were then compared and the following statistics were collected : 1.) Number of better solutions than LBI (that is, the number of instances where LBI+P, GP, and GP+A yielded shorter

solutions than LBI), 2.) Average percent improvement over LBI solution (that is, the average percent improvement in solution length of LBI+P, GP, and GP+A over LBI), and 3.) Average solution length (that is the average solution length for all procedures).

A summary of these results is provided in the following sections.

5.4.1. Basic Results

In *Table 5.2*, we report the key summary statistics for our experiments. For each of the four problem sizes (40, 60, 80, and 100 nodes), there are 675 observations. The main findings from this analysis are summarized below.

On average, LBI+P, GP, and GP+A outperform LBI on all of the problems sizes modeled. Furthermore, on average,

- LBI+P produced routes that were 7% to 12% shorter than LBI,
- GP produced routes that were 4% to 33% shorter LBI, and
- GP+A produced routes that were 6% to 35% shorter than LBI.

LBI+P, GP, and GP+A were more effective in finding lower-cost solutions than LBI for problems with less than 100 nodes. For example, GP outperformed LBI in 674 out of 675 instances of 40-node problems with an average improvement of 33%. On the other hand, GP outperformed LBI in 284 out of 675 instances of 100-node problems with an average improvement of just 4%.

LBI+P, GP, and GP+A do not perform as favorably vis-vis LBI for larger problems as they do for smaller problems. This can be attributed to two important factors. Mixed routes look more attractive as the number of nodes in the 250 unit by 250 unit Euclidean space increase. Therefore, the difference in performance between load-based-type approaches and generalized approaches narrows. As the number of nodes increases the quality of the initial solutions created by the Clarke-Wright algorithm used in this computational experiment deteriorates unless the number of links is increased dramatically. Our implementation of the Clarke-Wright algorithm contains a memory saving feature which allows the user to control the number of adjacent links defined per network node. As the number of nodes and the corresponding feasible links grow, this control parameter is adjusted to restrict the number of adjacent links defined. After a certain threshold, this memory saving technique causes the quality of solutions to deteriorate. Unfortunately, memory constraints in our processing environment prevented us from counteracting this problem. Even though the Clarke-Wright algorithm was used in all procedures, we believe its inefficiency with larger node problems affected the generalized-type procedures more adversely. Note that the Arcopt post -processor can not compensate for the Clarke-Wright inefficiency with large problems because it improves the individual routes, not the entire VRP which includes the vehicle assignment and the route formations.

Figures 5.7 through *5.9* illustrate the accuracy trends of the solution procedures as the problem size grows.

NUMBER OF SOLUTIONS BETTER THAN LBI

	40-NODES	60-NODES	80-NODES	100-NODES
LBI+P	559	298	272	240
GP	674	512	375	284
GP+A	674	542	396	322

AVERAGE PERCENT IMPROVEMENT OVER LBI

	40-NODES	60-NODES	80-NODES	100-NODES
LBI+P	12	7	7	7
GP	33	16	9	4
GP+A	35	19	11	6

AVERAGE SOLUTION LENGTH

	40-NODES	60-NODES	80-NODES	100-NODES
LBI	2,091	2,104	2,756	3,242
LBI+P	1,843	1,958	2,566	3,017
GP	1,410	1,772	2,499	3,098
GP+A	1,362	1,705	2,460	3,036

Table 5.2 Computational experiment summary (problem size clustering).

LBI+P, GP, and GP+A clearly outperform LBI for problem sizes of 40, 60, and 80 nodes. But this advantage becomes smaller for 100-node problems. For example, GP+A produces better solutions than LBI in 674 out of 675 40-node problems. Whereas, GP+A produces better solutions than LBI in 322 out of 675 100-node problems. Two reasons for this behavior are possible: 1.) as the number of nodes increases in the Euclidean space the attractiveness of mixed solutions increases, and 2.) as the number of nodes increases the Clarke-Wright heuristic produces "poorer" initial routes that adversely affect the generalized procedures more so than the load-based-type procedures.

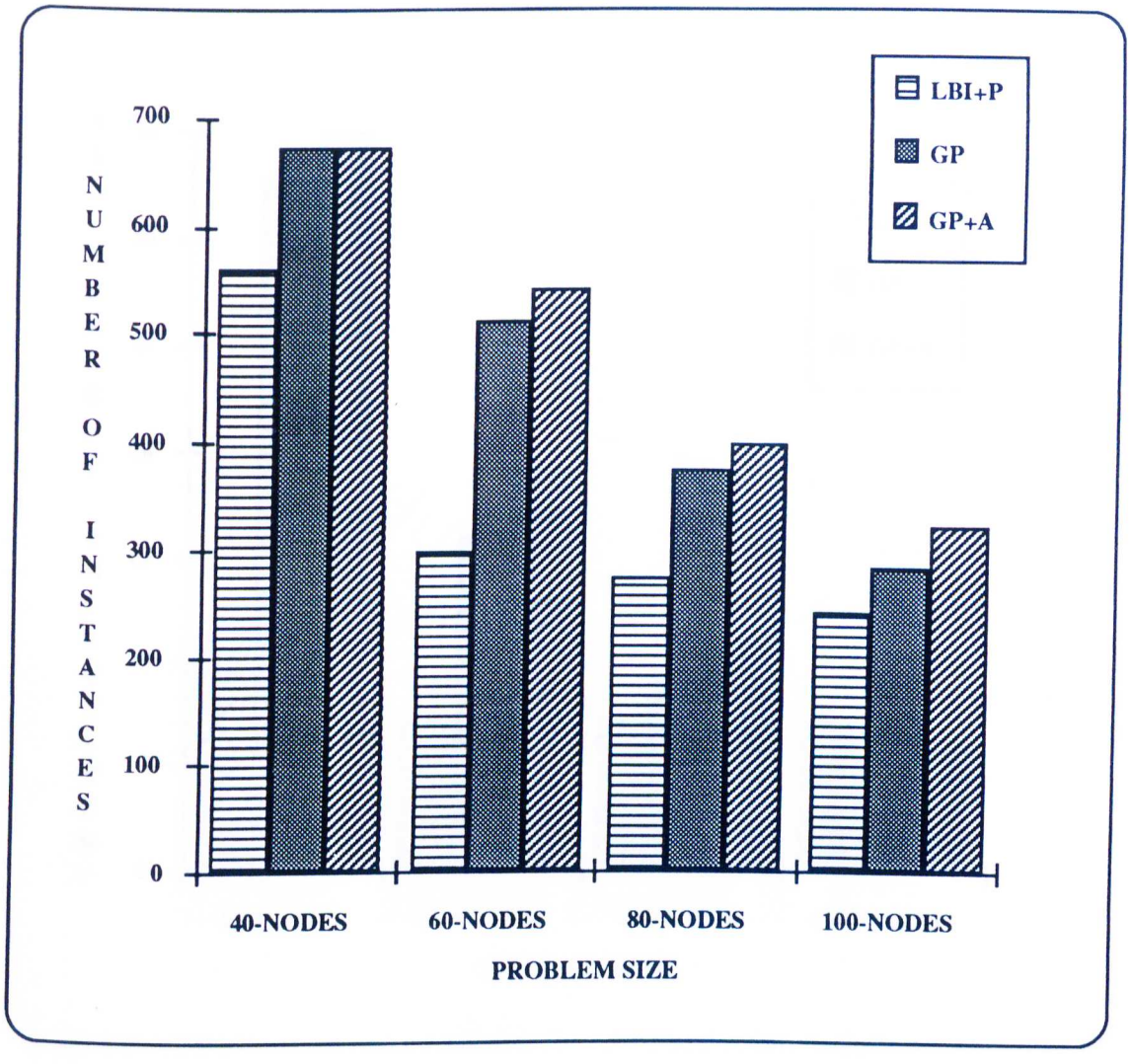


Figure 5.7 Number of solutions better than LBI (problem size clustering).

Counting the instances LBI+P, GP, and GP+A outperform LBI. Each column represents the number of solutions that were better than solutions produced by LBI.

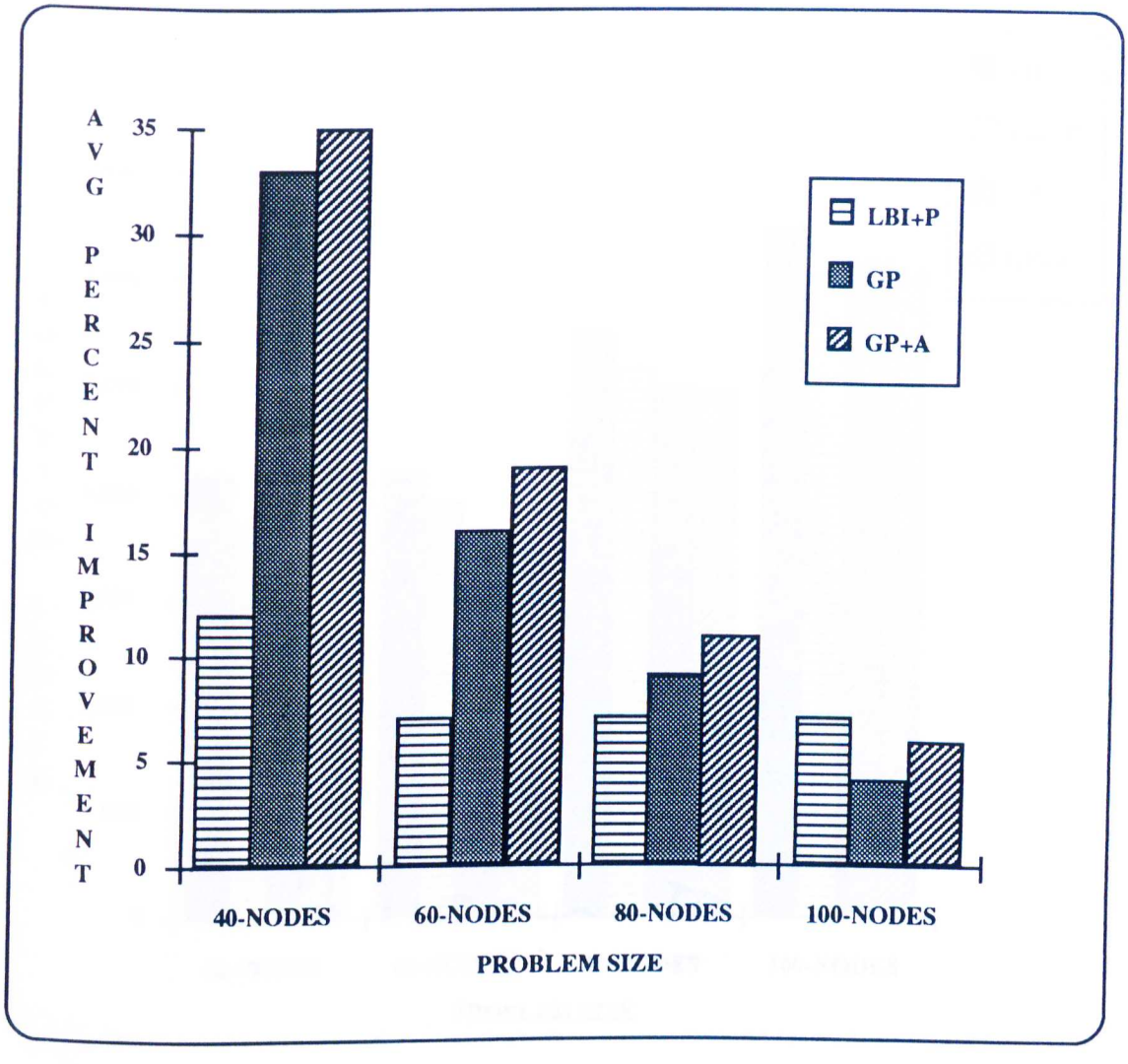


Figure 5.8 Avg. percent improvement over LBI (problem size clustering).

Each column represents the average percent improvement of LBI+P, GP, and GP+A solutions over LBI solutions.

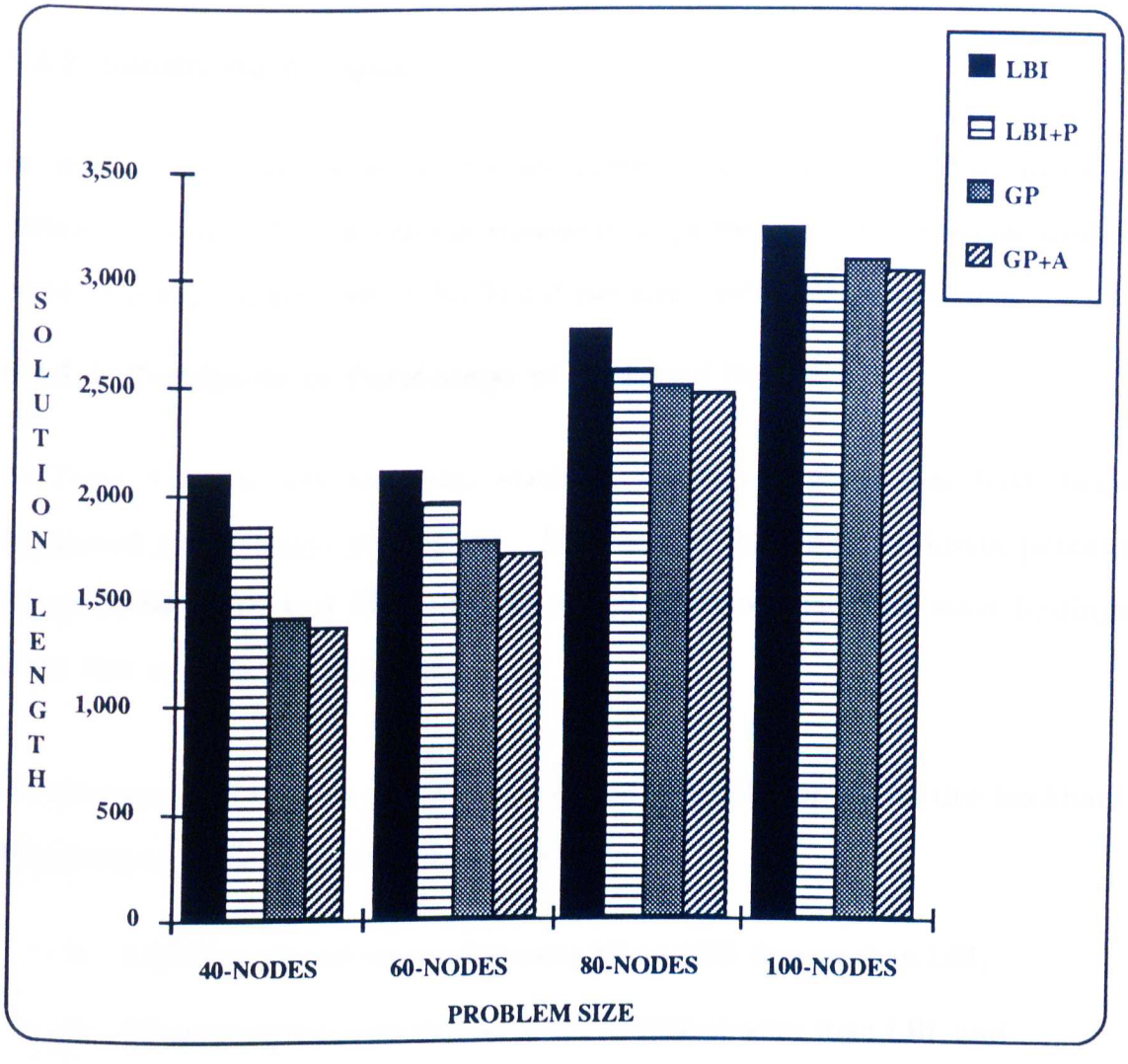


Figure 5.9 Avg. solution length (problem size clustering).

This figure shows how, on average, LBI+P, GP, and GP+A produce shorter routes than LBI.

5.4.2 Sensitivity Analysis

In this section, we examine the sensitivity of our solution procedures to different values of four critical parameters: percentage of backhaul nodes, delivery node concentration, backhaul penalty, and radial surcharges.

5.4.2.1 Sensitivity to Percentage of Backhaul Nodes

In *Table 5.3*, the key summary statistics for our experiments have been organized by backhaul percentage. For each of the three backhaul percent values (20%, 30%, and 40%), there are 900 observations. The main findings from this analysis are as follows:

On average, LBI+P, GP, and GP+A outperform LBI for all of the backhaul percentages used. Furthermore, on average,

- LBI+P produced routes that were 3% to 13% shorter than LBI,
- GP produced routes that were 8% to 23% shorter than LBI, and
- GP+A produced routes that were 10% to 25% shorter than LBI.

LBI+P, GP, and GP+A were more effective in finding lower-cost solutions than LBI for problems with a greater number of backhaul nodes. For example, GP outperformed LBI for the 40% backhaul node problems in 723 out of 900 instances by an average improvement of 23%. On the other hand, the GP outperformed LBI for the 20% backhaul node problems in 529 out of 900 instances by an average improvement of just 8%.

The increased effectiveness of LBI+P, GP, and GP+A on problems with a large number of backhauls can be attributed to the fact that as the number of backhaul nodes grows the attractiveness of dedicated backhaul routes (such as the ones featured by LBI+P, GP, and GP+A) increases. Furthermore, as the number of backhaul nodes decrease the attractiveness of dedicated backhaul routes decrease. Consequently, LBI+P, GP, and GP+A are not as effective in finding lower-cost solutions.

Figures 5.10 through *5.12* illustrate the accuracy trends of the solution procedures as the backhaul percentages change.

NUMBER OF SOLUTIONS BETTER THAN LBI

	BACKHAUL PERCENTAGE		
	20%	30%	40%
LBI+P	317	484	568
GP	529	592	723
GP+A	576	621	737

AVERAGE PERCENT IMPROVEMENT OVER LBI

	BACKHAUL PERCENTAGE		
	20%	30%	40%
LBI+P	3	7	13
GP	8	14	23
GP+A	10	16	25

AVERAGE SOLUTION LENGTH

	BACKHAUL PERCENTAGE		
	20%	30%	40%
LBI	2,278	2,538	2,830
LBI+P	2,206	2,348	2,485
GP	2,149	2,227	2,209
GP+A	2,090	2,173	2,159

Table 5.3 Computational experiment summary (backhaul percentage clustering).

These results show that on average LBI+P, GP, and GP+A outperform LBI over all backhaul percentages. As the number of backhauls increase, the attractiveness of dedicated backhaul routes increases. Thus GP and GP+A perform better than load-based procedures when the backhaul node percentage is higher. As the number of backhaul nodes decrease, the attractiveness of dedicated backhaul routes decreases. Thus GP and GP+A do not perform as well when the backhaul node percentage is lower.

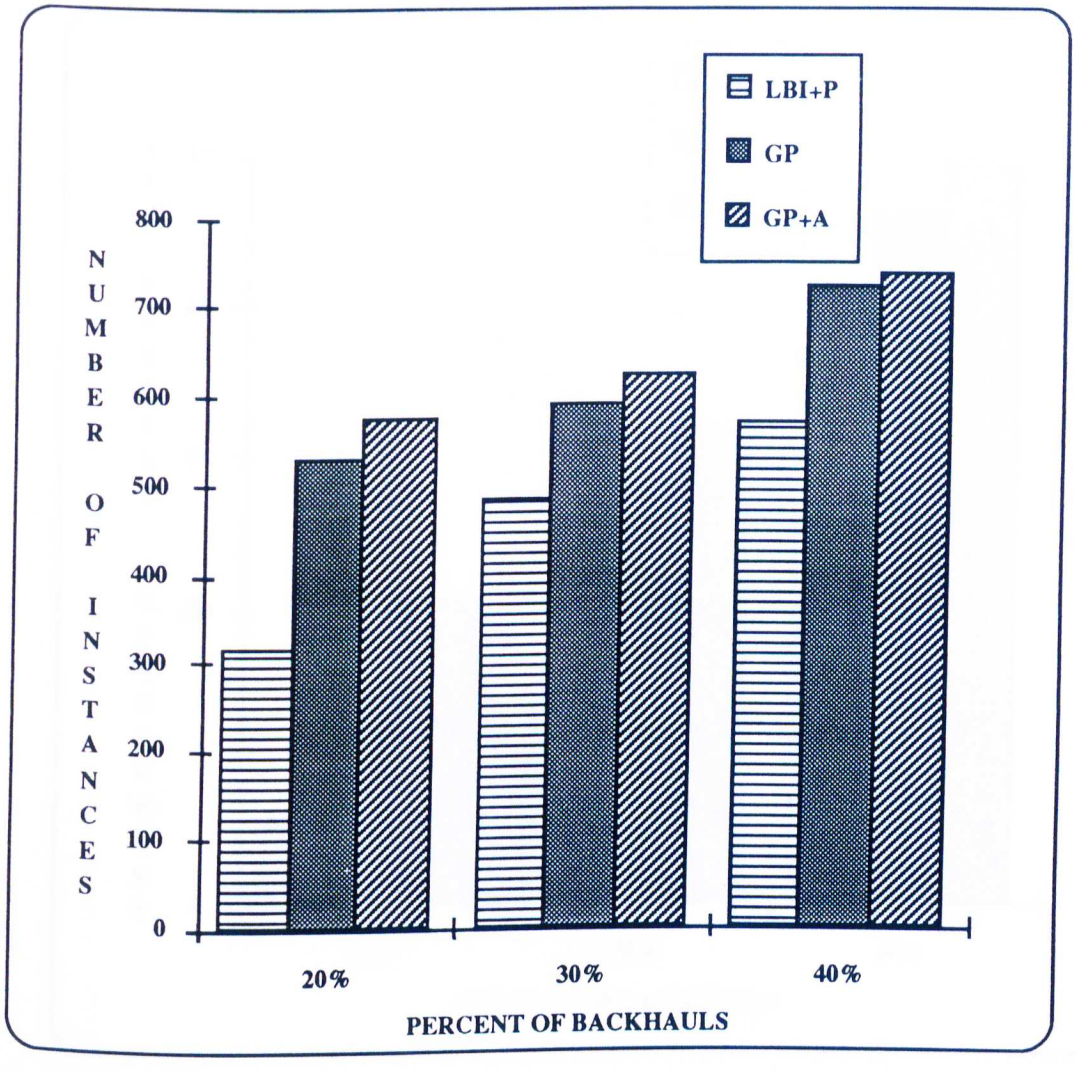


Figure 5.10 Number of solutions better than LBI (backhaul percentage clustering).

This figure illustrates the number of instances LBI+P, GP, and GP+A outperform LBI. Each column represents the number of solutions that were better than solutions produced by LBI.

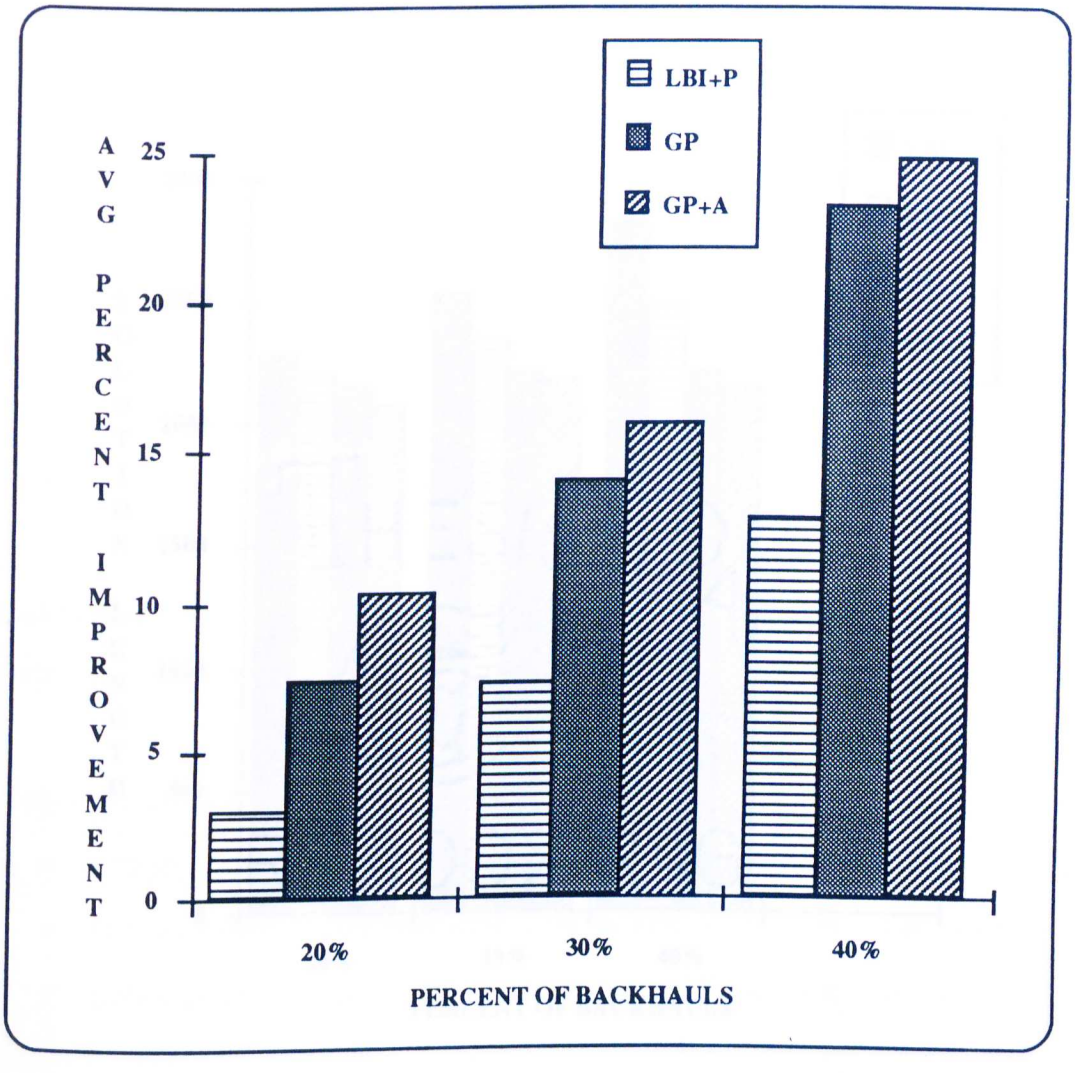


Figure 5.11 Avg. percent improvement over LBI (backhaul percentage clustering).

Each column represents the average percent improvement of LBI+P, GP, and GP+A solutions over LBI solutions.

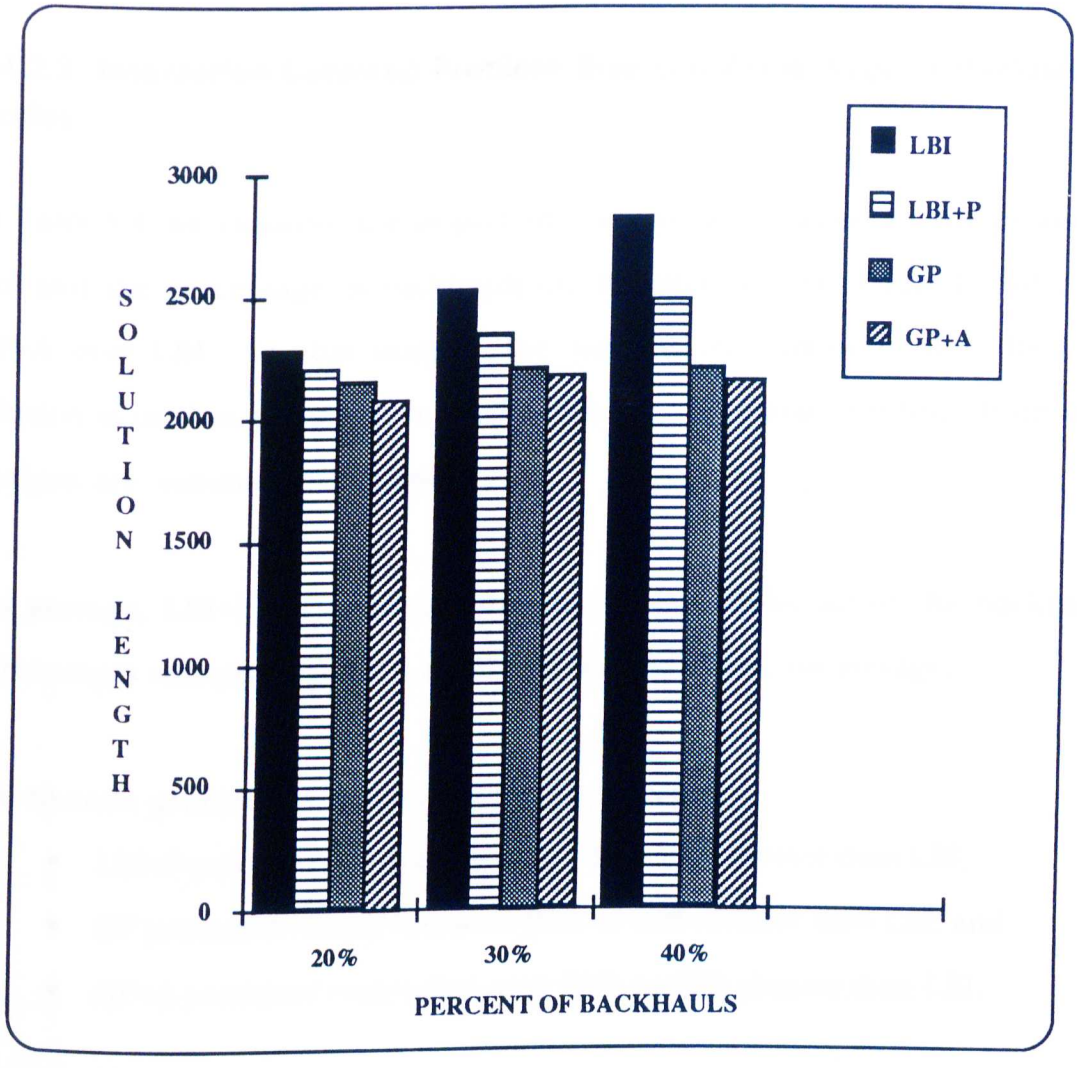


Figure 5.12 Avg. solution length (backhaul percentage clustering).

This figure shows how, on average, LBI+P, GP, and GP+A produce shorter routes than LBI. In addition, this figure illustrates how the solutions generated by generalized procedures are minimally impacted by the growth in the number of backhauls. Once the generalized procedures create a dedicated backhaul route(s), the cost of adding new backhauls tends to be low. On the other hand, adding backhauls to LBI generated mixed routes tends to be expensive due to potential for backhaul penalties.

5.4.2.2 Interaction between Problem Size and Percentage of Backhauls Nodes

In *Table 5.4*, we examine the impact of the interaction between the problem size and the percentage of backhauls on the effectiveness of LBI+P, GP, and GP+A over LBI. In this analysis, the percent improvement over the LBI solution is used as the measure of effectiveness. The main findings from this analysis are summarized below.

On average, LBI+P, GP, and GP+A outperform LBI for all of the backhaul percentages and problem sizes examined. Furthermore, on average,

For 40-node problems, and all backhaul percentages:

- LBI+P produced routes that were 4% to 18% shorter than LBI,
- GP produced routes that were 22% to 40% shorter than LBI, and
- GP+A produced routes that were 25% to 42% shorter than LBI.

For 60-node problems, and all backhaul percentages:

- LBI+P produced routes that were 2% to 11% shorter than LBI,
- GP produced routes that were 6% to 23% shorter than LBI, and
- GP+A produced routes that were 11% to 24% shorter than LBI.

For 80-node problems, and all backhaul percentages:

- LBI+P produced routes that were 3% to 12% shorter than LBI,
- GP produced routes that were 1% to 23% shorter than LBI, and

- GP+A produced routes that were 2% to 24% shorter than LBI.

For 100-node problems, and all backhaul percentages:

- LBI+P produced routes that were 3% to 10% shorter than LBI,
- GP produced routes that were 1% to 7% shorter than LBI, and
- GP+A produced routes that were 3% to 9% shorter than LBI.

On average, LBI+P, GP, and GP+A outperform LBI the smaller the number of nodes and the greater the number of backhauls. For example, in *Table 5.3* we show that for all problem sizes with 40% backhauls GP+A yields solutions on average 25% shorter than LBI. Table 5.4 shows that for all 40-node problems GP+A yields solutions on average 35% shorter than LBI. Furthermore, *Table 5.4* shows that for 40-node problems with 40% backhauls GP+A yields solutions on average 42% shorter than LBI.

Conversely, the greater the number of nodes and the smaller the number of backhauls the less effective LBI+P, GP, and GP+A are in obtaining lower-cost solutions than LBI. For example, in *Table 5.3* we show that for all problem sizes with 20% backhauls GP+A yields solutions on average 10% shorter than LBI. Table 5.4 shows that for all 100-node problems GP+A yields solutions on average 6% shorter than LBI. Furthermore, *Table 5.4* shows that for 100-node problems with 20% backhauls GP+A yields solutions on average 3% shorter than LBI.

The explanation for this pattern is simply that the smaller the number of backhauls nodes vis-a-vis problem size, the smaller the opportunity for procedures favoring dedicated routes (such as LBI+P, GP, and GP+A) to produce lower-cost solutions. This pattern is also illustrated in *Table 5.4* where the LBI+P, GP, and GP+A percent improvement over LBI decreases as the problem size increases. Note that for the all 100-node problem grouping, LBI+P outperforms GP, and GP+A because of the increased appeal of mixed routes.

Figures 5.13 through *5.16* illustrate the accuracy trends of the solution procedures as the problem size/backhaul percentage parameters change.

AVERAGE PERCENT IMPROVEMENT OVER LBI(40-NODES)

	BACKHAUL PERCENTAGE			
	20%	30%	40%	ALL 40-NODE PROBLEMS
LBI+P	4	11	18	12
GP	22	32	40	33
GP+A	25	34	42	35

AVERAGE PERCENT IMPROVEMENT OVER LBI(60-NODES)

	BACKHAUL PERCENTAGE			
	20%	30%	40%	ALL 60-NODE PROBLEMS
LBI+P	2	6	11	7
GP	6	16	23	16
GP+A	11	19	24	19

AVERAGE PERCENT IMPROVEMENT OVER LBI(80-NODES)

	BACKHAUL PERCENTAGE			
	20%	30%	40%	ALL 80-NODE PROBLEMS
LBI+P	3	5	12	7
GP	1	3	23	9
GP+A	2	4	24	11

AVERAGE PERCENT IMPROVEMENT OVER LBI(100-NODES)

	BACKHAUL PERCENTAGE			
	20%	30%	40%	ALL 100-NODE PROBLEMS
LBI+P	3	7	10	7
GP	1	5	7	4
GP+A	3	7	9	6

Table 5.4 Computational experiment summary (problem size/backhaul percentage clustering).

LBI+P, GP, and GP+A, which emphasize dedicated routes, yield greater solution improvements over LBI for problems of smaller size and with a larger number of backhauls. Furthermore, these procedures are less effective in obtaining solution improvements over LBI for problems of larger size and with a smaller number of backhauls. For example, GP+A produces solutions that are, on average, 42% shorter than LBI for 40-node problems with 40% backhauls nodes. On the other hand, GP+A yields solutions that are, on average, 3% shorter than LBI for 100-node problems with 20% backhauls nodes.

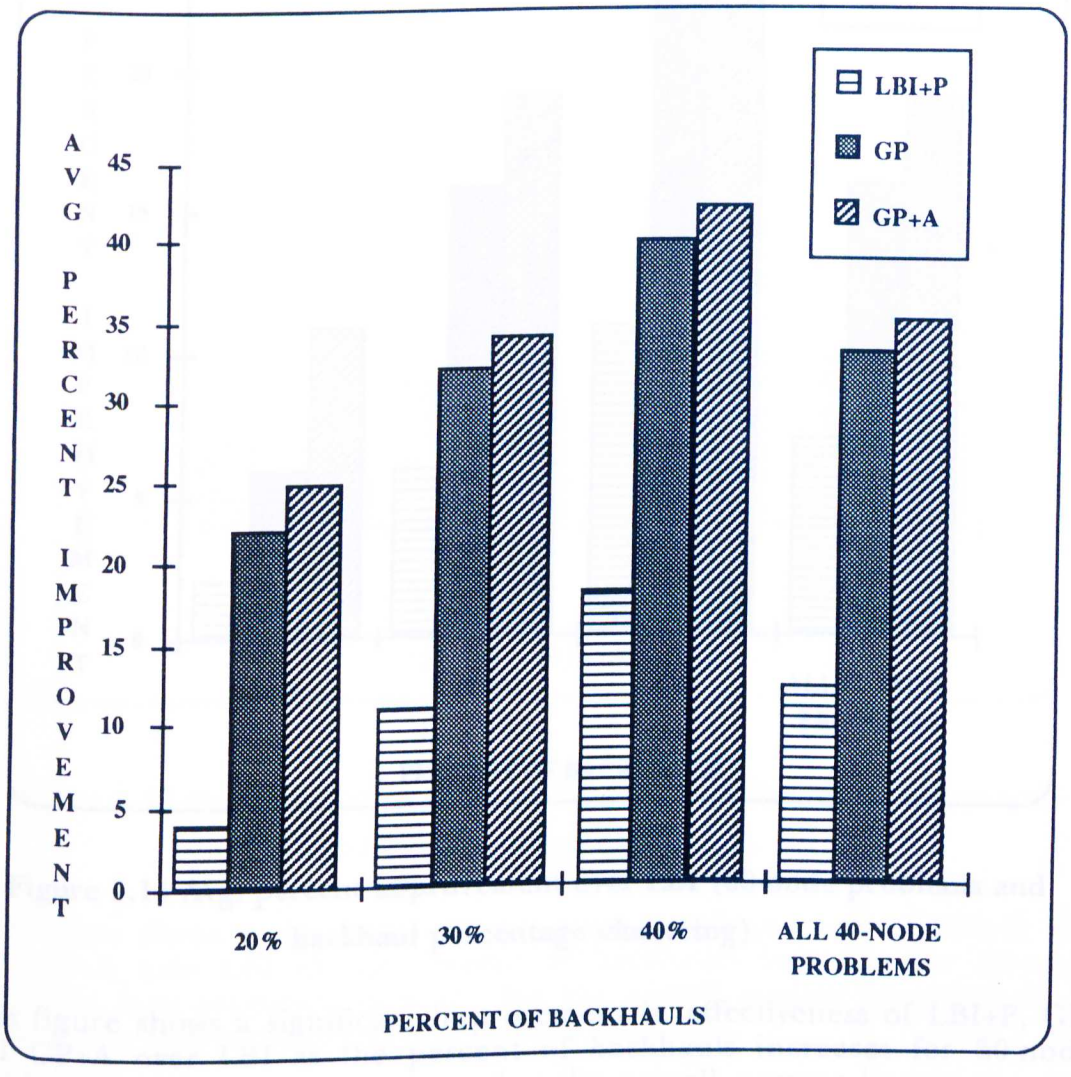


Figure 5.13 Avg. percent improvement over LBI (40-node problems and backhaul percentage clustering).

This figure shows the significant improvement in effectiveness of LBI+P, GP, and GP+A over LBI as the percent of backhauls increases for 40-node problems.

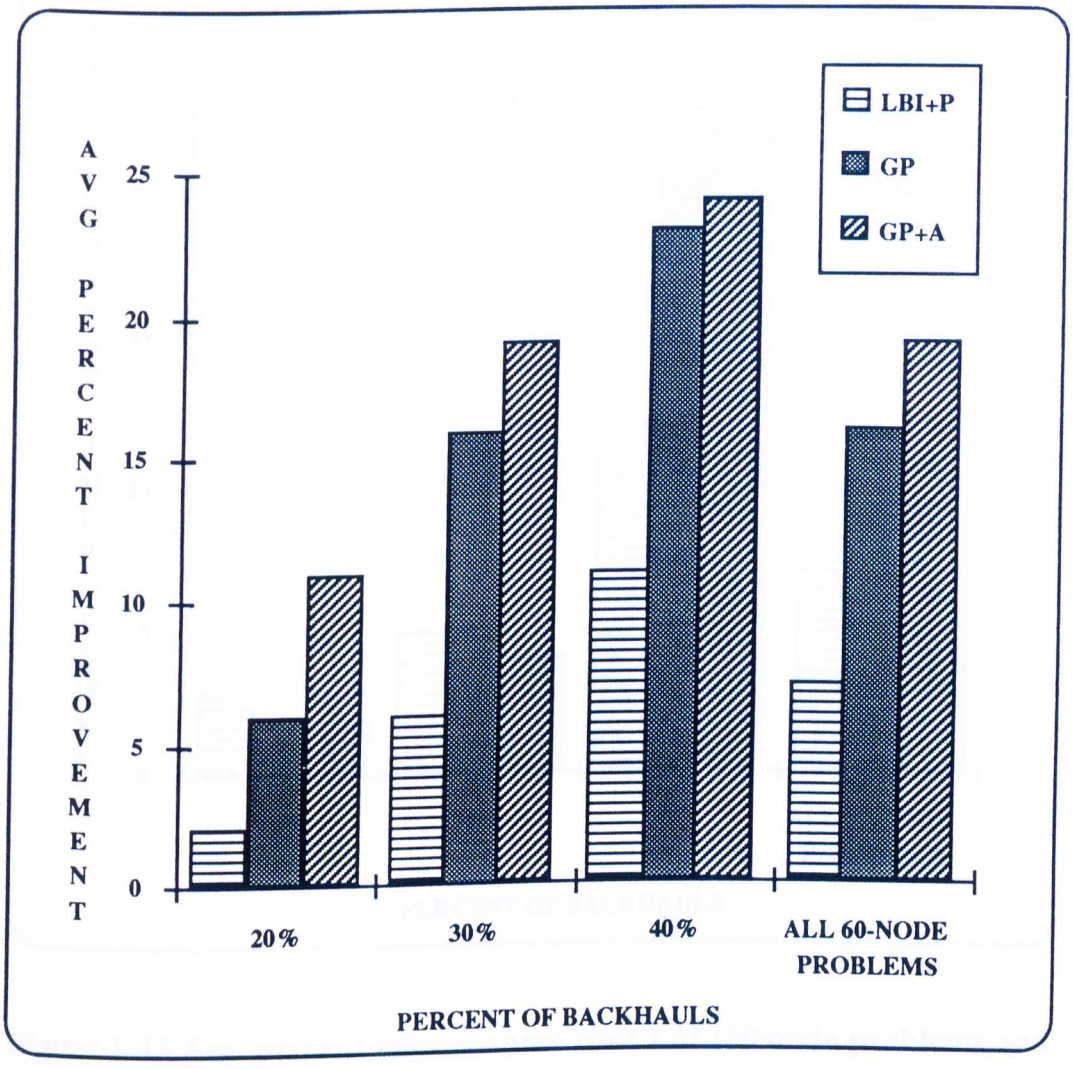


Figure 5.14 Avg. percent improvement over LBI (60-node problems and backhaul percentage clustering).

This figure shows a significant improvement in effectiveness of LBI+P, GP, and GP+A over LBI as the percent of backhauls increases for 60-node problems. It is important to note, that the overall percent improvement is smaller than that experienced with 40-node problems.

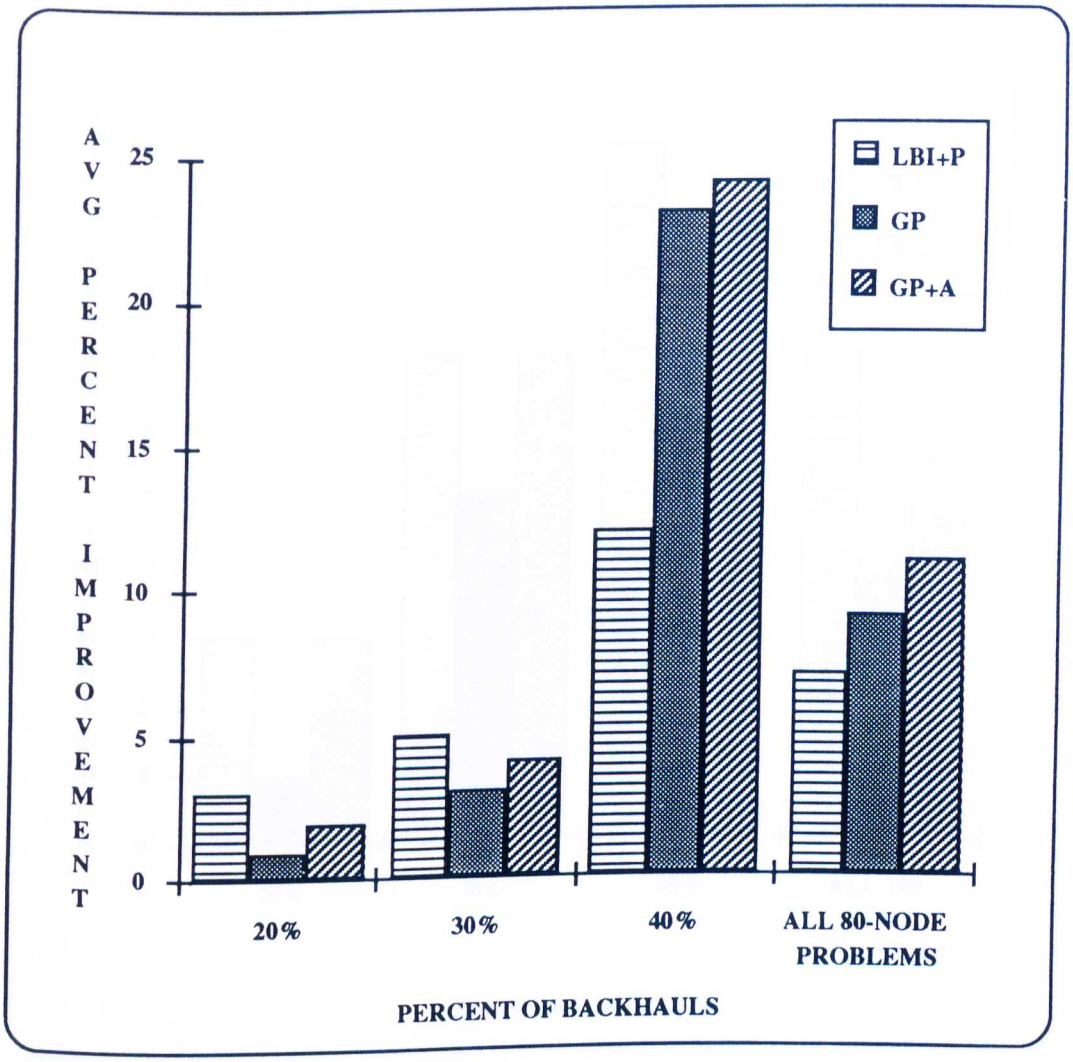


Figure 5.15 Avg. percent improvement over LBI (80-node problems and backhaul percentage clustering).

This figure shows a significant improvement in effectiveness of LBI+P, GP, and GP+A over LBI as the percent of backhauls increases for 80-node problems. It is important to note, that the overall percent improvement is smaller than that experienced with 60-node problems.

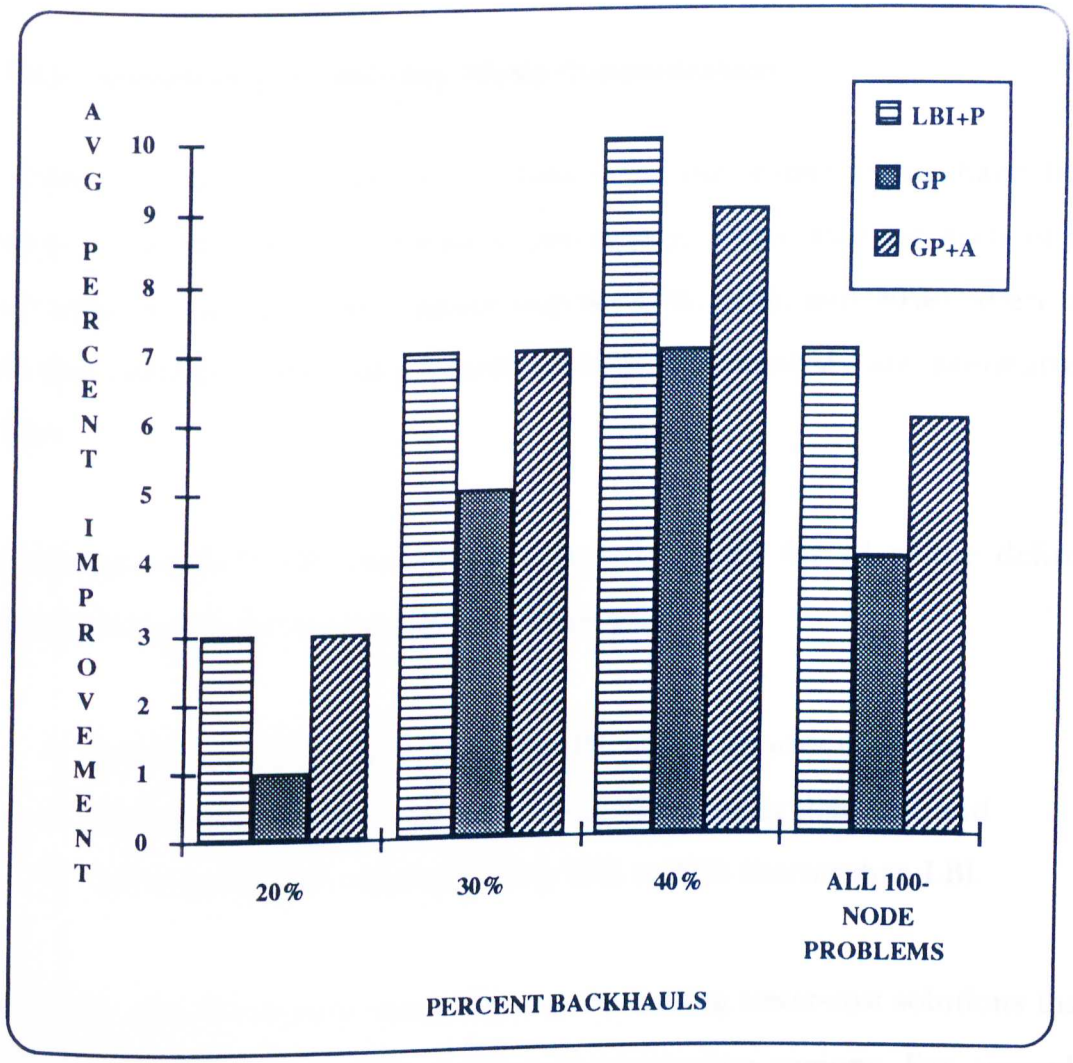


Figure 5.16 Avg. percent improvement over LBI (100-node problems and backhaul percentage clustering).

This figure shows a small improvement in the effectiveness of LBI+P, GP, and GP+A over the LBI as the percent of backhauls increases for 100-node problems. It is important to note, that the overall percent improvement is smaller than that experienced with 80-node problems

5.4.2.3 Sensitivity to Delivery Node Concentration

In *Table 5.5*, the key summary statistics for our experiments have been organized by delivery concentration percentage. Note that for each of the three delivery concentration values/regions (4%, 16%, and 40%), there are 900 observations. The main findings from this analysis are summarized below.

On average, LBI+P, GP, and GP+A outperform LBI for all of the delivery concentration values. Furthermore, on average,

- LBI+P produced routes that were 3% to 17% shorter than LBI,
- GP produced routes that were 8% to 29% shorter than LBI, and
- GP+A produced routes that were 10% to 31% shorter than LBI.

LBI+P, GP, and GP+A were more effective in finding lower-cost solutions than LBI for problems with smaller delivery concentration regions. For example, GP outperformed LBI on 839 out of 900 instances by an average improvement of 29% for the 4% delivery concentration problems. On the other hand as delivery concentration expands, the effectiveness of LBI+P, GP, and GP+A decreases. For example, GP outperforms LBI on 443 out of 900 instances by an average improvement of just 8% for the 40% delivery concentration problems.

The increased effectiveness of LBI+P, GP, and GP+A on problems with small delivery concentration values can be attributed to the fact that as the delivery

concentration region becomes smaller the attractiveness of dedicated backhaul routes (such as the ones featured by LBI+P, GP, and GP+A) increases. Furthermore, the reduced effectiveness of LBI+P, GP, and GP+A on problems with large delivery concentration values can be attributed to the fact that as the delivery concentration region becomes greater the opportunity for mixed routes increases and attractiveness of dedicated backhaul routes is reduced.

Figures *5.17* through *5.19* illustrate the accuracy trends of the solution procedures as the delivery concentration changes.

NUMBER OF SOLUTIONS BETTER THAN LBI

	CONCENTRATION PERCENTAGE		
	4%	16%	40%
LBI+P	686	417	266
GP	839	563	443
GP+A	849	621	464

AVERAGE PERCENT IMPROVEMENT OVER LBI

	CONCENTRATION PERCENTAGE		
	4%	16%	40%
LBI+P	17	6	3
GP	29	12	8
GP+A	31	15	10

AVERAGE SOLUTION LENGTH

	CONCENTRATION PERCENTAGE		
	4%	16%	40%
LBI	2,298	2,268	3,079
LBI+P	1,911	2,133	2,994
GP	1,632	2,026	2,927
GP+A	1,592	1,963	2,868

Table 5.5 Computational experiment summary (delivery concentration clustering).

This table shows several comparisons between LBI and the new approaches. These results show a clear advantage for LBI+P, GP, and GP+A for all delivery concentration values. It is important to note that as the delivery concentration region decreases the attractiveness of dedicated routes increases. Thus, the procedures that favor dedicated routes perform better. Furthermore, as the delivery concentration region increases the opportunity for mixed routes increases. Thus, the attractiveness of dedicated routes, and procedures which favor them decrease.

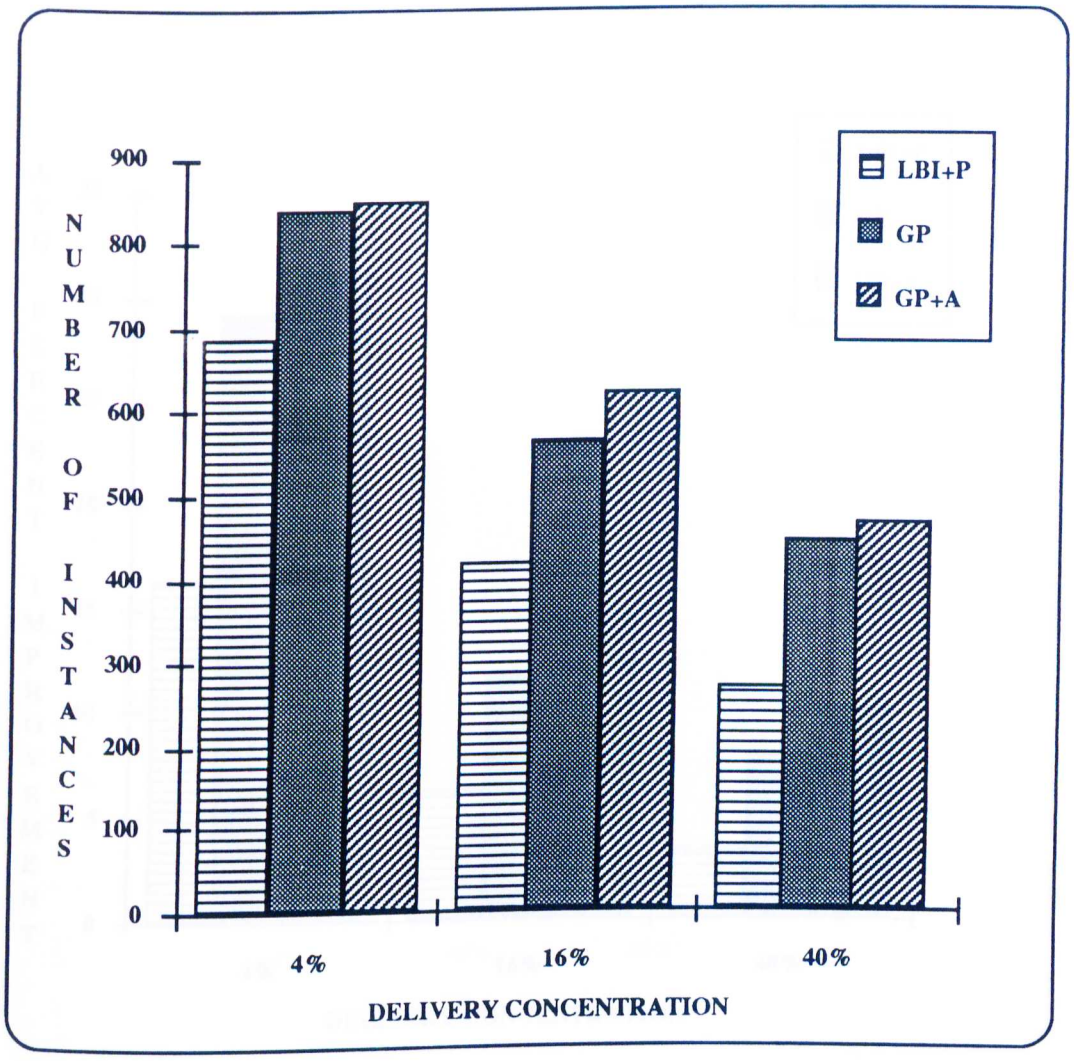


Figure 5.17 Number of solutions better than LBI (delivery concentration clustering).

LBI+P, GP, and GP+A have a clear advantage over LBI, especially for problems with small delivery concentration regions.

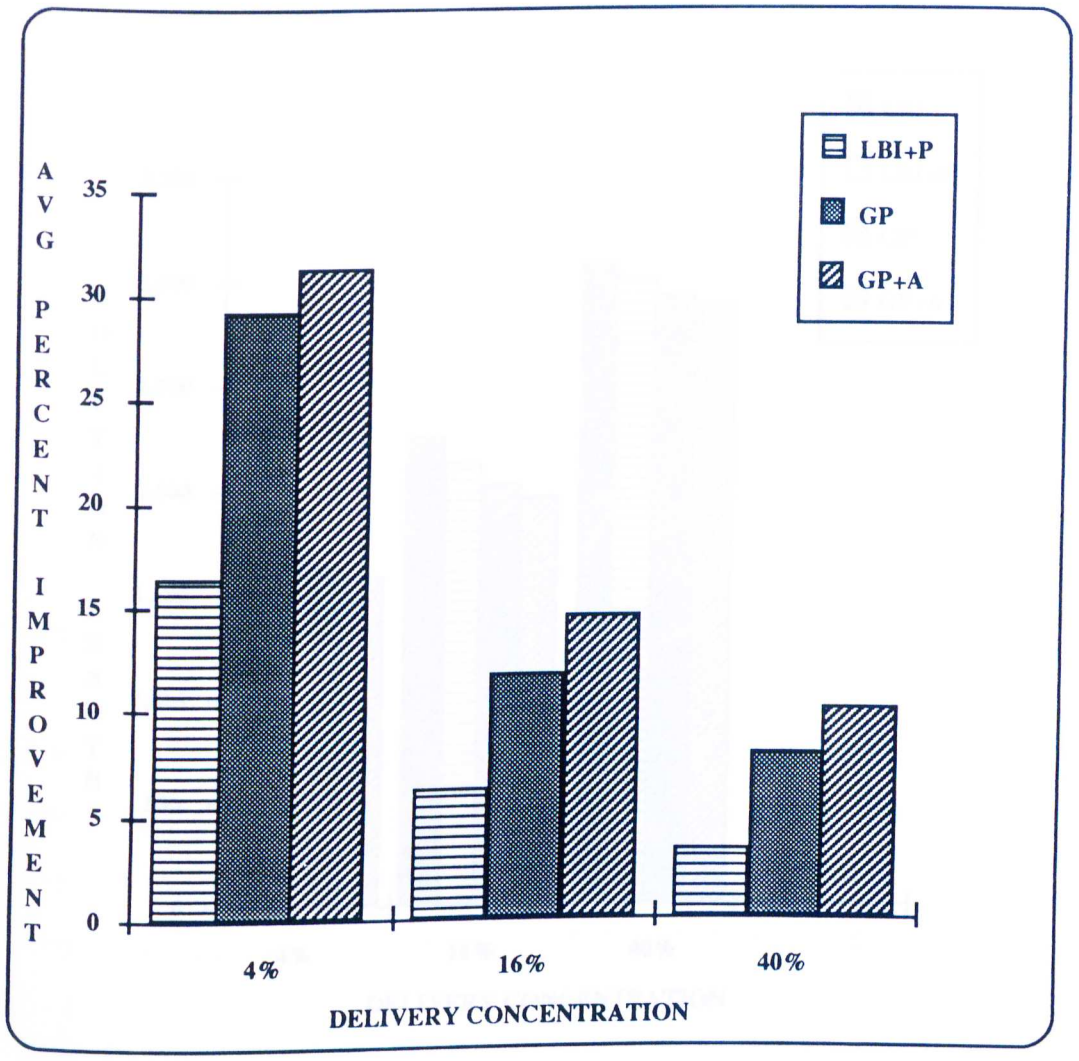


Figure 5.18 Avg. percent improvement over LBI (delivery concentration clustering).

LBI+P, GP, and GP+A clearly outperform LBI for problems with small delivery concentration regions.

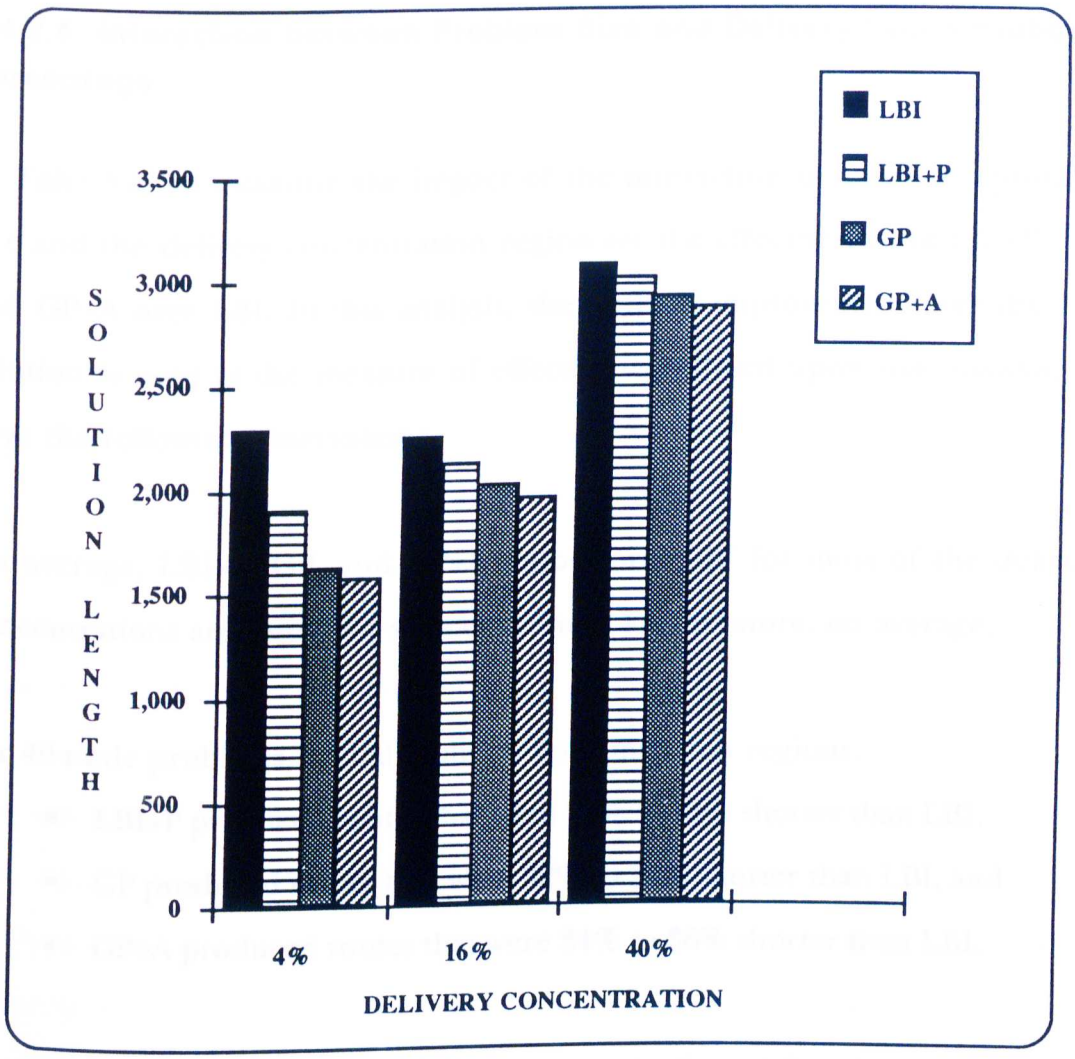


Figure 5.19 Avg. solution length (delivery concentration clustering).

Solution length increases as the delivery concentration region increases. Additionally, the relative improvement of LBI+P, GP, and GP+A over LBI diminishes as the delivery region grows.

5.4.2.4 Interaction between Problem Size and Delivery Concentration Percentage

In *Table 5.6*, we examine the impact of the interaction between the problem size and the delivery concentration region on the effectiveness of LBI+P, GP, and GP+A over LBI. In this analysis, the percent improvement over the LBI solution is used as the measure of effectiveness. Based upon our analysis, we have the following observations.

On average, LBI+P, GP, and GP+A outperform LBI for most of the delivery concentrations and problem sizes examined. Furthermore, on average,

For 40-node problems, and all delivery concentration regions:

- LBI+P produced routes that were 11% to 13% shorter than LBI,
- GP produced routes that were 32% to 34% shorter than LBI, and
- GP+A produced routes that were 34% to 36% shorter than LBI.

For 60-node problems, and all delivery concentration regions:

- LBI+P produced routes that were 1% to 15% shorter than LBI,
- GP produced routes that were 7% to 28% shorter than LBI, and
- GP+A produced routes that were 11% to 30% shorter than LBI.

For 80-node problems, and all delivery concentration regions:

- LBI+P produced routes that were 1% to 18% shorter than LBI,
- GP produced routes that were 0% to 29% shorter than LBI, and
- GP+A produced routes that were 1% to 31% shorter than LBI.

For 100-node problems, and all delivery concentration regions:

- LBI+P produced routes that were 0% to 20% shorter than LBI,
- GP produced routes that were -10% to 28% shorter than LBI, and
- GP+A produced routes that were -8% to 30% shorter than LBI.

On average, LBI+P, GP, and GP+A produce lower-cost solutions than LBI when the delivery concentration region is small. For example, Table 5.6 shows that for problems with 4% delivery concentration regions GP+A produces solutions 30% to 34% shorter on average than LBI regardless of problem size.

For the delivery concentration regions of 40% GP+A's effectiveness is reduced to the -8% to 10% improvement range for problem sizes of 60 nodes or more.

The decreased effectiveness of LBI+P, GP, and GP+A over LBI on problems with large delivery concentration values, as well as a large number of nodes can be attributed to the fact that as the delivery concentration region grows the attractiveness of dedicated backhaul routes(such as the ones featured by LBI+P, GP, and GP+A) decreases. In this environment, two factors make mixed routes more attractive than dedicated routes: 1.) due to the large number of delivery nodes, a large number of delivery routes are available for backhaul insertions, and 2) since the delivery routes span a greater portion of the Euclidean space, it becomes more economical to insert backhauls onto these routes than to create dedicated backhaul routes. Thus, LBI is better suited to find cost-effective solutions in this scenario.

Figures 5.20 through *5.23* illustrate the accuracy trends of the solution procedures as the problem size and delivery concentration parameters change.

AVERAGE PERCENT IMPROVEMENT OVER LBI (40-NODES)

	DELIVERY CONCENTRATION PERCENTAGE 4%	16%	40%	ALL 40-NODE PROBLEMS
LBI+P	13	13	11	12
GP	32	32	34	33
GP+A	34	34	36	35

AVERAGE PERCENT IMPROVEMENT OVER LBI (60-NODES)

	DELIVERY CONCENTRATION PERCENTAGE 4%	16%	40%	ALL 60-NODE PROBLEMS
LBI+P	15	5	1	7
GP	28	13	7	16
GP+A	30	17	11	19

AVERAGE PERCENT IMPROVEMENT OVER LBI (80-NODES)

	DELIVERY CONCENTRATION PERCENTAGE 4%	16%	40%	ALL 80-NODE PROBLEMS
LBI+P	18	5	1	7
GP	29	4	0	9
GP+A	31	6	1	11

AVERAGE PERCENT IMPROVEMENT OVER LBI (100-NODES)

	DELIVERY CONCENTRATION PERCENTAGE 4%	16%	40%	ALL 100-NODE PROBLEMS
LBI+P	20	2	0	7
GP	28	-2	-10	4
GP+A	30	1	-8	6

Table 5.6 Computational experiment summary (problem size/delivery concentration clustering).

The statistics assembled in this table allow us to examine the impact of problem size and delivery concentration percentage on the effectiveness of LBI+P, GP, and GP+A over LBI. Based on this data, LBI+P, GP, and GP+A produce lower-cost solutions (on average) the smaller the concentration region since they emphasize dedicated routes. Furthermore, LBI+P, GP, and GP+A are less effective vis-a-vis LBI the greater the concentration region, and the greater the number of nodes since mixed routes become more cost-effective in this environment. In this environment, two factors make mixed routes more attractive than dedicated routes: 1.) due to the large number of delivery nodes, a large number of delivery routes are available for backhaul insertions, and 2) since the delivery routes span a greater portion of the Euclidean space, it becomes more economical to insert backhauls onto these routes than to create dedicated backhaul routes. Thus, LBI is better suited to find cost-effective solutions in this scenario.

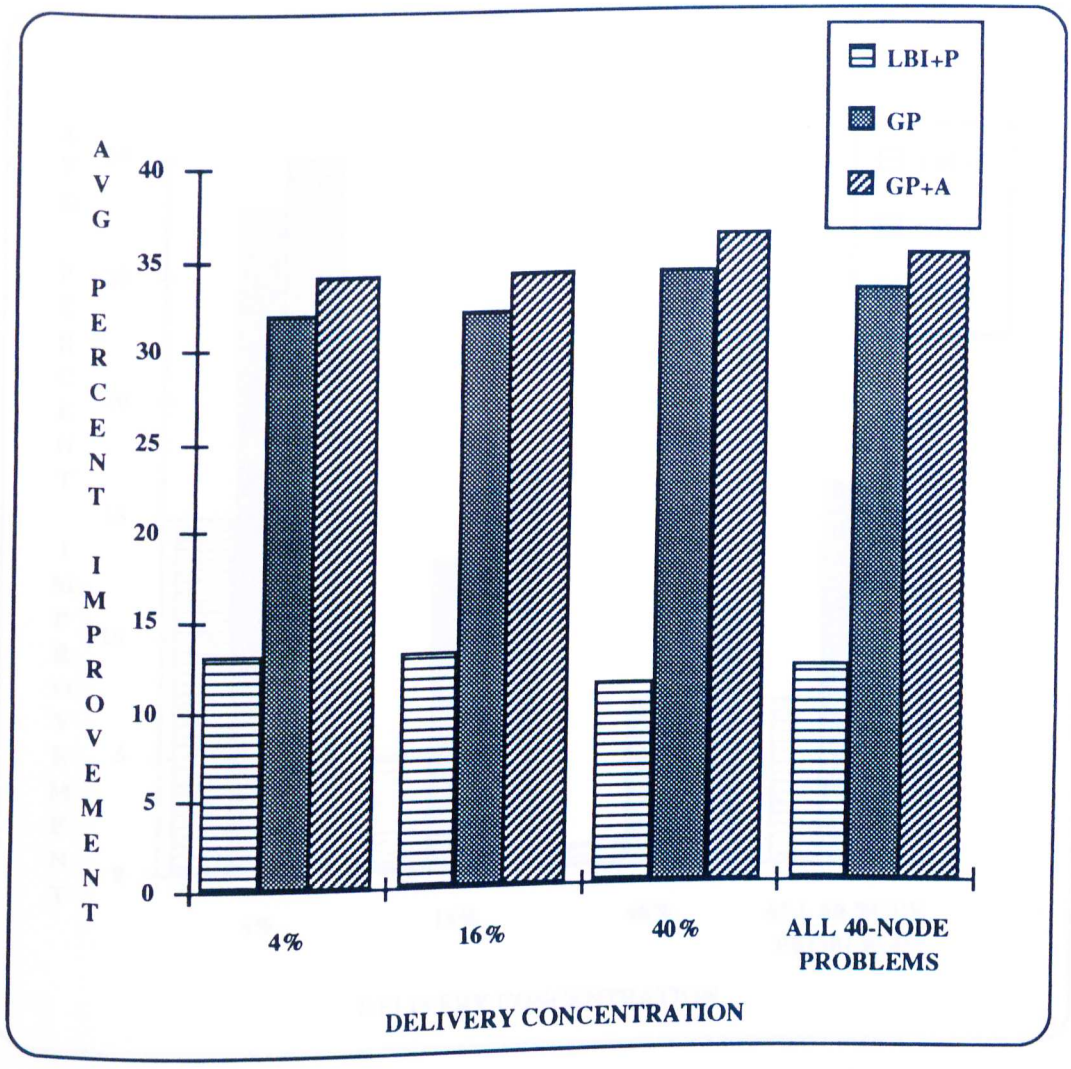


Figure 5.20 Avg. percent improvement over LBI (40-node problems and delivery concentration clustering).

LBI+P, GP, and GP+A outperform LBI across all delivery concentration regions for 40-node problems.

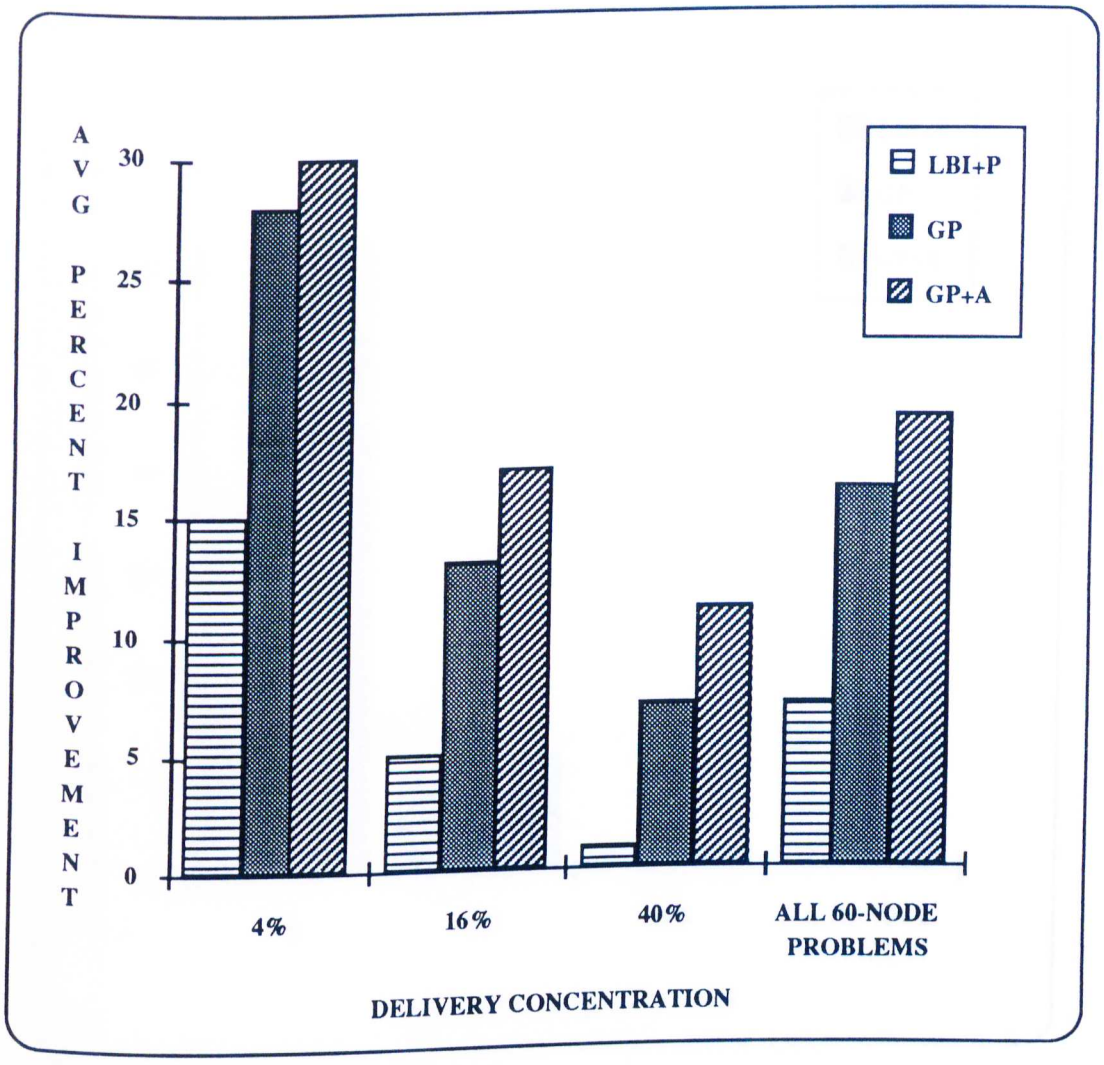


Figure 5.21 Avg. percent improvement over LBI (60-node problems and delivery concentration clustering).

As the number of nodes increases to 60, LBI+P, GP, and GP+A show a significant deterioration in their effectiveness to produce better solutions than LBI.

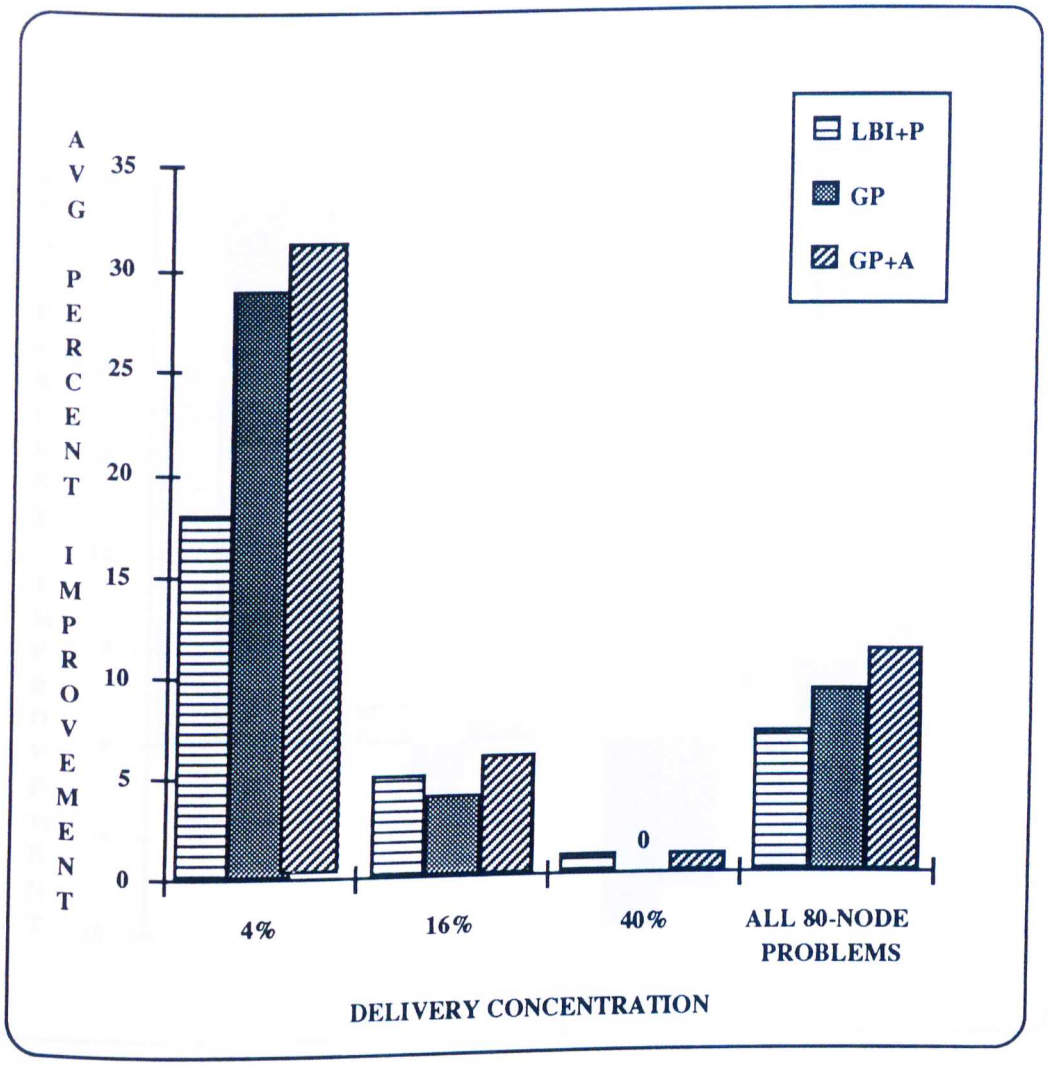


Figure 5.22 Avg. percent improvement over LBI (80-node problems and delivery concentration clustering).

As the number of nodes increases to 80, LBI+P, GP, and GP+A show a significant deterioration in their effectiveness to produce better solutions than LBI. Note that the average percent improvement of GP over LBI is zero for delivery concentration regions of 40%.

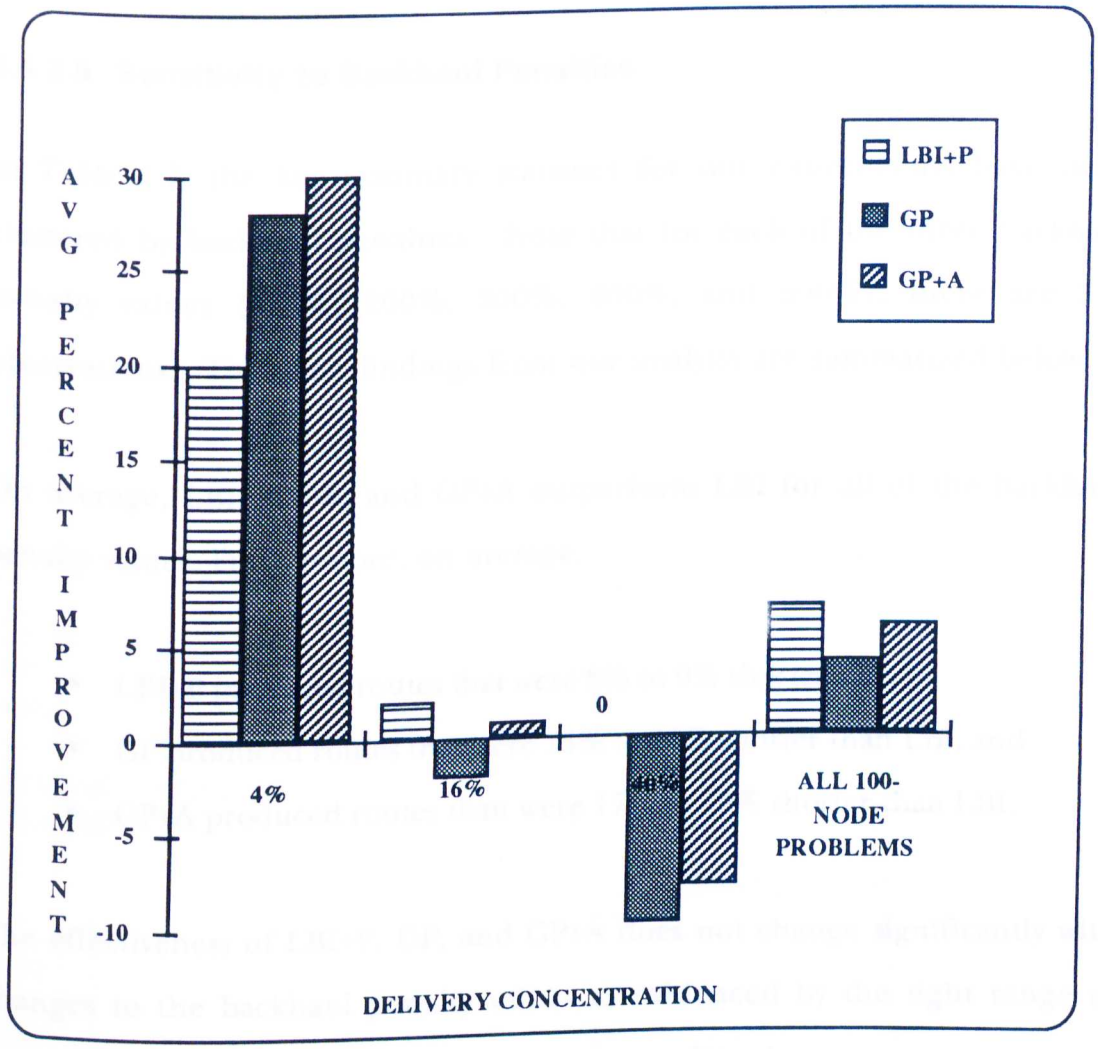


Figure 5.23 Avg. percent improvement over LBI (100-node problems and delivery concentration clustering).

As the number of nodes increases to 100, LBI+P, GP, and GP+A show a significant deterioration in their effectiveness to produce better solutions than LBI. Note that for delivery concentration regions of 16%, and 40%, GP produces solutions which are, on average, longer than LBI. Additionally, for delivery concentration regions of 40%, LBI+P produces solutions which are, on average, as long as LBI. And finally, for delivery concentration regions of 40%, GP+A produces solutions which are, on average, longer than LBI

5.4.2.5 Sensitivity to Backhaul Penalties

In *Table 5.7*, the key summary statistics for our experiments have been clustered by backhaul penalties. Note that for each of the three backhaul penalty values (100%, 200%, 300%, 400%, and 500%), there are 540 observations. The main findings from our analysis are summarized below.

On average, LBI+P, GP, and GP+A outperform LBI for all of the backhaul penalty values. Furthermore, on average,

- LBI+P produced routes that were 8% to 9% shorter LBI,
- GP produced routes that were 15% to 16% shorter than LBI, and
- GP+A produced routes than were 17% to 19% shorter than LBI.

The effectiveness of LBI+P, GP, and GP+A does not change significantly with changes to the backhaul penalty values as evidenced by the tight range of solution improvement percentages for each penalty values.

Figures 5.24 through *5.26* illustrate the accuracy trends of the solution procedures as the backhaul penalties change.

NUMBER OF SOLUTIONS BETTER THAN LBI

	BACKHAUL PENALTY				
	100%	200%	300%	400%	500%
LBI+P	260	274	284	278	273
GP	365	379	376	367	358
GP+A	386	390	386	384	376

AVERAGE PERCENT IMPROVEMENT OVER LBI

	BACKHAUL PENALTY				
	100%	200%	300%	400%	500%
LBI+P	8	8	9	8	9
GP	15	16	16	15	15
GP+A	17	18	19	18	17

AVERAGE SOLUTION LENGTH

	BACKHAUL PENALTY				
	100%	200%	300%	400%	500%
LBI	2,520	2,574	2,573	2,545	2,530
LBI+P	2,334	2,374	2,361	2,343	2,318
GP	2,172	2,203	2,200	2,202	2,198
GP+A	2,128	2,152	2,144	2,144	2,136

Table 5.7 Computational experiment summary (backhaul penalty clustering).

These results show a clear advantage for the LBI+P, GP, and GP+A across all backhaul penalty values. Note that the performance advantages of LBI+P, GP, and GP+A solutions over LBI solutions do not change significantly as the backhaul penalty values change.

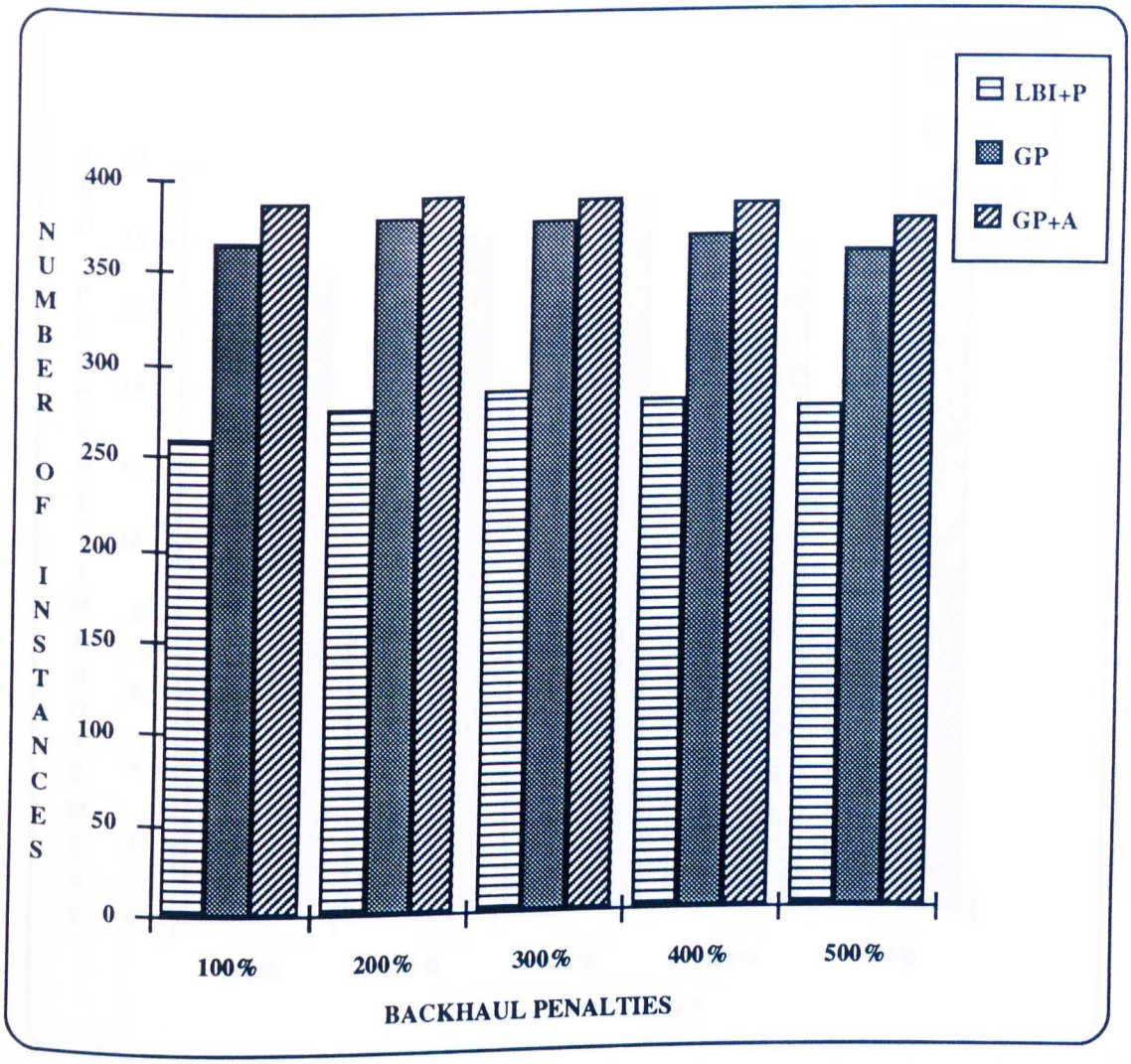


Figure 5.24 Number of solutions better than LBI (backhaul penalty clustering).

This figure illustrates the advantage of LBI+P, GP, and GP+A over LBI across all backhaul penalty values. Note that the number of instances where LBI+P, GP, and GP+A solutions are better than LBI solutions does not change significantly as the backhaul penalty values change.

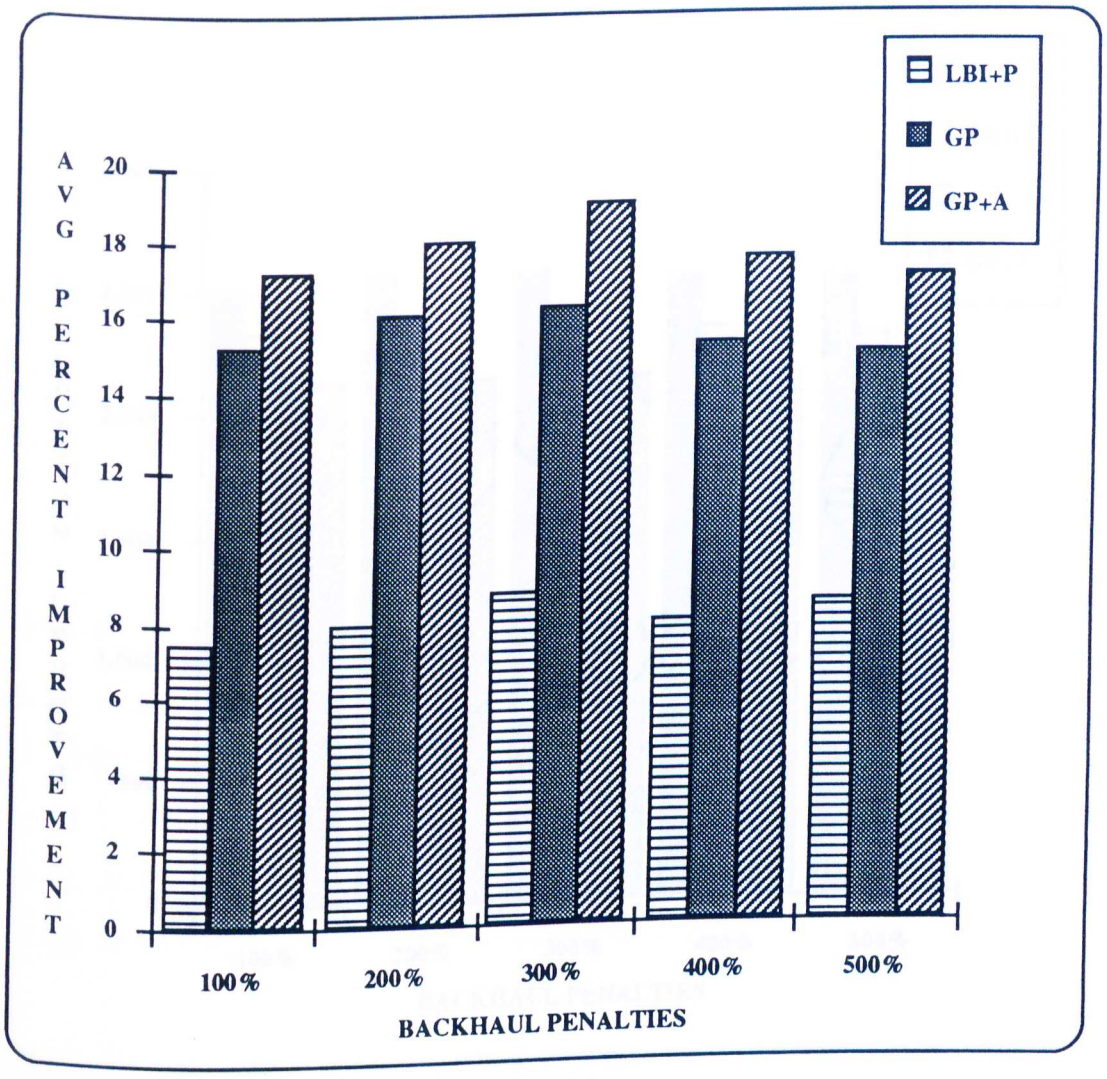


Figure 5.25 Avg. percent improvement over LBI (backhaul penalty clustering).

LBI+P, GP, and GP+A clearly outperform LBI across all backhaul penalty values.Note that the average percent improvement of LBI+P, GP, and GP+A solutions over LBI solutions does not change significantly as the backhaul penalty values change.

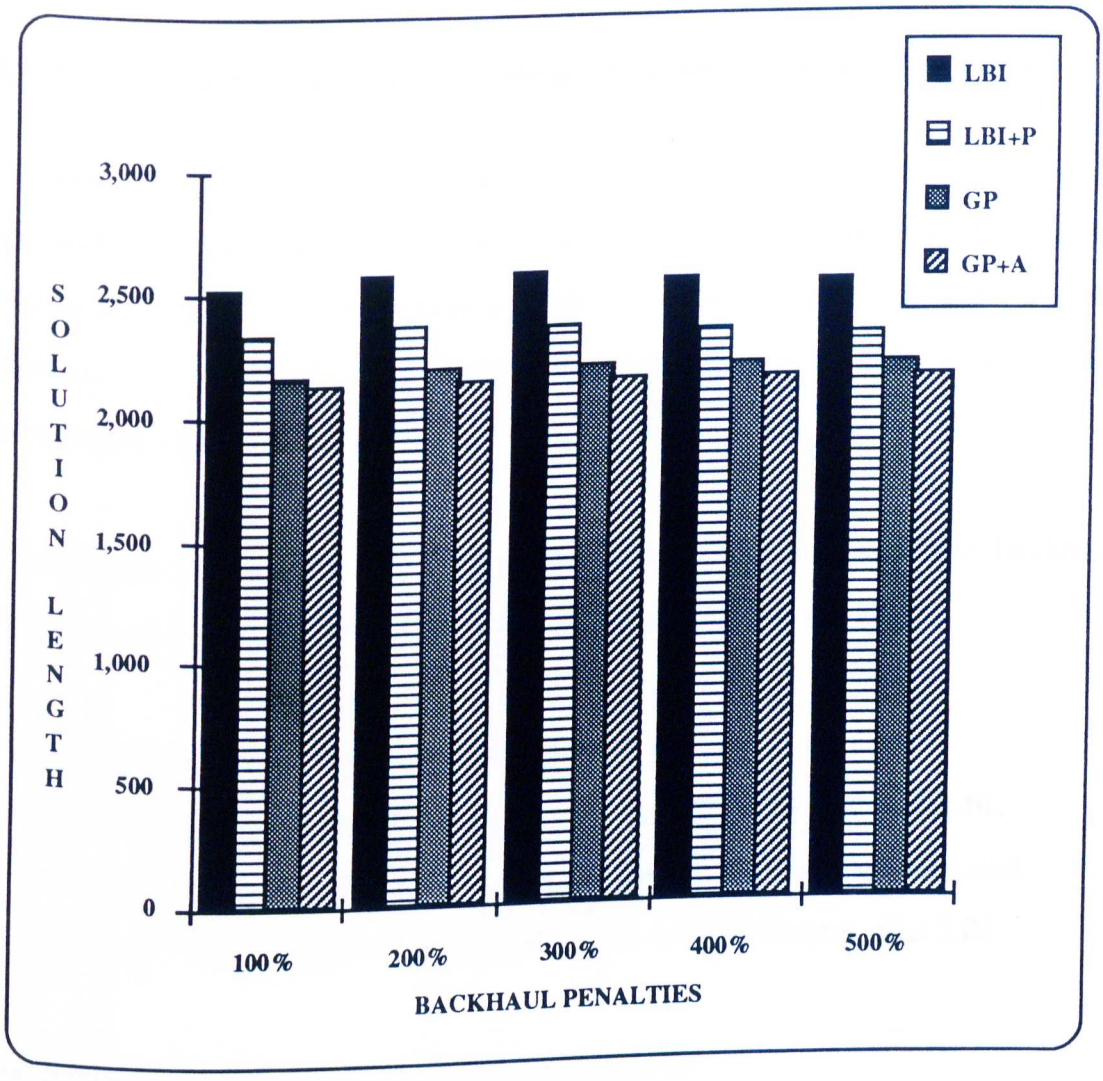


Figure 5.26 Avg. solution length (backhaul penalty clustering).

LBI+P, GP, and GP+A clearly outperform LBI across all backhaul penalty values.

5.4.2.6 Interaction between Problem Size and Backhaul Penalties

In *Table 5.8*, we examine the impact of the interaction of the problem size and the backhaul penalties on the effectiveness of LBI+P, GP, and GP+A over LBI. In this analysis, the percent improvement over the LBI solution is used as the measure of effectiveness. The main findings from this analysis are summarized below.

On average, LBI+P, GP, and GP+A outperform LBI for all of the backhaul penalties and problem sizes examined. Furthermore, on average,

For 40-node problems, and all backhaul penalties:

- LBI+P produced routes that were 11% to 13% shorter than LBI,
- GP produced routes that were 31% to 34% shorter than LBI, and
- GP+A produced routes that were 33% to 36% shorter than LBI.

For 60-node problems, and all backhaul penalties:

- LBI+P produced routes that were 6% to 8% shorter than LBI,
- GP produced routes that were 12% to 19% shorter than LBI, and
- GP+A produced routes that were 15% to 22% shorter than LBI.

For 80-node problems, and all backhaul penalties:

- LBI+P produced routes that were 5% to 7% shorter than LBI,
- GP produced routes that were 7% to 11% shorter than LBI, and
- GP+A produced routes that were 9% to 13% shorter than LBI.

For 100-node problems, and all backhaul penalties:

- LBI+P produced routes that were 7% shorter than LBI,
- GP produced routes that were 3% to 8% shorter than LBI, and
- GP+A produced routes that were 4% to 10% shorter than LBI.

Close examination of this data reveals that changes in the backhaul penalty values do not impact the effectiveness of LBI+P, GP, and GP+A. For example, *Table 5.8* shows that GP+A has average improvement over LBI for all 40-node problems of 35% which closely coincides with GP+A the average improvement values recorded for each penalty value in the 40-node problem set. This pattern is repeated for the 100-node problems where the overall GP+A average improvement over LBI is 6%, and where the CP+A average improvement over LBI recorded for each penalty value is in the 5% to 10% range.

It is important to note a slight decrease in effectiveness of GP, and GP+A over LBI as the backhaul penalties increase for 100-node problems. The decrease in effectiveness of GP and GP+A for 100-node problems with large backhaul penalty values can be attributed to the fact that LBI, and LBI+P can adjust to increases in backhaul penalty values earlier in the route construction process.

Figures 5.27 through *5.30* illustrate the accuracy trends of the solution procedures as the problem size and backhaul penalty parameters change.

AVERAGE PERCENT IMPROVEMENT OVER LBI (40-NODES)

	BACKHAUL PENALTY					
	100%	200%	300%	400%	500%	ALL 40-NODE PROBLEMS
LBI+P	12	12	13	11	12	12
GP	34	33	33	31	32	33
GP+A	35	36	36	33	34	35

AVERAGE PERCENT IMPROVEMENT OVER LBI (60-NODES)

	BACKHAUL PENALTY					
	100%	200%	300%	400%	500%	ALL 60-NODE PROBLEMS
LBI+P	6	6	8	7	8	7
GP	12	15	19	17	16	16
GP+A	15	17	22	21	19	19

AVERAGE PERCENT IMPROVEMENT OVER LBI (80-NODES)

	BACKHAUL PENALTY					
	100%	200%	300%	400%	500%	ALL 80-NODE PROBLEMS
LBI+P	5	7	7	7	7	7
GP	7	11	9	10	9	9
GP+A	9	13	10	12	10	11

AVERAGE PERCENT IMPROVEMENT OVER LBI (100 NODES)

	BACKHAUL PENALTY					
	100%	200%	300%	400%	500%	ALL 100-NODE PROBLEMS
LBI+P	7	7	7	7	7	7
GP	8	5	4	3	3	4
GP+A	10	6	6	4	5	6

Table 5.8 Computational experiment summary (problem size/backhaul penalty clustering).

These results reveal that changes in the backhaul penalty values do not impact the effectiveness of LBI+P, GP, and GP+A. For example, Table 5.8 shows that GP+A has average improvement over LBI for all 40-node problems of 35% which closely coincides with GP+A the average improvement values recorded for each penalty value in the 40-node problem set. This pattern is repeated for the 100-node problems where the overall GP+A average improvement over LBI is 6%, and where the CP+A average improvement over LBI recorded for each penalty value is in the 5% to 10% range.

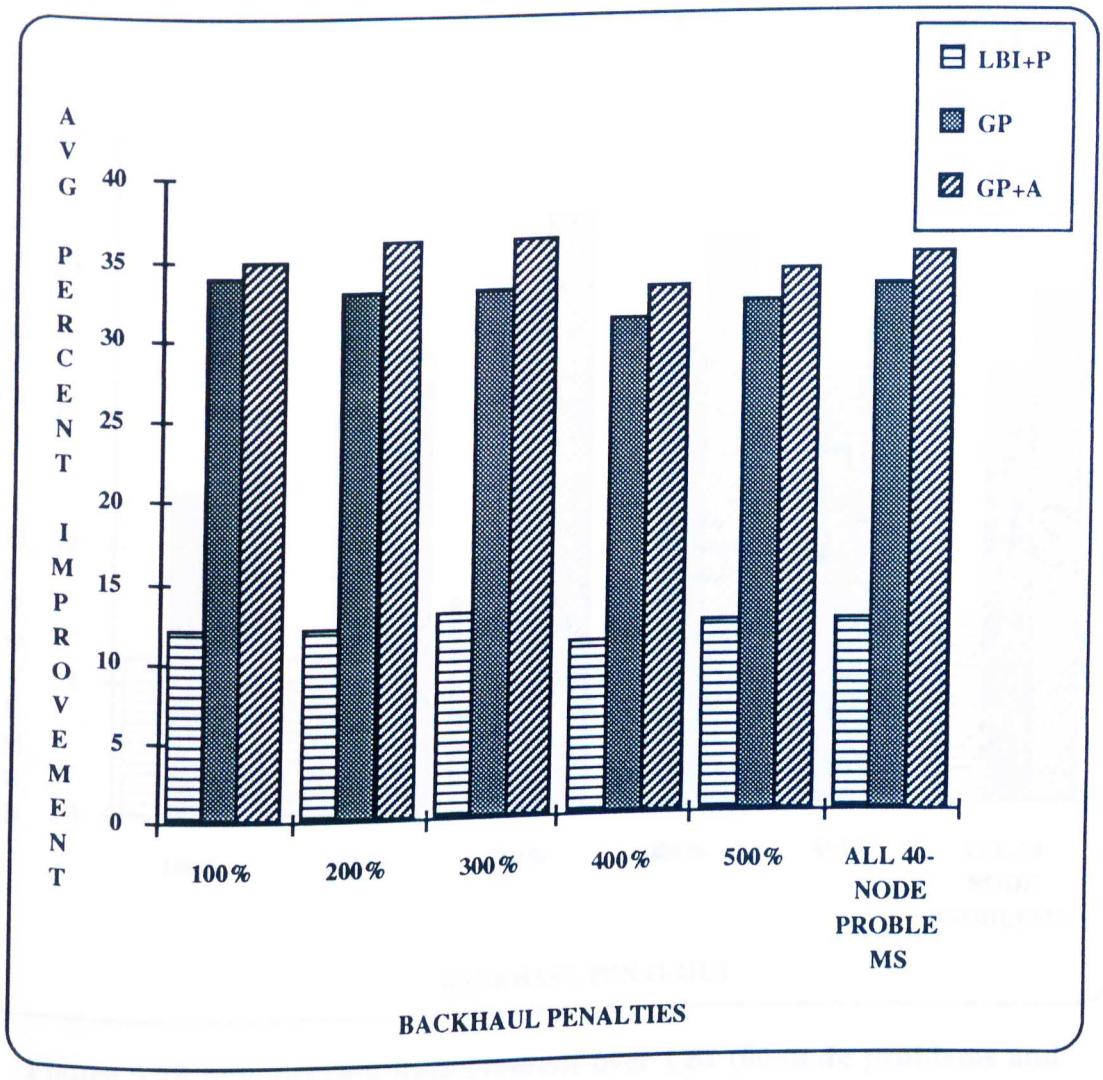


Figure 5.27 Avg. percent improvement over LBI (40-node problems and backhaul penalty clustering).

This figure shows no significant improvement in effectiveness of LBI+P, GP, and GP+A over LBI across all backhaul penalties for 40-node problems.

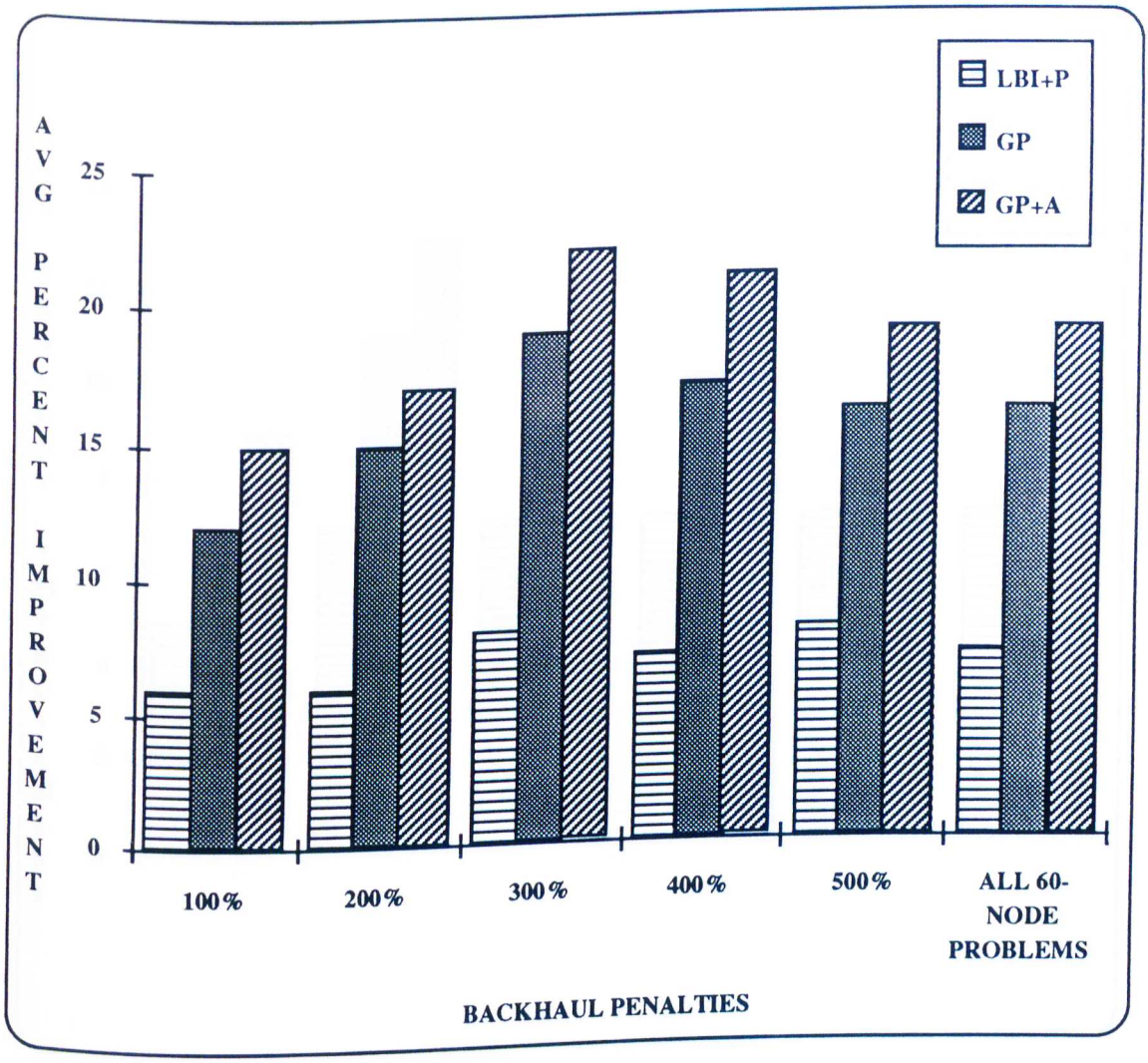


Figure 5.28 Avg. percent improvement over LBI (60-node problems and backhaul penalty clustering).

This figure shows no significant improvement in effectiveness of LBI+P, GP, and GP+A over LBI across all backhaul penalties for 60-node problems.

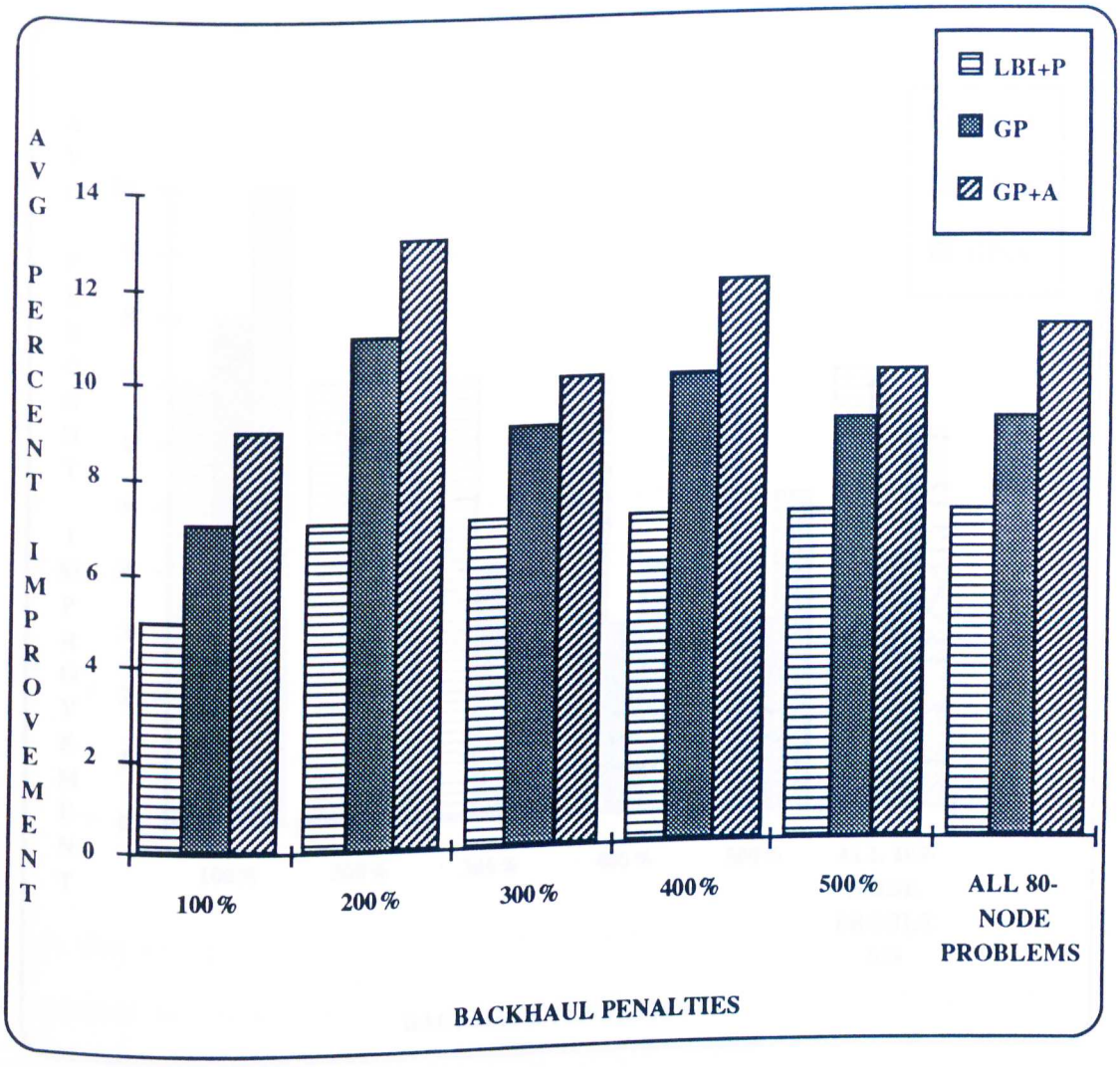


Figure 5.29 Avg. percent improvement over LBI (80-node problems and backhaul penalty clustering).

This figure shows no significant improvement in effectiveness of LBI+P, GP, and GP+A over LBI across all backhaul penalties for 80-node problems.

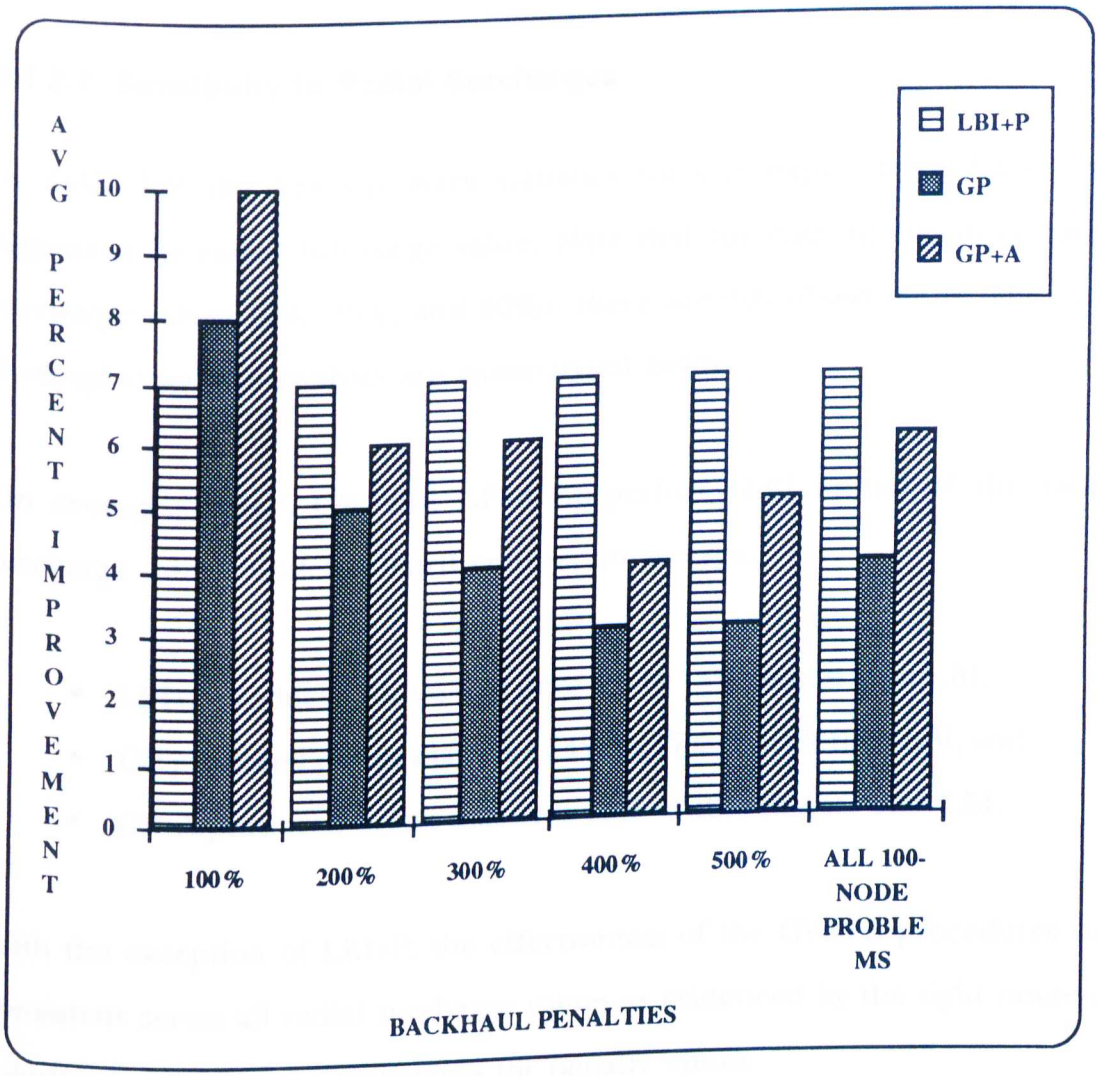


Figure 5.30 Avg. percent improvement over LBI (100-node problems and backhaul penalty clustering).

This figure shows a slight decrease in effectiveness of GP, and GP+A over LBI as the backhaul penalties increase for 100-node problems. The decrease in effectiveness of GP and GP+A for 100-node problems with large backhaul penalty values can be attributed to the fact that LBI, and LBI+P can adjust to increases in backhaul penalty values earlier in the route construction process.

5.4.2.7 Sensitivity to Radial Surcharges

In *Table 5.9*, the key summary statistics for our experiments have been organized by radial surcharge value. Note that for each of the three radial surcharge values (0%, 25%, and 50%), there are 900 observations. The main findings from this analysis are summarized below.

On average, LBI+P, GP, and GP+A outperform LBI for all of the radial surcharge values modeled. Furthermore, on average,

- LBI+P produced routes that were 4% to 13% shorter than LBI,
- GP produced routes that were 14% to 17% shorter than LBI, and
- GP+A produced routes that were 16% to 19% shorter than LBI.

With the exception of LBI+P, the effectiveness of the GVRPB procedures was consistent across all radial surcharge values as evidenced by the tight range of solution improvement percentages for penalty values.

LBI+P was more effective in finding lower-cost solutions than LBI for problems with smaller radial surcharges. For example, LBI+P outperformed LBI for the 0% radial surcharge problems in 584 out of 900 instances by an average improvement of 13%. On the other hand, LBI+P outperformed LBI for the 50% radial surcharge problems in 324 out of 900 instances by an average improvement of just 4%. This poor performance can be explained by the observation that higher radial surcharge values force backhauls into mixed

routes early in the LBI+P procedure, thus reducing the number of backhauls available during the post-processor procedure.

Figures 5.31 through *5.33* illustrate the accuracy trends of the solution procedures as the radial surcharges change.

NUMBER OF SOLUTIONS BETTER THAN LBI

	RADIAL SURCHARGE		
	0%	25%	50%
LBI+P	584	464	324
GP	640	628	580
GP+A	660	660	620

AVERAGE PERCENT IMPROVEMENT OVER LBI

	RADIAL SURCHARGE		
	0%	25%	50%
LBI+P	13	8	4
GP	14	17	16
GP+A	16	19	18

AVERAGE SOLUTION LENGTH

	RADIAL SURCHARGE		
	0%	25%	50%
LBI	2,512	2,560	2,573
LBI+P	2,195	2,372	2,472
GP	2,190	2,180	2,215
GP+A	2,145	2,126	2,152

Table 5.9 Computational experiment summary (radial surcharge clustering).

These results show a clear advantage for LBI+P, GP, and GP+A for all radial surcharge values. With the exception of LBI+P, the effectiveness of GP, and GP+A was relatively constant across all radial surcharge values. LBI+P performed poorly for higher radial surcharge values because the higher values force backhauls into mixed routes early in the procedure, thus reducing the effectiveness of the post-processor.

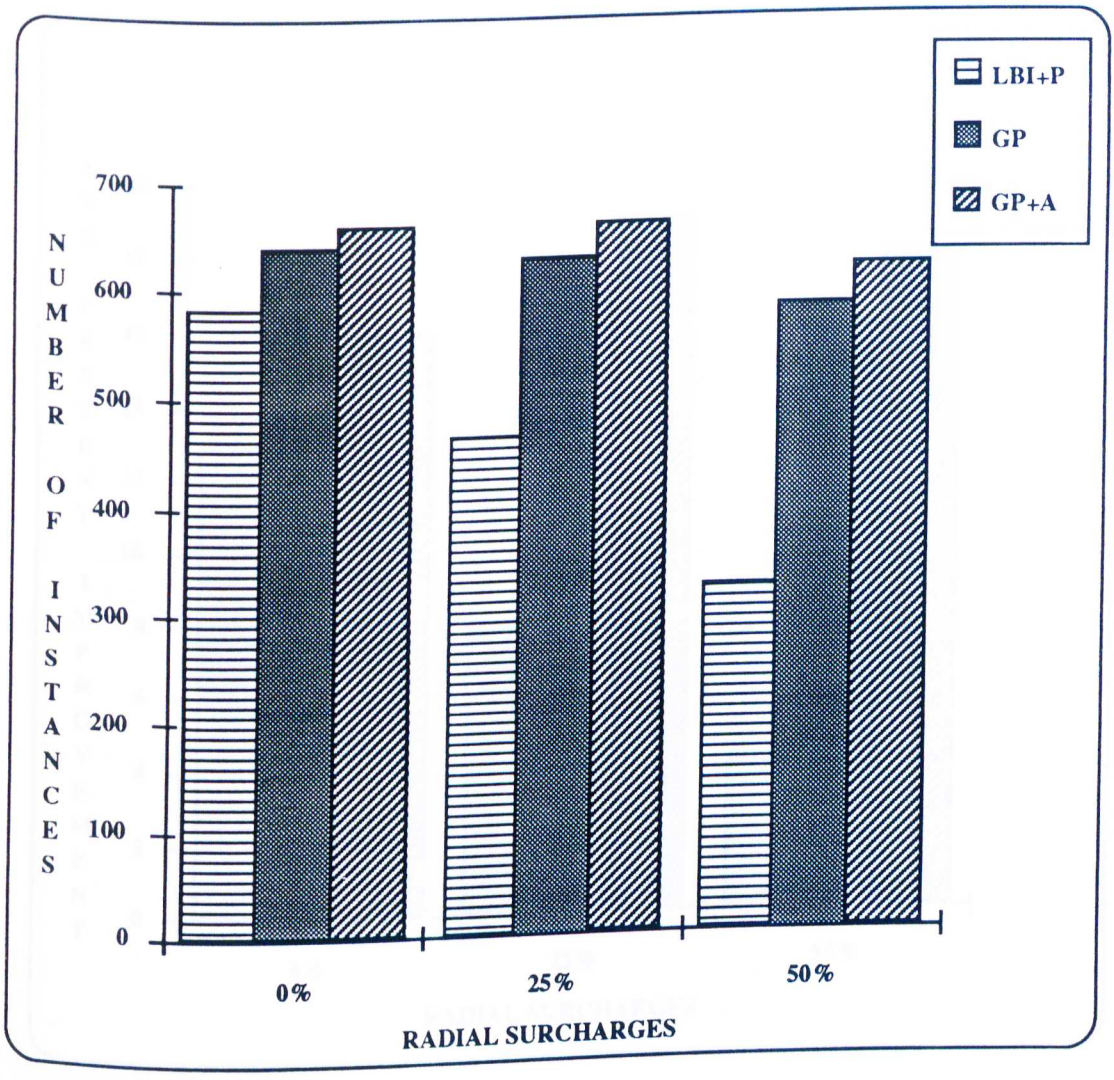


Figure 5.31 Number of solutions better than LBI (radial surcharge clustering).

LBI+P, GP, and GP+A outperform LBI across all radial surcharge values. LBI+P performed poorly for higher radial surcharge values because the higher values force backhauls into mixed routes early in the procedure, thus reducing the effectiveness of the post-processor.

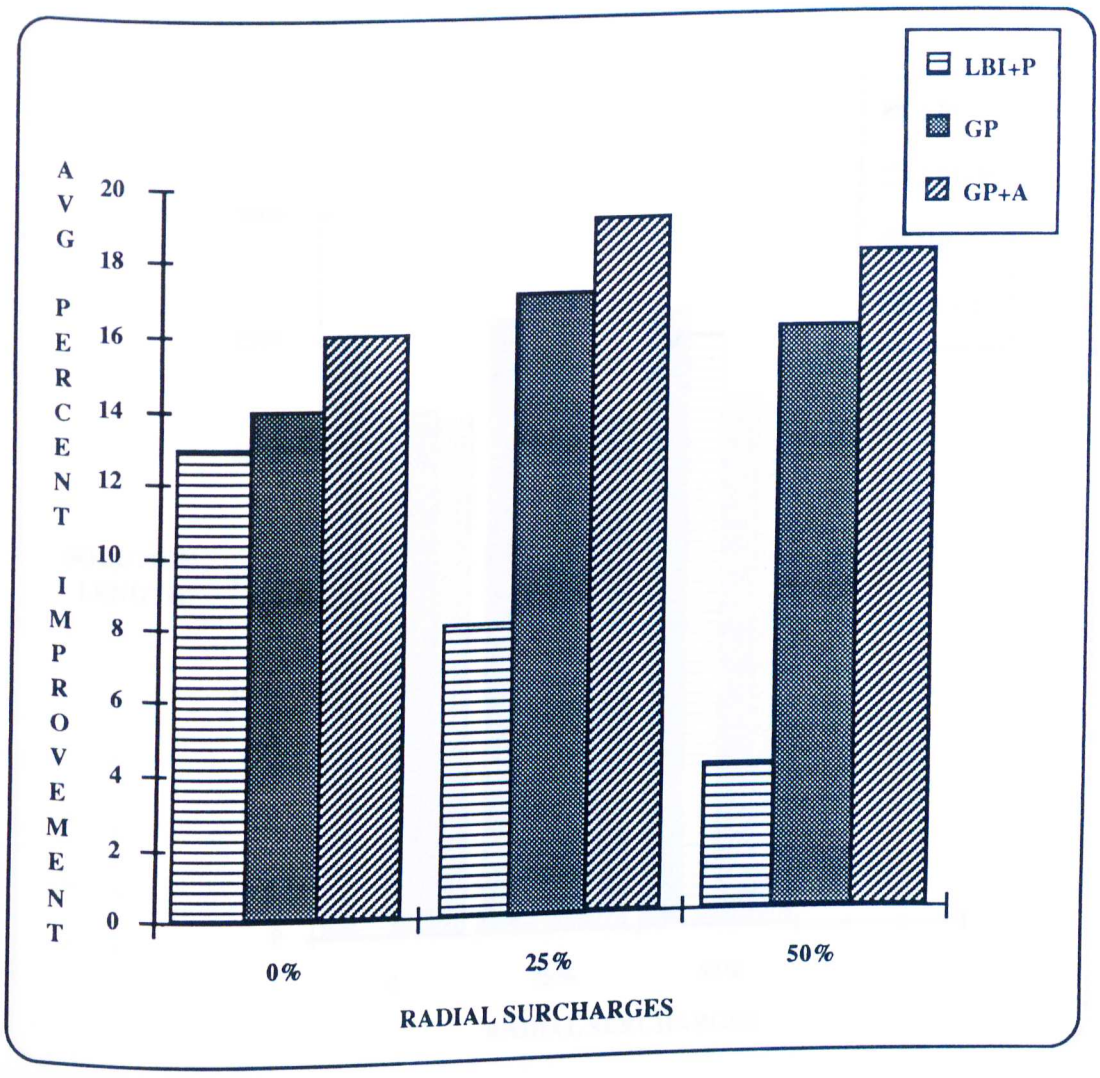


Figure 5.32 Avg. percent improvement over LBI (radial surcharge clustering).

LBI+P, GP, and GP+A outperform LBI across all radial surcharge values. LBI+P performed poorly for higher radial surcharge values because the higher values force backhauls into mixed routes early in the procedure, thus reducing the effectiveness of the post-processor.

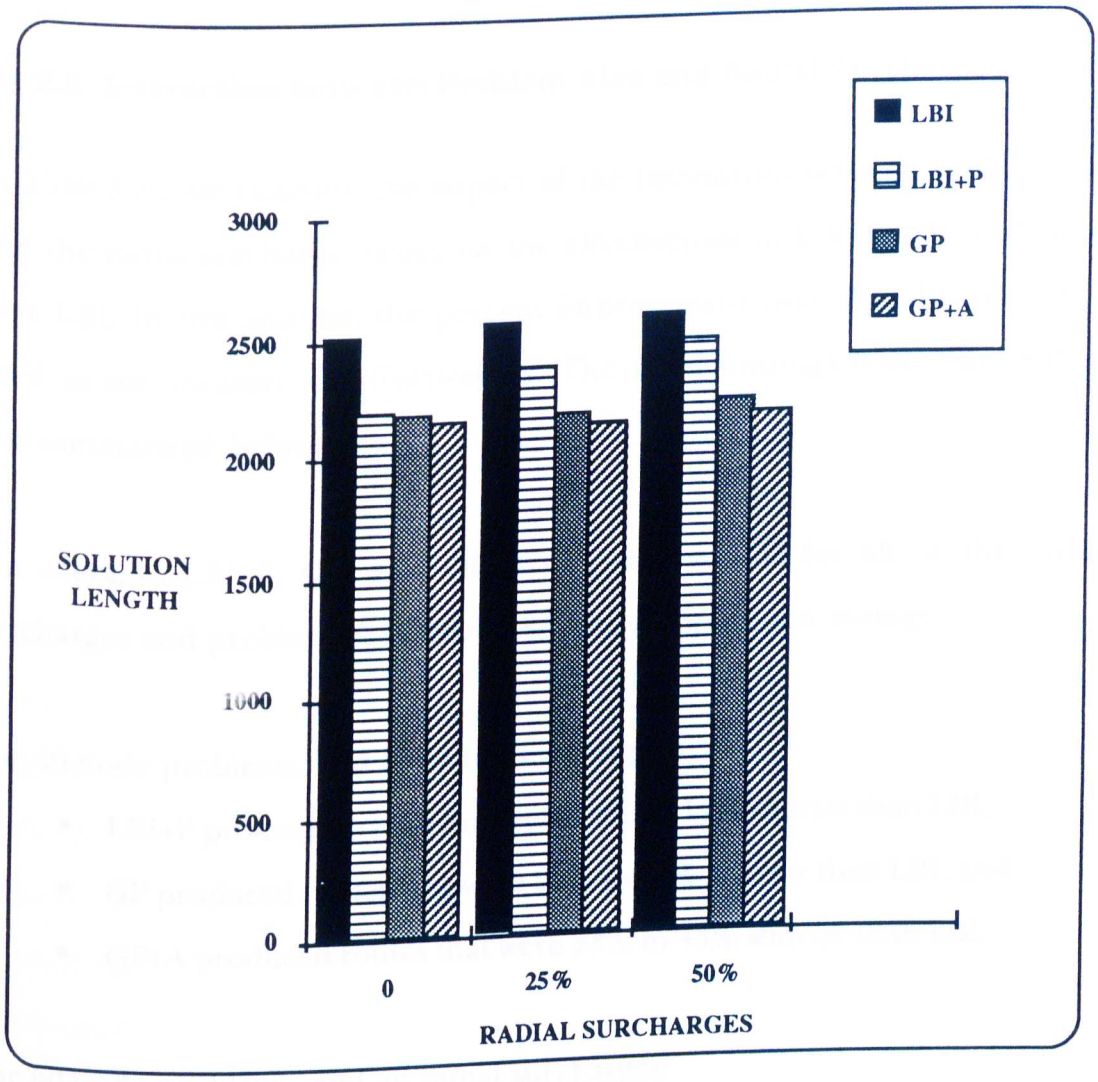


Figure 5.33 Avg. solution length (radial surcharge clustering).

Except LBI+P, the remaining procedures yield relatively constant solution lengths across all radial surcharge values. LBI+P performed poorly for higher radial surcharge values because the higher values force backhauls into mixed routes early in the procedure, thus reducing the effectiveness of the post-processor.

5.4.2.8 Interaction between Problem Size and Radial Surcharges

In *Table 5.10*, we examine the impact of the interaction between problem size and the radial surcharge values on the effectiveness of LBI+P, GP, and GP+A over LBI. In this analysis, the percent improvement over the LBI solution is used as the measure of effectiveness. The main findings from this analysis are summarized below.

On average, LBI+P, GP, and GP+A outperform LBI for all of the radial surcharges and problem sizes examined. Furthermore, on average,

For 40-node problems, and all radial surcharges:

- LBI+P produced routes that were 10% to 14% shorter than LBI,
- GP produced routes that were 21% to 41% shorter than LBI, and
- GP+A produced routes that were 23% to 43% shorter than LBI.

For 60-node problems, and all radial surcharges:

- LBI+P produced routes that were 2% to 12% shorter than LBI,
- GP produced routes that were 13% to 17% shorter than LBI, and
- GP+A produced routes that were 17% to 21% shorter than LBI.

For 80-node problems, and all radial surcharges:

- LBI+P produced routes that were 2% to 12% shorter than LBI,
- GP produced routes that were 7% to 11% shorter than LBI, and
- GP+A produced routes that were 8% to 12% shorter than LBI.

For 100-node problems, and all radial surcharges:

- LBI+P produced routes that were 2% to 13% shorter than LBI,
- GP produced routes that were 1% to 8% shorter than LBI, and
- GP+A produced routes that were 3% to 9% shorter than LBI.

Close examination of this data reveals that , with the exception of 40-node problems, the interaction of problem size and the radial surcharge values does not significantly impact the effectiveness of LBI+P, GP, and GP+A. For example, *Table 5.10* shows that for the 60-node problems the overall GP+A average improvement over LBI is 19% which is within the GP+A average improvement range of 17% to 21% for each radial surcharge value in the 40-node problem set. This pattern is repeated for the 100-node problems where the overall GP+A average improvement over LBI is 6%, and where the CP+A average improvement over LBI recorded for each penalty value is in the 3% to 9% range. On the other hand, the 40-node problems appear to react more to changes in the radial surcharge values. *Table 5.10* shows that for all 40-node problems the overall GP+A average improvement over LBI is 35% which significantly different than the GP+A average improvement range of 23% to 43% for each radial surcharge value. The reason for this behavior can be attributed to LBI's ability to better adapt to increases in radial surcharges as the number of nodes goes up. That is, for problems with a large number of nodes LBI will tend to find a number of attractive backhaul insertion options. Whereas, for problems with a small number of nodes LBI may be forced to

select between a common carrier solution with a high radial surcharge, and an unattractive backhaul insertion.

Figures 5.34 through *5.37* illustrate the accuracy trends of the solution procedures as the problem size and radial surcharge parameters change.

AVERAGE PERCENT IMPROVEMENT OVER LBI(40-NODES)

	RADIAL SURCHARGE			
	0%	25%	50%	ALL 40-NODE PROBLEMS
LBI+P	14	12	10	12
GP	21	33	41	33
GP+A	23	35	43	35

AVERAGE PERCENT IMPROVEMENT OVER LBI(60-NODES)

	RADIAL SURCHARGE			
	0%	25%	50%	ALL 60-NODE PROBLEMS
LBI+P	12	7	2	7
GP	17	18	13	16
GP+A	19	21	17	19

AVERAGE PERCENT IMPROVEMENT OVER LBI(80-NODES)

	RADIAL SURCHARGE			
	0%	25%	50%	ALL 80-NODE PROBLEMS
LBI+P	12	6	2	7
GP	11	11	7	9
GP+A	12	12	8	11

AVERAGE PERCENT IMPROVEMENT OVER LBI(100-NODES)

	RADIAL SURCHARGE			
	0%	25%	50%	ALL 100-NODE PROBLEMS
LBI+P	13	6	2	7
GP	8	5	1	4
GP+A	9	7	3	6

Table 5.10 Computational experiment summary (problem size/radial surcharge clustering).

This table reveals that with the exception of 40-node problems, the interaction of problem size and the radial surcharge values does not significantly impact the effectiveness of LBI+P, GP, and GP+A. For example, Table 5.10 shows that for the 60-node problems the overall GP+A average improvement over LBI is 19% which is within the GP+A average improvement range of 17% to 21% for each radial surcharge value in the 40-node problem set. This pattern is repeated for the 100-node problems where the overall GP+A average improvement over LBI is 6%, and where the CP+A average improvement over LBI recorded for each penalty value is in the 3% to 9% range. On the other

hand, the 40-node problems appear to react more to changes in the radial surcharge values. *Table 5.10* shows that for all 40-node problems the overall GP+A average improvement over LBI is 35% which significantly different than the GP+A average improvement range of 23% to 43% for each radial surcharge value. The reason for this behavior can be attributed to LBI's ability to better adapt to increases in radial surcharges as the number of nodes goes up. That is, for problems with a large number of nodes LBI will tend to find a number of attractive backhaul insertion options. Whereas, for problems with a small number of nodes LBI may be forced to select between a common carrier solution with a high radial surcharge, and an unattractive backhaul insertion.

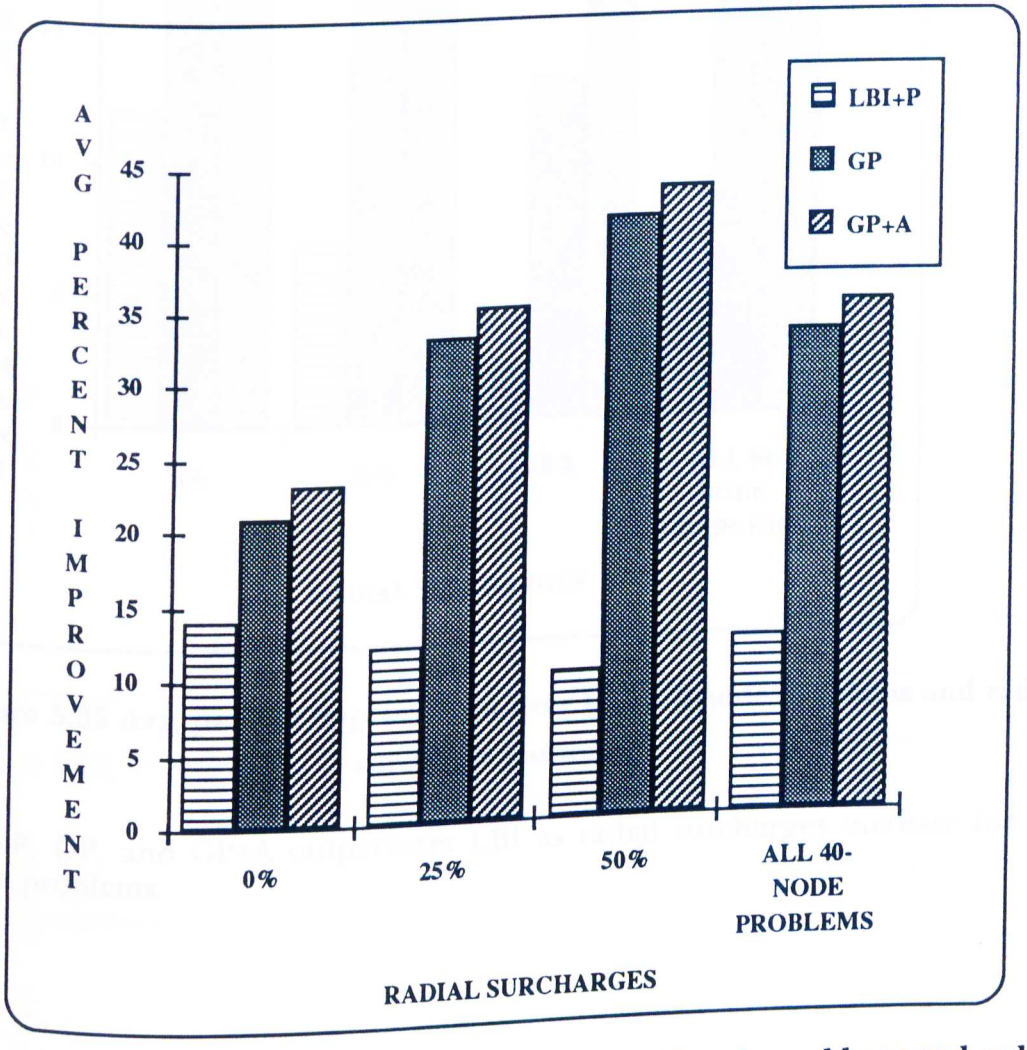


Figure 5.34 Avg. percent improvement over LBI (40-node problems and radial surcharge clustering).

LBI+P, GP, and GP+A outperform LBI as radial surcharges increase for 40-node problems.

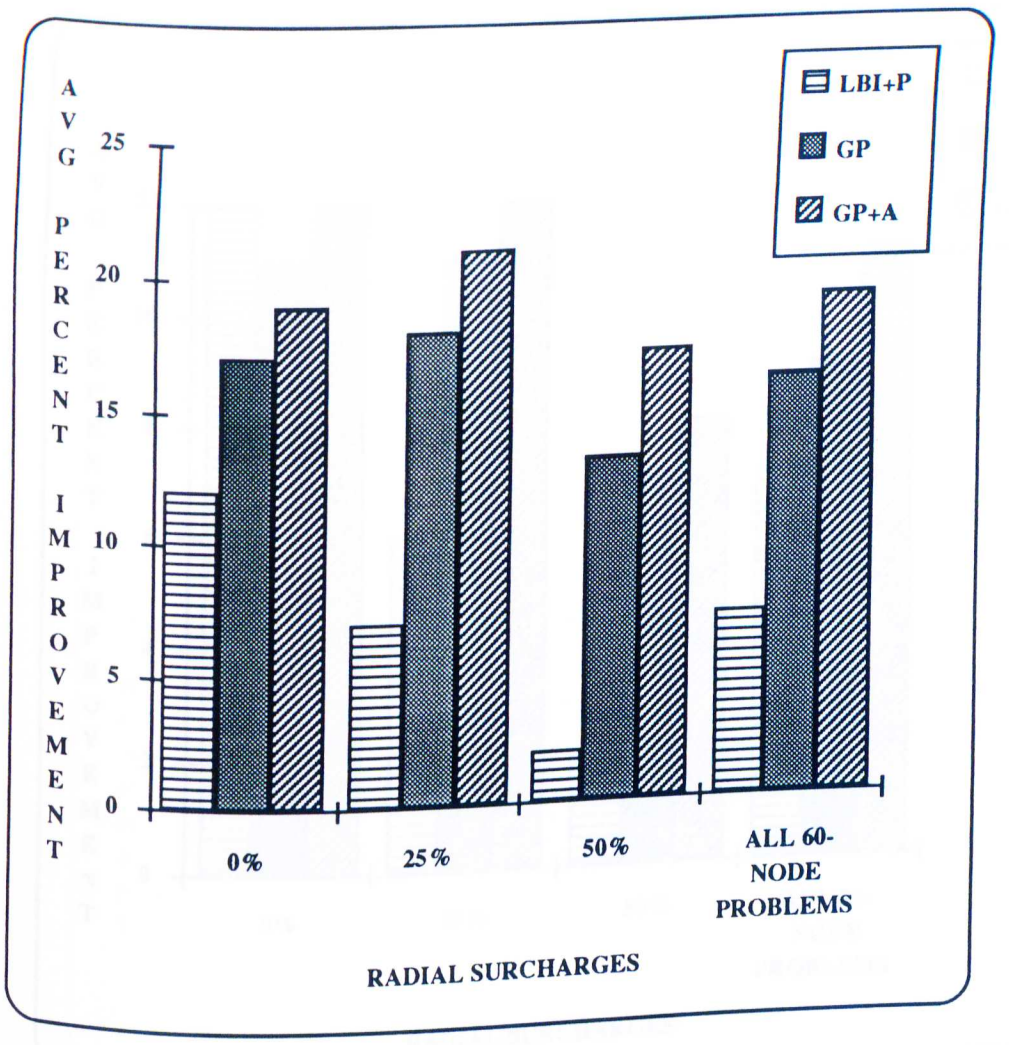


Figure 5.35 Avg. percent improvement over LBI (60-node problems and radial surcharge clustering).

LBI+P, GP, and GP+A outperform LBI as radial surcharges increase for 60-node problems.

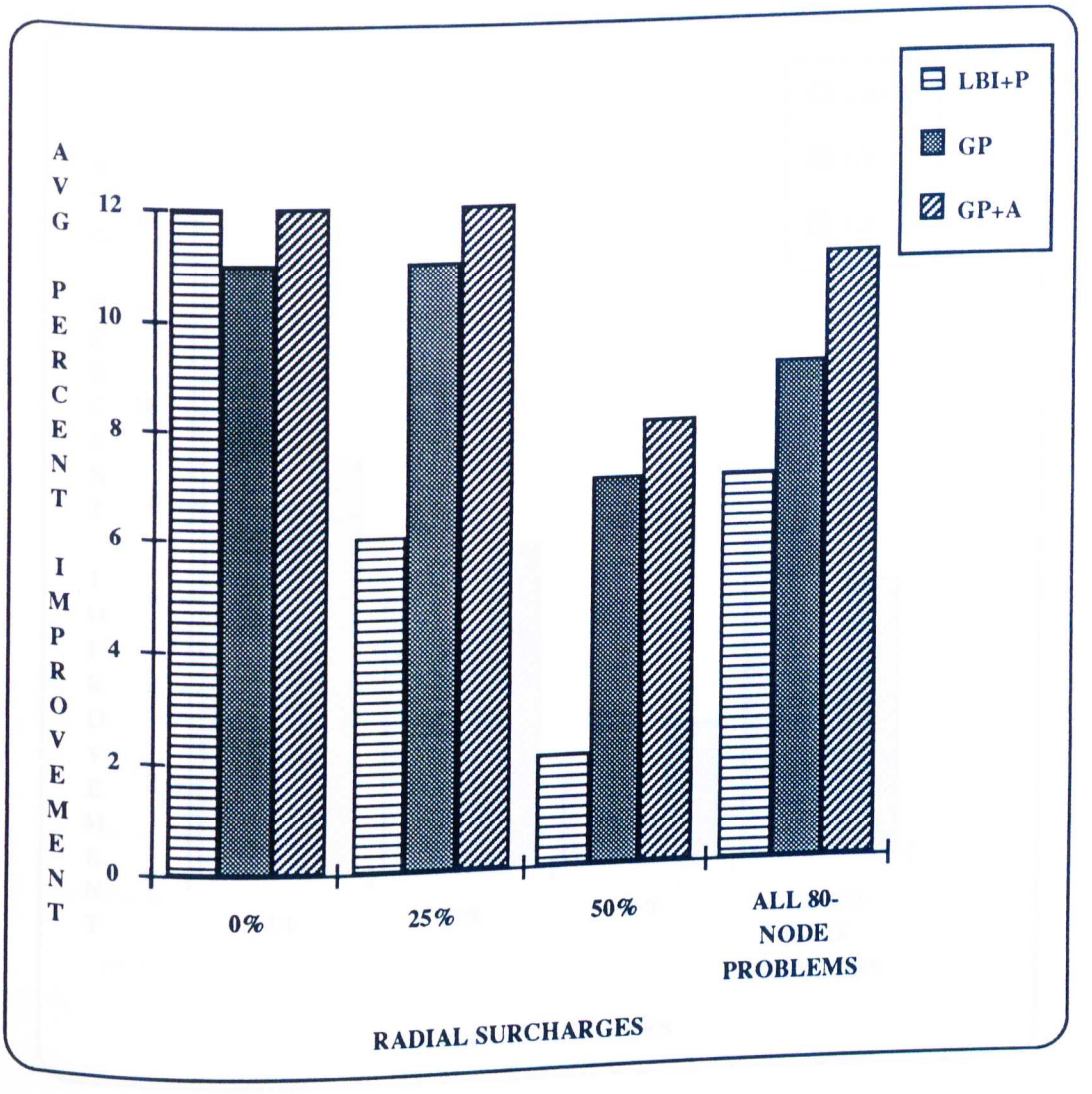


Figure 5.36 Avg. percent improvement over LBI (80-node problems and radial surcharge clustering).

LBI+P, GP, and GP+A outperform LBI as radial surcharges increase for 80-node problems.

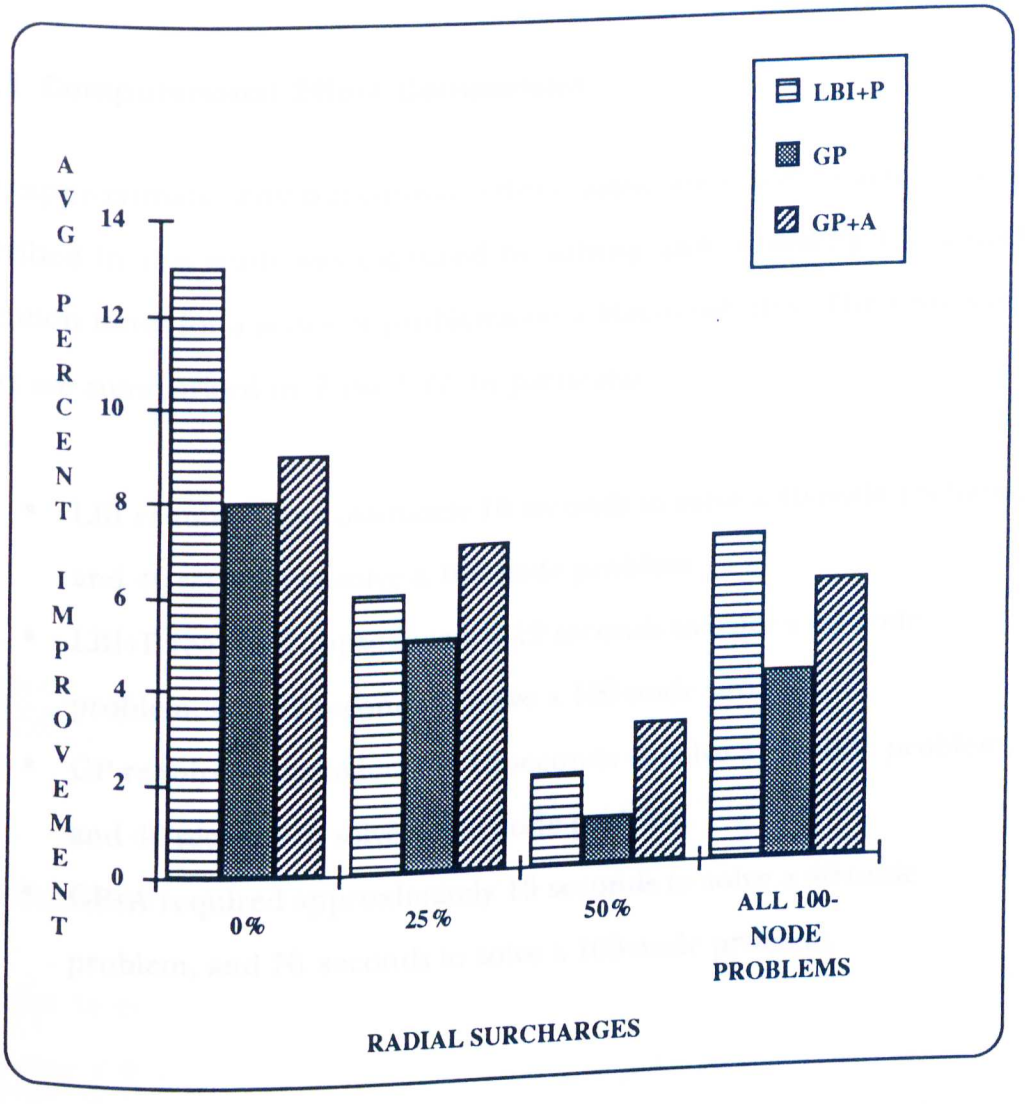


Figure 5.37 Avg. percent improvement over LBI (100-node problems and radial surcharge clustering).

LBI+P, GP, and GP+A outperform LBI as radial surcharges increase for 100-node problems.

5.4.3 Computational Effort Comparison

The approximate computational effort associated with each procedure described in this study was captured by solving and recording the wall-clock execution times for a series of problems on a Macintosh IIcx. The results of this effort are summarized in *Table 5.11*. In particular :

- LBI required approximately 10 seconds to solve a 40-node problem, and 46 seconds to solve a 100-node problem.
- LBI+P required approximately 12 seconds to solve a 40-node problem, and 47 seconds to solve a 100-node problem.
- GP required approximately 13 seconds to solve a 40-node problem, and 40 seconds to solve a 100-node problem.
- GP+A required approximately 10 seconds to solve a 40-node problem, and 76 seconds to solve a 100-node problem.

PROBLEM SIZE	LBI	LBI+P	GP	GP+A
40 NODES	10	12	13	19
50 NODES	15	16	15	23
60 NODES	21	21	21	37
70 NODES	26	26	21	35
80 NODES	34	34	27	45
90 NODES	36	37	31	53
100 NODES	46	47	40	76

(WALL-CLOCK EXECUTION TIME IN SECONDS MEASURED ON A MCINTOSH IIcx COMPUTER)

Table 5.11 Computational Effort Comparison.

5.5 SUMMARY OF COMPUTATIONAL EFFORT

In order to evaluate and compare the efficiency of VRPB procedures defined in *Chapter 4.0*, a series of computational experiments were performed using a Pascal compiler and a Macintosh platform. The computational experiments consisted of generating random problems for a number of possible combinations of critical VRPB parameters such as, scale, problem sizes, percentage of backhauls nodes, delivery node concentration, backhaul penalty, and radial surcharges.

The resulting 2700 problems were then solved by LBI, LBI+P, GP, and GP+A. The solutions generated by each procedure were then compared and the

following statistics were collected : number of better solutions better than LBI, percent improvement over LBI, and average solution length.

The main findings of our computational effort are listed below.

1) On average, LBI+P, GP, and GP+A outperform LBI by as much as 4% to 35% for all the critical parameters.

2) The effectiveness of the procedures which featured dedicated pickup routes, such as LBI+P, GP, and GP+A , was sensitive to the following VRPB conditions :

   a) As the number of nodes increased in a fixed size Euclidean space, the economic appeal of dedicated pickup routes decreased, and so did the effectiveness of the procedures which featured such options.

   b) As the number of backhaul nodes increased, the attractiveness of dedicated pickup routes increased, and so did the effectiveness of the procedures which featured such options. Furthermore, the computational data suggests that the converse is also true.

   c) As the delivery concentration region decreased the attractiveness of dedicated pickup routes increased, and so did the effectiveness of the procedures which featured such options. Furthermore, the computational data suggests that the converse is also true.

d) As radial surcharge values increased, backhauls are inserted into mixed routes early in LBI+P, thus reducing the number of backhauls available during the Post-Opt procedure and it's effectiveness.

3) The effectiveness of the procedures which featured dedicated pickup routes, such as LBI+P, GP, and GP+A , was not as sensitive to the following VRPB parameters :

a) The effectiveness of all the procedures was consistent across all backhaul penalty values as evidenced by the tight range of solution improvement percentages for penalty values.

b) With the exception of LBI+P, the effectiveness of the GVRPB procedures was consistent across all radial surcharge values as evidenced by the tight range of solution improvement percentages for penalty values.

4) The computational effort associated with each procedure was captured by solving and recording the wall-clock execution times for a series of problems on a Macintosh IIcx. The main finding of this experiment was that all procedures required at least 13 seconds to solve a 40-node and as many as 50 to 80 seconds to solve a 100-node problem. If one considers the complexity of the VRPB, and the processing platform used, even execution times of 80 seconds are very attractive. Future testing with a small number of larger problems should produce additional insight into improving the accuracy and the execution times of their solutions.

6.0 REAL-WORLD TEST CASE: ACE FIRE

In order to validate the solution procedures developed in *Chapter 4*, we used the data collected for the Ace Fire case study(see *Section 1.2)* and applied a number of modeling assumptions in order to develop solutions using some of the new procedures. The data collection process included meeting Ace Fire management to review their routing problem, reviewing the work-order activity for a typical week, the compilation of driver activity statistics, and the mapping of routing information into a computer-usable form (that is, a two-dimensional Euclidean space). It is important to note that the VRPB procedures developed in *Chapter 4* do not require Euclidean distances. Euclidean distances are used throughout this dissertation for the sake of convenience.

As described earlier in *Section 1.2,* Ace Fire has a fleet of trucks which delivers and services fire extinguishers in the greater Washington Metropolitan area. The fleet consists of ten trucks each of which is assigned to serve predetermined customer clusters. In a typical day, Ace Fire drivers average 120 customer visits. A customer visit schedule for Ace Fire for a typical day was used to test our the solution procedures. In order to facilitate the process, some simplifying assumptions were made. Some of the visits occurred within very concentrated zones within the metropolitan area. These sites were combined and the number of stops required within the zones was used as the pickup or delivery requirement. In this manner, the data was the reduced to the forty-three sites outlined in *Table 6.1.*

A graphical illustration of Ace Fire's service sites is presented in *Figure 6.1*. In addition, two sets of modeling assumptions were used to execute our procedures. These are shown in *Table 6.2*. Note that both test cases use a 500 by 500 Euclidean space as the routing region. This dimension was selected to fit a standard map of the Washington Metropolitan area. The number of nodes, their location, and their load is the same for both experiments (the process to develop this node informations was described earlier in this chapter). The percentage of backhauls was varied for each experiment to demonstrate the strengths of each procedure. The delivery concentration was estimated based on the location information, and the node classification (that is delivery or pickup). The backhaul and radial penalties were arbitrarily set at the same value for each experiment. The vehicle capacity was set at twenty, which happens to be the average number of stops an Ace Fire driver makes in a typical day.

The four procedures described in *Chapter 4* (LBI, LBI+P, GP, GP+A) were applied to the Ace Fire test problems under each of two sets of modeling assumptions to produce low-cost solutions. Note that the key difference between the set of assumptions is that Experiment 1 has more backhauls than Experiment 2. The results are summarized in *Table 6.3*. In the first experiment, with the smaller number of backhauls and a high degree of delivery concentration, LBI, and LBI+P (2,853 units for both) outperform GP, and GP+A (3,271, and 3,262 units, respectively). In the second experiment, with the greater number of backhauls and a high degree of delivery concentration, GP, and GP+A (2,789 units for both) outperformed LBI, and

LBI+P (2,935 units for both). The configurations for LBI and,GP are illustrated in *Figures 6.2.* through *6.5.*

The results for Ace Fire confirm our findings in *Chapter 5.* Mainly, that LBI performs better in cases where the delivery concentration region is large, and the number of backhauls is small. And, that GP performs better in cases where the delivery concentration region is small, and the number of backhauls is large.

As a final sensitivity test, the Ace Fire data was modified to study the impact of changes in vehicle capacity and pickup load requirements. First, the pickup loads were modified to be in the 5 to 15 unit range. Next, the parameters in experiments 1 and 2 were combined with vehicle capacities of 20, 25, 30, and 60 units to create eight new problems. These problems were solved by each of our VRPB procedures, and the results are sown in *Table 6.4.* For problems with a backhaul percentage of 20% increases in vehicle capacity resulted in lower distance solutions across all solution procedures. In addition, GP and GP+A produced better solutions than LBI and LBI+P. For problems with a backhaul percentage of 40%, increases in vehicle capacity resulted in longer distance solutions for the LBI and LBI+P procedures. In these cases, the bigger vehicle capacities allowed pickups with big loads to be inserted in routes where deliveries remained. The backhaul penalties associated with these insertions caused the higher objective values. For problems with a backhaul percentage of 40%, increases in vehicle capacity resulted in lower distance solutions for

the GP and GP+A procedures. In addition, GP and GP+A produced better solutions than LBI and LBI+P.

ACE FIRE SERVICE AREAS	COUNTY	No Of Stops Required
COLLEGE PARK,MD	MONTGOMERY	1
MONTGOMERY VILLAGE	MONTGOMERY	1
BROOKVILLE,MD	MONTGOMERY	1
GERMANTOWN,MD	MONTGOMERY	5
GAITHERSBURG,MD	MONTGOMERY	2
SYCAMORE ACRES	MONTGOMERY	1
SANDY SPRINGS	MONTGOMERY	1
SPENCERVILLE,MD	MONTGOMERY	1
ROCKVILLE,MD	MONTGOMERY	3
GARRET PARK,MD	MONTGOMERY	3
WHEATON,MD	MONTGOMERY	1
CALVERTON,MD	MONTGOMERY	1
CARDEROCK SPRINGS, MD	MONTGOMERY	3
BETHESDA,MD	MONTGOMERY	4
CHEVY CHASE,MD	MONTGOMERY	1
SILVER SPRING,MD	PRINCE GEORGES	1
LANGLEY PARK,MD	PRINCE GEORGES	3
GREENBELT,MD	PRINCE GEORGES	5
BLADENSBURG,MD	PRINCE GEORGES	4
NEW CARROLLTON,MD	PRINCE GEORGES	2
WOODMORE,MD	PRINCE GEORGES	1
BOWIE,MD	PRINCE GEORGES	2
SEAT PLEASENT,MD	PRINCE GEORGES	1
PATUXENT,MD	PRINCE GEORGES	1
TEMPLE HILLS,MD	PRINCE GEORGES	4
MORNINGSIDE,MD	PRINCE GEORGES	1
WESTPHALIA,MD	NORTHERN VA.	1
ARLINGTON,VA	NORTHERN VA.	5
CRYSTAL CITY,VA	NORTHERN VA.	2
BAILEYS CROSSROADS,VA	NORTHERN VA.	1
ANNANDALE,VA	NORTHERN VA.	1
VIENNA,VA	NORTHERN VA.	2
SPRINGFIELD,VA	NORTHERN VA.	2
ALEXANDRIA,VA	NORTHERN VA.	1
GROVETON,VA	NORTHERN VA.	1
WELLINGTON,VA	PRINCE WILLIAM	1
HAYMARKET,VA	PRINCE WILLIAM	2
MANASSAS,VA	PRINCE WILLIAM	1
WOODBRIDGE,VA	WASHINGTON,D.C.	7
NORTH EAST	WASHINGTON,D.C.	19
NORTH WEST	WASHINGTON,D.C.	15
SOUTH EAST	WASHINGTON,D.C.	4
SOUTH WEST		

Table 6.1 Ace Fire Locations

College Park, Md is the site for the Ace Fire depot, and the remaining 42 service areas are client sites.

CRITICAL MODELING ASSUMPTIONS	EXPERIMENT 1 VALUES	EXPERIMENT 2 VALUES
Network Scale	500 by 500	500 by 500
Number of Nodes	43	43
Percent of Backhauls	20%	40%
Estimated Delivery Concentration	40%	40%
Backhaul Penalty	100%	100%
Radial Surcharge	0%	0%
Vehicle Capacity	20 UNITS	20 UNITS

Table 6.2 Ace Fire Modeling Assumptions.

MODELING RESULTS	EXPERIMENT 1	EXPERIMENT 2
LBI	2853	2935
LBI+P	2853	2935
GP	3271	2789
GP+A	3262	2789

Table 6.3 Ace Fire Modeling Results.



Figure 6.1 Ace Fire Sites.

42 client locations and depot.



Figure 6.2 LBI solution for Ace fire.

Routes produced by LBI for first set of assumptions. Solution Length 2,853 units.



Figure 6.3 GP solution for Ace fire.

Routes produced by GP for first set of assumptions. Solution Length 3,271 units.



Figure 6.4 LBI solution for Ace fire.

Routes produced by LBI for second set of assumptions. Solution Length 2,935 units.



Figure 6.5 GP solution for Ace fire.

Routes produced by GP for second set of assumptions. Solution Length 2,789 units.

ACE FIRE SENSITIVITY ANALYSIS

CRITICAL ASSUMPTIONS
1. PICK-UP LOAD VARIES 5 TO 15 UNITS
2. VEHICLE CAPACITIES EVALUATED: 20 UNITS, 25 UNITS, 30 UNITS, AND 60 UNITS
3. RERUN EXPERIMENTS 1 AND 2 WITH ABOVE MODIFICATIONS

BACKHAUL PERCENTAGE	PROCEDURE	VEHICLE CAPACITY 20 UNITS	25 UNITS	30 UNITS	60 UNITS
20%	LBI	3,645	3,361	3,154	2,503
	LBI+P	3,645	3,361	3,154	2,503
	GP	3,438	3,295	3,122	2,933
	GP+A	3,438	3,295	3,122	2,933
40%	LBI	3,916	3,921	4,085	3,235
	LBI+P	3,485	3,921	4,085	3,235
	GP	3,357	3,297	3,097	2,553
	GP+A	3,357	3,297	3,097	2,553

Table 6.4 Ace Fire Sensitivity Analysis

The Ace Fire data has been modified to study the impact of changes in vehicle capacity and pickup load requirements. First, the pickup loads have been modified to be in the 5 to 15 unit range. Next, the parameters in experiments 1 and 2 have been combined with vehicle capacities of 20, 25, 30, and 60 units to create eight new problems. These problems were solved by each of our VRPB procedures. For problems with a backhaul percentage of 20% increases in vehicle capacity resulted in lower distance solutions across all solution procedures. In addition, GP and GP+A produced better solutions than LBI and LBI+P. For problems with a backhaul percentage of 40%, increases in vehicle capacity resulted in longer distance solutions for the LBI and LBI+P procedures. In these cases, the bigger vehicle capacities allowed pickups with big loads to be inserted in routes where deliveries remained. The backhaul penalties associated with these insertions caused the higher objective values. For problems with a backhaul percentage of 40%, increases in vehicle capacity resulted in lower distance solutions for the GP and GP+A procedures. In addition, GP and GP+A produced better solutions than LBI and LBI+P.

7.0 SUMMARY AND CONCLUSIONS

The problem studied in this dissertation concerns the routing of vehicles over a set of mixed customers. Some customers are delivery or linehaul points but others are pickup or backhaul points. In contrast to deliveries, when a vehicle services a pickup point, a load of product is accepted that is bound for the distribution center.

Given this as background, the vehicle routing problem with backhauls can be stated as follows: Find a set of vehicle routes that service the delivery and backhaul customers such that vehicle capacity is not violated and the total distance traveled is minimized.

In recent years, those responsible for physical distribution management have recognized the cost-cutting potential of servicing backhaul points as a part of all-delivery routes. A recent industry survey sponsored by the National Council of Physical Distribution Management reported that the number one program for improving productivity in logistics was the coordination of inbound with outbound freight distribution.

In this dissertation, we examine three diverse real-world routing applications with backhauls. Our review also focuses on several first-generation algorithms designed to solve VRPBs. These algorithms are limited :

- With respect to the ability to account for some critical real-world constraints;
- In that most procedures are too slow and cumbersome for microcomputer implementation;
- In that they solve the mixed route backhauling option (that is, the mixed delivery and backhaul route). The algorithms do not take into account the economic trade-offs that become available when common carrier services, supplier delivery services, or dedicated backhaul fleets are used.

With this in mind, the dissertation research objectives are three-fold :

- To develop new heuristics to solve the VRPB that redress the shortcomings of existing solution methods in dealing with real-world considerations,

- To enhance these new heuristics to produce integrated solutions that consist of complementary backhaul servicing options such as common carrier or supplier deliveries, dedicated backhaul fleets, and the most commonly modeled mixed route option, and

- To code the new heuristics on a microcomputer and then evaluate the accuracy, the performance, and the sensitivity of these new heuristics on a set of test problems.

Four heuristics were developed to achieve the research objectives :

1. A Load-Based Insertion Procedure (LBI) that consists of a two-stage process whereby the delivery and pickup nodes are initially placed in dedicated delivery routes and common carrier pickup routes, and then a load-based insertion process is used to cost-effectively create mixed routes.

2. A Load-Based Insertion Procedure plus Post-Processor Procedure (LBI+P) that post processes LBI by taking the common carrier tours that remain after the second stage, and creating cost-effective dedicated pickup tours with them.

3. A Generalized Procedure (GP) that consists of a multi-stage process whereby the economic merit of dedicated delivery and pickup routes, common carrier routes, and mixed routes is evaluated using a number of interchange operations during the backhaul insertion process.

4. A Generalized Procedure with an post processor ARCOPT (GP+A) that post processes GP by applying a traveling salesmen heuristic to the resulting dedicated routes produced by GP.

In order to evaluate and compare their performance, these heuristics were coded in Pascal and a series of computational experiments were performed on a Macintosh platform. The computational experiments consisted of generating random problems for a number of possible combinations of critical problem

parameters including problem scale, problem size, percentage of backhauls nodes, delivery node concentration, backhaul penalty, and radial surcharges.

Twenty seven hundred problems were solved by the four heuristics and the following summary statistics were collected : the number of solutions better than LBI, the percent improvement over LBI, and the average solution length.

The main findings of this empirical analysis are summarized below.

1. On average, LBI+P, GP, and GP+A outperformed LBI over all the critical parameter combinations.

2. The effectiveness of the procedures which featured dedicated pickup routes, such as LBI+P, GP, and GP+A , was sensitive to the following VRPB conditions :

   a. As the number of nodes increased in a fixed size Euclidean space, the economic appeal of dedicated pickup routes decreased, and so did the effectiveness of the procedures which featured such options.

   b. As the number of backhaul nodes increased, the attractiveness of dedicated pickup routes increased, and so did the effectiveness of the procedures which featured such options. Furthermore, the computational data suggests that the converse is also true.

c. As the delivery concentration region decreased the attractiveness of dedicated pickup routes increased, and so did the effectiveness of the procedures which featured such options. Furthermore, the computational data suggests that the converse is also true.

d. As radial surcharge values increased, backhauls are inserted into mixed routes early in LBI+P, thus reducing the number of backhauls available during the Post-Opt procedure and it's effectiveness.

3. The effectiveness of the procedures which featured dedicated pickup routes, such as LBI+P, GP, and GP+A , was not as sensitive to the following VRPB parameters :

a. The effectiveness of all the procedures was consistent across all backhaul penalty values as evidenced by the tight range of solution improvement percentages for penalty values.

b. With the exception of LBI+P, the effectiveness of the GVRPB procedures was consistent across all radial surcharge values as evidenced by the tight range of solution improvement percentages for penalty values.

4. The computational effort associated with each procedure was captured by solving and recording the wall-clock execution times for a series of problems on a Macintosh IIcx. The main finding of this experiment was that all procedures required at least 13 seconds to solve a 40-node and as many as 50 to 80 seconds to solve a 100-node problem. If one considers the

complexity of the VRPB, and the processing platform used, even execution times of 80 seconds are very attractive. Future testing with a small number of larger problems should produce additional insight into improving the accuracy and the execution times of their solutions.

As a final step in the validation the proposed network design procedures, the data collected for the Ace Fire case study was used in conjunction with a number of modeling assumptions to develop solutions using these procedures. The results of Ace Fire test case confirm our findings in *Chapter 5*. Mainly, that LBI performs better in cases where the delivery concentration region is large, and the number of backhauls is small. And, that GP performs better in cases where the delivery concentration region is small, and the number of backhauls is large.

In addition, this dissertation includes an empirical study of recently developed TSP heuristics which were used to develop the VRPB procedures which form the basis for our work(see *Appendix B*).

8.0 AREAS FOR FUTURE RESEARCH

A considerable challenge for future study is to extend the GVRPB procedures to solve large problems (that is, 200-node to 500-node problems). The main obstacle in this endeavor is the memory requirements associated with defining such problems. In our case, the memory constraints were attributed to the hardware and software environment, we selected. Note that the benefits associated with our microcomputer processing environment out weighed the costs. Especially in terms of the availability of real-time graphs capabilities which were critical to model development and testing. Although, some progress was made in this study with data compression techniques (especially with the ARCOPT routines which are capable of solving up 240-node problems). Further progress is necessary for the GVRPB procedures. Once the computational constraints are overcome, it will be possible to refine these procedures further, as we learn more about how large GVRPB behave under a variety of critical parameter scenarios. The Ace Fire data can provide a good basis for additional experimentation, as well. Further, refinement of these data will be necessary in order to contrast Ace's current routing strategy with those generated by the procedures developed in this study.

APPENDIX A. LOAD-BASED INSERTION (LBI) HIGH QUALITY SOLUTIONS

To illustrate the application of the high-quality rules specified in *Section 4.1* to candidate solutions, consider the configuration shown in *Figure 1*. This set of routes, known as Routes Configuration 1 (RC1), was obtained by inserting the backhauls onto the Clarke-Wright routes (shown in *Section 4.1, Figure 4.2*) using the load-based insertion procedure.

The total travel distance for these routes is 672, but one route violates Rule 2. On this route, 85% of the vehicle's capacity is used after accepting the backhaul.

We should also point out that, in *Figure 1*, the second and third routes are combined (by inspection) to provide the second driver with an average-duration workday.

In the initial solution phase, where the Clarke-Wright procedure generates routes that service a large number of delivery locations, it sometimes happens that the routes offer little flexibility for insertion of backhauls. For example, with only a few
long distance routes composed of many deliveries, there are very few low-cost places to insert backhauls; the backhauls are usually squeezed onto the end of the route.

To facilitate the insertion of backhauls, we decided to initiate the Clarke-Wright procedure with an artificial vehicle capacity that is smaller than the actual vehicle capacity. For example, vehicles might have an actual capacity of 160 units; however, in order to produce an initial solution that is composed of more routes with fewer deliveries per route, the capacity of the vehicles might be set at 120 units during the Clarke-Wright initial solution phase. More routes with fewer deliveries per route usually result in unused vehicle capacity, and backhauls can often be inserted more freely than in tightly constrained situations. In addition, since there are more routes, the number of backhauls per route decreases which, again reduces the complexity and interdependence of the insertions.

The effect of initiating the Clarke-Wright procedure with a reduced vehicle capacity is illustrated in *Figures 2, 3,* and *4* and summarized in *Table 1.* The initial vehicle capacity is chosen to be smaller than the actual capacity of 160. Once backhauls have been considered for insertion, the capacity of the vehicle "returns" to 160.

In *Table 1,* four route configurations (RC1 through RC4) are presented as candidate solutions to the problem illustrated in *Section 4.1, Figure 4.1.* Each of these solutions was generated from the load-based insertion procedure which consists of three phases—initial solution phase, insertion phase, and feasibility phase.

Configurations RC2 and RC3 were developed using a vehicle capacity of 120 during the initial phase. This resulted in an initial solution with more routes than in RC1 and a total distance of 1067 miles including radial distances and surcharges. During the insertion phase, backhaul penalties of 25 and 30 were used to generate the configurations shown in *Figures 2* and *3,* respectively. The RC2 configuration consists of eight routes with a total distance of 684 miles. Five drivers are expected to be on the road from 6.1 to 7.3 hours each to serve these routes. The RC3 configuration has a total distance of 686 miles after insertions, the same number of routes, a similar driver work schedule, and it also satisfies the specified rules.

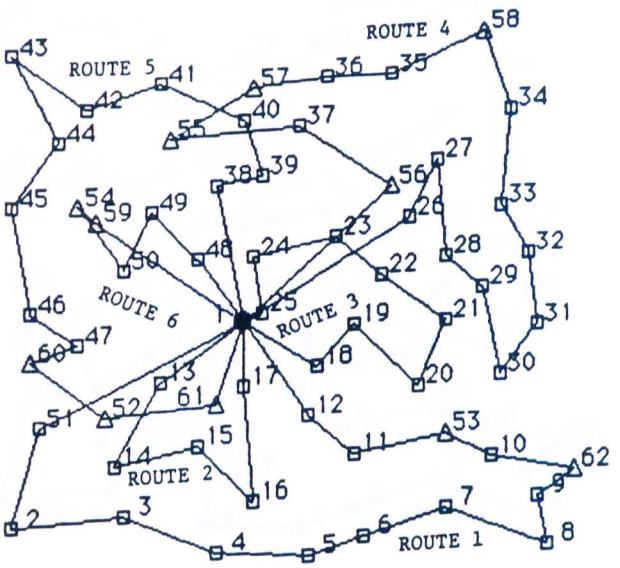
Configuration RC4 was developed using a vehicle capacity of 140 during the initial phase. This results in an initial solution with a total distance of 1066 miles including radial distances and surcharges. The insertion procedure yielded a configuration (see *Figure 4)* with six routes and total distance of 682 miles. Five drivers are expected to be on the road from 5.8 to 7.7 hours each, and all of the rules are fully satisfied.

A comparison of these four configurations leads us to favor the set of routes represented by either RC2 or RC4 for several reasons. First, the two configurations are cost-effective in the sense that the total distance traveled is nearly minimal. Second, the driver loads are balanced. Each driver does about the same amount of work. Third, the rules are fully satisfied.

Table 1. Summary of Computational Results with Load-based insertions.

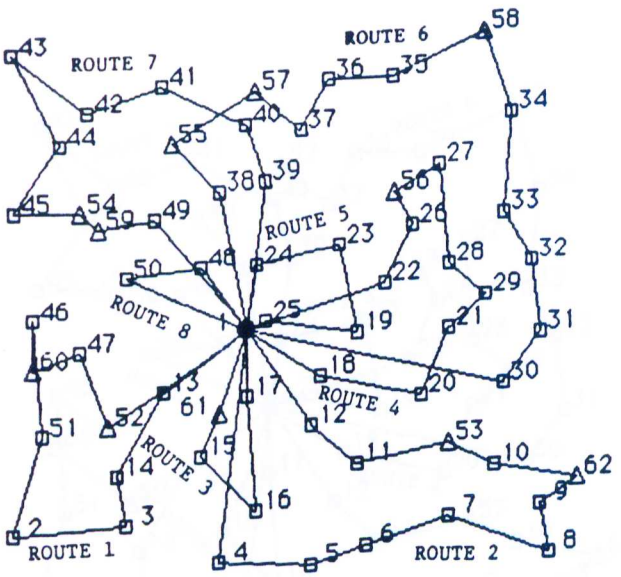
	Configurations			
	RC1	RC2	RC3	RC4
Initial Solution Phase				
Initial Vehicle Capacity	160	120	120	140
Clarke-Wright Distance	585	641	641	640
Radial Distance	284	284	284	284
Radial Surcharge (0.5 x radial distance)	142	142	142	142
Total Distance	1011	1067	1067	1066
Insertion Phase				
Backhaul Penalty	30	25	30	20
Number of Routes	6	8	8	6
Total Distance After Insertion	672	684	686	682
Average Hours per Driver	6.74	6.82	6.88	6.78
Range in Hours per Driver	6.7	1.2	1.3	1.9
Feasibility Phase				
Rule 1 Conformity	Yes	Yes	Yes	Yes
Rule 2 Conformity	No*	Yes	Yes	Yes
Rule 3 Conformity	Yes	Yes	Yes	Yes

* missed by 5%



ROUTE	DRIVER	HRS/DRIVER
1	1	7.7
2	2	6.6
3	2	-
4	3	9.6
5	4	6.9
6	5	2.9
	AVG 6.74 HRS/DRIVER	

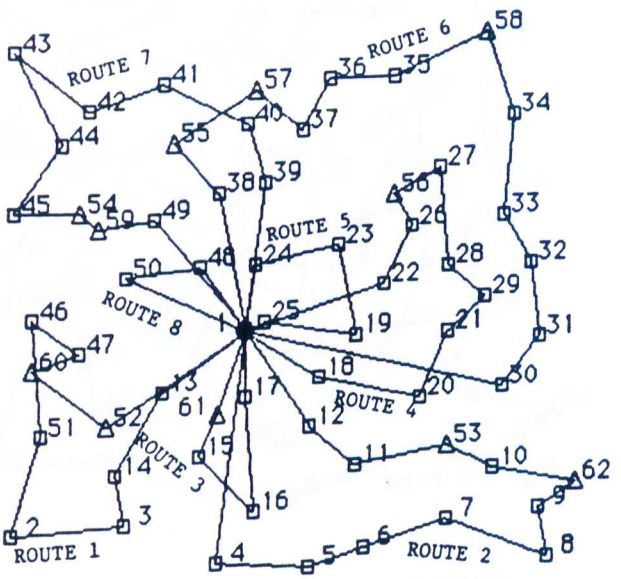
Figure 1. Routes Configuration 1 (RC1).



ROUTE	DRIVER	HRS/DRIVER
1	1	7.3
2	2	6.1
3	3	6.7
4	3	-
5	1	-
6	4	7.1
7	5	7.1
8	5	-

Avg 6.82 HRS/DRIVER

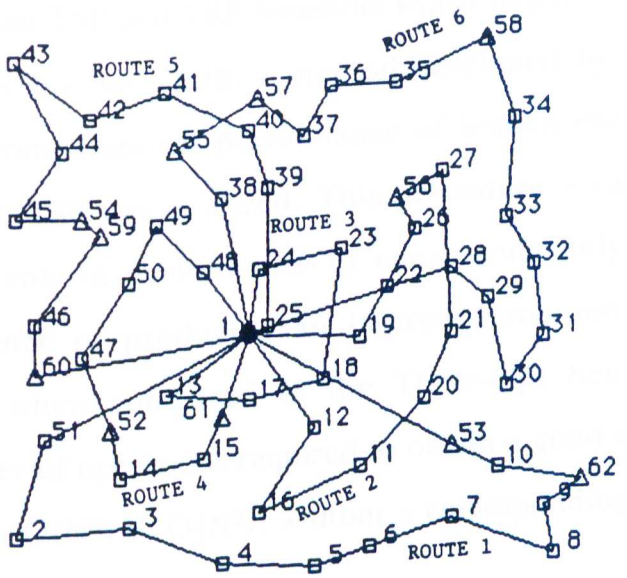
Figure 2. Routes Configuration 2 (RC2).



ROUTE	DRIVER	HRS/DRIVER
1	1	7.4
2	2	6.1
3	3	6.7
4	3	-
5	1	-
6	4	7.1
7	5	7.1
8	5	-

AVG 6.88 HRS/DRIVER

Figure 3. Routes Configuration 3 (RC3).



ROUTE	DRIVER	HRS/DRIVER
1	1	7.2
2	2	5.8
3	3	6.7
4	3	-
5	4	6.5
6	5	7.7

AVG 6.78 HRS/DRIVER

Figure 4. Routes Configuration 4 (RC4).

APPENDIX B ARCOPT EMPIRICAL ANALYSIS

1. Introduction

In the search for fast TSP and VRP heuristics which might be used as part of a solution algorithm for the VRPB, a method developed by Stewart [16] for accelerating the computational performance of branch exchange heuristics for the symmetric TSP was analyzed. This procedure -- called ARCOPT -- derives improvements in performance by considering only exchanges that have a good chance of producing an improved solution. Stewart shows empirically that, when compared to the Three-Opt heuristic, ARCOPT reduces the number of operations required to obtain a good solution to a TSP with N nodes from $O(N^3)$ to $O(N^2)$, without a corresponding degradation in the quality of the solution.

A series of computational experiments was performed to compare the efficiency of ARCOPT to other well-known TSP heuristics.

2. Definitions

In this section, we define the pre-processors, the TSP heuristics , and the post-processor used during our analysis.

2.1 Pre-processors

Our pre-processors generated three type of starting solutions:

- Random Starting Solution (RSS)
- Nearest Neighbor Starting Solution (NNSS)
- Space-filling Starting Solution (SCSS).

2.2 TSP Heuristics

Two-Opt procedure[18]

This branch-exchange procedure uses the full distance matrix and examines the potential cost savings of replacing any two arcs in the current feasible solution with two arcs not in the solution set. This procedure is repeated until no improvement can be found by performing such an exchange. The Two-Opt procedure has been modified to improve its speed and accuracy. Only a subset of possible exchanges of two arcs are examined before an exchange is selected and a new route is developed. This approach reduces the number of iterations required to reach two-optimality.

Three-Opt procedure[18]

This branch-exchange procedure uses the full distance matrix and examines the potential cost savings of replacing any three arcs in the current feasible solution with three arcs not in the solution set. This procedure is repeated until no improvement can be found by executing such an exchange.

ARCOPT procedure[16]

Stewart recommends reducing the number of arcs to be considered in a branch-exchange procedure. Specifically, he suggests using only the arcs found in a set of K mutually exclusive minimal spanning trees (on the network under under analysis) during the branch-exchange operations. Based on this premise, he has developed a very efficient Three-Opt heuristic (ARCOPT) to solve the TSP problem quickly.

2.3 Post-processor

Annealing Process[11]

Any of the heuristics defined in *Section 2.2* can be adapted to accept a number of non-improving exchanges[11]. This ascent procedure is then followed by another application of the descent heuristic. This sequence is repeated several times during a preset time interval. This approach is intended to force the algorithm away from certain local minima and hopefully to find other more promising directions of descent.

3.1 Modeling procedures

In this analysis, the following modeling procedures were implemented.

1. The ARCOPT heuristic using a random starting solution (**RSS+A**).
2. The ARCOPT heuristic using a nearest neighbor starting solution (**NNSS+A**).
3. The ARCOPT heuristic using a space-filling curve starting solution (**SCSS+A**).
4. The Two-Opt heuristic using a random starting solution (**RSS+TWO**).
5. The Two-Opt heuristic using an Annealing post-processor (**TWO+An**).
6. The Three-Opt heuristic using a random starting solution (**THREE-OPT**).
7. The ARCOPT heuristic using a random starting solution, and an estimated value of **K**, (**RSS+K+A**).
8. The ARCOPT heuristic using a random starting solution, an estimated value of **K**, and an Annealing post-processor (**RSS+K+A+An**).

4. EXPERIMENTAL DESIGN AND PROCESSING ENVIRONMENT

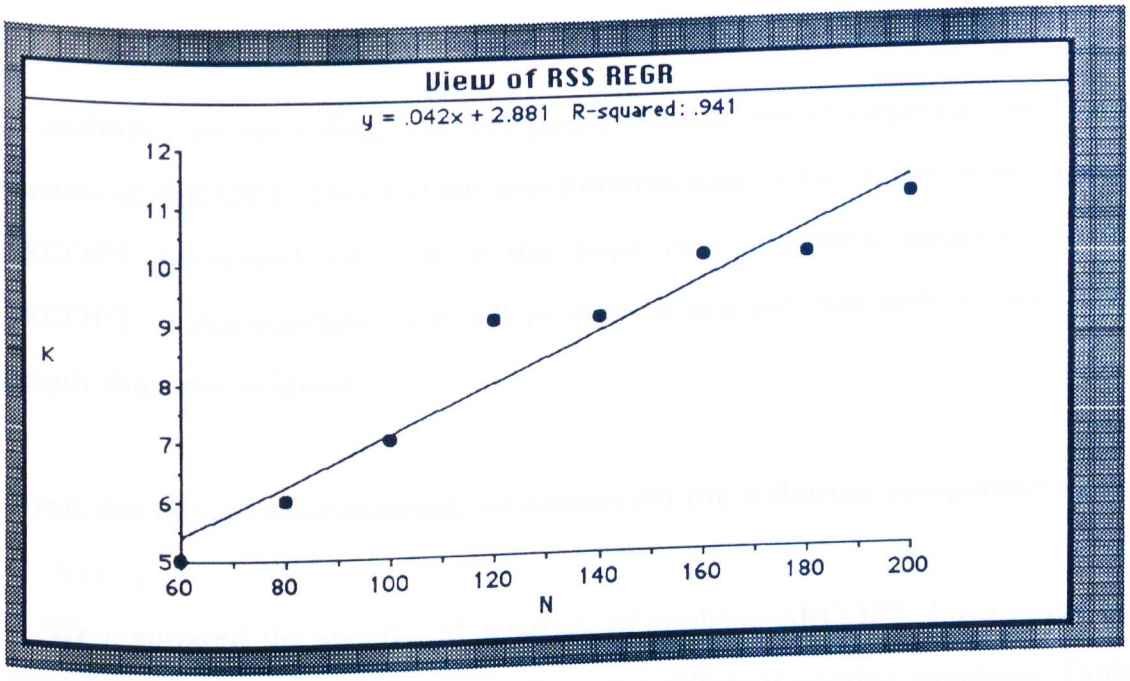
Eighty test problems were randomly generated in Euclidean space. Problem sizes included 60, 80, 100, 120, 140, 160, 180, and 200-node problems. Ten samples were taken for each network size yielding a total of 80 test problems. These problems were solved by each of the eight procedures described earlier. This gives a total of 640 solutions. Execution times were also collected.

Computational testing of the heuristics was done on a Macintosh Plus with 1 Meg of RAM. This base unit is supplemented with a 16MHZ Processor Card (RADIUS Accelerator 16) and a math co-processor (4 X MacPlus). All computer codes were written in Pascal using Turbo Pascal (compiler Version 1.1 by Borland). Storage compression techniques were required to extend user data limits.

5. Computational results and analysis

During the implementation of ARCOPT, we determined that the parameter K was critical to solution quality and operational performance. K is defined as the number of mutually exclusive minimal spanning trees that are used in determining the arcs to be considered during the branch-exchange process. Given the sensitivity of this parameter to ARCOPT's performance, and the lack of firm guidelines provided by Stewart [16] for determining this parameter, a regression model was developed to forecast the value of K which, on average, produces high quality solutions given the number of nodes in the

network. The regression model for K is shown in *Figure 7*. In this model the independent variable, N, is the number of nodes in a TSP. The dependent value , K, is the number of mutually exclusive MSTs required by ARCOPT to get a good TSP solution for problem of size N. The desired value of K was obtained for each problem in the test set (keep in mind that sets of 60, 80, 100, 120, 140, 160, 180, and 200-node problems were used) by solving the TSP using ARCOPT for a range of values for K, and selecting the value of K which yielded the best solution. The resulting ten values of K were then averaged to produce the desired value of K for that problems size set. This process was repeated for each problem set. Finally, the desired values of K for each problem size category were paired with the appropriate TSP problem size to produce the regression data sets. *Figure 7* shows that a linear relation accurately(R-squared of 94%) describes the interaction between N and K for these data set.



View of RSS REGR

Simple - Y : K X : N

DF:	R-squared:	Std. Err.:	Coef. Var.:
7	.941	.558	6.664

Beta Coefficient Table

Parameter:	Value:	Std. Err.:	Variance:	T-Value:
INTERCEPT	2.881	.594	.352	4.854
SLOPE	.042	.004	.00001854	9.815

Analysis of Variance Table

Source	DF:	Sum Squares:	Mean Square:	F-test:
REGRESSION	1	30.006	30.006	96.325
RESIDUAL	6	1.869	.312	$.0001 < p \leq .005$
TOTAL	7	31.875		

Residual Information Table

SS[e(i)-e(i-1)]:	$e \geq 0$:	$e < 0$:	DW test:
2.858	3	5	1.529

Figure 7. Simple Regression Model for K as Function of N

In addition, an annealing-type[11] post-processor was incorporated into our version of ARCOPT. This feature re-introduces some unattractive links into an ARCOPT generated solution in the hope that additional applications of ARCOPT to this configuration will produce a new solution with a lower TSP length than the original.

Given the above enhancements, we conducted the following comparisons.

1) We compared the speed and accuracy of applying ARCOPT, for a range of K values, to our test problems while using different starting solutions. *Table 1* shows that **RSS+A** is more accurate than **NNSS+A** and **SCSS+A** by 1% to 2%. Moreover, **RSS+A** is more time consuming than it's counterparts by as much as 21%.

2) We compared the speed and accuracy of applying **TWO+An** and **RSS+TWO** to our test problems. *Table 2* shows that **TWO+An** is more accurate than **RSS+TWO** by 3%. Moreover, **TWO+An** is slower than it's counterpart by 1%.

3) We compared the speed and accuracy of applying **TWO+An**, **RSS+A**, and **THREE-OPT** to our test problems. *Table 3* shows that **TWO+An** and **RSS+A** are both faster (75% to 84%) and more accurate (1% to 2%) than **THREE-OPT**.

4) We compared the speed and accuracy of applying **RSS+A**, **RSS+K+A**, and **RSS+K+A+An** to our test problems. *Table 4* shows that **RSS+K+A** and **RSS+K+A+An** are both faster than **RRS+A**(89% to 92%). Moreover, **RSS+K+A** and **RSS+K+A+An** are both less accurate than **RRS+A**(1% to 5%).

5) We compared the speed and accuracy of applying **TWO+An**, **RSS+K+A**, and **RSS+K+A+An** to our test problems. *Table 5* shows that **RSS+K+A** and **RSS+K+A+An** are both faster than **TWO+An**(87% to 90%). Moreover, **RSS+K+A** and **RSS+K+A+An** are both less accurate than **TWO+An**(1% to 4%).

Given the above results, it is clear that the ARCOPT variation which includes the random starting solution, the K forecasting feature, and the Annealing post-processor (**RSS+K+A+An**) significantly outperforms most classical TSP heuristics, as well as enhances the original ARCOPT procedure.

AVERAGE SOLUTION LENGTHS

NODES	RSS+A	NNSS+A	SCSS+A
60	1086	1091	1093
80	1277	1288	1278
100	1413	1423	1471
120	1566	1569	1549
140	1663	1718	1673
160	1760	1846	1782
180	1852	1882	1872
200	1977	2065	2006
AVERAGE	1574	1610	1591
PERCENT CHANGE		2%	1%

AVERAGE EXECUTION TIMES (IN WALL-CLOCK SECONDS)

NODES	RSS+A	NNSS+A	SCSS+A
60	408	323	355
80	703	560	597
100	1096	861	924
120	1584	1242	1323
140	2172	1720	1818
160	2886	2277	2408
180	3692	2954	3080
200	4679	3734	3895
AVERAGE	2153	1709	1800
PERCENT CHANGE		-21%	-16%

Table 1. ARCOPT Comparison

Table 1 summarizes the average solution lengths and execution times for **ARCOPT** with different starting procedures (**RSS, NNSS, and SCSS**). For each problem size a set of ten random problems were generated and their solutions and execution times averaged. The results show that for a range of **K** values **RSS+A** is more accurate than **NNSS+A**, and **SCSS+A** by 1% to 2%. However, **RSS+A** is more time consuming than it's counterparts by as much as 21%.

AVERAGE SOLUTION LENGTHS

NODES	RSS+TWO	TWO+An
60	1115	1097
80	1317	1285
100	1462	1423
120	1597	1561
140	1740	1682
160	1827	1766
180	1946	1872
200	2075	2008
AVERAGE	1635	1587
PERCENT CHANGE		-3%

AVERAGE EXECUTION TIMES (IN WALL-CLOCK SECONDS)

NODES	RSS+TWO	TWO+An
60	304	301
80	607	603
100	916	908
120	1319	1311
140	1833	1814
160	2436	2416
180	3197	3144
200	3937	3882
AVERAGE	1819	1797
PERCENT CHANGE		-1%

Table 2. Two-opt comparison.

Table 2 compares the average solution lengths and execution times of two-opt with random starting solutions (**RSS+TWO**) and two-opt with an annealing post-processor (**TWO+An**). For each problem size a set of ten random problems were generated and their solutions and execution times averaged. The results show that **TWO+An** is more accurate than **RSS+TWO** by 3%. However, **TWO+An** is slower than it's counterpart by 1%.

AVERAGE SOLUTION LENGTHS

NODES	THREE-OPT	RSS+A	TWO+An
60	1116	1086	1097
80	1309	1277	1285
100	1450	1413	1423
120	1573	1566	1561
140	1694	1663	1682
160	1791	1760	1766
180	N/A	1852	1872
200	N/A	1977	2008
AVERAGE	1489	1461	1469
PERCENT CHANGE		-2%	-1%

AVERAGE EXECUTION TIMES (IN WALL-CLOCK SECONDS)

NODES	THREE-OPT	RSS+A	TWO+An
60	284	408	301
80	898	703	603
100	2349	1096	908
120	5053	1584	1311
140	9616	2172	1814
160	16541	2886	2416
180	N/A	3692	3144
200	N/A	4679	3882
AVERAGE	5790	1475	919
PERCENT CHANGE		-75%	-84%

FOR PROBLEM SET 180, AND 200, THREE-OPT RAN OUT OF TIME. CONSEQUENTLY DATA WAS NOT INCLUDED IN OUR COMPARISON.

Table 3. TSP heuristics comparison.

Table 3 compares the average solution lengths and execution times of three-opt (**THREE-OPT**), **ARCOPT** with random starting solutions (**RSS+A**), and two-opt with an annealing post-processor (**TWO+An**). For each problem size a set of ten random problems were generated and their solutions and execution times averaged. The results show that **TWO+An**, and **RSS+A** are both faster (75% to 84%) and more accurate (1% to 2%) than **THREE-OPT**.

AVERAGE SOLUTION LENGTHS			
NODES	RSS+A	RSS+K+A	RSS+K+A+An
60	1086	1157	1094
80	1277	1352	1289
100	1413	1483	1433
120	1566	1662	1578
140	1663	1752	1716
160	1760	1832	1803
180	1852	1940	1879
200	1977	2055	1990
AVERAGE	1574	1654	1598
PERCENT CHANGE		5%	1%

AVERAGE EXECUTION TIMES (IN WALL-CLOCK SECONDS)			
NODES	RSS+A	RSS+K+A	RSS+K+A+An
60	408	16	43
80	703	33	72
100	1096	62	108
120	1584	108	157
140	2172	168	216
160	2886	256	332
180	3692	329	405
200	4679	474	572
AVERAGE	2153	181	238
PERCENT CHANGE		-92%	-89%

Table 4. ARCOPT refinements comparisons.

Table 4 compares the average solution lengths and execution times of **ARCOPT** with random starting solutions (**RSS+A**), **ARCOPT** with random starting solutions and an estimated value for **K** (**RSS+K+A**), and **ARCOPT** with random starting solutions, an estimated value for K, and an annealing post-processor (**RSS+K+A+An**). For each problem size a set of ten random problems were generated and their solutions and execution times averaged. The results show that **RSS+K+A**, and **RSS+K+A+An** are both faster than **RSS+A** (89% to 92%). Moreover, **RSS+K+A**, and **RSS+K+A+An** are only slightly less accurate (1% to 5%) than **RSS+A**.

AVERAGE SOLUTION LENGTHS

NODES	TWO+An	RSS+K+A	RSS+K+A+An
60	1097	1157	1094
80	1285	1352	1289
100	1423	1483	1433
120	1561	1662	1578
140	1682	1752	1716
160	1766	1832	1803
180	1872	1940	1879
200	2008	2055	1990
AVERAGE	1587	1654	1598
PERCENT CHANGE		4%	1%

AVERAGE EXECUTION TIMES (IN WALL-CLOCK SECONDS)

NODES	TWO+An	RSS+K+A	RSS+K+A+An
60	301	16	43
80	603	33	72
100	908	62	108
120	1311	108	157
140	1814	168	216
160	2416	256	332
180	3144	329	405
200	3882	474	572
AVERAGE	1797	181	238
PERCENT CHANGE		-90%	-87%

Table 5. Final Two-Opt and ARCOPT comparison.

Table 5 compares the average solution lengths and execution times of two-opt with an annealing post-processor (**TWO+An**), **ARCOPT** with random starting solutions and an estimated value for **K** (**RSS+K+A**), and **ARCOPT** with random starting solutions, an estimated value for K, and an annealing post-processor (**RSS+K+A+An**). For each problem size a set of ten random problems were generated and their solutions and execution times averaged. The results show that **RSS+K+A**, and **RSS+K+A+An** are both faster than **TWO+An** (87% to 90%). Moreover, **RSS+K+A**, and **RSS+K+A+An** are only slightly less accurate (1% to 4%) than **TWO+An**.

7. Conclusions

In this appendix, the performance characteristics of ARCOPT were examined. We studied the sensitivity of ARCOPT's performance to the parameter K, the number of mutually exclusive MSTs required to generate the arcs available for the branch-exchange procedure. A regression model was developed to forecast the value of K which, on average, produces high quality solutions given the number of nodes in the network. The R-squared for the regression model is in the 90% range. Using forecasted values of K for ARCOPT, TSP solutions were produced of quality comparable to solving the ARCOPT procedure for a range of K, but at a fraction of the computational effort.

In addition, an Annealing post-processor was added to the ARCOPT procedure to yield improved solutions in exchange for a negligible amount of additional processing effort.

APPENDIX C Computational Results
(Note that concentration percentages 60%, 70%, and 80% in this printout should be 4%, 16%, and 40%, respectively)

NO. OF NODES	% OF BACKHAULS	% OF CONCENTR	BKHAUL PENALTY	RADIAL SURCHARGE	PROBL NO.	LOAD BASED SOL	G(A) SOL	G(B) SOL	G(B)/ARC SOL
40	20	60	100	0	1	1108	1033	1001	992
40	20	60	100	0	2	968	913	874	866
40	20	60	100	0	3	1075	1011	972	971
40	20	60	100	0	4	1055	1006	974	964
40	20	60	100	0	5	908	908	866	866
40	20	60	100	25	1	911	911	786	786
40	20	60	100	25	2	1172	1172	891	884
40	20	60	100	25	3	1255	1255	1030	1027
40	20	60	100	25	4	1278	1278	1125	1125
40	20	60	100	25	5	1120	1120	953	795
40	20	60	100	50	1	1311	1311	1123	1123
40	20	60	100	50	2	1290	1290	912	912
40	20	60	100	50	3	1611	1438	994	972
40	20	60	100	50	4	1360	1360	970	962
40	20	60	100	50	5	1284	1284	912	614
40	20	60	200	0	1	1090	1090	1064	1064
40	20	60	200	0	2	1073	1073	1048	794
40	20	60	200	0	3	909	909	901	901
40	20	60	200	0	4	1128	1128	1110	1110
40	20	60	200	0	5	920	920	871	864
40	20	60	200	25	1	1273	1273	1077	1070
40	20	60	200	25	2	1486	1486	1161	1154
40	20	60	200	25	3	1154	1154	1080	1074
40	20	60	200	25	4	946	946	773	772
40	20	60	200	25	5	1477	1477	1203	713
40	20	60	200	50	1	1417	1417	1325	1325
40	20	60	200	50	2	1469	1469	1035	1024
40	20	60	200	50	3	1210	1210	867	862
40	20	60	200	50	4	1299	1210	924	911
40	20	60	200	50	5	1398	1396	1020	1017
40	20	60	300	0	1	1018	997	851	848
40	20	60	300	0	2	1154	1061	959	953
40	20	60	300	0	3	977	977	908	697
40	20	60	300	0	4	1119	1029	863	859
40	20	60	300	0	5	1053	960	960	954
40	20	60	300	25	1	1193	1193	1080	1078
40	20	60	300	25	2	1199	1064	871	865
40	20	60	300	25	3	1133	1120	941	937
40	20	60	300	25	4	1148	1148	955	920
40	20	60	300	25	5	1200	1200	1103	1103
40	20	60	300	50	1	1397	1347	1032	957
40	20	60	300	50	2	1544	1370	857	853
40	20	60	300	50	3	1514	1514	1423	1423
40	20	60	300	50	4	1210	1177	873	808
40	20	60	300	50	5	1482	1482	1427	1427
40	20	60	400	0	1	1049	1049	1049	1049
40	20	60	400	0	2	1112	1108	1018	1018
40	20	60	400	0	3	1054	1028	937	921
40	20	60	400	0	4	1129	1067	1035	1028
40	20	60	400	0	5	1110	905	803	796
40	20	60	400	25	1	1175	1168	964	956
40	20	60	400	25	2	1413	1403	1100	855
40	20	60	400	25	3	1323	1323	1118	991
40	20	60	400	25	4	1404	1372	1114	1105
40	20	60	400	25	5	1364	1364	1239	1234
40	20	60	400	50	1	1471	1471	1463	1463
40	20	60	400	50	2	1447	1447	1428	1428
40	20	60	400	50	3	1323	1238	850	842
40	20	60	400	50	4	1229	1229	1144	1144
40	20	60	400	50	5	1674	1487	1031	1031
40	20	60	500	0	1	1022	1022	972	969
40	20	60	500	0	2	1133	1133	1103	1103
40	20	60	500	0	3	1055	1055	1022	1012

40	20	60	500	0	4	1086	1086	999	993
40	20	60	500	0	5	811	811	769	769
40	20	60	500	25	1	1315	1315	1292	1292
40	20	60	500	25	2	1256	1169	944	791
40	20	60	500	25	3	1336	1320	1022	1007
40	20	60	500	25	4	1407	1353	1042	1031
40	20	60	500	25	5	1235	1235	1076	882
40	20	60	500	50	1	1483	1420	942	927
40	20	60	500	50	2	1487	1308	847	842
40	20	60	500	50	3	1475	1475	968	784
40	20	60	500	50	4	1014	1014	905	905
40	20	60	500	50	5	1451	1443	1151	1132
40	20	60	500	50	1	1772	1518	1466	1461
40	20	70	100	0	2	1395	1310	1137	1133
40	20	70	100	0	3	1579	1579	1329	1329
40	20	70	100	0	4	1275	1275	1207	1201
40	20	70	100	0	5	1537	1537	1397	1334
40	20	70	100	0	1	1580	1580	1416	1416
40	20	70	100	25	2	1673	1626	1422	1421
40	20	70	100	25	3	1831	1654	1266	1266
40	20	70	100	25	4	1406	1314	1092	1092
40	20	70	100	25	5	1544	1544	1295	1295
40	20	70	100	25	1	1819	1819	1314	1311
40	20	70	100	50	2	2207	2207	1492	1492
40	20	70	100	50	3	1920	1743	1192	1189
40	20	70	100	50	4	1879	1549	1145	1136
40	20	70	100	50	5	1716	1716	1292	1292
40	20	70	100	50	1	1305	1305	1286	1286
40	20	70	200	0	2	1271	1232	1111	1111
40	20	70	200	0	3	1396	1343	1267	1259
40	20	70	200	0	4	1505	1334	1298	1287
40	20	70	200	0	5	1279	1223	1124	1094
40	20	70	200	0	1	1859	1679	1446	1420
40	20	70	200	25	2	1452	1269	1022	1022
40	20	70	200	25	3	1695	1520	1209	1207
40	20	70	200	25	4	1620	1582	1294	1117
40	20	70	200	25	5	1992	1992	1436	1299
40	20	70	200	25	1	1720	1720	1276	1276
40	20	70	200	50	2	1901	1901	1331	1000
40	20	70	200	50	3	2025	2025	1483	1483
40	20	70	200	50	4	1604	1567	1164	1149
40	20	70	200	50	5	2011	1876	1250	946
40	20	70	200	50	1	1378	1378	1286	1283
40	20	70	300	0	2	1408	1408	1348	1344
40	20	70	300	0	3	1374	1258	1219	1219
40	20	70	300	0	4	1563	1553	1475	1469
40	20	70	300	0	5	1399	1310	1247	1247
40	20	70	300	0	1	1325	1262	1009	1009
40	20	70	300	25	2	1509	1509	1303	1301
40	20	70	300	25	3	1732	1518	1171	1163
40	20	70	300	25	4	2001	1939	1596	1594
40	20	70	300	25	5	1498	1421	1181	1168
40	20	70	300	25	1	2324	2281	1369	1009
40	20	70	300	50	2	1628	1628	1239	1239
40	20	70	300	50	3	1443	1443	1151	1151
40	20	70	300	50	4	2006	1979	1312	972
40	20	70	300	50	5	1768	1763	1369	1369
40	20	70	300	50	1	1464	1464	1359	1128
40	20	70	400	0	2	1478	1457	1345	1344
40	20	70	400	0	3	1466	1435	1215	1211
40	20	70	400	0	4	1385	1385	1341	1341
40	20	70	400	0	5	1608	1533	1396	1165
40	20	70	400	0	1	1426	1426	1143	1143
40	20	70	400	25	2	1478	1478	1179	1160
40	20	70	400	25	3	1962	1630	1337	1330
40	20	70	400	25					

40	20	70	400	25	4	1591	1528	1222	1127
40	20	70	400	25	5	1786	1624	1391	1375
40	20	70	400	50	1	1704	1586	1148	1145
40	20	70	400	50	2	1717	1717	1261	916
40	20	70	400	50	3	1971	1971	1286	892
40	20	70	400	50	4	1743	1740	1449	1448
40	20	70	400	50	5	1808	1808	1440	1440
40	20	70	500	0	1	1374	1374	1319	1319
40	20	70	500	0	2	1414	1292	1291	1284
40	20	70	500	0	3	1756	1405	1234	1218
40	20	70	500	0	4	1253	1095	1091	1083
40	20	70	500	0	5	1421	1421	1416	1416
40	20	70	500	25	1	1835	1586	1232	1209
40	20	70	500	25	2	1640	1640	1305	1121
40	20	70	500	25	3	1319	1300	1107	1106
40	20	70	500	25	4	1673	1673	1385	1352
40	20	70	500	25	5	1717	1668	1358	1350
40	20	70	500	50	1	1725	1725	1405	1405
40	20	70	500	50	2	2263	2154	1550	1219
40	20	70	500	50	3	1951	1779	1234	1028
40	20	70	500	50	4	1950	1792	1242	1232
40	20	70	500	50	5	1686	1686	1307	1307
40	20	80	100	0	1	1976	1976	1694	1693
40	20	80	100	0	2	1730	1721	1310	1310
40	20	80	100	0	3	1653	1533	1323	1317
40	20	80	100	0	4	1784	1750	1557	1555
40	20	80	100	0	5	1655	1512	1353	1350
40	20	80	100	25	1	2220	2220	1761	1761
40	20	80	100	25	2	2108	2106	1478	1478
40	20	80	100	25	3	1869	1855	1263	1263
40	20	80	100	25	4	1903	1903	1351	1351
40	20	80	100	25	5	2369	2130	1387	1387
40	20	80	100	50	1	2014	1957	1283	1283
40	20	80	100	50	2	2021	1949	1514	1492
40	20	80	100	50	3	2720	2524	1764	1748
40	20	80	100	50	4	1920	1920	1272	1272
40	20	80	100	50	5	2154	2000	1291	1291
40	20	80	200	0	1	1814	1579	1556	1540
40	20	80	200	0	2	1802	1720	1361	1361
40	20	80	200	0	3	1648	1648	1297	1297
40	20	80	200	0	4	1715	1715	1406	1406
40	20	80	200	0	5	1781	1781	1495	1479
40	20	80	200	25	1	2126	1967	1525	1525
40	20	80	200	25	2	2138	2138	1535	1535
40	20	80	200	25	3	2101	2101	1558	1151
40	20	80	200	25	4	2222	1989	1525	1156
40	20	80	200	25	5	2102	2102	1623	1395
40	20	80	200	50	1	2503	2499	1523	1523
40	20	80	200	50	2	2432	2275	1572	1571
40	20	80	200	50	3	2596	2596	1661	1661
40	20	80	200	50	4	1948	1948	1419	1419
40	20	80	200	50	5	2688	2620	1770	1770
40	20	80	300	0	1	2236	1972	1497	1497
40	20	80	300	0	2	1790	1790	1579	1579
40	20	80	300	0	3	1615	1615	1412	1412
40	20	80	300	0	4	2061	1871	1581	1274
40	20	80	300	0	5	1560	1560	1384	1384
40	20	80	300	25	1	2097	2097	1533	1533
40	20	80	300	25	2	1783	1721	1472	1472
40	20	80	300	25	3	1666	1666	1440	1440
40	20	80	300	25	4	1931	1771	1395	1380
40	20	80	300	25	5	2262	2163	1754	1753
40	20	80	300	50	1	2577	2577	1475	1475
40	20	80	300	50	2	2477	2329	1514	1514
40	20	80	300	50	3	2402	2325	1607	1607

40	20	80	300	50	4	2168	2168	1491	1491
40	20	80	300	50	5	2195	2057	1494	1494
40	20	80	400	0	1	2036	1685	1591	1591
40	20	80	400	0	2	1704	1629	1414	1414
40	20	80	400	0	3	1719	1719	1394	1394
40	20	80	400	0	4	1915	1915	1624	1624
40	20	80	400	0	5	1459	1459	1339	1339
40	20	80	400	25	1	1882	1882	1577	1577
40	20	80	400	25	2	1889	1741	1323	1323
40	20	80	400	25	3	2141	2046	1659	1659
40	20	80	400	25	4	2312	2204	1551	1550
40	20	80	400	25	5	1991	1991	1557	1557
40	20	80	400	50	1	1847	1847	1336	1336
40	20	80	400	50	2	1910	1767	1362	1360
40	20	80	400	50	3	2479	2479	1688	1688
40	20	80	400	50	4	2055	1970	1499	1499
40	20	80	400	50	5	2341	2341	1723	1723
40	20	80	500	0	1	1662	1558	1485	1485
40	20	80	500	0	2	1800	1652	1537	1537
40	20	80	500	0	3	1813	1813	1635	1635
40	20	80	500	0	4	2017	1581	1543	1529
40	20	80	500	0	5	2071	1772	1775	1761
40	20	80	500	25	1	1932	1932	1709	1709
40	20	80	500	25	2	2314	2293	1617	1595
40	20	80	500	25	3	1713	1604	1554	1554
40	20	80	500	25	4	2044	2044	1624	1624
40	20	80	500	25	5	2447	2447	1689	1689
40	20	80	500	50	1	2668	2668	1584	1580
40	20	80	500	50	2	2129	2014	1594	1592
40	20	80	500	50	3	1864	1759	1224	1224
40	20	80	500	50	4	1869	1869	1362	1362
40	20	80	500	50	5	2082	2008	1526	1526
40	30	60	100	0	1	1444	1373	1183	1080
40	30	60	100	0	2	1521	1222	1221	1201
40	30	60	100	0	3	1771	1562	1282	1275
40	30	60	100	0	4	1579	1262	1260	1244
40	30	60	100	0	5	1479	1167	1050	1030
40	30	60	100	25	1	1730	1313	1019	1008
40	30	60	100	25	2	1562	1344	1070	1070
40	30	60	100	25	3	1619	1471	1225	1201
40	30	60	100	25	4	1616	1508	1161	1154
40	30	60	100	25	5	1518	1310	1050	1046
40	30	60	100	50	1	1837	1543	858	789
40	30	60	100	50	2	2101	1917	1213	1203
40	30	60	100	50	3	1935	1656	1109	1097
40	30	60	100	50	4	1839	1758	1135	1123
40	30	60	100	50	5	1987	1578	1024	1011
40	30	60	200	0	1	1302	1302	1221	1218
40	30	60	200	0	2	1615	1401	1354	1352
40	30	60	200	0	3	1516	1458	1435	1432
40	30	60	200	0	4	1292	1105	1106	1098
40	30	60	200	0	5	1040	849	770	770
40	30	60	200	25	1	1559	1278	1012	1010
40	30	60	200	25	2	1464	1333	1101	1096
40	30	60	200	25	3	1764	1487	1191	1187
40	30	60	200	25	4	1502	1371	986	986
40	30	60	200	25	5	1675	1462	975	875
40	30	60	200	50	1	1970	1792	1027	1024
40	30	60	200	50	2	1811	1687	914	882
40	30	60	200	50	3	2259	2037	1382	1379
40	30	60	200	50	4	2172	2035	1402	1399
40	30	60	200	50	5	2027	1705	1151	1148
40	30	60	300	0	1	1565	1409	1286	1100
40	30	60	300	0	2	1472	1368	1275	1274
40	30	60	300	0	3	1264	1159	1156	1154

40	30	60	300	0	4	1436	1096	1095	1094
40	30	60	300	0	5	1518	1325	1325	1315
40	30	60	300	25	1	1574	1230	970	956
40	30	60	300	25	2	1544	1302	1051	1044
40	30	60	300	25	3	1690	1541	1266	1243
40	30	60	300	25	4	1443	1355	1049	1040
40	30	60	300	25	5	1936	1680	1479	1476
40	30	60	300	50	1	1865	1729	1078	1070
40	30	60	300	50	2	1737	1590	1103	1103
40	30	60	300	50	3	1823	1627	1173	1104
40	30	60	300	50	4	1735	1505	1020	996
40	30	60	300	50	5	1878	1509	1028	1023
40	30	60	400	0	1	1292	1239	1141	1140
40	30	60	400	0	2	1564	1084	1086	1082
40	30	60	400	0	3	1229	1170	1088	1086
40	30	60	400	0	4	1507	1334	1329	1324
40	30	60	400	0	5	1404	1156	1155	1146
40	30	60	400	25	1	1855	1758	1403	1291
40	30	60	400	25	2	1723	1701	1336	1288
40	30	60	400	25	3	1949	1781	1453	1441
40	30	60	400	25	4	1806	1612	1257	1245
40	30	60	400	25	5	1609	1460	1194	1169
40	30	60	400	50	1	2118	1876	1197	832
40	30	60	400	50	2	2141	1790	1175	1169
40	30	60	400	50	3	2167	2069	1611	1491
40	30	60	400	50	4	1913	1536	996	991
40	30	60	400	50	5	2277	2013	1334	1330
40	30	60	500	0	1	1433	1330	1129	1121
40	30	60	500	0	2	1347	1180	1183	1177
40	30	60	500	0	3	1262	1152	1152	1147
40	30	60	500	0	4	1357	1285	1253	1173
40	30	60	500	0	5	1433	1433	1314	1313
40	30	60	500	25	1	1589	1395	1055	958
40	30	60	500	25	2	1751	1463	1177	1175
40	30	60	500	25	3	1784	1482	1178	1175
40	30	60	500	25	4	1589	1357	1096	1096
40	30	60	500	25	5	1524	1380	1131	1118
40	30	60	500	50	1	1912	1706	994	988
40	30	60	500	50	2	2067	1724	1107	1083
40	30	60	500	50	3	2290	2074	1236	927
40	30	60	500	50	4	1878	1564	1027	1007
40	30	60	500	50	5	2054	2021	1061	831
40	30	70	100	0	1	1572	1418	1334	1334
40	30	70	100	0	2	1680	1583	1359	1313
40	30	70	100	0	3	1798	1495	1420	1418
40	30	70	100	0	4	1524	1450	1354	1324
40	30	70	100	0	5	1629	1435	1326	1322
40	30	70	100	25	1	2000	1832	1471	1326
40	30	70	100	25	2	1844	1542	1257	1243
40	30	70	100	25	3	1969	1963	1516	1516
40	30	70	100	25	4	2226	1961	1532	1277
40	30	70	100	25	5	2080	1987	1529	1310
40	30	70	100	50	1	2566	2147	1295	1295
40	30	70	100	50	2	1971	1700	1205	1182
40	30	70	100	50	3	2541	2369	1701	1596
40	30	70	100	50	4	2211	2135	1338	1136
40	30	70	100	50	5	2704	2394	1354	1295
40	30	70	200	0	1	1819	1614	1424	1424
40	30	70	200	0	2	1868	1352	1250	1236
40	30	70	200	0	3	1851	1644	1564	1282
40	30	70	200	0	4	1682	1682	1561	1561
40	30	70	200	0	5	1928	1637	1572	1502
40	30	70	200	25	1	2269	2057	1534	1533
40	30	70	200	25	2	1846	1642	1331	1300
40	30	70	200	25	3	1954	1701	1425	1421

40	30	70	200	25	4	1903	1648	1402	1401
40	30	70	200	25	5	2042	1821	1429	1429
40	30	70	200	50	1	2365	1902	1303	1302
40	30	70	200	50	2	2235	2050	1332	1325
40	30	70	200	50	3	2046	1938	1514	1497
40	30	70	200	50	4	2829	2392	1624	1594
40	30	70	200	50	5	2553	2417	1584	1571
40	30	70	300	0	1	1606	1315	1315	1278
40	30	70	300	0	2	1854	1697	1459	1204
40	30	70	300	0	3	1985	1615	1492	1468
40	30	70	300	0	4	2053	1762	1614	1594
40	30	70	300	0	5	2075	1766	1604	1416
40	30	70	300	25	1	1764	1583	1330	1323
40	30	70	300	25	2	2066	2065	1735	1735
40	30	70	300	25	3	2361	2157	1589	1471
40	30	70	300	25	4	2357	1946	1522	1506
40	30	70	300	25	5	2158	1977	1449	1197
40	30	70	300	50	1	2234	2141	1493	1285
40	30	70	300	50	2	2719	2131	1416	1412
40	30	70	300	50	3	2140	1872	1205	1114
40	30	70	300	50	4	2919	2560	1612	1203
40	30	70	300	50	5	2041	1878	1274	1266
40	30	70	400	0	1	1893	1599	1415	1415
40	30	70	400	0	2	1710	1596	1597	1596
40	30	70	400	0	3	1735	1698	1504	1504
40	30	70	400	0	4	1711	1509	1506	1500
40	30	70	400	0	5	1682	1606	1530	1517
40	30	70	400	25	1	2180	2043	1552	1534
40	30	70	400	25	2	2037	1920	1515	1506
40	30	70	400	25	3	2459	2085	1639	1619
40	30	70	400	25	4	2139	2015	1399	1240
40	30	70	400	25	5	2435	2016	1530	1510
40	30	70	400	50	1	2698	2451	1594	1594
40	30	70	400	50	2	2317	1981	1411	1411
40	30	70	400	50	3	2421	2230	1589	1573
40	30	70	400	50	4	2655	2287	1526	1525
40	30	70	400	50	5	2586	2106	1420	1412
40	30	70	500	0	1	1800	1424	1425	1412
40	30	70	500	0	2	1747	1466	1479	1460
40	30	70	500	0	3	1934	1806	1627	1627
40	30	70	500	0	4	1976	1479	1379	1362
40	30	70	500	0	5	1500	1300	1310	1303
40	30	70	500	25	1	2181	1818	1365	1358
40	30	70	500	25	2	1872	1590	1195	1195
40	30	70	500	25	3	2116	1702	1397	1395
40	30	70	500	25	4	1839	1531	1144	846
40	30	70	500	25	5	2500	1902	1526	1499
40	30	70	500	50	1	2062	1856	1379	1365
40	30	70	500	50	2	2343	1987	1399	1398
40	30	70	500	50	3	2563	2376	1311	1294
40	30	70	500	50	4	2020	1888	1392	1392
40	30	70	500	50	5	2180	2107	1478	1299
40	30	80	100	0	1	2106	1706	1482	1482
40	30	80	100	0	2	2568	2115	1862	1860
40	30	80	100	0	3	1943	1786	1594	1587
40	30	80	100	0	4	2366	1695	1692	1692
40	30	80	100	0	5	2051	1876	1514	1514
40	30	80	100	25	1	2428	2428	1774	1774
40	30	80	100	25	2	2411	2305	1689	1403
40	30	80	100	25	3	2058	2016	1272	1272
40	30	80	100	25	4	2252	2033	1445	1444
40	30	80	100	25	5	3367	3219	1826	1826
40	30	80	100	50	1	2667	2473	1528	1528
40	30	80	100	50	2	2475	2332	1473	1466
40	30	80	100	50	3	2738	2582	1381	1381

40	30	80	100	50	4	2643	2264	1523	1523
40	30	80	100	50	5	2208	1924	1383	1383
40	30	80	200	0	1	2363	2162	1762	1762
40	30	80	200	0	2	2320	1983	1722	1711
40	30	80	200	0	3	1903	1590	1565	1565
40	30	80	200	0	4	2254	1997	1700	1669
40	30	80	200	0	5	2361	1977	1711	1504
40	30	80	200	25	1	2986	2529	1825	1825
40	30	80	200	25	2	2631	2258	1734	1727
40	30	80	200	25	3	2574	2367	1792	1791
40	30	80	200	25	4	2585	2341	1715	1715
40	30	80	200	25	5	2361	1854	1405	1405
40	30	80	200	50	1	3056	2682	1780	1780
40	30	80	200	50	2	3469	2875	1698	1698
40	30	80	200	50	3	2876	2876	1726	1726
40	30	80	200	50	4	2607	2375	1630	1428
40	30	80	200	50	5	3121	2786	1703	1499
40	30	80	300	0	1	2150	1831	1653	1647
40	30	80	300	0	2	2084	1820	1662	1399
40	30	80	300	0	3	2152	1971	1773	1773
40	30	80	300	0	4	2468	2025	1749	1749
40	30	80	300	0	5	1959	1652	1584	1580
40	30	80	300	25	1	2474	2440	1590	1590
40	30	80	300	25	2	2460	2051	1472	1469
40	30	80	300	25	3	2381	2132	1614	1596
40	30	80	300	25	4	2278	2165	1672	1672
40	30	80	300	25	5	2473	1993	1500	1500
40	30	80	300	50	1	3172	2725	1809	1796
40	30	80	300	50	2	2960	2364	1498	1498
40	30	80	300	50	3	3307	3104	1908	1899
40	30	80	300	50	4	2931	2382	1627	1614
40	30	80	300	50	5	2981	2283	1424	1418
40	30	80	400	0	1	2443	2061	2055	2055
40	30	80	400	0	2	1749	1565	1341	1323
40	30	80	400	0	3	2157	1926	1799	1553
40	30	80	400	0	4	2001	1915	1706	1706
40	30	80	400	0	5	2075	1917	1695	1695
40	30	80	400	25	1	2235	2004	1578	1578
40	30	80	400	25	2	2610	2291	1641	1641
40	30	80	400	25	3	3049	2616	1646	1646
40	30	80	400	25	4	2361	2124	1705	1698
40	30	80	400	25	5	2044	2015	1741	1741
40	30	80	400	50	1	2726	2510	1778	1428
40	30	80	400	50	2	2503	2392	1784	1784
40	30	80	400	50	3	2724	2584	1565	1565
40	30	80	400	50	4	2776	2686	1785	1203
40	30	80	400	50	5	2169	1948	1706	1706
40	30	80	500	0	1	1952	1705	1523	1516
40	30	80	500	0	2	1961	1815	1684	1683
40	30	80	500	0	3	2154	1970	1838	1838
40	30	80	500	0	4	2163	1951	1865	1865
40	30	80	500	0	5	2277	2277	1932	1926
40	30	80	500	25	1	2597	2597	1620	1620
40	30	80	500	25	2	2048	1788	1511	1511
40	30	80	500	25	3	2710	2092	1536	1521
40	30	80	500	25	4	2606	2188	1751	1654
40	30	80	500	25	5	2559	2178	1729	1729
40	30	80	500	50	1	2484	2368	1635	1635
40	30	80	500	50	2	2831	2502	1614	1163
40	30	80	500	50	3	3108	2740	1611	1611
40	30	80	500	50	4	3164	2725	1989	1766
40	30	80	500	50	5	2421	2151	1516	1512
40	30	80	500	50	1	1822	1322	1274	1271
40	40	60	100	0	2	1637	1271	1206	1200
40	40	60	100	0	3	1620	1244	1138	1137
40	40	60	100	0					

40	40	60	100	0	4	1833	1435	1409	1388
40	40	60	100	0	5	2074	1414	1365	1362
40	40	60	100	25	1	2495	1997	1380	1348
40	40	60	100	25	2	1856	1575	1013	1011
40	40	60	100	25	3	1842	1454	1135	1132
40	40	60	100	25	4	1540	1476	914	912
40	40	60	100	25	5	2173	1665	1278	1268
40	40	60	100	50	1	2448	1908	1169	1165
40	40	60	100	50	2	2467	2052	1057	1033
40	40	60	100	50	3	2467	2016	1296	1296
40	40	60	100	50	4	2194	1944	1253	1251
40	40	60	100	50	5	2536	2153	1276	1224
40	40	60	200	0	1	1671	1224	1224	1222
40	40	60	200	0	2	1847	1246	1351	1322
40	40	60	200	0	3	1898	1493	1361	1358
40	40	60	200	0	4	1583	1189	1189	1182
40	40	60	200	0	5	1679	1233	1163	1158
40	40	60	200	25	1	2466	2122	1548	1539
40	40	60	200	25	2	2343	1804	1282	1274
40	40	60	200	25	3	2342	1784	1367	1338
40	40	60	200	25	4	2059	1651	1292	1281
40	40	60	200	25	5	2120	1722	1193	1189
40	40	60	200	50	1	3053	2302	1258	1254
40	40	60	200	50	2	2568	2242	1478	1478
40	40	60	200	50	3	2564	2329	1602	1163
40	40	60	200	50	4	2736	2227	1271	1256
40	40	60	200	50	5	2511	2032	1271	1260
40	40	60	300	0	1	1828	1312	1262	1244
40	40	60	300	0	2	2030	1459	1372	1369
40	40	60	300	0	3	1800	1324	1294	1273
40	40	60	300	0	4	1869	1350	1229	1224
40	40	60	300	0	5	1892	1343	1345	1334
40	40	60	300	25	1	2537	1892	1470	1388
40	40	60	300	25	2	2011	1592	1171	1155
40	40	60	300	25	3	2075	1616	1248	1248
40	40	60	300	25	4	2263	1782	1134	1091
40	40	60	300	25	5	2461	2039	1293	1097
40	40	60	300	50	1	2479	2085	1270	1238
40	40	60	300	50	2	2421	2224	1483	1470
40	40	60	300	50	3	2155	1578	848	848
40	40	60	300	50	4	2973	2463	1554	1554
40	40	60	300	50	5	2477	2319	1397	1391
40	40	60	400	0	1	1720	1260	1260	1258
40	40	60	400	0	2	1813	1342	1346	1337
40	40	60	400	0	3	2031	1482	1482	1480
40	40	60	400	0	4	1655	1261	1192	1169
40	40	60	400	0	5	1661	1311	1295	1290
40	40	60	400	25	1	2202	2009	1450	1447
40	40	60	400	25	2	2376	2043	1445	1414
40	40	60	400	25	3	2180	1918	1461	1156
40	40	60	400	25	4	2055	1786	1240	1236
40	40	60	400	25	5	2410	1801	1210	1207
40	40	60	400	50	1	2609	2239	1317	1281
40	40	60	400	50	2	2397	2187	1396	1017
40	40	60	400	50	3	2785	2310	1382	1369
40	40	60	400	50	4	2533	2245	1482	1442
40	40	60	400	50	5	2579	2218	1326	1322
40	40	60	500	0	1	1834	1342	1361	1350
40	40	60	500	0	2	1451	1000	1051	1045
40	40	60	500	0	3	1572	1161	1159	1151
40	40	60	500	0	4	1824	1409	1273	1247
40	40	60	500	0	5	1830	1349	1288	1203
40	40	60	500	25	1	2598	2160	1659	1397
40	40	60	500	25	2	1966	1536	1173	1173
40	40	60	500	25	3	2136	1749	1376	1358

40	40	60	500	25	4	1803	1562	1260	1241
40	40	60	500	25	5	2021	1717	1158	1144
40	40	60	500	50	1	2557	2128	1359	1359
40	40	60	500	50	2	2294	1962	1268	1264
40	40	60	500	50	3	2887	2508	1627	1627
40	40	60	500	50	4	2319	2046	1251	1251
40	40	60	500	50	5	2320	1995	1216	1202
40	40	70	100	0	1	2168	1788	1788	1784
40	40	70	100	0	2	2086	1575	1254	1220
40	40	70	100	0	3	1890	1337	1338	1322
40	40	70	100	0	4	2293	1924	1622	1182
40	40	70	100	0	5	1952	1565	1565	1559
40	40	70	100	25	1	2761	2352	1910	1910
40	40	70	100	25	2	2493	2198	1638	1636
40	40	70	100	25	3	2629	2154	1728	1723
40	40	70	100	25	4	2067	1802	1400	1400
40	40	70	100	25	5	3072	2402	1900	1870
40	40	70	100	50	1	2408	2169	1527	1518
40	40	70	100	50	2	2227	1931	1219	1219
40	40	70	100	50	3	2813	2480	1571	1336
40	40	70	100	50	4	2340	1999	1383	1383
40	40	70	100	50	5	3295	2792	1633	1450
40	40	70	200	0	1	1986	1528	1249	1193
40	40	70	200	0	2	2215	1552	1545	1535
40	40	70	200	0	3	2368	1748	1658	1644
40	40	70	200	0	4	2005	1601	1554	1503
40	40	70	200	0	5	2125	1776	1639	1174
40	40	70	200	25	1	3156	2324	1684	1676
40	40	70	200	25	2	2370	1850	1484	1462
40	40	70	200	25	3	2671	1987	1467	1438
40	40	70	200	25	4	2147	1715	1364	1358
40	40	70	200	25	5	2831	2143	1547	1527
40	40	70	200	50	1	3037	2423	1520	1520
40	40	70	200	50	2	3394	2976	1653	1156
40	40	70	200	50	3	3159	2636	1756	1741
40	40	70	200	50	4	2804	2377	1596	1593
40	40	70	200	50	5	2479	2388	1446	1079
40	40	70	300	0	1	1871	1431	1395	1393
40	40	70	300	0	2	2280	1615	1513	1494
40	40	70	300	0	3	1794	1479	1296	1290
40	40	70	300	0	4	2390	1765	1761	1759
40	40	70	300	0	5	2152	1824	1635	1635
40	40	70	300	25	1	2621	2159	1730	1730
40	40	70	300	25	2	2482	1835	1467	1467
40	40	70	300	25	3	3078	2290	1609	1561
40	40	70	300	25	4	2629	2113	1530	1314
40	40	70	300	25	5	2624	2399	1578	1561
40	40	70	300	50	1	2525	2423	1668	1603
40	40	70	300	50	2	3268	2572	1653	1653
40	40	70	300	50	3	2357	2028	1419	1375
40	40	70	300	50	4	2706	2288	1482	1482
40	40	70	300	50	5	2721	2092	1316	1309
40	40	70	400	0	1	2019	1590	1590	1582
40	40	70	400	0	2	1995	1542	1542	1503
40	40	70	400	0	3	1984	1543	1468	1328
40	40	70	400	0	4	1929	1588	1446	1380
40	40	70	400	0	5	1967	1550	1573	1573
40	40	70	400	25	1	2347	1935	1512	1263
40	40	70	400	25	2	2966	2222	1736	1728
40	40	70	400	25	3	2503	2127	1482	1480
40	40	70	400	25	4	2645	1980	1502	1502
40	40	70	400	25	5	2721	2268	1309	1308
40	40	70	400	50	1	3089	2608	1763	1751
40	40	70	400	50	2	3331	2359	1411	1397
40	40	70	400	50	3	2822	2463	1604	1452

40	40	70	400	50	4	3006	2433	1517	1504
40	40	70	400	50	5	2599	2258	1380	1379
40	40	70	500	0	1	2123	1701	1695	1682
40	40	70	500	0	2	2065	1703	1645	1495
40	40	70	500	0	3	2082	1632	1548	1519
40	40	70	500	0	4	2255	1638	1545	1545
40	40	70	500	0	5	2093	1674	1618	1607
40	40	70	500	25	1	2612	1887	1329	1308
40	40	70	500	25	2	2303	1909	1427	1412
40	40	70	500	25	3	2317	2077	1550	1543
40	40	70	500	25	4	2444	2055	1663	1663
40	40	70	500	25	5	2756	2198	1716	1635
40	40	70	500	50	1	2989	2488	1712	1529
40	40	70	500	50	2	3333	3176	1751	1291
40	40	70	500	50	3	3264	2757	1864	1864
40	40	70	500	50	4	2634	2054	1338	1330
40	40	70	500	50	5	3037	2570	1486	1277
40	40	70	500	50	1	2462	1822	1463	1463
40	40	80	100	0	2	2553	2265	1781	1675
40	40	80	100	0	3	2767	2139	1832	1818
40	40	80	100	0	4	3073	2328	2189	2004
40	40	80	100	0	5	2448	1960	1674	1674
40	40	80	100	0	1	3066	2628	1724	1724
40	40	80	100	25	2	2813	2336	1715	1715
40	40	80	100	25	3	2372	1982	1512	1512
40	40	80	100	25	4	2749	2128	1586	1586
40	40	80	100	25	5	2721	2218	1616	1538
40	40	80	100	25	1	3528	2875	1773	1592
40	40	80	100	50	2	3939	3381	1620	1611
40	40	80	100	50	3	3434	2917	1793	1793
40	40	80	100	50	4	3150	2820	1737	1737
40	40	80	100	50	5	3676	3111	1840	1813
40	40	80	100	50	1	3487	3487	2088	2056
40	40	80	200	0	2	2199	1766	1424	1389
40	40	80	200	0	3	2227	1867	1780	1486
40	40	80	200	0	4	2585	2169	1878	1455
40	40	80	200	0	5	1908	1641	1626	1626
40	40	80	200	0	1	2887	2412	1892	1817
40	40	80	200	25	2	2334	1898	1532	1532
40	40	80	200	25	3	2729	2326	1920	1765
40	40	80	200	25	4	3076	2560	1904	1866
40	40	80	200	25	5	3939	3344	2199	2199
40	40	80	200	25	1	3490	2688	1567	1567
40	40	80	200	50	2	3464	3108	2059	2059
40	40	80	200	50	3	3831	3807	1912	1910
40	40	80	200	50	4	3154	2688	1888	1884
40	40	80	200	50	5	3375	2896	1840	1832
40	40	80	200	50	1	2436	2018	1735	1380
40	40	80	300	0	2	2514	1956	1736	1736
40	40	80	300	0	3	2263	1962	1750	1542
40	40	80	300	0	4	2738	2018	1795	1427
40	40	80	300	0	5	2643	2060	1799	1778
40	40	80	300	0	1	3155	2649	2045	2032
40	40	80	300	25	2	2792	2379	1544	1544
40	40	80	300	25	3	3189	2370	1572	1569
40	40	80	300	25	4	3300	2829	1854	1854
40	40	80	300	25	5	2768	1985	1569	1559
40	40	80	300	25	1	3734	3555	1956	1956
40	40	80	300	50	2	4659	4456	1966	1186
40	40	80	300	50	3	3008	2583	1827	1826
40	40	80	300	50	4	2940	2643	1513	1467
40	40	80	300	50	5	3224	2850	1807	1515
40	40	80	300	50	1	2581	2543	1713	1713
40	40	80	400	0	2	2238	1859	1512	1512
40	40	80	400	0	3	2582	1995	1714	1467
40	40	80	400	0					

40	40	80	400	0	4	2443	1588	1267	1259
40	40	80	400	0	5	2360	2018	1739	1355
40	40	80	400	25	1	3082	2713	1717	1701
40	40	80	400	25	2	2784	2212	1479	1232
40	40	80	400	25	3	2813	2233	1932	1932
40	40	80	400	25	4	2616	2204	1913	1737
40	40	80	400	25	5	2579	2102	1722	1720
40	40	80	400	25	1	2797	2529	1606	1599
40	40	80	400	50	2	2853	2504	1685	1639
40	40	80	400	50	3	3274	2858	1877	1849
40	40	80	400	50	4	3396	2962	1808	1796
40	40	80	400	50	5	3505	3180	1772	1764
40	40	80	400	50	1	2055	1831	1672	1672
40	40	80	500	0	2	2763	2234	1972	1921
40	40	80	500	0	3	2162	1772	1623	1600
40	40	80	500	0	4	2193	1740	1627	1615
40	40	80	500	0	5	2607	2167	2120	2101
40	40	80	500	0	1	3299	2944	1751	1312
40	40	80	500	25	2	3223	2600	2063	2041
40	40	80	500	25	3	2853	2284	1374	1169
40	40	80	500	25	4	3000	2469	1741	1564
40	40	80	500	25	5	2504	2001	1507	1469
40	40	80	500	25	1	3547	3297	1753	1612
40	40	80	500	50	2	3562	3044	1830	1669
40	40	80	500	50	3	3353	2595	1491	1489
40	40	80	500	50	4	3443	3180	1975	1975
40	40	80	500	50	5	3928	3399	2079	1689

NUMBER OF NODES 40 NUMBER OF OBSERVATIONS 675

	LOAD BASED	G(A)	G(B)	G(B)/ARCOPT
AVG SOLUTION LENGTH	2091	1843	1410	1362
NO OF BETTER SOLUTIONS		559	674	674
% IMPROVEMENT OVER LOAD BASED		12	33	35

% OF BACKHAULS IS 20 NUMBER OF OBSERVATIONS 225

	LOAD BASED	G(A)	G(B)	G(B)/ARCOPT
AVG SOLUTION LENGTH	1638	1580	1272	1235
NO OF BETTER SOLUTIONS		117	224	224
% IMPROVEMENT OVER LOAD BASED		4	22	25

% OF BACKHAULS IS 30 NUMBER OF OBSERVATIONS 225

	LOAD BASED	G(A)	G(B)	G(B)/ARCOPT
AVG SOLUTION LENGTH	2091	1851	1422	1378
NO OF BETTER SOLUTIONS		218	225	225
% IMPROVEMENT OVER LOAD BASED		11	32	34

% OF BACKHAULS IS 40 NUMBER OF OBSERVATIONS 225

	LOAD BASED	G(A)	G(B)	G(B)/ARCOPT
AVG SOLUTION LENGTH	2544	2096	1535	1473
NO OF BETTER SOLUTIONS		224	225	225
% IMPROVEMENT OVER LOAD BASED		18	40	42

% OF CONCENTRATION IS 60 NUMBER OF OBSERVATIONS 225

	LOAD BASED	G(A)	G(B)	G(B)/ARCOPT
AVG SOLUTION LENGTH	1694	1479	1158	1124
NO OF BETTER SOLUTIONS		179	224	224
% IMPROVEMENT OVER LOAD BASED		13	32	34

% OF CONCENTRATION IS 70 NUMBER OF OBSERVATIONS 225

	LOAD BASED	G(A)	G(B)	G(B)/ARCOPT
AVG SOLUTION LENGTH	2088	1827	1429	1372
NO OF BETTER SOLUTIONS		193	225	225
% IMPROVEMENT OVER LOAD BASED		13	32	34

% OF CONCENTRATION IS 80 NUMBER OF OBSERVATIONS 225

	LOAD BASED	G(A)	G(B)	G(B)/ARCOPT
AVG SOLUTION LENGTH	2491	2222	1641	1591
NO OF BETTER SOLUTIONS		187	225	225
% IMPROVEMENT OVER LOAD BASED		11	34	36

BACKHAUL PENALTY IS 100 NUMBER OF OBSERVATIONS 135

	LOAD BASED	G(A)	G(B)	G(B)/ARCOPT
AVG SOLUTION LENGTH	2054	1813	1365	1334
NO OF BETTER SOLUTIONS		112	135	135
% IMPROVEMENT OVER LOAD BASED		12	34	35

BACKHAUL PENALTY IS 200 NUMBER OF OBSERVATIONS 135

	LOAD BASED	G(A)	G(B)	G(B)/ARCOPT
AVG SOLUTION LENGTH	2126	1875	1423	1369
NO OF BETTER SOLUTIONS		105	135	135
% IMPROVEMENT OVER LOAD BASED		12	33	36

BACKHAUL PENALTY IS 300 NUMBER OF OBSERVATIONS 135

	LOAD BASED	G(A)	G(B)	G(B)/ARCOPT
AVG SOLUTION LENGTH	2109	1843	1407	1356
NO OF BETTER SOLUTIONS		117	135	135
% IMPROVEMENT OVER LOAD BASED		13	33	36

BACKHAUL PENALTY IS 400 NUMBER OF OBSERVATIONS 135

	LOAD BASED	G(A)	G(B)	G(B)/ARCOPT
AVG SOLUTION LENGTH	2083	1850	1437	1387
NO OF BETTER SOLUTIONS		114	134	134
% IMPROVEMENT OVER LOAD BASED		11	31	33

BACKHAUL PENALTY IS 500 NUMBER OF OBSERVATIONS 135

	LOAD BASED	G(A)	G(B)	G(B)/ARCOPT
AVG SOLUTION LENGTH	2082	1832	1416	1364
NO OF BETTER SOLUTIONS		111	135	135
% IMPROVEMENT OVER LOAD BASED		12	32	34

RADIAL SURCHARGE IS 0 NUMBER OF OBSERVATIONS 225

	LOAD BASED	G(A)	G(B)	G(B)/ARCOPT
AVG SOLUTION LENGTH	1779	1531	1404	1366
NO OF BETTER SOLUTIONS		186	224	224
% IMPROVEMENT OVER LOAD BASED		14	21	23

RADIAL SURCHARGE IS 25 NUMBER OF OBSERVATIONS 225

	LOAD BASED	G(A)	G(B)	G(B)/ARCOPT
AVG SOLUTION LENGTH	2100	1846	1409	1367
NO OF BETTER SOLUTIONS		186	225	225
% IMPROVEMENT OVER LOAD BASED		12	33	35

RADIAL SURCHARGE IS 50 NUMBER OF OBSERVATIONS 225

	LOAD BASED	G(A)	G(B)	G(B)/ARCOPT
AVG SOLUTION LENGTH	2393	2150	1416	1354
NO OF BETTER SOLUTIONS		187	225	225
% IMPROVEMENT OVER LOAD BASED		10	41	43

NO. OF NODES	% OF BACKHAULS	% OF CONCENTR	BKHAUL PENALTY	RADIAL SURCHARGE	PROBL NO.	LOAD BASED SOL	G(A) SOL	G(B) SOL	G(B)/ARC SOL
60	20	60	100	0	1	1437	1216	1219	1197
60	20	60	100	0	2	1510	1241	1161	1131
60	20	60	100	0	3	1442	1391	1102	1087
60	20	60	100	0	4	1334	1330	1082	943
60	20	60	100	0	5	1483	1452	1289	1282
60	20	60	100	25	1	1749	1670	1316	935
60	20	60	100	25	2	1474	1474	1133	1120
60	20	60	100	25	3	1297	1209	1069	1029
60	20	60	100	25	4	1664	1592	1275	1261
60	20	60	100	25	5	1480	1480	1178	1170
60	20	60	100	50	1	1879	1879	1388	1374
60	20	60	100	50	2	1235	1235	1445	1445
60	20	60	100	50	3	1301	1301	1236	1119
60	20	60	100	50	4	1348	1348	1319	1305
60	20	60	100	50	5	1360	1360	1276	934
60	20	60	200	0	1	1068	1033	1008	1008
60	20	60	200	0	2	1408	1375	1203	1107
60	20	60	200	0	3	1545	1349	1213	1198
60	20	60	200	0	4	1526	1423	1362	1234
60	20	60	200	0	5	1561	1209	1156	1117
60	20	60	200	25	1	1516	1516	1350	1297
60	20	60	200	25	2	1553	1534	1135	993
60	20	60	200	25	3	1185	1185	1284	1260
60	20	60	200	25	4	1755	1711	1133	983
60	20	60	200	25	5	1453	1301	1023	1013
60	20	60	200	50	1	1093	1093	1177	1165
60	20	60	200	50	2	1580	1580	1367	870
60	20	60	200	50	3	1560	1560	1233	1224
60	20	60	200	50	4	1698	1698	1353	995
60	20	60	200	50	5	1650	1650	1298	1293
60	20	60	300	0	1	1377	1209	1188	1168
60	20	60	300	0	2	1514	1313	1268	1262
60	20	60	300	0	3	1437	1275	1231	1213
60	20	60	300	0	4	1424	1295	1174	1164
60	20	60	300	0	5	1591	1327	1322	1314
60	20	60	300	25	1	1863	1623	1299	1289
60	20	60	300	25	2	1831	1580	1573	1570
60	20	60	300	25	3	1802	1776	1244	846
60	20	60	300	25	4	1948	1618	1049	1047
60	20	60	300	25	5	2120	1850	1495	1477
60	20	60	300	50	1	1484	1484	1302	1288
60	20	60	300	50	2	1163	1163	1154	1128
60	20	60	300	50	3	1174	1174	1344	1067
60	20	60	300	50	4	1364	1364	1431	1035
60	20	60	300	50	5	2315	2121	1367	897
60	20	60	400	0	1	1574	1243	1198	1188
60	20	60	400	0	2	1341	1111	1109	1091
60	20	60	400	0	3	1532	1253	1252	1246
60	20	60	400	0	4	1600	1037	1036	1021
60	20	60	400	0	5	1441	1281	1168	1137
60	20	60	400	25	1	1349	1349	1527	1181
60	20	60	400	25	2	1912	1829	1545	1533
60	20	60	400	25	3	1644	1483	1263	1244
60	20	60	400	25	4	1974	1871	1472	1225
60	20	60	400	25	5	1342	1342	1301	1294
60	20	60	400	50	1	1336	1176	984	961
60	20	60	400	50	2	1450	1450	1325	1097
60	20	60	400	50	3	2109	2031	1395	1002
60	20	60	400	50	4	1058	1058	1123	1109
60	20	60	400	50	5	1349	1349	1477	1389
60	20	60	500	0	1	1465	1312	1306	1299
60	20	60	500	0	2	1596	1335	1333	1327
60	20	60	500	0	3	1340	1198	1182	1169

60	20	60	500	0	4	1490	1313	1314	1302
60	20	60	500	0	5	1257	1162	1066	1039
60	20	60	500	25	1	1521	1383	1317	1040
60	20	60	500	25	2	1773	1596	1421	1420
60	20	60	500	25	3	1822	1718	1425	1410
60	20	60	500	25	4	1523	1517	1089	1081
60	20	60	500	25	5	1817	1745	1350	1320
60	20	60	500	50	1	984	984	1025	902
60	20	60	500	50	2	1527	1463	1212	1199
60	20	60	500	50	3	1289	1289	1328	1314
60	20	60	500	50	4	1083	1083	1229	1212
60	20	60	500	50	5	1219	1219	1477	1211
60	20	70	100	0	1	1447	1447	1560	1541
60	20	70	100	0	2	1784	1784	2011	1933
60	20	70	100	0	3	1445	1445	1445	1445
60	20	70	100	0	4	1560	1560	1516	1503
60	20	70	100	0	5	1449	1449	1454	1454
60	20	70	100	25	1	1503	1503	1511	1511
60	20	70	100	25	2	1646	1646	1748	1576
60	20	70	100	25	3	1566	1566	1574	1545
60	20	70	100	25	4	1762	1762	1692	1679
60	20	70	100	25	5	1618	1618	1481	1470
60	20	70	100	50	1	1355	1355	1414	1398
60	20	70	100	50	2	1657	1657	1999	1947
60	20	70	100	50	3	1387	1387	1492	1459
60	20	70	100	50	4	1672	1672	1586	1578
60	20	70	100	50	5	2207	2207	2139	1892
60	20	70	200	0	1	1960	1960	2040	2008
60	20	70	200	0	2	1964	1767	1353	1091
60	20	70	200	0	3	2166	2162	2246	2208
60	20	70	200	0	4	1814	1814	2000	1996
60	20	70	200	0	5	1826	1826	1950	1939
60	20	70	200	25	1	1833	1833	2011	1923
60	20	70	200	25	2	1838	1809	1754	1752
60	20	70	200	25	3	1743	1743	1640	1629
60	20	70	200	25	4	2041	1988	1603	1550
60	20	70	200	25	5	1676	1676	1647	1589
60	20	70	200	50	1	2280	2280	2365	2331
60	20	70	200	50	2	1811	1811	1477	1466
60	20	70	200	50	3	2069	2069	2069	2069
60	20	70	200	50	4	1940	1940	1799	1220
60	20	70	200	50	5	1774	1774	1625	1615
60	20	70	300	0	1	1758	1758	1758	1758
60	20	70	300	0	2	1977	1977	1738	1723
60	20	70	300	0	3	1593	1593	1839	1828
60	20	70	300	0	4	1797	1797	1778	1774
60	20	70	300	0	5	2228	2228	2241	2241
60	20	70	300	25	1	1818	1818	2175	2124
60	20	70	300	25	2	1653	1568	1485	1479
60	20	70	300	25	3	1938	1938	1990	1950
60	20	70	300	25	4	1550	1550	1473	1177
60	20	70	300	25	5	2155	2155	2201	2191
60	20	70	300	50	1	2201	2018	1638	1621
60	20	70	300	50	2	1905	1905	1676	1277
60	20	70	300	50	3	1974	1974	2022	1946
60	20	70	300	50	4	1928	1928	2137	2134
60	20	70	300	50	5	1932	1932	1711	1701
60	20	70	400	0	1	1872	1521	1565	1544
60	20	70	400	0	2	2048	1937	2156	2060
60	20	70	400	0	3	1753	1753	1938	1928
60	20	70	400	0	4	2065	2065	2131	1609
60	20	70	400	0	5	1890	1890	1744	1684
60	20	70	400	25	1	1740	1740	1504	1281
60	20	70	400	25	2	1676	1644	1561	1544
60	20	70	400	25	3	1406	1406	1313	1311

60	20	70	400	25	4	1510	1510	1547	1537
60	20	70	400	25	5	1813	1746	1422	1416
60	20	70	400	50	1	1838	1838	2133	2121
60	20	70	400	50	2	1805	1805	1830	1795
60	20	70	400	50	3	1943	1943	1416	1022
60	20	70	400	50	4	2239	2239	2372	2372
60	20	70	400	50	5	2133	2133	1806	1518
60	20	70	500	0	1	2039	2039	2110	1761
60	20	70	500	0	2	1917	1917	2065	2065
60	20	70	500	0	3	1917	1810	1719	1638
60	20	70	500	0	4	1916	1862	1752	1744
60	20	70	500	0	5	2079	2079	2109	1933
60	20	70	500	25	1	1786	1786	1561	1552
60	20	70	500	25	2	1611	1611	1679	1661
60	20	70	500	25	3	1648	1648	1623	1255
60	20	70	500	25	4	1601	1601	1807	1466
60	20	70	500	25	5	1323	1323	1410	1407
60	20	70	500	50	1	1771	1771	1838	1190
60	20	70	500	50	2	2177	2177	2296	1893
60	20	70	500	50	3	2195	2195	2524	2475
60	20	70	500	50	4	1689	1689	1859	1805
60	20	70	500	50	5	1503	1503	1673	1665
60	20	80	100	0	1	1990	1990	2060	2055
60	20	80	100	0	2	1774	1774	2003	1994
60	20	80	100	0	3	1887	1887	1963	1795
60	20	80	100	0	4	1484	1484	1654	1654
60	20	80	100	0	5	1551	1551	1617	1604
60	20	80	100	25	1	1810	1810	1810	1810
60	20	80	100	25	2	1789	1789	1901	1875
60	20	80	100	25	3	1704	1704	1704	1704
60	20	80	100	25	4	1757	1757	1697	1557
60	20	80	100	25	5	1599	1599	1599	1599
60	20	80	100	50	1	1861	1861	1861	1861
60	20	80	100	50	2	1879	1879	2034	1882
60	20	80	100	50	3	1672	1672	1813	1632
60	20	80	100	50	4	1767	1767	1767	1767
60	20	80	100	50	5	1793	1793	1765	1297
60	20	80	200	0	1	1672	1672	1672	1672
60	20	80	200	0	2	1857	1857	1865	1865
60	20	80	200	0	3	1837	1837	1837	1837
60	20	80	200	0	4	1899	1899	1751	1737
60	20	80	200	0	5	1812	1812	1701	1522
60	20	80	200	25	1	1759	1759	1771	1771
60	20	80	200	25	2	1972	1972	2010	1991
60	20	80	200	25	3	1671	1671	1671	1671
60	20	80	200	25	4	1921	1921	1873	1850
60	20	80	200	25	5	2211	2211	2056	2049
60	20	80	200	50	1	1670	1670	1670	1670
60	20	80	200	50	2	1897	1897	1986	1945
60	20	80	200	50	3	1669	1669	1772	1759
60	20	80	200	50	4	1769	1769	1769	1769
60	20	80	200	50	5	1665	1665	1644	1642
60	20	80	300	0	1	1795	1795	1706	1699
60	20	80	300	0	2	1791	1791	1896	1896
60	20	80	300	0	3	1925	1925	1702	1695
60	20	80	300	0	4	1909	1909	1909	1909
60	20	80	300	0	5	2020	2020	1861	1553
60	20	80	300	25	1	1875	1875	1811	1811
60	20	80	300	25	2	1959	1959	1959	1959
60	20	80	300	25	3	2233	2233	1789	1584
60	20	80	300	25	4	1974	1974	1781	1467
60	20	80	300	25	5	2448	2448	2054	2011
60	20	80	300	50	1	2005	2005	2026	1740
60	20	80	300	50	2	2031	2031	2031	2031
60	20	80	300	50	3	1988	1988	1936	1750

60	20	80	300	50	4	2035	2035	1695	1388
60	20	80	300	50	5	2069	2069	2069	2069
60	20	80	400	0	1	1874	1844	1874	1874
60	20	80	400	0	2	2343	2343	1991	1958
60	20	80	400	0	3	1784	1784	1908	1907
60	20	80	400	0	4	1715	1715	1815	1815
60	20	80	400	0	5	2235	2235	1882	1231
60	20	80	400	25	1	2081	2081	1944	1939
60	20	80	400	25	2	1866	1866	1887	1887
60	20	80	400	25	3	2312	2312	1875	1482
60	20	80	400	25	4	1939	1939	1939	1939
60	20	80	400	25	5	1816	1816	1793	1408
60	20	80	400	50	1	1995	1995	2189	2171
60	20	80	400	50	2	1957	1957	1984	1984
60	20	80	400	50	3	1770	1770	1541	1531
60	20	80	400	50	4	2157	2157	2126	1821
60	20	80	400	50	5	2352	2352	1855	1848
60	20	80	500	0	1	2190	2190	2256	2252
60	20	80	500	0	2	2170	2170	2088	2062
60	20	80	500	0	3	2275	2181	2120	2098
60	20	80	500	0	4	2171	2171	2079	2079
60	20	80	500	0	5	2500	2500	2014	1942
60	20	80	500	25	1	1995	1995	1995	1995
60	20	80	500	25	2	2126	2105	1992	1971
60	20	80	500	25	3	1639	1639	1747	1746
60	20	80	500	25	4	1940	1940	1684	1466
60	20	80	500	25	5	1965	1965	2028	2028
60	20	80	500	50	1	1818	1818	1812	1809
60	20	80	500	50	2	2369	2369	2038	1503
60	20	80	500	50	3	1912	1912	1860	1860
60	20	80	500	50	4	2030	2030	1730	1428
60	20	80	500	50	5	2074	2074	2150	2150
60	20	80	500	50	1	2285	1599	1602	1602
60	30	60	100	0	2	2026	1406	1303	1294
60	30	60	100	0	3	1874	1514	1447	1444
60	30	60	100	0	4	2009	1406	1423	1420
60	30	60	100	0	5	1934	1638	1403	1395
60	30	60	100	0	1	1731	1568	1374	1353
60	30	60	100	25	2	1963	1831	1480	1467
60	30	60	100	25	3	2045	1923	1388	1381
60	30	60	100	25	4	2126	2126	1407	1383
60	30	60	100	25	5	2689	2217	1461	1395
60	30	60	100	25	1	1734	1734	1570	1563
60	30	60	100	50	2	1778	1778	1558	1519
60	30	60	100	50	3	2756	2386	1587	1579
60	30	60	100	50	4	1541	1541	1269	1159
60	30	60	100	50	5	1714	1714	1360	1326
60	30	60	100	50	1	1999	1532	1497	1494
60	30	60	200	0	2	1843	1604	1341	1165
60	30	60	200	0	3	1829	1630	1305	1290
60	30	60	200	0	4	2215	1483	1416	1403
60	30	60	200	0	5	2002	1877	1733	1721
60	30	60	200	0	1	1407	1407	1373	1353
60	30	60	200	25	2	1913	1688	1446	1440
60	30	60	200	25	3	1657	1657	1319	1249
60	30	60	200	25	4	2735	2067	1579	1573
60	30	60	200	25	5	2473	1754	1379	1354
60	30	60	200	25	1	1824	1824	1497	1393
60	30	60	200	50	2	1995	1763	1376	1353
60	30	60	200	50	3	2068	2068	1287	1260
60	30	60	200	50	4	2335	2335	1559	1538
60	30	60	200	50	5	1988	1872	1484	1460
60	30	60	200	50	1	2178	1457	1585	1574
60	30	60	300	0	2	1800	1619	1598	1576
60	30	60	300	0	3	1910	1409	1380	1327
60	30	60	300	0					

60	30	60	300	0	4	2170	1623	1585	1583
60	30	60	300	0	5	1836	1422	1350	917
60	30	60	300	25	1	2236	1825	1296	1295
60	30	60	300	25	2	2456	2093	1400	1327
60	30	60	300	25	3	1832	1665	1268	1250
60	30	60	300	25	4	2744	2188	1586	1542
60	30	60	300	25	5	2414	1867	1348	1326
60	30	60	300	50	1	1496	1496	1515	1498
60	30	60	300	50	2	1190	1190	1432	1430
60	30	60	300	50	3	1541	1541	1274	1263
60	30	60	300	50	4	1713	1713	1259	1245
60	30	60	300	50	5	2653	2241	1559	1544
60	30	60	400	0	1	2030	1469	1434	1415
60	30	60	400	0	2	1187	1155	1370	1320
60	30	60	400	0	3	1993	1576	1448	1439
60	30	60	400	0	4	2220	1621	1550	1537
60	30	60	400	0	5	2053	1495	1347	1335
60	30	60	400	25	1	1476	1476	1583	1499
60	30	60	400	25	2	2233	1964	1635	1616
60	30	60	400	25	3	1826	1647	1766	1762
60	30	60	400	25	4	2122	1807	1471	1471
60	30	60	400	25	5	1793	1793	1374	1365
60	30	60	400	50	1	1396	1396	1442	1347
60	30	60	400	50	2	2009	1971	1499	1492
60	30	60	400	50	3	2414	2323	1398	1150
60	30	60	400	50	4	1300	1300	1295	1290
60	30	60	400	50	5	1355	1355	1303	1054
60	30	60	400	50	1	2085	1487	1343	1334
60	30	60	500	0	2	1948	1629	1595	1595
60	30	60	500	0	3	2202	1589	1588	1568
60	30	60	500	0	4	1732	1511	1471	1301
60	30	60	500	0	5	1814	1381	1250	1249
60	30	60	500	0	1	2815	2134	1623	1588
60	30	60	500	25	2	2733	2170	1544	1500
60	30	60	500	25	3	2318	1597	1186	1168
60	30	60	500	25	4	1623	1623	1532	1363
60	30	60	500	25	5	2476	1875	1404	1390
60	30	60	500	25	1	1231	1231	1326	1272
60	30	60	500	50	2	1426	1426	1365	1270
60	30	60	500	50	3	2328	2218	1409	1036
60	30	60	500	50	4	1655	1655	1543	1536
60	30	60	500	50	5	1263	1263	1308	1301
60	30	60	500	50	1	1577	1577	1656	1644
60	30	70	100	0	2	1982	1892	1639	1611
60	30	70	100	0	3	2222	2068	2043	1791
60	30	70	100	0	4	1997	1978	1890	1857
60	30	70	100	0	5	2284	2185	1903	1878
60	30	70	100	0	1	1532	1532	1476	1457
60	30	70	100	25	2	1959	1959	1682	1671
60	30	70	100	25	3	1832	1832	1648	1630
60	30	70	100	25	4	1848	1848	1712	1687
60	30	70	100	25	5	1739	1739	1624	1600
60	30	70	100	25	1	2412	2412	1964	1939
60	30	70	100	50	2	1849	1849	1765	1765
60	30	70	100	50	3	1908	1908	1764	1753
60	30	70	100	50	4	2237	2237	1805	1791
60	30	70	100	50	5	2125	2125	2095	1676
60	30	70	100	50	1	2081	2023	1786	1781
60	30	70	200	0	2	2573	2398	2023	1985
60	30	70	200	0	3	2241	2175	1749	1705
60	30	70	200	0	4	1997	1783	1837	1662
60	30	70	200	0	5	2190	2079	1841	1819
60	30	70	200	0	1	2037	2037	1848	1687
60	30	70	200	25	2	2163	2007	1707	1449
60	30	70	200	25	3	2044	2044	1745	1551
60	30	70	200	25					

60	30	70	200	25	4	2095	2095	1688	1687
60	30	70	200	25	5	1899	1822	1766	1746
60	30	70	200	50	1	2293	2293	2006	1976
60	30	70	200	50	2	1870	1870	1663	1661
60	30	70	200	50	3	2110	2110	1755	1440
60	30	70	200	50	4	2160	2118	1645	1635
60	30	70	200	50	5	1992	1992	1860	1847
60	30	70	300	0	1	2640	1786	1602	1586
60	30	70	300	0	2	2370	1919	1798	1775
60	30	70	300	0	3	1960	1939	1582	1529
60	30	70	300	0	4	2303	2213	1825	1822
60	30	70	300	0	5	2068	1869	1637	1598
60	30	70	300	25	1	2175	2175	1749	1744
60	30	70	300	25	2	2262	2221	1860	1851
60	30	70	300	25	3	2092	2092	1988	1944
60	30	70	300	25	4	1854	1703	1662	1638
60	30	70	300	25	5	2410	2193	1741	1725
60	30	70	300	50	1	2682	2637	2023	2007
60	30	70	300	50	2	2291	2251	1639	1612
60	30	70	300	50	3	2336	2336	1911	1897
60	30	70	300	50	4	2798	2767	2071	2043
60	30	70	300	50	5	2035	2035	1757	1742
60	30	70	400	0	1	2206	1827	1788	1752
60	30	70	400	0	2	2443	1914	1782	1765
60	30	70	400	0	3	2057	2022	1780	1743
60	30	70	400	0	4	2077	1909	1627	1588
60	30	70	400	0	5	1692	1692	1711	1543
60	30	70	400	25	1	2273	2158	1813	1799
60	30	70	400	25	2	1984	1842	1687	1687
60	30	70	400	25	3	1979	1979	1945	1453
60	30	70	400	25	4	2197	2197	1985	1657
60	30	70	400	25	5	2778	2563	1816	1796
60	30	70	400	50	1	1890	1890	1612	1322
60	30	70	400	50	2	1818	1800	1852	1834
60	30	70	400	50	3	1832	1832	1508	1501
60	30	70	400	50	4	2393	2266	1786	1759
60	30	70	400	50	5	2168	2145	1972	1661
60	30	70	500	0	1	2508	1966	1960	1922
60	30	70	500	0	2	2434	1861	1764	1740
60	30	70	500	0	3	1994	1710	1604	1591
60	30	70	500	0	4	2434	2053	1856	1849
60	30	70	500	0	5	1893	1857	1592	1575
60	30	70	500	25	1	1685	1685	1470	1459
60	30	70	500	25	2	1342	1342	1750	1730
60	30	70	500	25	3	1869	1869	2238	2125
60	30	70	500	25	4	2398	2398	1932	1564
60	30	70	500	25	5	2096	1897	1729	1720
60	30	70	500	50	1	3212	2649	1764	1751
60	30	70	500	50	2	1741	1741	1485	1277
60	30	70	500	50	3	1701	1701	1990	1986
60	30	70	500	50	4	1816	1816	1879	1879
60	30	70	500	50	5	2134	2109	2011	1985
60	30	80	100	0	1	1914	1914	1981	1961
60	30	80	100	0	2	1904	1904	2070	2056
60	30	80	100	0	3	1874	1874	2167	1825
60	30	80	100	0	4	1780	1780	2021	2019
60	30	80	100	0	5	1978	1978	2110	2097
60	30	80	100	25	1	2160	2131	2258	2247
60	30	80	100	25	2	1840	1840	1905	1886
60	30	80	100	25	3	1867	1867	1961	1961
60	30	80	100	25	4	1844	1844	1906	1906
60	30	80	100	25	5	2284	2284	2539	2094
60	30	80	100	50	1	1887	1887	2016	2016
60	30	80	100	50	2	2328	2274	2035	2026
60	30	80	100	50	3	1901	1901	2156	2156

60	30	80	100	50	4	1861	1861	1847	1836
60	30	80	100	50	5	1730	1730	1752	1748
60	30	80	200	0	1	2189	2189	1989	1750
60	30	80	200	0	2	2351	2351	2212	2084
60	30	80	200	0	3	1856	1856	1886	1867
60	30	80	200	0	4	1802	1802	1976	1952
60	30	80	200	0	5	2235	2086	2197	2188
60	30	80	200	25	1	1942	1942	2316	2309
60	30	80	200	25	2	2141	2081	2181	2171
60	30	80	200	25	3	2279	2279	2338	2338
60	30	80	200	25	4	1907	1907	1962	1938
60	30	80	200	25	5	2326	2326	2612	2578
60	30	80	200	25	1	2196	2196	2217	2193
60	30	80	200	50	2	2368	2368	2363	2363
60	30	80	200	50	3	2179	2179	2340	2340
60	30	80	200	50	4	2157	2157	2232	2232
60	30	80	200	50	5	2336	2336	2077	1838
60	30	80	200	50	1	2234	2209	1793	1793
60	30	80	300	0	2	2126	2126	2129	1851
60	30	80	300	0	3	2475	2458	2097	2073
60	30	80	300	0	4	2344	2344	2184	1992
60	30	80	300	0	5	2277	2127	2071	2068
60	30	80	300	0	1	2031	2031	2049	1594
60	30	80	300	25	2	2306	2306	2066	1734
60	30	80	300	25	3	2500	2500	2257	2239
60	30	80	300	25	4	2151	2151	1977	1944
60	30	80	300	25	5	2505	2195	1982	1946
60	30	80	300	25	1	2196	2196	2027	2016
60	30	80	300	50	2	2395	2395	2446	2445
60	30	80	300	50	3	2391	2391	2476	2465
60	30	80	300	50	4	2162	2162	2162	2128
60	30	80	300	50	5	2297	2297	2039	1770
60	30	80	300	50	1	2356	2325	2223	1846
60	30	80	400	0	2	2465	2465	2127	2059
60	30	80	400	0	3	2312	2312	2039	2015
60	30	80	400	0	4	2862	2862	2387	2374
60	30	80	400	0	5	2775	2775	2430	2419
60	30	80	400	0	1	2326	2326	2268	2101
60	30	80	400	25	2	2551	2371	1949	1606
60	30	80	400	25	3	2451	2451	2161	2100
60	30	80	400	25	4	2455	2455	1959	1947
60	30	80	400	25	5	2315	2315	2209	2196
60	30	80	400	25	1	2170	2170	2438	2427
60	30	80	400	50	2	2779	2779	2024	2015
60	30	80	400	50	3	1760	1760	1594	1591
60	30	80	400	50	4	2181	2181	2087	2074
60	30	80	400	50	5	2129	2129	1856	1482
60	30	80	400	50	1	2816	2345	2070	1817
60	30	80	500	0	2	2152	2152	1920	1689
60	30	80	500	0	3	1982	1982	1889	1693
60	30	80	500	0	4	2588	2578	2388	2382
60	30	80	500	0	5	2525	2435	1996	1768
60	30	80	500	0	1	2617	2617	2297	2258
60	30	80	500	25	2	2375	2375	1969	1915
60	30	80	500	25	3	2419	2383	2038	1985
60	30	80	500	25	4	2372	2372	2487	2487
60	30	80	500	25	5	2712	2712	2222	1605
60	30	80	500	25	1	3033	2836	1947	1937
60	30	80	500	50	2	2434	2434	2133	1899
60	30	80	500	50	3	2364	2364	2116	2086
60	30	80	500	50	4	2837	2837	2096	1362
60	30	80	500	50	5	2530	2530	2320	2306
60	30	80	500	50	1	2247	2213	1659	1655
60	40	60	100	0	2	2341	1628	1534	1513
60	40	60	100	0	3	2395	1730	1795	1794
60	40	60	100	0					

60	40	60	100	0	4	2565	1567	1565	1549
60	40	60	100	0	5	2487	1693	1600	1579
60	40	60	100	25	1	3171	2200	1522	1497
60	40	60	100	25	2	2920	2133	1566	1536
60	40	60	100	25	3	2602	2242	1597	1574
60	40	60	100	25	4	2556	1936	1491	1478
60	40	60	100	25	5	2309	1937	1254	1205
60	40	60	100	25	1	2853	2561	1626	1612
60	40	60	100	50	2	1718	1718	1561	1561
60	40	60	100	50	3	2484	2446	1622	1603
60	40	60	100	50	4	3514	2963	1739	1702
60	40	60	100	50	5	1859	1859	1450	1415
60	40	60	100	50	1	2559	1708	1565	1536
60	40	60	200	0	2	2184	1345	1237	1211
60	40	60	200	0	3	1949	1555	1331	1300
60	40	60	200	0	4	2566	1728	1386	1377
60	40	60	200	0	5	2732	1699	1637	1615
60	40	60	200	0	1	2345	1834	1404	1397
60	40	60	200	25	2	2291	2263	1731	1731
60	40	60	200	25	3	3162	2317	1676	1620
60	40	60	200	25	4	3109	1957	1376	1364
60	40	60	200	25	5	2612	2170	1570	1551
60	40	60	200	25	1	2634	2431	1744	1707
60	40	60	200	50	2	2160	2034	1700	1672
60	40	60	200	50	3	2088	1910	1709	1671
60	40	60	200	50	4	1955	1955	1536	1520
60	40	60	200	50	5	2005	1935	1493	1475
60	40	60	200	50	1	2520	1837	1762	1758
60	40	60	300	0	2	2608	1729	1529	1449
60	40	60	300	0	3	2371	1733	1597	1583
60	40	60	300	0	4	2655	1709	1707	1705
60	40	60	300	0	5	2992	1605	1601	1588
60	40	60	300	0	1	2858	2400	1623	1613
60	40	60	300	25	2	1911	1911	1869	1860
60	40	60	300	25	3	3744	2305	1324	1309
60	40	60	300	25	4	3262	2293	1698	1690
60	40	60	300	25	5	1414	1414	1484	1481
60	40	60	300	25	1	1912	1869	1640	1582
60	40	60	300	50	2	2732	2183	1490	1466
60	40	60	300	50	3	2414	2222	1485	1483
60	40	60	300	50	4	2116	1782	1293	1274
60	40	60	300	50	5	3076	2559	1672	1606
60	40	60	300	50	1	2571	1503	1550	1543
60	40	60	400	0	2	2300	1560	1541	1537
60	40	60	400	0	3	2574	1715	1718	1711
60	40	60	400	0	4	2385	1783	1349	1345
60	40	60	400	0	5	2485	1746	1762	1739
60	40	60	400	0	1	3408	2284	1647	1642
60	40	60	400	25	2	1575	1543	1537	1536
60	40	60	400	25	3	2644	1926	1450	1420
60	40	60	400	25	4	2070	1912	1540	1524
60	40	60	400	25	5	2899	2254	1711	1696
60	40	60	400	25	1	1986	1986	1697	1685
60	40	60	400	50	2	1712	1712	1655	1650
60	40	60	400	50	3	2124	1964	1675	1662
60	40	60	400	50	4	1814	1814	1707	1685
60	40	60	400	50	5	2332	2010	1745	1716
60	40	60	400	50	1	1979	1703	1740	1615
60	40	60	500	0	2	2510	1683	1586	1570
60	40	60	500	0	3	2815	1849	1730	1704
60	40	60	500	0	4	2657	1598	1597	1561
60	40	60	500	0	5	2599	1617	1616	1605
60	40	60	500	0	1	1599	1599	1576	1572
60	40	60	500	25	2	2837	1977	1391	1342
60	40	60	500	25	3	2952	1954	1374	1365
60	40	60	500	25					

60	40	60	500	25	4	1825	1825	1754	1725
60	40	60	500	25	5	3354	2131	1331	1324
60	40	60	500	50	1	1728	1728	1279	1276
60	40	60	500	50	2	1466	1466	1365	1360
60	40	60	500	50	3	2536	2405	1591	1573
60	40	60	500	50	4	1973	1852	1695	1687
60	40	60	500	50	5	1781	1763	1609	1559
60	40	60	500	50	1	1934	1934	2127	2107
60	40	70	100	0	2	2294	1931	1930	1904
60	40	70	100	0	3	2290	2246	1818	1813
60	40	70	100	0	4	2678	2323	1955	1921
60	40	70	100	0	5	2262	1830	1617	1612
60	40	70	100	0	1	2760	2409	1835	1834
60	40	70	100	25	2	2247	2173	1719	1702
60	40	70	100	25	3	2359	2218	1857	1552
60	40	70	100	25	4	2089	2089	1904	1849
60	40	70	100	25	5	2447	2125	1897	1876
60	40	70	100	25	1	1885	1885	2045	2029
60	40	70	100	50	2	2345	2345	2010	2003
60	40	70	100	50	3	2514	2514	1727	1709
60	40	70	100	50	4	1816	1816	1910	1900
60	40	70	100	50	5	2477	2328	1938	1911
60	40	70	100	50	1	2705	2291	1976	1952
60	40	70	200	0	2	2322	2167	1941	1928
60	40	70	200	0	3	2495	1998	1871	1849
60	40	70	200	0	4	2554	2554	2145	2104
60	40	70	200	0	5	2641	2259	2188	2184
60	40	70	200	0	1	2557	2518	2103	2085
60	40	70	200	25	2	2440	2328	1754	1496
60	40	70	200	25	3	2521	2320	1931	1926
60	40	70	200	25	4	2024	2024	1705	1666
60	40	70	200	25	5	2203	2203	1834	1812
60	40	70	200	25	1	2210	2210	2116	2107
60	40	70	200	50	2	2683	2683	2118	2072
60	40	70	200	50	3	2134	2134	2080	2039
60	40	70	200	50	4	2280	2280	1896	1883
60	40	70	200	50	5	2431	2431	1967	1960
60	40	70	200	50	1	2403	1974	1790	1774
60	40	70	300	0	2	3089	2221	1945	1888
60	40	70	300	0	3	2414	2204	2101	2076
60	40	70	300	0	4	2662	1993	1794	1792
60	40	70	300	0	5	1894	1894	1687	1681
60	40	70	300	0	1	2718	2360	1893	1892
60	40	70	300	25	2	1872	1872	1830	1820
60	40	70	300	25	3	2163	2113	1855	1845
60	40	70	300	25	4	2146	2146	1792	1742
60	40	70	300	25	5	2665	2654	1889	1887
60	40	70	300	25	1	2146	2146	1630	1630
60	40	70	300	50	2	2892	2656	1940	1912
60	40	70	300	50	3	2419	2372	1761	1731
60	40	70	300	50	4	2270	2270	2167	2131
60	40	70	300	50	5	1669	1669	1876	1808
60	40	70	300	50	1	2587	2380	2066	2062
60	40	70	400	0	2	2557	1989	1939	1928
60	40	70	400	0	3	2943	2049	1977	1927
60	40	70	400	0	4	2593	1832	1591	1584
60	40	70	400	0	5	2584	2001	1939	1932
60	40	70	400	0	1	2904	2904	1981	1910
60	40	70	400	25	2	1961	1961	1978	1698
60	40	70	400	25	3	3464	2485	1726	1726
60	40	70	400	25	4	1907	1907	1891	1882
60	40	70	400	25	5	2484	2210	1787	1739
60	40	70	400	25	1	2556	2556	2014	1981
60	40	70	400	50	2	2133	2133	1866	1836
60	40	70	400	50	3	2372	2086	2003	1910
60	40	70	400	50					

60	40	70	400	50	4	1901	1901	1764	1750
60	40	70	400	50	5	2650	2519	1723	1705
60	40	70	500	0	1	1902	1902	1943	1916
60	40	70	500	0	2	3201	2017	1886	1835
60	40	70	500	0	3	3098	2321	1982	1979
60	40	70	500	0	4	2942	2035	2039	1966
60	40	70	500	0	5	1644	1644	1679	1668
60	40	70	500	25	1	3047	2436	2022	2022
60	40	70	500	25	2	2292	2104	1781	1770
60	40	70	500	25	3	2551	2431	2198	2137
60	40	70	500	25	4	2587	2208	2061	2058
60	40	70	500	25	5	2495	2156	1847	1825
60	40	70	500	50	1	2039	2039	2061	1428
60	40	70	500	50	2	1813	1813	1766	1760
60	40	70	500	50	3	2621	2563	1981	1976
60	40	70	500	50	4	2802	2699	1962	1962
60	40	70	500	50	5	1845	1845	1911	1755
60	40	80	100	0	1	2280	2042	2110	2090
60	40	80	100	0	2	2578	2177	2126	2118
60	40	80	100	0	3	2169	2169	2164	2136
60	40	80	100	0	4	2056	2056	2300	2139
60	40	80	100	0	5	2040	1993	2044	2044
60	40	80	100	25	1	2708	2423	2252	2252
60	40	80	100	25	2	2323	2323	2108	2102
60	40	80	100	25	3	2269	2269	2409	2395
60	40	80	100	25	4	2097	2097	2653	2653
60	40	80	100	25	5	2073	2073	2199	2171
60	40	80	100	50	1	2160	2160	2010	1973
60	40	80	100	50	2	2252	2252	2115	2061
60	40	80	100	50	3	2315	2315	2264	2259
60	40	80	100	50	4	1956	1956	1963	1963
60	40	80	100	50	5	2101	2101	2421	2397
60	40	80	200	0	1	2981	2487	2176	2164
60	40	80	200	0	2	2201	2201	2122	1923
60	40	80	200	0	3	2217	2217	1928	1914
60	40	80	200	0	4	2086	2082	2031	2026
60	40	80	200	0	5	2119	2119	2037	2021
60	40	80	200	25	1	2685	2685	2473	2473
60	40	80	200	25	2	2309	2309	2047	2040
60	40	80	200	25	3	2477	2477	2201	2192
60	40	80	200	25	4	2497	2497	2333	2296
60	40	80	200	25	5	2390	2390	2388	2386
60	40	80	200	50	1	2465	2465	1961	1957
60	40	80	200	50	2	2250	2250	2602	2345
60	40	80	200	50	3	2260	2260	2026	2026
60	40	80	200	50	4	2529	2529	2405	2392
60	40	80	200	50	5	2482	2482	2143	2109
60	40	80	300	0	1	2429	2429	2342	2340
60	40	80	300	0	2	2909	2560	2357	2110
60	40	80	300	0	3	2456	2456	2233	2233
60	40	80	300	0	4	2541	2404	2073	2055
60	40	80	300	0	5	2274	2274	1806	1796
60	40	80	300	25	1	2911	2911	2238	2230
60	40	80	300	25	2	2768	2731	2067	2044
60	40	80	300	25	3	2556	2556	2288	1967
60	40	80	300	25	4	2511	2511	2362	2130
60	40	80	300	25	5	2596	2596	2352	2351
60	40	80	300	50	1	2702	2702	2004	1992
60	40	80	300	50	2	2642	2642	2113	2105
60	40	80	300	50	3	3001	3001	2179	2175
60	40	80	300	50	4	2710	2710	2187	2138
60	40	80	300	50	5	2258	2258	1980	1931
60	40	80	400	0	1	2618	2421	2300	2289
60	40	80	400	0	2	2881	2785	2343	2323
60	40	80	400	0	3	2576	2576	2198	2173

60	40	80	400	0	4	2690	2690	2183	2115
60	40	80	400	0	5	2388	2315	2118	2118
60	40	80	400	25	1	3042	3042	2016	2011
60	40	80	400	25	2	2413	2413	2396	2374
60	40	80	400	25	3	2572	2535	2111	1909
60	40	80	400	25	4	2575	2575	2334	2025
60	40	80	400	25	5	2999	2999	2043	2015
60	40	80	400	25	1	2604	2604	2048	1699
60	40	80	400	50	2	2449	2449	2297	2257
60	40	80	400	50	3	2458	2428	2103	2103
60	40	80	400	50	4	2700	2447	1983	1654
60	40	80	400	50	5	2973	2973	2550	2543
60	40	80	400	50	1	3243	2698	2189	2158
60	40	80	500	0	2	2211	2108	2309	2300
60	40	80	500	0	3	3120	2797	2008	1993
60	40	80	500	0	4	2944	2369	2109	2083
60	40	80	500	0	5	2979	2818	2221	2218
60	40	80	500	0	1	2623	2623	2135	2094
60	40	80	500	25	2	2041	2041	1951	1888
60	40	80	500	25	3	1940	1940	2389	2372
60	40	80	500	25	4	2523	2523	2536	2536
60	40	80	500	25	5	2032	2032	2117	2084
60	40	80	500	25	1	2916	2823	2287	2223
60	40	80	500	50	2	2298	2298	1812	1807
60	40	80	500	50	3	2530	2530	2180	2142
60	40	80	500	50	4	2928	2928	2136	2126
60	40	80	500	50	5	1949	1949	2038	1555
60	40	80	500	50					

NUMBER OF NODES 60 NUMBER OF OBSERVATIONS 675

	LOAD BASED	G(A)	G(B)	G(B)/ARCOPT
AVG SOLUTION LENGTH	2104	1958	1772	1705
NO OF BETTER SOLUTIONS		298	512	542
% IMPROVEMENT OVER LOAD BASED		7	16	19

% OF BACKHAULS IS 20 NUMBER OF OBSERVATIONS 225

	LOAD BASED	G(A)	G(B)	G(B)/ARCOPT
AVG SOLUTION LENGTH	1753	1713	1643	1555
NO OF BETTER SOLUTIONS		63	129	147
% IMPROVEMENT OVER LOAD BASED		2	6	11

% OF BACKHAULS IS 30 NUMBER OF OBSERVATIONS 225

	LOAD BASED	G(A)	G(B)	G(B)/ARCOPT
AVG SOLUTION LENGTH	2114	1982	1783	1711
NO OF BETTER SOLUTIONS		107	181	189
% IMPROVEMENT OVER LOAD BASED		6	16	19

% OF BACKHAULS IS 40 NUMBER OF OBSERVATIONS 225

	LOAD BASED	G(A)	G(B)	G(B)/ARCOPT
AVG SOLUTION LENGTH	2443	2181	1890	1850
NO OF BETTER SOLUTIONS		128	202	206
% IMPROVEMENT OVER LOAD BASED		11	23	24

% OF CONCENTRATION IS 60 NUMBER OF OBSERVATIONS 225

	LOAD BASED	G(A)	G(B)	G(B)/ARCOPT
AVG SOLUTION LENGTH	1971	1677	1424	1372
NO OF BETTER SOLUTIONS		162	205	211
% IMPROVEMENT OVER LOAD BASED		15	28	30

% OF CONCENTRATION IS 70 NUMBER OF OBSERVATIONS 225

	LOAD BASED	G(A)	G(B)	G(B)/ARCOPT
AVG SOLUTION LENGTH	2112	1999	1831	1762
NO OF BETTER SOLUTIONS		97	166	180
% IMPROVEMENT OVER LOAD BASED		5	13	17

% OF CONCENTRATION IS 80 NUMBER OF OBSERVATIONS 225

	LOAD BASED	G(A)	G(B)	G(B)/ARCOPT
AVG SOLUTION LENGTH	2228	2199	2061	1982
NO OF BETTER SOLUTIONS		39	141	151
% IMPROVEMENT OVER LOAD BASED		1	7	11

BACKHAUL PENALTY IS 100 NUMBER OF OBSERVATIONS 135

	LOAD BASED	G(A)	G(B)	G(B)/ARCOPT
AVG SOLUTION LENGTH	1978	1863	1736	1689
NO OF BETTER SOLUTIONS		50	87	93
% IMPROVEMENT OVER LOAD BASED		6	12	15

BACKHAUL PENALTY IS 200 NUMBER OF OBSERVATIONS 135

	LOAD BASED	G(A)	G(B)	G(B)/ARCOPT
AVG SOLUTION LENGTH	2081	1964	1777	1724
NO OF BETTER SOLUTIONS		55	105	106
% IMPROVEMENT OVER LOAD BASED		6	15	17

BACKHAUL PENALTY IS 300 NUMBER OF OBSERVATIONS 135

	LOAD BASED	G(A)	G(B)	G(B)/ARCOPT
AVG SOLUTION LENGTH	2185	2019	1777	1712
NO OF BETTER SOLUTIONS		64	110	93
% IMPROVEMENT OVER LOAD BASED		8	19	22

BACKHAUL PENALTY IS 400 NUMBER OF OBSERVATIONS 135

	LOAD BASED	G(A)	G(B)	G(B)/ARCOPT
AVG SOLUTION LENGTH	2137	1986	1779	1694
NO OF BETTER SOLUTIONS		64	111	117
% IMPROVEMENT OVER LOAD BASED		7	17	21

BACKHAUL PENALTY IS 500 NUMBER OF OBSERVATIONS 135

	LOAD BASED	G(A)	G(B)	G(B)/ARCOPT
AVG SOLUTION LENGTH	2137	1960	1792	1707
NO OF BETTER SOLUTIONS		65	99	109
% IMPROVEMENT OVER LOAD BASED		8	16	20

RADIAL SURCHARGE IS 0 NUMBER OF OBSERVATIONS 225

	LOAD BASED	G(A)	G(B)	G(B)/ARCOPT
AVG SOLUTION LENGTH	2126	1871	1768	1719
NO OF BETTER SOLUTIONS		149	177	184
% IMPROVEMENT OVER LOAD BASED		12	17	19

RADIAL SURCHARGE IS 25 NUMBER OF OBSERVATIONS 225

	LOAD BASED	G(A)	G(B)	G(B)/ARCOPT
AVG SOLUTION LENGTH	2140	1997	1762	1696
NO OF BETTER SOLUTIONS		99	176	183
% IMPROVEMENT OVER LOAD BASED		7	18	21

RADIAL SURCHARGE IS 50 NUMBER OF OBSERVATIONS 225

	LOAD BASED	G(A)	G(B)	G(B)/ARCOPT
AVG SOLUTION LENGTH	2045	2007	1786	1701
NO OF BETTER SOLUTIONS		50	159	175
% IMPROVEMENT OVER LOAD BASED		2	13	17

NO. OF NODES	% OF BACKHAULS	% OF CONCENTR	BKHAUL PENALTY	RADIAL SURCHARGE	PROBL NO.	LOAD BASED SOL	G(A) SOL	G(B) SOL	G(B)/ARC SOL
80	20	60	100	0	1	1704	1370	1282	1250
80	20	60	100	0	2	2057	1210	1207	1169
80	20	60	100	0	3	1568	1335	1307	1292
80	20	60	100	0	4	2064	1792	1481	1476
80	20	60	100	0	5	1814	1300	1276	1268
80	20	60	100	25	1	1749	1749	1265	1253
80	20	60	100	25	2	1787	1699	1377	1362
80	20	60	100	25	3	2240	1851	1281	1256
80	20	60	100	25	4	1696	1696	1509	1485
80	20	60	100	25	5	1522	1522	1380	1367
80	20	60	100	50	1	2068	1939	1537	1528
80	20	60	100	50	2	1635	1448	1102	1097
80	20	60	100	50	3	1863	1863	1520	1508
80	20	60	100	50	4	1745	1745	1503	1467
80	20	60	100	50	5	1543	1543	1369	1358
80	20	60	200	0	1	1887	1490	1333	1306
80	20	60	200	0	2	1788	1483	1324	1308
80	20	60	200	0	3	2063	1519	1366	1349
80	20	60	200	0	4	1645	1376	1406	1390
80	20	60	200	0	5	1902	1615	1589	1573
80	20	60	200	25	1	1675	1664	1521	1496
80	20	60	200	25	2	1944	1723	1464	1450
80	20	60	200	25	3	2166	1760	1401	1387
80	20	60	200	25	4	2074	1652	1339	1330
80	20	60	200	25	5	1469	1469	1382	1351
80	20	60	200	50	1	1822	1822	1488	1481
80	20	60	200	50	2	2311	2140	1536	1500
80	20	60	200	50	3	2639	2533	1773	1754
80	20	60	200	50	4	1603	1548	1407	1374
80	20	60	200	50	5	2632	2486	1608	1595
80	20	60	300	0	1	1995	1489	1489	1484
80	20	60	300	0	2	1911	1533	1530	1505
80	20	60	300	0	3	1974	1576	1544	1531
80	20	60	300	0	4	1873	1450	1281	1247
80	20	60	300	0	5	1415	1415	1519	1499
80	20	60	300	25	1	2341	1977	1412	1398
80	20	60	300	25	2	1667	1431	1096	1081
80	20	60	300	25	3	2553	2041	1495	1454
80	20	60	300	25	4	2668	2198	1648	1630
80	20	60	300	25	5	1929	1841	1449	1396
80	20	60	300	50	1	1691	1662	1476	1459
80	20	60	300	50	2	1203	1203	1218	1195
80	20	60	300	50	3	1890	1890	1684	1612
80	20	60	300	50	4	1368	1368	1497	1483
80	20	60	300	50	5	1793	1793	1592	1565
80	20	60	400	0	1	1650	1251	1196	1190
80	20	60	400	0	2	1759	1394	1396	1384
80	20	60	400	0	3	1923	1304	1302	1300
80	20	60	400	0	4	2054	1422	1418	1406
80	20	60	400	0	5	1973	1623	1454	1420
80	20	60	400	25	1	2341	1942	1525	1508
80	20	60	400	25	2	1575	1516	1327	1322
80	20	60	400	25	3	1706	1706	1421	1386
80	20	60	400	25	4	1488	1478	1428	1405
80	20	60	400	25	5	2320	1860	1463	1376
80	20	60	400	50	1	1712	1712	1396	1374
80	20	60	400	50	2	1818	1747	1532	1507
80	20	60	400	50	3	2061	1896	1459	1447
80	20	60	400	50	4	1443	1443	1319	1309
80	20	60	400	50	5	1457	1457	1390	1355
80	20	60	500	0	1	1802	1586	1557	1547
80	20	60	500	0	2	1768	1637	1569	1553
80	20	60	500	0	3	1466	1446	1456	1441

80	20	60	500	0	4	2173	1498	1372	1359
80	20	60	500	0	5	2194	1682	1725	1717
80	20	60	500	25	1	2484	1858	1314	1301
80	20	60	500	25	2	1855	1775	1440	1407
80	20	60	500	25	3	2017	1790	1406	1392
80	20	60	500	25	4	2552	1737	1192	1137
80	20	60	500	25	5	1625	1625	1653	1640
80	20	60	500	50	1	2106	1932	1353	1340
80	20	60	500	50	2	2658	2290	1383	1345
80	20	60	500	50	3	1232	1232	1458	1328
80	20	60	500	50	4	1547	1547	1498	1473
80	20	60	500	50	5	1609	1609	1476	1446
80	20	60	500	50	1	2359	2359	2224	2224
80	20	70	100	0	2	1919	1830	1869	1818
80	20	70	100	0	3	1862	1862	2155	2098
80	20	70	100	0	4	2294	2294	2412	2385
80	20	70	100	0	5	2195	2182	2155	2152
80	20	70	100	0	1	2174	2110	1846	1818
80	20	70	100	25	2	2037	2037	2105	1967
80	20	70	100	25	3	2139	2139	2157	2076
80	20	70	100	25	4	1874	1874	2058	2004
80	20	70	100	25	5	1936	1936	2267	2183
80	20	70	100	25	1	2465	2465	2432	2340
80	20	70	100	50	2	2108	2108	2132	2055
80	20	70	100	50	3	2044	2044	2055	1992
80	20	70	100	50	4	2161	2161	2451	2432
80	20	70	100	50	5	1672	1672	1931	1829
80	20	70	100	50	1	2315	2315	2737	2687
80	20	70	200	0	2	2330	2330	2389	2343
80	20	70	200	0	3	1986	1986	2102	2031
80	20	70	200	0	4	2556	2552	2367	2364
80	20	70	200	0	5	2044	1955	2159	2135
80	20	70	200	0	1	2261	2228	2085	2077
80	20	70	200	25	2	2099	2034	1967	1933
80	20	70	200	25	3	2083	2083	2061	2039
80	20	70	200	25	4	1871	1871	2134	2112
80	20	70	200	25	5	1848	1848	2100	2053
80	20	70	200	25	1	2394	2394	2304	2048
80	20	70	200	50	2	2312	2312	2526	2338
80	20	70	200	50	3	2299	2273	2309	2309
80	20	70	200	50	4	1951	1951	2287	2224
80	20	70	200	50	5	2258	2258	2064	2039
80	20	70	200	50	1	1920	1920	2129	2085
80	20	70	300	0	2	2259	2259	2544	2421
80	20	70	300	0	3	2247	2247	2404	2391
80	20	70	300	0	4	1855	1855	2199	2112
80	20	70	300	0	5	2463	2463	2423	2379
80	20	70	300	0	1	1975	1975	2193	2139
80	20	70	300	25	2	1873	1873	1912	1828
80	20	70	300	25	3	2042	2042	2446	2404
80	20	70	300	25	4	2027	2027	2238	2222
80	20	70	300	25	5	2035	2035	2040	2008
80	20	70	300	25	1	2189	2189	2122	2068
80	20	70	300	50	2	1859	1859	1896	1811
80	20	70	300	50	3	2133	2133	2324	2262
80	20	70	300	50	4	1806	1806	2017	1953
80	20	70	300	50	5	2220	2220	2240	2200
80	20	70	300	50	1	2371	2182	2500	2411
80	20	70	400	0	2	2179	2017	1847	1798
80	20	70	400	0	3	2209	2209	2453	2368
80	20	70	400	0	4	2074	2074	2214	2125
80	20	70	400	0	5	2086	2086	1995	1942
80	20	70	400	0	1	1867	1867	2202	2135
80	20	70	400	25	2	1812	1812	2255	2224
80	20	70	400	25	3	2284	2284	2506	2472
80	20	70	400	25					

80	20	70	400	25	4	2219	2088	2010	1949
80	20	70	400	25	5	2426	2426	2448	2340
80	20	70	400	50	1	1872	1872	2127	2095
80	20	70	400	50	2	1935	1935	2259	2218
80	20	70	400	50	3	1861	1861	1980	1954
80	20	70	400	50	4	1966	1966	2163	2083
80	20	70	400	50	5	1969	1969	2088	2006
80	20	70	500	0	1	2271	2102	1896	1796
80	20	70	500	0	2	1874	1874	2185	2161
80	20	70	500	0	3	1974	1974	2285	2222
80	20	70	500	0	4	2243	2183	2481	2425
80	20	70	500	0	5	1958	1958	2192	2171
80	20	70	500	25	1	1628	1628	2134	2100
80	20	70	500	25	2	2013	2013	2238	2185
80	20	70	500	25	3	1921	1921	2118	2026
80	20	70	500	25	4	2095	2095	2427	2407
80	20	70	500	25	5	1867	1867	2132	2108
80	20	70	500	50	1	1918	1918	2144	2105
80	20	70	500	50	2	1903	1903	2060	2019
80	20	70	500	50	3	1919	1919	2125	2028
80	20	70	500	50	4	1786	1786	1985	1973
80	20	70	500	50	5	1908	1908	2015	2015
80	20	70	500	50	1	3927	3927	4341	4255
80	20	80	100	0	2	3729	3729	3971	3932
80	20	80	100	0	3	3136	3136	3354	3335
80	20	80	100	0	4	4059	4059	4293	4287
80	20	80	100	0	5	4032	4032	4345	4305
80	20	80	100	0	1	3531	3531	3881	3806
80	20	80	100	25	2	4137	4137	4641	4603
80	20	80	100	25	3	3921	3921	4262	4237
80	20	80	100	25	4	4507	4507	4740	4739
80	20	80	100	25	5	4404	4404	4766	4739
80	20	80	100	25	1	4644	4644	5063	5059
80	20	80	100	50	2	3404	3404	3669	3648
80	20	80	100	50	3	4578	4578	4819	4788
80	20	80	100	50	4	2979	2979	3108	3020
80	20	80	100	50	5	3981	3981	4334	4320
80	20	80	100	50	1	3053	3053	3302	3270
80	20	80	200	0	2	3748	3748	4100	4021
80	20	80	200	0	3	5465	5465	5679	5672
80	20	80	200	0	4	3685	3685	4376	4307
80	20	80	200	0	5	4014	4014	4029	4029
80	20	80	200	0	1	2784	2784	3060	3050
80	20	80	200	25	2	3085	3085	3412	3392
80	20	80	200	25	3	3127	3127	3392	3369
80	20	80	200	25	4	2938	2938	3034	3022
80	20	80	200	25	5	3620	3620	3859	3844
80	20	80	200	25	1	3899	3899	4380	4346
80	20	80	200	50	2	5211	5211	5866	5824
80	20	80	200	50	3	3479	3479	3738	3689
80	20	80	200	50	4	3143	3143	3490	3431
80	20	80	200	50	5	3808	3808	4050	4038
80	20	80	200	50	1	4460	4460	4904	4853
80	20	80	300	0	2	3412	3412	3678	3618
80	20	80	300	0	3	3254	3254	3519	3508
80	20	80	300	0	4	4405	4405	4841	4788
80	20	80	300	0	5	2925	2925	3168	3107
80	20	80	300	0	1	3767	3767	3955	3931
80	20	80	300	25	2	3091	3091	3091	3091
80	20	80	300	25	3	3373	3373	3552	3458
80	20	80	300	25	4	3912	3912	4437	4418
80	20	80	300	25	5	3518	3518	4028	3978
80	20	80	300	25	1	3094	3094	3326	3298
80	20	80	300	50	2	3253	3253	3605	3580
80	20	80	300	50	3	4698	4698	4903	4885
80	20	80	300	50					

80	20	80	300	50	4	2616	2616	2908	2836
80	20	80	300	50	5	3463	3463	3587	3559
80	20	80	400	0	1	3538	3523	3538	3538
80	20	80	400	0	2	2410	2410	2522	2456
80	20	80	400	0	3	3854	3854	4049	3831
80	20	80	400	0	4	2650	2650	3106	3086
80	20	80	400	0	5	4571	4571	4709	4667
80	20	80	400	25	1	4516	4516	4750	4743
80	20	80	400	25	2	3822	3822	4064	4042
80	20	80	400	25	3	3070	3070	3141	3113
80	20	80	400	25	4	3145	3145	3145	3145
80	20	80	400	25	5	4013	4013	4496	4487
80	20	80	400	50	1	3680	3680	4099	4051
80	20	80	400	50	2	4038	4038	4182	4151
80	20	80	400	50	3	3866	3866	4021	3984
80	20	80	400	50	4	3561	3561	3992	3959
80	20	80	400	50	5	3437	3437	3573	3544
80	20	80	500	0	1	4313	4313	4331	4331
80	20	80	500	0	2	3909	3909	4101	4076
80	20	80	500	0	3	2874	2874	2978	2952
80	20	80	500	0	4	4263	4263	4494	4410
80	20	80	500	0	5	3812	3812	4040	4032
80	20	80	500	25	1	4070	4070	4385	4371
80	20	80	500	25	2	3447	3447	3766	3573
80	20	80	500	25	3	3112	3112	3463	3442
80	20	80	500	25	4	3755	3755	4195	4125
80	20	80	500	25	5	3676	3676	3743	3732
80	20	80	500	50	1	4331	4331	4331	4331
80	20	80	500	50	2	4005	4005	4200	4095
80	20	80	500	50	3	4052	4052	4237	4213
80	20	80	500	50	4	4066	4066	4535	4455
80	20	80	500	50	5	4044	4044	4578	4560
80	20	80	500	50	1	2378	1977	1858	1816
80	30	60	100	0	2	2849	1949	1907	1871
80	30	60	100	0	3	2764	1820	1605	1511
80	30	60	100	0	4	2399	1628	1514	1492
80	30	60	100	0	5	2584	1641	1551	1487
80	30	60	100	0	1	2908	2161	1653	1622
80	30	60	100	25	2	2452	2074	1632	1603
80	30	60	100	25	3	1748	1748	1606	1545
80	30	60	100	25	4	2557	2432	1742	1723
80	30	60	100	25	5	3059	2194	1606	1595
80	30	60	100	25	1	1835	1835	1722	1622
80	30	60	100	50	2	2619	2619	1913	1885
80	30	60	100	50	3	3187	2752	1742	1731
80	30	60	100	50	4	1739	1739	1758	1714
80	30	60	100	50	5	1730	1730	1641	1595
80	30	60	100	50	1	2975	1780	1702	1670
80	30	60	200	0	2	1734	1734	1831	1763
80	30	60	200	0	3	2728	1658	1646	1633
80	30	60	200	0	4	2314	1826	1527	1494
80	30	60	200	0	5	2829	1876	1566	1532
80	30	60	200	0	1	3035	2323	1730	1706
80	30	60	200	25	2	2570	2042	1509	1494
80	30	60	200	25	3	2242	2102	1578	1559
80	30	60	200	25	4	3428	2491	1707	1662
80	30	60	200	25	5	2822	2181	1701	1684
80	30	60	200	25	1	2137	2137	1602	1559
80	30	60	200	50	2	2611	2413	1790	1733
80	30	60	200	50	3	1791	1791	1693	1665
80	30	60	200	50	4	2443	2247	1588	1579
80	30	60	200	50	5	2742	2529	1849	1839
80	30	60	200	50	1	2631	1448	1468	1368
80	30	60	300	0	2	2435	1897	1868	1836
80	30	60	300	0	3	2960	1931	1926	1925
80	30	60	300	0					

80	30	60	300	0	4	2464	1680	1582	1572
80	30	60	300	0	5	2385	1626	1557	1531
80	30	60	300	25	1	2182	2101	1749	1737
80	30	60	300	25	2	2045	1823	1539	1539
80	30	60	300	25	3	2428	1930	1767	1749
80	30	60	300	25	4	2696	2129	2097	2067
80	30	60	300	25	5	1819	1728	1574	1550
80	30	60	300	50	1	1856	1856	1770	1760
80	30	60	300	50	2	1660	1660	1481	1471
80	30	60	300	50	3	1511	1511	1683	1664
80	30	60	300	50	4	1572	1572	1723	1696
80	30	60	300	50	5	3191	2722	1783	1743
80	30	60	400	0	1	2804	1462	1503	1488
80	30	60	400	0	2	2610	1724	1595	1585
80	30	60	400	0	3	2716	1841	1657	1652
80	30	60	400	0	4	2836	1797	1638	1569
80	30	60	400	0	5	2992	1813	1811	1784
80	30	60	400	25	1	2173	2173	2025	1977
80	30	60	400	25	2	1874	1761	1750	1630
80	30	60	400	25	3	2230	1993	1591	1546
80	30	60	400	25	4	1705	1705	1536	1522
80	30	60	400	25	5	1818	1703	1738	1693
80	30	60	400	50	1	1690	1690	1455	1428
80	30	60	400	50	2	2038	1943	1742	1733
80	30	60	400	50	3	1573	1573	1816	1737
80	30	60	400	50	4	1657	1657	1768	1731
80	30	60	400	50	5	2765	2131	1536	1453
80	30	60	500	0	1	2704	1718	1697	1680
80	30	60	500	0	2	2500	1691	1670	1644
80	30	60	500	0	3	2523	1636	1458	1449
80	30	60	500	0	4	2762	1894	1893	1864
80	30	60	500	0	5	2781	1918	1893	1881
80	30	60	500	25	1	1604	1604	1697	1689
80	30	60	500	25	2	2675	2088	1734	1726
80	30	60	500	25	3	1982	1916	1659	1642
80	30	60	500	25	4	1738	1738	1649	1521
80	30	60	500	25	5	2314	1913	1664	1621
80	30	60	500	50	1	2646	2033	1506	1486
80	30	60	500	50	2	1939	1897	1791	1748
80	30	60	500	50	3	1642	1642	1773	1748
80	30	60	500	50	4	1792	1792	1895	1881
80	30	60	500	50	5	1926	1904	1640	1624
80	30	70	100	0	1	2596	2596	2611	2580
80	30	70	100	0	2	2409	2409	2256	2192
80	30	70	100	0	3	2605	2470	2331	2267
80	30	70	100	0	4	1761	1761	1865	1853
80	30	70	100	0	5	2385	2360	2164	2130
80	30	70	100	25	1	2523	2523	2720	2605
80	30	70	100	25	2	2444	2349	2342	2307
80	30	70	100	25	3	2263	2263	2496	2417
80	30	70	100	25	4	2082	2082	2344	2289
80	30	70	100	25	5	1936	1936	1974	1958
80	30	70	100	50	1	2078	2078	2056	1993
80	30	70	100	50	2	2414	2414	2523	2432
80	30	70	100	50	3	2922	2922	3017	2980
80	30	70	100	50	4	2223	2223	2580	2541
80	30	70	100	50	5	2030	2030	2265	2164
80	30	70	200	0	1	2416	2416	2333	2285
80	30	70	200	0	2	2354	2316	2466	2458
80	30	70	200	0	3	2745	2406	2407	2383
80	30	70	200	0	4	2117	2117	2613	2558
80	30	70	200	0	5	2301	2244	2570	2505
80	30	70	200	25	1	2415	2415	3014	2953
80	30	70	200	25	2	2588	2588	2618	2592
80	30	70	200	25	3	2445	2308	2204	2150

80	30	70	200	25	4	2225	2225	2520	2471
80	30	70	200	25	5	2126	2083	2173	2085
80	30	70	200	50	1	2733	2733	3168	3139
80	30	70	200	50	2	2363	2363	2573	2474
80	30	70	200	50	3	2219	2219	2271	2239
80	30	70	200	50	4	2433	2433	2514	2393
80	30	70	200	50	5	2140	2140	2310	2243
80	30	70	300	0	1	2631	2479	2627	2597
80	30	70	300	0	2	2570	2560	2799	2570
80	30	70	300	0	3	2387	2387	2463	2463
80	30	70	300	0	4	2199	2199	2408	2390
80	30	70	300	0	5	2187	2187	2495	2449
80	30	70	300	25	1	2226	2226	2298	2262
80	30	70	300	25	2	2481	2481	2456	2322
80	30	70	300	25	3	2111	2111	2296	2275
80	30	70	300	25	4	2261	2259	2625	2592
80	30	70	300	25	5	2516	2516	2559	2511
80	30	70	300	50	1	2462	2317	2179	2179
80	30	70	300	50	2	2470	2414	2328	2252
80	30	70	300	50	3	2257	2257	2540	2504
80	30	70	300	50	4	2343	2343	2598	2533
80	30	70	300	50	5	1981	1981	1982	1914
80	30	70	400	0	1	2337	2337	2561	2509
80	30	70	400	0	2	2216	2216	2551	2527
80	30	70	400	0	3	2216	2216	2464	2376
80	30	70	400	0	4	1980	1980	2043	1980
80	30	70	400	0	5	1995	1995	2117	2103
80	30	70	400	25	1	1968	1968	2186	2169
80	30	70	400	25	2	2073	2073	2436	2427
80	30	70	400	25	3	2303	2303	2254	2214
80	30	70	400	25	4	2045	2045	2437	2426
80	30	70	400	25	5	1813	1813	2326	2277
80	30	70	400	50	1	2148	2148	2450	2383
80	30	70	400	50	2	2010	2010	2370	2314
80	30	70	400	50	3	2052	2052	2350	2304
80	30	70	400	50	4	2210	2210	2067	1980
80	30	70	400	50	5	2169	2169	2086	2066
80	30	70	500	0	1	2931	2542	2600	2519
80	30	70	500	0	2	2472	2472	2798	2749
80	30	70	500	0	3	2377	2344	2343	2341
80	30	70	500	0	4	2202	2202	2660	2587
80	30	70	500	0	5	2178	2178	2294	2288
80	30	70	500	25	1	2035	2035	2828	2768
80	30	70	500	25	2	2275	2275	2455	2425
80	30	70	500	25	3	2132	2132	2557	2526
80	30	70	500	25	4	2559	2415	2289	2264
80	30	70	500	25	5	2212	2212	2582	2527
80	30	70	500	50	1	2230	2230	2317	2276
80	30	70	500	50	2	2135	2135	2574	2535
80	30	70	500	50	3	2017	2017	2054	2029
80	30	70	500	50	4	1894	1894	2275	2203
80	30	70	500	50	5	2263	2263	2243	2216
80	30	80	100	0	1	3693	3638	4123	4110
80	30	80	100	0	2	3332	3332	4023	3958
80	30	80	100	0	3	4398	4398	4645	4588
80	30	80	100	0	4	3234	3234	3368	3361
80	30	80	100	0	5	5631	5568	6010	6008
80	30	80	100	25	1	3708	3708	3979	3950
80	30	80	100	25	2	2434	2434	2389	2349
80	30	80	100	25	3	3866	3866	4336	4321
80	30	80	100	25	4	3590	3590	3906	3902
80	30	80	100	25	5	3681	3681	4179	4159
80	30	80	100	50	1	3838	3838	4003	3955
80	30	80	100	50	2	3969	3969	4133	4133
80	30	80	100	50	3	4095	4095	4434	4403

80	30	80	100	50	4	3304	3304	3534	3436
80	30	80	100	50	5	4184	4184	4287	4227
80	30	80	200	0	1	3272	3272	3510	3469
80	30	80	200	0	2	3679	3679	3717	3695
80	30	80	200	0	3	3944	3858	4114	4081
80	30	80	200	0	4	3798	3798	4061	3975
80	30	80	200	0	5	3290	3290	3716	3700
80	30	80	200	25	1	5652	5652	5815	5790
80	30	80	200	25	2	3828	3828	4072	4039
80	30	80	200	25	3	4362	4362	4596	4584
80	30	80	200	25	4	3675	3675	4029	3784
80	30	80	200	25	5	2752	2752	2246	2202
80	30	80	200	50	1	4468	4468	5093	5069
80	30	80	200	50	2	3680	3680	3815	3749
80	30	80	200	50	3	4535	4535	4858	4814
80	30	80	200	50	4	3524	3524	3841	3802
80	30	80	200	50	5	4321	4321	4745	4696
80	30	80	300	0	1	3952	3952	4372	4345
80	30	80	300	0	2	3581	3581	4012	3995
80	30	80	300	0	3	4646	4646	5752	5725
80	30	80	300	0	4	4050	4050	4860	4852
80	30	80	300	0	5	3652	3652	3669	3667
80	30	80	300	25	1	3614	3614	4083	4052
80	30	80	300	25	2	3390	3390	3992	3955
80	30	80	300	25	3	3278	3278	3539	3502
80	30	80	300	25	4	4944	4944	5356	5350
80	30	80	300	25	5	2786	2786	2891	2850
80	30	80	300	50	1	4110	4110	4357	4339
80	30	80	300	50	2	4091	4091	4559	4525
80	30	80	300	50	3	4653	4653	5078	5070
80	30	80	300	50	4	2736	2736	2207	2139
80	30	80	300	50	5	4023	4023	4505	4494
80	30	80	400	0	1	3737	3737	4215	4171
80	30	80	400	0	2	3988	3988	4336	4322
80	30	80	400	0	3	3523	3523	3896	3882
80	30	80	400	0	4	4873	4873	5235	5233
80	30	80	400	0	5	3649	3649	4222	4172
80	30	80	400	25	1	3750	3750	4018	4014
80	30	80	400	25	2	2617	2617	2642	2573
80	30	80	400	25	3	3139	3139	2477	2386
80	30	80	400	25	4	4288	4253	4683	4654
80	30	80	400	25	5	4078	4078	4342	4313
80	30	80	400	50	1	3819	3819	4171	4128
80	30	80	400	50	2	4292	4292	4689	4637
80	30	80	400	50	3	3307	3307	3604	3570
80	30	80	400	50	4	3910	3910	4638	4634
80	30	80	400	50	5	3421	3421	3519	3509
80	30	80	500	0	1	3991	3991	4467	4456
80	30	80	500	0	2	3333	3333	3698	3668
80	30	80	500	0	3	3029	3029	2985	2969
80	30	80	500	0	4	3425	3425	4071	3982
80	30	80	500	0	5	4460	4460	4927	4889
80	30	80	500	25	1	3258	3258	2646	2619
80	30	80	500	25	2	3760	3760	3989	3928
80	30	80	500	25	3	3197	3197	3307	3291
80	30	80	500	25	4	4272	4272	4791	4762
80	30	80	500	25	5	4272	4272	5040	5009
80	30	80	500	50	1	2928	2928	2737	2736
80	30	80	500	50	2	3715	3715	3778	3745
80	30	80	500	50	3	3868	3868	4363	4355
80	30	80	500	50	4	3413	3413	2190	2151
80	30	80	500	50	5	3677	3677	4059	4051
80	30	80	500	50	1	3389	1746	1852	1846
80	40	60	100	0	2	1915	1844	1790	1743
80	40	60	100	0	3	3047	1856	1773	1755
80	40	60	100	0					

80	40	60	100	0	4	2546	2251	1808	1798
80	40	60	100	0	5	3110	1812	1749	1722
80	40	60	100	25	1	3767	2996	2014	1984
80	40	60	100	25	2	2238	2238	2041	2018
80	40	60	100	25	3	4310	2794	1788	1758
80	40	60	100	25	4	2755	2383	1707	1682
80	40	60	100	25	5	3023	2382	1728	1701
80	40	60	100	50	1	2361	2361	1576	1551
80	40	60	100	50	2	2697	2397	1799	1775
80	40	60	100	50	3	2438	2321	1765	1755
80	40	60	100	50	4	2630	2471	1804	1795
80	40	60	100	50	5	2263	2263	2101	2091
80	40	60	100	50	1	2580	1860	1964	1866
80	40	60	200	0	2	3382	1882	1753	1668
80	40	60	200	0	3	1879	1879	1802	1778
80	40	60	200	0	4	3511	1882	1810	1785
80	40	60	200	0	5	2911	2006	1908	1899
80	40	60	200	0	1	2887	2315	1705	1626
80	40	60	200	25	2	4115	2572	1771	1751
80	40	60	200	25	3	4508	2894	2516	2449
80	40	60	200	25	4	2536	2155	1933	1925
80	40	60	200	25	5	3317	2229	1566	1545
80	40	60	200	25	1	3206	2674	1651	1627
80	40	60	200	50	2	4161	3254	1803	1773
80	40	60	200	50	3	2245	2215	2056	2018
80	40	60	200	50	4	2314	2145	1834	1810
80	40	60	200	50	5	2628	2628	1775	1773
80	40	60	200	50	1	3007	1719	1818	1800
80	40	60	300	0	2	3134	1992	1977	1923
80	40	60	300	0	3	3121	1702	1644	1627
80	40	60	300	0	4	1859	1859	1762	1715
80	40	60	300	0	5	3393	1893	1894	1876
80	40	60	300	0	1	2341	1949	1859	1835
80	40	60	300	25	2	3992	2685	1905	1901
80	40	60	300	25	3	2258	2029	1659	1652
80	40	60	300	25	4	2782	2429	1983	1934
80	40	60	300	25	5	2221	2110	1678	1649
80	40	60	300	25	1	2274	2274	2012	1986
80	40	60	300	50	2	1846	1846	1814	1780
80	40	60	300	50	3	2424	2402	1798	1768
80	40	60	300	50	4	2106	1934	1704	1611
80	40	60	300	50	5	1994	1994	1813	1792
80	40	60	300	50	1	3520	1721	1853	1821
80	40	60	400	0	2	2103	2103	1773	1761
80	40	60	400	0	3	2587	1767	1734	1715
80	40	60	400	0	4	1893	1619	1696	1670
80	40	60	400	0	5	1998	1899	1837	1776
80	40	60	400	0	1	2332	2015	1952	1910
80	40	60	400	25	2	1739	1739	1977	1922
80	40	60	400	25	3	3544	2556	1763	1750
80	40	60	400	25	4	4064	2715	1758	1740
80	40	60	400	25	5	3990	2639	1599	1523
80	40	60	400	25	1	2544	2544	1603	1581
80	40	60	400	50	2	3193	2555	2076	2011
80	40	60	400	50	3	2346	2346	1979	1952
80	40	60	400	50	4	2167	2118	1701	1687
80	40	60	400	50	5	4548	3264	1874	1855
80	40	60	400	50	1	2699	1665	1817	1806
80	40	60	500	0	2	2190	1864	1897	1845
80	40	60	500	0	3	3784	1887	1809	1750
80	40	60	500	0	4	3785	1783	1758	1722
80	40	60	500	0	5	2028	1879	1996	1946
80	40	60	500	0	1	2970	2371	1980	1977
80	40	60	500	25	2	1938	1891	1773	1730
80	40	60	500	25	3	1995	1983	1883	1775
80	40	60	500	25					

80	40	60	500	25	4	2053	2006	1897	1895
80	40	60	500	25	5	3122	2200	2016	1926
80	40	60	500	50	1	2667	2599	2020	1983
80	40	60	500	50	2	2357	2341	1910	1872
80	40	60	500	50	3	3651	3026	1930	1902
80	40	60	500	50	4	1993	1993	1946	1890
80	40	60	500	50	5	1714	1714	1837	1798
80	40	70	100	0	1	2563	2563	2172	2128
80	40	70	100	0	2	2903	2213	1985	1980
80	40	70	100	0	3	3027	2819	2670	2589
80	40	70	100	0	4	2081	2081	2203	2161
80	40	70	100	0	5	2548	2435	2653	2626
80	40	70	100	25	1	2255	2255	2808	2726
80	40	70	100	25	2	2437	2437	2711	2600
80	40	70	100	25	3	2628	2628	2278	2269
80	40	70	100	25	4	2793	2793	2792	2741
80	40	70	100	25	5	2866	2774	2306	2197
80	40	70	100	50	1	2589	2589	2332	2265
80	40	70	100	50	2	2631	2631	2131	2078
80	40	70	100	50	3	2682	2625	2339	2315
80	40	70	100	50	4	2390	2390	2583	2582
80	40	70	100	50	5	2919	2919	2241	2210
80	40	70	200	0	1	3360	2257	2016	2004
80	40	70	200	0	2	3226	2846	2401	2348
80	40	70	200	0	3	2545	2490	2858	2813
80	40	70	200	0	4	2112	2064	2234	2158
80	40	70	200	0	5	2778	2413	2528	2495
80	40	70	200	25	1	2708	2708	2396	2394
80	40	70	200	25	2	2376	2376	2550	2431
80	40	70	200	25	3	3151	3151	2330	2290
80	40	70	200	25	4	2323	2323	2038	2014
80	40	70	200	25	5	3786	3367	2263	2243
80	40	70	200	50	1	2918	2532	2274	2187
80	40	70	200	50	2	3483	3483	2479	2443
80	40	70	200	50	3	2736	2736	2341	2319
80	40	70	200	50	4	2617	2617	2888	2850
80	40	70	200	50	5	3654	3254	2124	2095
80	40	70	300	0	1	3153	2335	2165	2085
80	40	70	300	0	2	3530	1971	1874	1854
80	40	70	300	0	3	3526	2448	2071	2048
80	40	70	300	0	4	3120	2469	1933	1869
80	40	70	300	0	5	3211	2097	1925	1900
80	40	70	300	25	1	2837	2665	2044	1928
80	40	70	300	25	2	2423	2423	3124	3114
80	40	70	300	25	3	2613	2545	1990	1977
80	40	70	300	25	4	2680	2680	2466	2454
80	40	70	300	25	5	4177	2900	1959	1946
80	40	70	300	50	1	2458	2458	2818	2807
80	40	70	300	50	2	2351	2351	2135	2086
80	40	70	300	50	3	2955	2901	2005	1955
80	40	70	300	50	4	2479	2479	2828	2788
80	40	70	300	50	5	2381	2381	2831	2765
80	40	70	400	0	1	2305	2234	2439	2416
80	40	70	400	0	2	3451	2445	2220	2170
80	40	70	400	0	3	2793	2344	2202	2164
80	40	70	400	0	4	3715	2149	2015	1974
80	40	70	400	0	5	3738	1935	1814	1766
80	40	70	400	25	1	3536	3378	2037	2013
80	40	70	400	25	2	2290	2290	2513	2462
80	40	70	400	25	3	3512	2904	2149	2133
80	40	70	400	25	4	3504	2897	2267	2180
80	40	70	400	25	5	2984	2857	2966	2866
80	40	70	400	50	1	2496	2496	2418	2394
80	40	70	400	50	2	2718	2718	2105	2069
80	40	70	400	50	3	2356	2356	2133	2107

80	40	70	400	50	4	2408	2408	2105	2105
80	40	70	400	50	5	2635	2496	2143	2098
80	40	70	500	0	1	3959	2156	2115	2064
80	40	70	500	0	2	2494	2489	2962	2960
80	40	70	500	0	3	2768	2768	2677	2639
80	40	70	500	0	4	2703	2307	2587	2551
80	40	70	500	0	5	3899	2170	2170	2152
80	40	70	500	25	1	2256	2101	2179	2148
80	40	70	500	25	2	3164	2556	2340	2240
80	40	70	500	25	3	2804	2767	1939	1905
80	40	70	500	25	4	2201	2201	2393	2218
80	40	70	500	25	5	2420	2420	2591	2486
80	40	70	500	50	1	2750	2750	2136	2101
80	40	70	500	50	2	2477	2477	2440	2389
80	40	70	500	50	3	2398	2398	2182	2143
80	40	70	500	50	4	2290	2290	1751	1500
80	40	70	500	50	5	3126	2946	2103	2083
80	40	70	500	50	1	2841	2730	2642	2635
80	40	80	100	0	2	2391	2391	2196	2182
80	40	80	100	0	3	2729	2729	2508	2501
80	40	80	100	0	4	2521	2521	2619	2571
80	40	80	100	0	5	2249	2249	2352	2343
80	40	80	100	0	1	2819	2574	2190	2128
80	40	80	100	25	2	2521	2521	2464	2456
80	40	80	100	25	3	2615	2615	2590	2590
80	40	80	100	25	4	2568	2568	2375	2354
80	40	80	100	25	5	2373	2356	2234	2217
80	40	80	100	25	1	2622	2622	2624	2564
80	40	80	100	50	2	2857	2857	2567	2543
80	40	80	100	50	3	2453	2453	2587	2580
80	40	80	100	50	4	2423	2423	2277	2276
80	40	80	100	50	5	2360	2360	2332	2315
80	40	80	100	50	1	2875	2754	2592	2537
80	40	80	200	0	2	3218	3002	2250	2186
80	40	80	200	0	3	3520	3022	2414	2381
80	40	80	200	0	4	2856	2810	2525	2507
80	40	80	200	0	5	3283	3215	2567	2550
80	40	80	200	0	1	2706	2706	2216	2198
80	40	80	200	25	2	3346	3110	2467	2430
80	40	80	200	25	3	3127	3127	2289	2205
80	40	80	200	25	4	3304	2892	2476	2436
80	40	80	200	25	5	2883	2818	2476	2455
80	40	80	200	25	1	2308	2308	2455	2422
80	40	80	200	50	2	2789	2789	2895	2849
80	40	80	200	50	3	2736	2736	2836	2775
80	40	80	200	50	4	3371	3371	2622	2602
80	40	80	200	50	5	2799	2799	2496	2495
80	40	80	200	50	1	3315	3190	2837	2837
80	40	80	300	0	2	3003	2707	2221	2179
80	40	80	300	0	3	3018	3018	3021	2997
80	40	80	300	0	4	3116	3116	2739	2708
80	40	80	300	0	5	3165	2850	2588	2531
80	40	80	300	0	1	3708	3708	2291	2255
80	40	80	300	25	2	3145	3145	2460	2425
80	40	80	300	25	3	3867	3846	2857	2807
80	40	80	300	25	4	2754	2754	2694	2661
80	40	80	300	25	5	2874	2874	2517	2463
80	40	80	300	25	1	4095	3507	2442	2364
80	40	80	300	50	2	3657	3393	2499	2488
80	40	80	300	50	3	3018	3018	2166	2152
80	40	80	300	50	4	3869	3781	2494	2482
80	40	80	300	50	5	3049	3049	2676	2542
80	40	80	300	50	1	3150	2839	2645	2584
80	40	80	400	0	2	3118	3056	2774	2763
80	40	80	400	0	3	2997	2997	2599	2549
80	40	80	400	0					

80	40	80	400	0	4	2438	2438	2544	2458
80	40	80	400	0	5	3415	2537	2610	2610
80	40	80	400	25	1	2862	2862	2866	2841
80	40	80	400	25	2	3732	3732	2481	2468
80	40	80	400	25	3	3360	3360	2602	2558
80	40	80	400	25	4	3518	3518	2737	2630
80	40	80	400	25	5	3292	3222	2388	2388
80	40	80	400	50	1	3608	3608	2820	2771
80	40	80	400	50	2	3469	3416	2544	2522
80	40	80	400	50	3	3592	3402	2494	2480
80	40	80	400	50	4	3042	3042	2304	2296
80	40	80	400	50	5	2789	2789	2390	2353
80	40	80	500	0	1	3679	3047	2476	2441
80	40	80	500	0	2	3309	3224	2705	2638
80	40	80	500	0	3	2915	2915	2479	2459
80	40	80	500	0	4	3168	3168	2577	2445
80	40	80	500	0	5	3231	3210	2777	2735
80	40	80	500	25	1	4492	2872	2138	2095
80	40	80	500	25	2	2819	2819	2463	2427
80	40	80	500	25	3	2954	2748	2449	2422
80	40	80	500	25	4	3373	2928	2475	2440
80	40	80	500	25	5	3255	3255	2566	2561
80	40	80	500	50	1	3598	3598	2758	2720
80	40	80	500	50	2	3448	3448	2907	2848
80	40	80	500	50	3	3278	3278	3035	2686
80	40	80	500	50	4	3039	3039	2852	2692
80	40	80	500	50	5	3747	3747	2708	2540

NUMBER OF NODES 80 NUMBER OF OBSERVATIONS 675

	LOAD BASED	G(A)	G(B)	G(B)/ARCOPT
AVG SOLUTION LENGTH	2756	2566	2499	2460
NO OF BETTER SOLUTIONS		272	375	396
% IMPROVEMENT OVER LOAD BASED		7	9	11

% OF BACKHAULS IS 20 NUMBER OF OBSERVATIONS 225

	LOAD BASED	G(A)	G(B)	G(B)/ARCOPT
AVG SOLUTION LENGTH	2558	2477	2539	2500
NO OF BETTER SOLUTIONS		68	87	98
% IMPROVEMENT OVER LOAD BASED		3	1	2

% OF BACKHAULS IS 30 NUMBER OF OBSERVATIONS 225

	LOAD BASED	G(A)	G(B)	G(B)/ARCOPT
AVG SOLUTION LENGTH	2805	2651	2724	2685
NO OF BETTER SOLUTIONS		74	93	100
% IMPROVEMENT OVER LOAD BASED		5	3	4

% OF BACKHAULS IS 40 NUMBER OF OBSERVATIONS 225

	LOAD BASED	G(A)	G(B)	G(B)/ARCOPT
AVG SOLUTION LENGTH	2906	2571	2236	2195
NO OF BETTER SOLUTIONS		130	195	198
% IMPROVEMENT OVER LOAD BASED		12	23	24

% OF CONCENTRATION IS 60 NUMBER OF OBSERVATIONS 225

	LOAD BASED	G(A)	G(B)	G(B)/ARCOPT
AVG SOLUTION LENGTH	2339	1920	1654	1624
NO OF BETTER SOLUTIONS		168	209	211
% IMPROVEMENT OVER LOAD BASED		18	29	31

% OF CONCENTRATION IS 70 NUMBER OF OBSERVATIONS 225

	LOAD BASED	G(A)	G(B)	G(B)/ARCOPT
AVG SOLUTION LENGTH	2396	2283	2312	2261
NO OF BETTER SOLUTIONS		69	93	107
% IMPROVEMENT OVER LOAD BASED		5	4	6

% OF CONCENTRATION IS 80	NUMBER OF OBSERVATIONS		225
	LOAD BASED	G(A)	G(B)	G(B)/ARCOPT
AVG SOLUTION LENGTH	3534	3496	3533	3495
NO OF BETTER SOLUTIONS		35	73	78
% IMPROVEMENT OVER LOAD BASED		1	0	1

BACKHAUL PENALTY IS 100	NUMBER OF OBSERVATIONS		135
	LOAD BASED	G(A)	G(B)	G(B)/ARCOPT
AVG SOLUTION LENGTH	2710	2562	2508	2472
NO OF BETTER SOLUTIONS		47	76	83
% IMPROVEMENT OVER LOAD BASED		5	7	9

BACKHAUL PENALTY IS 200	NUMBER OF OBSERVATIONS		135
	LOAD BASED	G(A)	G(B)	G(B)/ARCOPT
AVG SOLUTION LENGTH	2853	2646	2529	2489
NO OF BETTER SOLUTIONS		65	77	79
% IMPROVEMENT OVER LOAD BASED		7	11	13

BACKHAUL PENALTY IS 300	NUMBER OF OBSERVATIONS		135
	LOAD BASED	G(A)	G(B)	G(B)/ARCOPT
AVG SOLUTION LENGTH	2749	2554	2504	2466
NO OF BETTER SOLUTIONS		53	72	83
% IMPROVEMENT OVER LOAD BASED		7	9	10

BACKHAUL PENALTY IS 400	NUMBER OF OBSERVATIONS		135
	LOAD BASED	G(A)	G(B)	G(B)/ARCOPT
AVG SOLUTION LENGTH	2736	2531	2456	2417
NO OF BETTER SOLUTIONS		53	75	79
% IMPROVEMENT OVER LOAD BASED		7	10	12

BACKHAUL PENALTY IS 500	NUMBER OF OBSERVATIONS		135
	LOAD BASED	G(A)	G(B)	G(B)/ARCOPT
AVG SOLUTION LENGTH	2734	2538	2501	2455
NO OF BETTER SOLUTIONS		54	75	75
% IMPROVEMENT OVER LOAD BASED		7	9	10

RADIAL SURCHARGE IS 0	NUMBER OF OBSERVATIONS		225
	LOAD BASED	G(A)	G(B)	G(B)/ARCOPT
AVG SOLUTION LENGTH	2820	2477	2519	2482
NO OF BETTER SOLUTIONS		128	130	133
% IMPROVEMENT OVER LOAD BASED		12	11	12

RADIAL SURCHARGE IS 25	NUMBER OF OBSERVATIONS		225
	LOAD BASED	G(A)	G(B)	G(B)/ARCOPT
AVG SOLUTION LENGTH	2757	2584	2465	2426
NO OF BETTER SOLUTIONS		94	128	137
% IMPROVEMENT OVER LOAD BASED		6	11	12

RADIAL SURCHARGE IS 50	NUMBER OF OBSERVATIONS		225
	LOAD BASED	G(A)	G(B)	G(B)/ARCOPT
AVG SOLUTION LENGTH	2692	2638	2514	2471
NO OF BETTER SOLUTIONS		50	117	126
% IMPROVEMENT OVER LOAD BASED		2	7	8

NO. OF NODES	% OF BACKHAULS	% OF CONCENTR	BKHAUL PENALTY	RADIAL SURCHARGE	PROBL NO.	LOAD BASED SOL	G(A) SOL	G(B) SOL	G(B)/ARC SOL
100	20	60	100	0	1	2237	1773	1583	1548
100	20	60	100	0	2	2342	1774	1702	1676
100	20	60	100	0	3	2564	1777	1637	1326
100	20	60	100	0	4	2178	1459	1540	1513
100	20	60	100	0	5	5967	5967	6138	6127
100	20	60	100	25	1	4401	4277	4042	4041
100	20	60	100	25	2	2609	2320	1607	1605
100	20	60	100	25	3	2445	2017	1696	1635
100	20	60	100	25	4	2547	2035	1522	1505
100	20	60	100	25	5	3918	3890	3893	3890
100	20	60	100	50	1	4112	4112	4244	4238
100	20	60	100	50	2	1877	1877	1577	1540
100	20	60	100	50	3	2851	2413	1643	1613
100	20	60	100	50	4	2451	2451	1745	1695
100	20	60	100	50	5	5604	5604	5598	5580
100	20	60	200	0	1	5577	5331	5464	5463
100	20	60	200	0	2	2751	1895	2082	2072
100	20	60	200	0	3	4877	4484	4280	4279
100	20	60	200	0	4	2124	1617	1514	1501
100	20	60	200	0	5	2379	1613	1617	1584
100	20	60	200	25	1	2571	2231	1693	1622
100	20	60	200	25	2	2056	1937	1608	1587
100	20	60	200	25	3	4471	4102	4108	4012
100	20	60	200	25	4	5335	5093	4795	4768
100	20	60	200	25	5	2805	2110	1593	1582
100	20	60	200	50	1	3715	3715	3688	3679
100	20	60	200	50	2	2335	2335	1726	1702
100	20	60	200	50	3	1481	1481	1617	1598
100	20	60	200	50	4	1802	1802	1428	1403
100	20	60	200	50	5	4223	4223	4018	4003
100	20	60	300	0	1	1901	1620	1691	1645
100	20	60	300	0	2	2494	1726	1756	1695
100	20	60	300	0	3	2153	1937	1690	1675
100	20	60	300	0	4	5914	5907	6158	6143
100	20	60	300	0	5	2513	1647	1586	1507
100	20	60	300	25	1	5410	5410	5547	5432
100	20	60	300	25	2	3912	3907	3804	3804
100	20	60	300	25	3	4750	4118	3805	3777
100	20	60	300	25	4	5998	5878	5943	5913
100	20	60	300	25	5	2893	2357	1697	1345
100	20	60	300	50	1	2883	2502	1635	1593
100	20	60	300	50	2	2034	2034	1705	1660
100	20	60	300	50	3	4434	4434	4516	4516
100	20	60	300	50	4	2328	2190	1492	1464
100	20	60	300	50	5	4510	4510	4505	4455
100	20	60	400	0	1	4218	3912	3967	3924
100	20	60	400	0	2	1879	1538	1681	1651
100	20	60	400	0	3	2250	1756	1675	1668
100	20	60	400	0	4	5466	4710	4600	4577
100	20	60	400	0	5	2357	1643	1664	1633
100	20	60	400	25	1	4947	4850	4876	4829
100	20	60	400	25	2	1732	1717	1873	1827
100	20	60	400	25	3	2817	1978	1370	1354
100	20	60	400	25	4	1877	1815	1885	1854
100	20	60	400	25	5	5573	5239	4630	4600
100	20	60	400	50	1	4937	4937	5130	5115
100	20	60	400	50	2	5274	5274	5482	5428
100	20	60	400	50	3	1908	1908	2056	1916
100	20	60	400	50	4	2128	2018	1577	1544
100	20	60	400	50	5	5674	5674	5860	5807
100	20	60	400	50	1	2212	1598	1479	1452
100	20	60	500	0	2	2078	1556	1540	1529
100	20	60	500	0	3	2272	1574	1655	1634
100	20	60	500	0					

100	20	60	500	0	4	2536	1862	1730	1703
100	20	60	500	0	5	4920	4409	4520	4503
100	20	60	500	25	1	4344	4344	4348	4311
100	20	60	500	25	2	3327	2489	1815	1782
100	20	60	500	25	3	2888	2162	1532	1522
100	20	60	500	25	4	5468	5468	5433	5427
100	20	60	500	25	5	1962	1829	1735	1403
100	20	60	500	50	1	5197	5197	5325	5322
100	20	60	500	50	2	5544	5019	4618	4607
100	20	60	500	50	3	1442	1442	1555	1486
100	20	60	500	50	4	4246	4246	4287	4258
100	20	60	500	50	5	4653	4609	4541	4521
100	20	70	100	0	1	2717	2592	2536	2466
100	20	70	100	0	2	2310	2310	2581	2545
100	20	70	100	0	3	2137	2137	2110	2097
100	20	70	100	0	4	2305	2305	2489	2373
100	20	70	100	0	5	2376	2142	2036	1993
100	20	70	100	25	1	2078	2078	2305	2275
100	20	70	100	25	2	2653	2199	2062	2050
100	20	70	100	25	3	2553	2523	1899	1845
100	20	70	100	25	4	2440	2440	2381	2094
100	20	70	100	25	5	2508	2508	2433	2272
100	20	70	100	50	1	2587	2587	2487	2390
100	20	70	100	50	2	2026	2026	1915	1909
100	20	70	100	50	3	2406	2406	2357	1919
100	20	70	100	50	4	2295	2295	2576	2326
100	20	70	100	50	5	1945	1945	2194	2180
100	20	70	200	0	1	2400	2400	2318	2283
100	20	70	200	0	2	2331	2331	2235	2088
100	20	70	200	0	3	2160	2139	2170	2166
100	20	70	200	0	4	1776	1776	1819	1617
100	20	70	200	0	5	2129	2129	2152	2117
100	20	70	200	25	1	2234	2234	2189	2080
100	20	70	200	25	2	2023	2023	2214	2123
100	20	70	200	25	3	2106	2048	2207	2183
100	20	70	200	25	4	2000	2000	2000	1990
100	20	70	200	25	5	2180	2180	2289	2223
100	20	70	200	50	1	2059	2059	2334	2275
100	20	70	200	50	2	2594	2594	2583	2496
100	20	70	200	50	3	2044	2044	2091	1959
100	20	70	200	50	4	2476	2476	2358	2318
100	20	70	200	50	5	2467	2467	2394	2357
100	20	70	300	0	1	2403	2189	2486	2486
100	20	70	300	0	2	1660	1660	1792	1758
100	20	70	300	0	3	1975	1975	2213	2211
100	20	70	300	0	4	2494	2245	2080	2047
100	20	70	300	0	5	2421	2178	2188	2188
100	20	70	300	25	1	2079	2079	2365	2315
100	20	70	300	25	2	1975	1975	2337	2264
100	20	70	300	25	3	1929	1929	2469	2302
100	20	70	300	25	4	2024	2024	2194	2076
100	20	70	300	25	5	2365	2196	2175	2079
100	20	70	300	50	1	2304	2304	2295	1977
100	20	70	300	50	2	2149	2149	2331	2028
100	20	70	300	50	3	2298	2298	2671	2582
100	20	70	300	50	4	2298	2298	2120	2108
100	20	70	300	50	5	2372	2372	2853	2711
100	20	70	400	0	1	1815	1815	2167	2064
100	20	70	400	0	2	2353	2167	2240	2175
100	20	70	400	0	3	2139	2138	2233	2103
100	20	70	400	0	4	2192	2192	2349	2244
100	20	70	400	0	5	2549	2544	2508	2483
100	20	70	400	25	1	2215	2215	2504	2411
100	20	70	400	25	2	2512	2512	2669	2620
100	20	70	400	25	3	2488	2430	2225	2087

100	20	70	400	25	4	2571	2571	2332	2274
100	20	70	400	25	5	2004	2004	2365	2278
100	20	70	400	50	1	2034	2034	2064	2018
100	20	70	400	50	2	2120	2120	2226	2101
100	20	70	400	50	3	1974	1974	2140	2100
100	20	70	400	50	4	2264	2264	2279	2167
100	20	70	400	50	5	2043	2043	2202	2091
100	20	70	500	0	1	2157	2157	2185	2151
100	20	70	500	0	2	1991	1991	2249	2189
100	20	70	500	0	3	2247	2050	2089	2058
100	20	70	500	0	4	1894	1894	2007	1987
100	20	70	500	0	5	2217	2217	2491	2433
100	20	70	500	25	1	2471	2471	2663	2613
100	20	70	500	25	2	2160	2160	2196	2140
100	20	70	500	25	3	2128	2128	2164	2093
100	20	70	500	25	4	1983	1974	2017	1693
100	20	70	500	25	5	2062	2062	2061	1971
100	20	70	500	50	1	1970	1970	2151	2115
100	20	70	500	50	2	2208	2208	2320	2242
100	20	70	500	50	3	1904	1904	1982	1853
100	20	70	500	50	4	2033	2033	2130	2061
100	20	70	500	50	5	2283	2283	2174	2141
100	20	80	100	0	1	3615	3615	3877	3833
100	20	80	100	0	2	4166	4166	4550	4484
100	20	80	100	0	3	4013	4013	4443	4362
100	20	80	100	0	4	3822	3822	4189	4162
100	20	80	100	0	5	3095	3095	3331	3311
100	20	80	100	25	1	4242	4242	4536	4460
100	20	80	100	25	2	3349	3349	3793	3738
100	20	80	100	25	3	3226	3226	3475	3222
100	20	80	100	25	4	3885	3885	4354	4305
100	20	80	100	25	5	3917	3917	4151	4114
100	20	80	100	50	1	4930	4930	5284	5232
100	20	80	100	50	2	4231	4231	4834	4831
100	20	80	100	50	3	3289	3289	3765	3708
100	20	80	100	50	4	4438	4438	4825	4525
100	20	80	100	50	5	3224	3224	3520	3489
100	20	80	200	0	1	4180	4180	4905	4868
100	20	80	200	0	2	3903	3903	4375	4345
100	20	80	200	0	3	4099	4099	4569	4283
100	20	80	200	0	4	4510	4510	4380	4354
100	20	80	200	0	5	3411	3411	3688	3636
100	20	80	200	25	1	3235	3235	3439	3407
100	20	80	200	25	2	4993	4993	5720	5611
100	20	80	200	25	3	3626	3626	3804	3797
100	20	80	200	25	4	3530	3530	3850	3796
100	20	80	200	25	5	4008	4008	4349	4256
100	20	80	200	50	1	4841	4841	4879	4879
100	20	80	200	50	2	4107	4107	4638	4579
100	20	80	200	50	3	3285	3285	3903	3838
100	20	80	200	50	4	3588	3588	3795	3783
100	20	80	200	50	5	3406	3406	3581	3520
100	20	80	300	0	1	3671	3671	3794	3760
100	20	80	300	0	2	3832	3832	3979	3943
100	20	80	300	0	3	3753	3753	4405	4337
100	20	80	300	0	4	3424	3424	3671	3621
100	20	80	300	0	5	3262	3262	3622	3560
100	20	80	300	25	1	4278	4278	4457	4343
100	20	80	300	25	2	3577	3577	3828	3731
100	20	80	300	25	3	3131	3131	3289	3263
100	20	80	300	25	4	3196	3196	3430	3171
100	20	80	300	25	5	3731	3731	4030	4007
100	20	80	300	50	1	3594	3594	3873	3819
100	20	80	300	50	2	3648	3648	3822	3719
100	20	80	300	50	3	3526	3526	4262	4143

100	20	80	300	50	4	3588	3588	4011	3991
100	20	80	300	50	5	3338	3338	3583	3493
100	20	80	400	0	1	4526	4526	5292	5219
100	20	80	400	0	2	3917	3917	4294	4145
100	20	80	400	0	3	4451	4451	4840	4599
100	20	80	400	0	4	4070	4070	4283	4249
100	20	80	400	0	5	4266	4266	4647	4490
100	20	80	400	25	1	4521	4521	5007	4976
100	20	80	400	25	2	3327	3327	3544	3514
100	20	80	400	25	3	4558	4558	4997	4746
100	20	80	400	25	4	3522	3522	3855	3785
100	20	80	400	25	5	3273	3273	3594	3520
100	20	80	400	50	1	3802	3802	4173	4100
100	20	80	400	50	2	4316	4316	4675	4529
100	20	80	400	50	3	4354	4354	4231	4191
100	20	80	400	50	4	3257	3257	3629	3598
100	20	80	400	50	5	3986	3986	4222	4191
100	20	80	500	0	1	3316	3316	3606	3534
100	20	80	500	0	2	3202	3202	3766	3717
100	20	80	500	0	3	5259	5193	5438	5388
100	20	80	500	0	4	4199	4199	4681	4336
100	20	80	500	0	5	3869	3869	4162	4014
100	20	80	500	25	1	4455	4455	4707	4596
100	20	80	500	25	2	3182	3182	3443	3358
100	20	80	500	25	3	3732	3732	4480	4103
100	20	80	500	25	4	4047	4047	4418	4385
100	20	80	500	25	5	3769	3769	4252	4174
100	20	80	500	50	1	4558	4558	4675	4627
100	20	80	500	50	2	3858	3858	3994	3968
100	20	80	500	50	3	3245	3245	3608	3537
100	20	80	500	50	4	3828	3828	4133	4051
100	20	80	500	50	5	3838	3838	4560	4499
100	30	60	100	0	1	3419	1943	1874	1849
100	30	60	100	0	2	2744	2015	1947	1757
100	30	60	100	0	3	3407	1917	1531	1519
100	30	60	100	0	4	2376	2066	1782	1764
100	30	60	100	0	5	3152	1846	1906	1872
100	30	60	100	25	1	3491	2878	1981	1969
100	30	60	100	25	2	3504	2512	1934	1926
100	30	60	100	25	3	2662	2362	1722	1697
100	30	60	100	25	4	2281	2262	1875	1861
100	30	60	100	25	5	3133	2314	1764	1721
100	30	60	100	50	1	2470	2378	1898	1862
100	30	60	100	50	2	4252	3200	1621	1594
100	30	60	100	50	3	2402	2301	1796	1760
100	30	60	100	50	4	4186	3443	1961	1944
100	30	60	100	50	5	2540	2481	1708	1678
100	30	60	200	0	1	3287	2095	1848	1830
100	30	60	200	0	2	3292	1887	1885	1872
100	30	60	200	0	3	3243	1909	1932	1878
100	30	60	200	0	4	3101	1712	1867	1857
100	30	60	200	0	5	2742	1832	2352	2330
100	30	60	200	25	1	3668	2858	2000	1918
100	30	60	200	25	2	2312	1900	1763	1754
100	30	60	200	25	3	2045	1883	1651	1620
100	30	60	200	25	4	3893	2510	1583	1496
100	30	60	200	25	5	3192	2440	1746	1717
100	30	60	200	50	1	3033	2532	2619	2576
100	30	60	200	50	2	2209	2158	1812	1797
100	30	60	200	50	3	2292	2292	2055	1886
100	30	60	200	50	4	2954	2495	1988	1917
100	30	60	200	50	5	3771	3209	1854	1818
100	30	60	300	0	1	2089	1872	1831	1774
100	30	60	300	0	2	3836	2085	1963	1913
100	30	60	300	0	3	3203	1941	1899	1864

100	30	60	300	0	4	2327	1874	1971	1905
100	30	60	300	0	5	2613	1879	1808	1782
100	30	60	300	25	1	1583	1583	1576	1528
100	30	60	300	25	2	2381	2023	1963	1908
100	30	60	300	25	3	3547	2589	1780	1757
100	30	60	300	25	4	4209	2748	1924	1895
100	30	60	300	25	5	2364	2100	1719	1698
100	30	60	300	50	1	5003	3479	1797	1787
100	30	60	300	50	2	1978	1978	1882	1858
100	30	60	300	50	3	2093	2030	1632	1609
100	30	60	300	50	4	3089	2434	1832	1700
100	30	60	300	50	5	2207	2207	1814	1782
100	30	60	400	0	1	2436	1594	1580	1559
100	30	60	400	0	2	2624	1840	1811	1766
100	30	60	400	0	3	3117	1989	2025	1967
100	30	60	400	0	4	3172	2254	1993	1967
100	30	60	400	0	5	2390	1782	1785	1704
100	30	60	400	25	1	2186	1749	1620	1576
100	30	60	400	25	2	2182	1986	1903	1839
100	30	60	400	25	3	3731	2623	1947	1926
100	30	60	400	25	4	4170	2653	1912	1865
100	30	60	400	25	5	1968	1813	1762	1730
100	30	60	400	50	1	1783	1783	1707	1657
100	30	60	400	50	2	1737	1737	1906	1851
100	30	60	400	50	3	1737	1737	1831	1767
100	30	60	400	50	4	1539	1539	1815	1750
100	30	60	400	50	5	3183	2633	1716	1689
100	30	60	500	0	1	2393	1781	1946	1929
100	30	60	500	0	2	3178	2051	1993	1946
100	30	60	500	0	3	3180	1796	2001	1974
100	30	60	500	0	4	2351	1899	1800	1786
100	30	60	500	0	5	3668	1950	1824	1809
100	30	60	500	25	1	4438	2789	1798	1728
100	30	60	500	25	2	2417	2039	1824	1785
100	30	60	500	25	3	4208	2650	1780	1723
100	30	60	500	25	4	1895	1895	1860	1764
100	30	60	500	25	5	1778	1778	1923	1784
100	30	60	500	50	1	3312	2796	1687	1670
100	30	60	500	50	2	2626	2197	3223	3187
100	30	60	500	50	3	2936	2509	1883	1855
100	30	60	500	50	4	2949	2599	1976	1913
100	30	60	500	50	5	2110	2110	1966	1945
100	30	70	100	0	1	2211	2117	2291	2180
100	30	70	100	0	2	2493	2493	2695	2296
100	30	70	100	0	3	2631	2603	2870	2863
100	30	70	100	0	4	2697	2697	2715	2686
100	30	70	100	0	5	2668	2555	2644	2547
100	30	70	100	25	1	2695	2538	2590	2519
100	30	70	100	25	2	2283	2283	2336	2245
100	30	70	100	25	3	2316	2316	2476	2441
100	30	70	100	25	4	2691	2691	2333	2295
100	30	70	100	25	5	2312	2312	2760	2707
100	30	70	100	50	1	2476	2476	2337	2259
100	30	70	100	50	2	2939	2939	2209	2180
100	30	70	100	50	3	2394	2394	2070	2026
100	30	70	100	50	4	3077	3077	2791	2721
100	30	70	100	50	5	2459	2459	2477	2407
100	30	70	200	0	1	2648	2340	2351	2273
100	30	70	200	0	2	2906	2900	2982	2941
100	30	70	200	0	3	2311	2113	2402	2334
100	30	70	200	0	4	2442	2373	2651	2581
100	30	70	200	0	5	2353	2343	2266	2264
100	30	70	200	25	1	2487	2487	2736	2710
100	30	70	200	25	2	2415	2415	2524	1970
100	30	70	200	25	3	2233	2198	2105	2065

100	30	70	200	25	4	2644	2636	2986	2782
100	30	70	200	25	5	2067	2067	2191	2166
100	30	70	200	50	1	2369	2369	2439	2337
100	30	70	200	50	2	2360	2360	2936	2840
100	30	70	200	50	3	2704	2704	3048	2983
100	30	70	200	50	4	2343	2343	2410	2378
100	30	70	200	50	5	2513	2513	2572	2498
100	30	70	200	50	1	2532	2532	2632	2622
100	30	70	300	0	2	3181	2883	2749	2731
100	30	70	300	0	3	2771	2557	2335	2265
100	30	70	300	0	4	2470	2467	2239	2220
100	30	70	300	0	5	2488	2488	2535	2460
100	30	70	300	0	1	2553	2553	2424	2361
100	30	70	300	25	2	2679	2679	2417	2327
100	30	70	300	25	3	2478	2236	2026	2022
100	30	70	300	25	4	2468	2468	2894	2796
100	30	70	300	25	5	3282	2929	2948	2910
100	30	70	300	25	1	2653	2653	2934	2894
100	30	70	300	50	2	2240	2240	2381	2316
100	30	70	300	50	3	2226	2226	2667	2595
100	30	70	300	50	4	2305	2305	2835	2818
100	30	70	300	50	5	2845	2845	3022	2973
100	30	70	300	50	1	2473	2167	2183	2120
100	30	70	400	0	2	2581	2544	2871	2827
100	30	70	400	0	3	2232	2232	2655	2649
100	30	70	400	0	4	2402	2402	2770	2762
100	30	70	400	0	5	2043	2043	2189	2189
100	30	70	400	0	1	2473	2473	2612	2579
100	30	70	400	25	2	2785	2785	2738	2585
100	30	70	400	25	3	2338	2338	2719	2620
100	30	70	400	25	4	2330	2330	2534	2508
100	30	70	400	25	5	2145	2145	2604	2521
100	30	70	400	25	1	2402	2402	2596	2562
100	30	70	400	50	2	2262	2262	2500	2483
100	30	70	400	50	3	2350	2350	2701	2608
100	30	70	400	50	4	2169	2169	2280	2238
100	30	70	400	50	5	2313	2313	2643	2486
100	30	70	400	50	1	2286	2286	2520	2432
100	30	70	500	0	2	2354	2354	2443	2420
100	30	70	500	0	3	2083	2083	2334	2256
100	30	70	500	0	4	2465	2465	2573	2546
100	30	70	500	0	5	2205	2205	2499	2454
100	30	70	500	0	1	2264	2264	2368	2097
100	30	70	500	25	2	2407	2407	2486	2441
100	30	70	500	25	3	2238	2238	2345	2277
100	30	70	500	25	4	2312	2312	2629	2502
100	30	70	500	25	5	2458	2336	2381	2335
100	30	70	500	25	1	2352	2352	2481	2411
100	30	70	500	50	2	2392	2392	2544	2483
100	30	70	500	50	3	2423	2423	3047	2945
100	30	70	500	50	4	2235	2235	2248	2220
100	30	70	500	50	5	2493	2493	2424	2334
100	30	70	500	50	1	4443	4443	5059	5022
100	30	80	100	0	2	3623	3623	3866	3853
100	30	80	100	0	3	4260	4260	4623	4564
100	30	80	100	0	4	4870	4870	5805	5754
100	30	80	100	0	5	3905	3905	4379	4365
100	30	80	100	0	1	3568	3568	3906	3894
100	30	80	100	25	2	3386	3386	3763	3750
100	30	80	100	25	3	4034	4034	4282	4232
100	30	80	100	25	4	4566	4566	4693	4589
100	30	80	100	25	5	4119	4119	4337	4312
100	30	80	100	25	1	4963	4963	5523	5409
100	30	80	100	50	2	4324	4324	4364	4294
100	30	80	100	50	3	3766	3766	4257	4203
100	30	80	100	50					

100	30	80	100	50	4	3963	3963	4394	4319
100	30	80	100	50	5	4061	4061	4883	4857
100	30	80	200	0	1	4544	4540	4881	4798
100	30	80	200	0	2	3630	3630	3929	3904
100	30	80	200	0	3	3885	3885	4185	4129
100	30	80	200	0	4	4047	4047	4352	4295
100	30	80	200	0	5	3369	3369	3682	3623
100	30	80	200	25	1	3448	3448	3286	3204
100	30	80	200	25	2	4226	4226	4736	4679
100	30	80	200	25	3	3388	3388	3697	3648
100	30	80	200	25	4	3312	3312	3414	3393
100	30	80	200	25	5	4128	4128	4752	4655
100	30	80	200	50	1	3870	3870	4554	4512
100	30	80	200	50	2	4131	4131	4936	4911
100	30	80	200	50	3	3924	3924	4256	4238
100	30	80	200	50	4	4959	4959	5229	5182
100	30	80	200	50	5	4435	4435	4704	4697
100	30	80	300	0	1	3523	3523	3885	3857
100	30	80	300	0	2	4159	4159	4652	4550
100	30	80	300	0	3	3507	3507	3867	3852
100	30	80	300	0	4	3621	3563	3591	3550
100	30	80	300	0	5	5690	5690	6385	6097
100	30	80	300	25	1	4059	4059	4584	4532
100	30	80	300	25	2	4360	4360	5031	5020
100	30	80	300	25	3	4033	4033	4333	3999
100	30	80	300	25	4	4082	4082	4727	4596
100	30	80	300	25	5	3686	3686	3871	3817
100	30	80	300	50	1	4501	4501	5025	4993
100	30	80	300	50	2	3920	3920	4513	4025
100	30	80	300	50	3	5550	5550	6020	5918
100	30	80	300	50	4	3004	3004	3131	3111
100	30	80	300	50	5	4726	4726	5204	5158
100	30	80	400	0	1	5169	5169	5468	5425
100	30	80	400	0	2	3846	3846	4176	4128
100	30	80	400	0	3	3395	3395	3674	3632
100	30	80	400	0	4	4685	4685	5130	5111
100	30	80	400	0	5	3997	3997	4672	4662
100	30	80	400	25	1	4880	4880	5183	5133
100	30	80	400	25	2	3649	3649	3981	3940
100	30	80	400	25	3	4717	4717	5473	5441
100	30	80	400	25	4	3637	3637	4085	3989
100	30	80	400	25	5	4572	4572	4808	4708
100	30	80	400	50	1	3345	3345	3401	3324
100	30	80	400	50	2	4285	4285	4949	4921
100	30	80	400	50	3	3727	3727	4249	4206
100	30	80	400	50	4	4576	4576	5455	5134
100	30	80	400	50	5	4263	4263	4498	4432
100	30	80	500	0	1	4630	4630	5395	5373
100	30	80	500	0	2	4033	4033	4597	4476
100	30	80	500	0	3	4131	4131	4571	4501
100	30	80	500	0	4	3890	3890	4600	4529
100	30	80	500	0	5	4577	4577	5246	5197
100	30	80	500	25	1	3722	3722	4102	3992
100	30	80	500	25	2	4202	4202	4397	4320
100	30	80	500	25	3	4312	4312	4755	4725
100	30	80	500	25	4	4119	4119	4570	4464
100	30	80	500	25	5	3910	3910	4091	4071
100	30	80	500	50	1	3536	3536	3841	3798
100	30	80	500	50	2	3993	3993	4539	4493
100	30	80	500	50	3	3499	3499	3994	3955
100	30	80	500	50	4	3764	3764	4020	3996
100	30	80	500	50	5	4280	4280	5084	5048
100	30	80	500	50	1	4284	2024	2010	1963
100	40	60	100	0	2	4208	2060	2024	2004
100	40	60	100	0	3	3277	2153	1771	1759
100	40	60	100	0					

100	40	60	100	0	4	4425	2110	2098	2083
100	40	60	100	0	5	4237	2026	2011	1972
100	40	60	100	25	1	3750	3013	2234	2211
100	40	60	100	25	2	2971	2515	1843	1768
100	40	60	100	25	3	4015	2737	2148	2109
100	40	60	100	25	4	3213	2694	1955	1920
100	40	60	100	25	5	3094	2495	1810	1786
100	40	60	100	50	1	2080	2080	1776	1737
100	40	60	100	50	2	2549	2520	2187	2128
100	40	60	100	50	3	5118	3923	2011	1945
100	40	60	100	50	4	4714	4003	2182	2158
100	40	60	100	50	5	3134	2899	2065	1997
100	40	60	200	0	1	4331	1902	1810	1765
100	40	60	200	0	2	4195	2073	1963	1951
100	40	60	200	0	3	4012	2155	2017	1971
100	40	60	200	0	4	4189	1941	1852	1803
100	40	60	200	0	5	3633	1965	1916	1876
100	40	60	200	25	1	2518	2503	2081	2037
100	40	60	200	25	2	2144	2144	2016	1975
100	40	60	200	25	3	3004	2405	2187	2083
100	40	60	200	25	4	3198	2483	1987	1975
100	40	60	200	25	5	2303	2134	1765	1749
100	40	60	200	50	1	3697	3120	2100	2031
100	40	60	200	50	2	2379	2379	2072	2036
100	40	60	200	50	3	2728	2714	1998	1973
100	40	60	200	50	4	3294	2956	2096	2012
100	40	60	200	50	5	2243	2243	2113	2044
100	40	60	300	0	1	4212	2142	2225	2225
100	40	60	300	0	2	3171	2246	2168	2141
100	40	60	300	0	3	3880	2035	2077	2023
100	40	60	300	0	4	3089	2109	2143	2107
100	40	60	300	0	5	4366	1879	2173	2142
100	40	60	300	25	1	2486	2275	1877	1871
100	40	60	300	25	2	4653	2847	1926	1858
100	40	60	300	25	3	2954	2262	1944	1863
100	40	60	300	25	4	2300	2124	1882	1843
100	40	60	300	25	5	2259	2259	2082	2015
100	40	60	300	50	1	2508	2508	2070	2052
100	40	60	300	50	2	1912	1912	1853	1814
100	40	60	300	50	3	3428	2677	1797	1772
100	40	60	300	50	4	4113	3328	1978	1957
100	40	60	300	50	5	3201	2906	2160	2123
100	40	60	400	0	1	4428	2010	2012	2012
100	40	60	400	0	2	2499	2044	1880	1857
100	40	60	400	0	3	4102	1923	1928	1906
100	40	60	400	0	4	3999	1966	1944	1923
100	40	60	400	0	5	4593	1981	1983	1953
100	40	60	400	25	1	2880	2680	2089	2010
100	40	60	400	25	2	2098	2098	2149	2105
100	40	60	400	25	3	5356	3186	2193	2105
100	40	60	400	25	4	3553	2669	2115	2038
100	40	60	400	25	5	2453	2236	2229	2171
100	40	60	400	50	1	2847	2799	2112	2064
100	40	60	400	50	2	2921	2562	2252	2200
100	40	60	400	50	3	1976	1976	2267	2114
100	40	60	400	50	4	2259	2185	1948	1908
100	40	60	400	50	5	2199	2199	1919	1846
100	40	60	500	0	1	2417	1872	1871	1853
100	40	60	500	0	2	4231	1915	1913	1907
100	40	60	500	0	3	4463	2018	1971	1962
100	40	60	500	0	4	4303	2000	2024	1996
100	40	60	500	0	5	3758	2295	2043	2018
100	40	60	500	25	1	2466	2466	2064	1997
100	40	60	500	25	2	2695	2242	2093	2042
100	40	60	500	25	3	3526	2572	2091	2053

100	40	60	500	25	4	2884	2511	2282	2134
100	40	60	500	25	5	4048	2612	1795	1717
100	40	60	500	50	1	2333	2333	1855	1847
100	40	60	500	50	2	2243	2139	2075	1941
100	40	60	500	50	3	2370	2370	1822	1779
100	40	60	500	50	4	1752	1752	2160	2140
100	40	60	500	50	5	2445	2282	2115	2041
100	40	70	100	0	1	2790	2780	2720	2609
100	40	70	100	0	2	2664	2664	2979	2888
100	40	70	100	0	3	2521	2521	2824	2611
100	40	70	100	0	4	3022	2612	2554	2494
100	40	70	100	0	5	2873	2873	2594	2516
100	40	70	100	25	1	2545	2545	3049	2987
100	40	70	100	25	2	2763	2763	2860	2705
100	40	70	100	25	3	2575	2575	2111	2077
100	40	70	100	25	4	2990	2947	3201	3108
100	40	70	100	25	5	3533	3199	2760	2743
100	40	70	100	50	1	2931	2931	2978	2954
100	40	70	100	50	2	3175	3175	3033	2972
100	40	70	100	50	3	2505	2505	3171	2936
100	40	70	100	50	4	3015	3015	2764	2720
100	40	70	100	50	5	2737	2737	3053	3041
100	40	70	200	0	1	2794	2794	2995	2968
100	40	70	200	0	2	2736	2736	3064	2957
100	40	70	200	0	3	2627	2288	2452	2416
100	40	70	200	0	4	2926	2674	2436	2421
100	40	70	200	0	5	2773	2773	3066	2996
100	40	70	200	25	1	2789	2789	2489	2463
100	40	70	200	25	2	3035	3035	3357	3298
100	40	70	200	25	3	2761	2761	2775	2752
100	40	70	200	25	4	3282	2958	2721	2606
100	40	70	200	25	5	2499	2499	3054	2917
100	40	70	200	50	1	2803	2803	2779	2773
100	40	70	200	50	2	2321	2321	2481	2428
100	40	70	200	50	3	2682	2682	2632	2583
100	40	70	200	50	4	2115	2115	2316	2297
100	40	70	200	50	5	2797	2797	2441	2427
100	40	70	300	0	1	2952	2873	3249	3164
100	40	70	300	0	2	2401	2401	2709	2644
100	40	70	300	0	3	3519	3067	2803	2723
100	40	70	300	0	4	2675	2656	2499	2470
100	40	70	300	0	5	2702	2416	2714	2700
100	40	70	300	25	1	2934	2934	3000	2905
100	40	70	300	25	2	3244	3244	3155	3093
100	40	70	300	25	3	2589	2589	2880	2796
100	40	70	300	25	4	2638	2638	2456	2410
100	40	70	300	25	5	2382	2382	2679	2629
100	40	70	300	50	1	2746	2746	2852	2810
100	40	70	300	50	2	2678	2678	3104	3075
100	40	70	300	50	3	2565	2565	2827	2745
100	40	70	300	50	4	2340	2340	2681	2594
100	40	70	300	50	5	2915	2915	3245	3194
100	40	70	400	0	1	3217	3042	2812	2761
100	40	70	400	0	2	3729	2778	2623	2491
100	40	70	400	0	3	3110	2460	2573	2559
100	40	70	400	0	4	2742	2588	2763	2686
100	40	70	400	0	5	3335	2572	2597	2572
100	40	70	400	25	1	2452	2452	2742	2660
100	40	70	400	25	2	2735	2735	2696	2627
100	40	70	400	25	3	3035	3020	3172	3152
100	40	70	400	25	4	2580	2580	2664	2617
100	40	70	400	25	5	2530	2530	2813	2774
100	40	70	400	50	1	2366	2366	2800	2740
100	40	70	400	50	2	2458	2458	2736	2680
100	40	70	400	50	3	2334	2334	2517	2480

100	40	70	400	50	4	2184	2184	2420	2387
100	40	70	400	50	5	2493	2493	2385	2329
100	40	70	500	0	1	3016	2726	2799	2742
100	40	70	500	0	2	3637	3108	3004	2976
100	40	70	500	0	3	3218	2635	3011	2954
100	40	70	500	0	4	2802	2727	2720	2608
100	40	70	500	0	5	2159	2159	2650	2566
100	40	70	500	25	1	2776	2582	2550	2503
100	40	70	500	25	2	2345	2345	2669	2647
100	40	70	500	25	3	2299	2299	2446	2392
100	40	70	500	25	4	2603	2603	2832	2791
100	40	70	500	25	5	2780	2724	2624	2589
100	40	70	500	50	1	2502	2502	2624	2602
100	40	70	500	50	2	2536	2536	3240	3086
100	40	70	500	50	3	2665	2665	2906	2846
100	40	70	500	50	4	2384	2384	2829	2702
100	40	70	500	50	5	2632	2632	3012	2924
100	40	80	100	0	1	3785	3785	3989	3951
100	40	80	100	0	2	4303	4303	4497	4495
100	40	80	100	0	3	4814	4814	5541	5518
100	40	80	100	0	4	4731	4731	5136	5113
100	40	80	100	0	5	4302	4302	4661	4640
100	40	80	100	25	1	3717	3717	3862	3853
100	40	80	100	25	2	4084	4084	4814	4754
100	40	80	100	25	3	5616	5616	6089	6057
100	40	80	100	25	4	4819	4819	5550	5515
100	40	80	100	25	5	4104	4104	4790	4704
100	40	80	100	50	1	4963	4963	5278	5270
100	40	80	100	50	2	4152	4152	4591	4525
100	40	80	100	50	3	3904	3904	4584	4559
100	40	80	100	50	4	4872	4872	5266	5215
100	40	80	100	50	5	3659	3659	3833	3799
100	40	80	200	0	1	4352	4352	4622	4621
100	40	80	200	0	2	3629	3629	4360	4331
100	40	80	200	0	3	5093	5093	5745	5613
100	40	80	200	0	4	4915	4915	5408	5374
100	40	80	200	0	5	4925	4925	5394	5283
100	40	80	200	25	1	4653	4653	5511	5402
100	40	80	200	25	2	4933	4933	5216	5178
100	40	80	200	25	3	4634	4634	5168	5127
100	40	80	200	25	4	3745	3745	4403	4378
100	40	80	200	25	5	5168	5168	5567	5463
100	40	80	200	50	1	3855	3855	4295	4266
100	40	80	200	50	2	4487	4487	4968	4879
100	40	80	200	50	3	4347	4347	4703	4616
100	40	80	200	50	4	3951	3951	4456	4416
100	40	80	200	50	5	3864	3864	4655	4635
100	40	80	300	0	1	4790	4790	5036	5017
100	40	80	300	0	2	4611	4581	4679	4650
100	40	80	300	0	3	4277	4277	4723	4713
100	40	80	300	0	4	4650	4650	6193	5660
100	40	80	300	0	5	3515	3515	4049	4012
100	40	80	300	25	1	4169	4169	4537	4501
100	40	80	300	25	2	3580	3580	4014	3983
100	40	80	300	25	3	4259	4259	4270	4177
100	40	80	300	25	4	3899	3899	4049	4023
100	40	80	300	25	5	4510	4510	5015	4946
100	40	80	300	50	1	3900	3900	4197	4126
100	40	80	300	50	2	5012	5012	5498	5479
100	40	80	300	50	3	5048	5048	5732	5645
100	40	80	300	50	4	4095	4095	4590	4552
100	40	80	300	50	5	3622	3622	4627	4586
100	40	80	400	0	1	3909	3909	4502	4499
100	40	80	400	0	2	4376	4304	4489	4403
100	40	80	400	0	3	4361	4361	5020	5007

100	40	80	400	0	4	4111	4111	4682	4585
100	40	80	400	0	5	3652	3652	3797	3778
100	40	80	400	25	1	5479	5479	5695	5686
100	40	80	400	25	2	4882	4882	5518	5488
100	40	80	400	25	3	4180	4180	4621	4602
100	40	80	400	25	4	3253	3253	3385	3334
100	40	80	400	25	5	4970	4970	5699	5658
100	40	80	400	50	1	4168	4168	4402	4386
100	40	80	400	50	2	4327	4327	4808	4743
100	40	80	400	50	3	3734	3734	4110	4066
100	40	80	400	50	4	4361	4361	4508	4489
100	40	80	400	50	5	3348	3348	3733	3685
100	40	80	500	0	1	3015	3015	3390	3319
100	40	80	500	0	2	4925	4925	5307	5244
100	40	80	500	0	3	3833	3833	4275	4263
100	40	80	500	0	4	3746	3746	4229	4170
100	40	80	500	0	5	4031	4031	4844	4815
100	40	80	500	25	1	3772	3772	4039	4004
100	40	80	500	25	2	4367	4367	4848	4836
100	40	80	500	25	3	4322	4322	4950	4527
100	40	80	500	25	4	3652	3652	3950	3935
100	40	80	500	25	5	3552	3552	4100	4065
100	40	80	500	50	1	4349	4349	4911	4896
100	40	80	500	50	2	3727	3727	4424	4416
100	40	80	500	50	3	3870	3870	4120	4075
100	40	80	500	50	4	4379	4379	5033	4974
100	40	80	500	50	5	3706	3706	3991	3945

NUMBER OF NODES 100 NUMBER OF OBSERVATIONS 675

	LOAD BASED	G(A)	G(B)	G(B)/ARCOPT
AVG SOLUTION LENGTH	3242	3017	3098	3036
NO OF BETTER SOLUTIONS		240	284	322
% IMPROVEMENT OVER LOAD BASED		7	4	6

% OF BACKHAULS IS 20 NUMBER OF OBSERVATIONS 225

	LOAD BASED	G(A)	G(B)	G(B)/ARCOPT
AVG SOLUTION LENGTH	3161	3052	3143	3071
NO OF BETTER SOLUTIONS		69	89	107
% IMPROVEMENT OVER LOAD BASED		3	1	3

% OF BACKHAULS IS 30 NUMBER OF OBSERVATIONS 225

	LOAD BASED	G(A)	G(B)	G(B)/ARCOPT
AVG SOLUTION LENGTH	3140	2906	2978	2917
NO OF BETTER SOLUTIONS		85	93	107
% IMPROVEMENT OVER LOAD BASED		7	5	7

% OF BACKHAULS IS 40 NUMBER OF OBSERVATIONS 225

	LOAD BASED	G(A)	G(B)	G(B)/ARCOPT
AVG SOLUTION LENGTH	3426	3093	3173	3119
NO OF BETTER SOLUTIONS		86	102	108
% IMPROVEMENT OVER LOAD BASED		10	7	9

% OF CONCENTRATION IS 60 NUMBER OF OBSERVATIONS 225

	LOAD BASED	G(A)	G(B)	G(B)/ARCOPT
AVG SOLUTION LENGTH	3188	2566	2291	2247
NO OF BETTER SOLUTIONS		177	201	203
% IMPROVEMENT OVER LOAD BASED		20	28	30

% OF CONCENTRATION IS 70 NUMBER OF OBSERVATIONS 225

	LOAD BASED	G(A)	G(B)	G(B)/ARCOPT
AVG SOLUTION LENGTH	2476	2424	2532	2457
NO OF BETTER SOLUTIONS		58	79	109
% IMPROVEMENT OVER LOAD BASED		2	-2	1

% OF CONCENTRATION IS 80	NUMBER OF OBSERVATIONS		225
LOAD BASED	G(A)	G(B)	G(B)/ARCOPT
AVG SOLUTION LENGTH 4061	4060	4472	4403
NO OF BETTER SOLUTIONS	5	4	10
% IMPROVEMENT OVER LOAD BASED	0	-10	-8

BACKHAUL PENALTY IS 100	NUMBER OF OBSERVATIONS		135
LOAD BASED	G(A)	G(B)	G(B)/ARCOPT
AVG SOLUTION LENGTH 3338	3099	3080	3018
NO OF BETTER SOLUTIONS	51	67	75
% IMPROVEMENT OVER LOAD BASED	7	8	10

BACKHAUL PENALTY IS 200	NUMBER OF OBSERVATIONS		135
LOAD BASED	G(A)	G(B)	G(B)/ARCOPT
AVG SOLUTION LENGTH 3235	3011	3082	3026
NO OF BETTER SOLUTIONS	49	62	70
% IMPROVEMENT OVER LOAD BASED	7	5	6

BACKHAUL PENALTY IS 300	NUMBER OF OBSERVATIONS		135
LOAD BASED	G(A)	G(B)	G(B)/ARCOPT
AVG SOLUTION LENGTH 3247	3027	3110	3042
NO OF BETTER SOLUTIONS	50	59	75
% IMPROVEMENT OVER LOAD BASED	7	4	6

BACKHAUL PENALTY IS 400	NUMBER OF OBSERVATIONS		135
LOAD BASED	G(A)	G(B)	G(B)/ARCOPT
AVG SOLUTION LENGTH 3222	3005	3137	3078
NO OF BETTER SOLUTIONS	47	47	54
% IMPROVEMENT OVER LOAD BASED	7	3	4

BACKHAUL PENALTY IS 500	NUMBER OF OBSERVATIONS		135
LOAD BASED	G(A)	G(B)	G(B)/ARCOPT
AVG SOLUTION LENGTH 3168	2942	3082	3016
NO OF BETTER SOLUTIONS	43	49	57
% IMPROVEMENT OVER LOAD BASED	7	3	5

RADIAL SURCHARGE IS 0	NUMBER OF OBSERVATIONS		225
LOAD BASED	G(A)	G(B)	G(B)/ARCOPT
AVG SOLUTION LENGTH 3323	2899	3068	3013
NO OF BETTER SOLUTIONS	119	107	117
% IMPROVEMENT OVER LOAD BASED	13	8	9

RADIAL SURCHARGE IS 25	NUMBER OF OBSERVATIONS		225
LOAD BASED	G(A)	G(B)	G(B)/ARCOPT
AVG SOLUTION LENGTH 3243	3060	3084	3014
NO OF BETTER SOLUTIONS	83	97	113
% IMPROVEMENT OVER LOAD BASED	6	5	7

RADIAL SURCHARGE IS 50	NUMBER OF OBSERVATIONS		225
LOAD BASED	G(A)	G(B)	G(B)/ARCOPT
AVG SOLUTION LENGTH 3160	3091	3142	3081
NO OF BETTER SOLUTIONS	38	80	92
% IMPROVEMENT OVER LOAD BASED	2	1	3

APPENDIX D LBI and LBI+P Code

Options	File (by segment)	Size
	Runtime.lib	18256
	Interface.lib	8104
	Segment 1	26364
D N V R	BCKINSXX(A)	24500
	Segment 2	24500
D N V R	CWUNIT(A)2	17848
	Segment 3	17848
D N V R	BASECASEUNIT(A)	21608
	Segment 4	21608

```
program BCKINSERT;
  uses
    CWUNIT, BASECASE;
  label
    1011, 1313, 383, 195, 197, 935, 777, 778, 296, 289, 188, 301, 38, 36, 39, 40, 100, 111, 112, 114, 116, 50, 115, 120, 117,
        118, 133, 134, 144, 145, 3333, 335, 336, 388, 201, 1212, 1213;
  type
    MATRIX3 = array[1..100, 1..11] of INTEGER;
    MATRIX4 = array[1..100, 1..5] of INTEGER;
    DATETIMEREC = record
       YEAR, MONTH, DAY, HOUR, MINUTE, SECOND, DAYOFWEEK: INTEGER;
      end;
  var
    NPT1: COUNTER;
    NPT: COUNTER;
    NARC: COUNTER;
    NTOURS: COUNTER;
    MAXDROP: COUNTER;
    NVT: COUNTER;
    MAXTIME: COUNTER;
    MILMIN: REAL1;
    ID: MATRIX1;
    PTOUR: MATRIX2;
    Q: IVECT100;
    X: VECT100;
    Y: VECT100;
    TRCAP: MATRIX5;
    REJARR: matrix9;
    rejecount: INTEGER;
    II, IIII, III2, III3, III4: INTEGER;
    DATETIME: DATETIMEREC;
    T0, T1, T2, T3, T4, T5, T6, ELAPS, HRS, MINS, SECS, AHEAD, BEHIND: INTEGER;
    NARC1, NARC3: COUNTER;
    ROUTDATA: MATRIX3;
    CC: MATRIX4;
    V, NTOUR, III: COUNTER;
{   C: array[1..10, 1..10] of INTEGER;}
    I, J, CMIN1, CMIN2, CMIN3, CMIN4, CMIN5, CALC, BACKHAUL, JPOINT, IPOINT, DELIVER: INTEGER;
    PENALTY1, PERCENT1, INFLATION1, INCREASE: REAL;
    PENALTY, PERCENT, INFLATION: INTEGER;
    REJECTED, AFTER, AFTER1, NLINKS, MIDPOINT, YYYZ, XOFF: INTEGER;
    NM1, IPLUS1, TEMP1, DIST1, DIST2, TEMP2, TEMP3, TEMP4, TEMP5, TEMPX, TEMPXX, NODES, K, L, XXXX, TDIST1, TDIST2, P0, P1,
        P2, P3, P4: INTEGER;
    TOTTIME1, TOTTIME, SAVINGS: INTEGER;
    JJ, DIRECTION, KK: INTEGER;
    TEMP, ICAP, ICAP1, ICAP2, ICAP5, ISUM, JJKK: INTEGER;
    NEWSCALE, XXX, TOURS, YYY: COUNTER;
    CCXX, CCXX2: INTEGER;
    NUM: INTEGER;
  function C1 (X: VECT100; Y: VECT100; I: INTEGER; J: INTEGER): INTEGER;
    var
      CC1: INTEGER;
  begin
    CC1 := ROUND(SQRT(SQR(X[I] - X[J]) + SQR(Y[I] - Y[J])));
{WRITELN('CC1  ', CC1);}
    C1 := CC1;
  end;
  procedure GETTIME (var DATETIME: DATETIMEREC);
  EXTERNAL;
  procedure CAPSET (var ID: MATRIX1; var PTOUR: MATRIX2; var Q: IVECT100; var ROUTDATA: MATRIX3; var CC: MATRIX4; var
       K: COUNTER; var NARC, NVT: COUNTER; var TRCAP: MATRIX5);
    label
      236, 235, 288;
    var
      ICAP, I, J, JJ, IPOINT, YYYZ, DIRECTION, KK, II: INTEGER;
  begin
    ICAP := 0;
    I := 0;
    J := 0;
```

```
    JJ := 0;
    IPOINT := 0;
    YYYZ := 0;
    DIRECTION := 0;
    KK := 0;
    II := 0;
    for J := 1 to NARC do
      begin
        JJ := CC[K, 2];
        if ((ID[J, 3] = 2) and (J = JJ)) then
          goto 236;
        if ID[J, 1] <> 1 then
          goto 288;
        if ((ROUTDATA[PTOUR[ID[JJ, 2], 1], 7] = ID[J, CC[K, 5]]) and (PTOUR[ID[JJ, 2], 1] = PTOUR[ID[J, 2], 1])) then
          goto 236
        else
          goto 235;
235:
      end;
236:
    YYYZ := PTOUR[ID[J, 2], 1];
    DIRECTION := J;
    ROUTDATA[YYYZ, 5] := DIRECTION;
    ROUTDATA[YYYZ, 10] := CC[K, 5];
{WRITELN('ROUTE', YYYZ, 'ROUTDATA[5]', ROUTDATA[YYYZ, 5], 'R[10]', ROUTDATA[YYYZ, 10]);}
288:
  end;
{CAPACITY CHECK SUBROUTINE}
  procedure CAPCHK (var ID: MATRIX1; var PTOUR: MATRIX2; var Q: IVECT100; var ROUTDATA: MATRIX3; var CC: MATRIX4; var
        K: COUNTER; var NARC, NVT: COUNTER; var TRCAP: MATRIX5; var V: COUNTER);
    label
      934;
    var
      ICAP, I, J, JJ, IPOINT, YYYZ, DIRECTION, KK, JJKK: INTEGER;
  begin
    ICAP := 0;
    I := 0;
    J := 0;
    JJ := 0;
    IPOINT := 0;
    YYYZ := 0;
    DIRECTION := 0;
    KK := 0;
    JJKK := 0;
    V := 0;
    JJ := CC[K, 2];
    JJKK := PTOUR[ID[JJ, 2], 1];
    YYYZ := ROUTDATA[JJKK, 5];
    ICAP := ROUTDATA[JJKK, 7];
    IPOINT := ID[YYYZ, 2];
    I := YYYZ;
{WRITELN('CC[K,2]', JJ, ' PTOUR[ID[JJ, 2], 1]', JJKK, 'ROUTDATA[JJKK, 5]', YYYZ);}
{WRITELN('ROUTDATA [ JJKK , 7 ]', ICAP, 'ID[YYYZ, 2]', IPOINT, 'YYYZ', I);}
    repeat
      ICAP := ICAP - Q[IPOINT];
      if ID[YYYZ, 5] = 0 then
        I := ID[I, 7]
      else
        I := ID[I, 8];
      while ICAP > TRCAP[NVT] do
        begin
          V := 1;
          goto 934;
        end;
      if IPOINT = ID[I, 1] then
        IPOINT := ID[I, 2]
      else
        IPOINT := ID[I, 1];
{WRITELN('IPOINT', IPOINT);}
```

```
  until IPOINT = 1;
934:
 end;
 procedure CAPREV (var ID: MATRIX1; var PTOUR: MATRIX2; var Q: IVECT100; var ROUTDATA: MATRIX3; var CC: MATRIX4; var
     K: COUNTER; var NARC, NVT: COUNTER; var TRCAP: MATRIX5; var NTOUR: COUNTER);
  label
   196, 189, 235, 236, 288, 101;
  var
   ICAP, I, J, JJ, IPOINT, YYYZ, DIRECTION, KK, JJKK, II: INTEGER;
 begin
  ICAP := 0;
  I := 0;
  J := 0;
  JJ := 0;
  IPOINT := 0;
  YYYZ := 0;
  DIRECTION := 0;
  KK := 0;
  JJKK := 0;
  II := 0;
  for II := 1 to NTOUR do
   begin
    JJKK := II;
    YYYZ := ROUTDATA[JJKK, 5];
    if ROUTDATA[JJKK, 9] = 3 then
     goto 288;
    while YYYZ = 0 do
     begin
      for J := 1 to NARC do
       begin
        if ID[J, 3] = 0 then
         goto 235;
        if JJKK = PTOUR[ID[J, 2], 1] then
         goto 236
        else
         goto 235;
236:
        YYYZ := J;
        while ID[J, 3] = 2 do
         begin
          ID[YYYZ, 6] := Q[ID[YYYZ, 2]];
          goto 288;
         end;
        goto 101;
235:
       end;
     end;
101:
    ICAP := ROUTDATA[JJKK, 7];
    IPOINT := ID[YYYZ, 2];
    while ID[YYYZ, 3] = 2 do
     begin
      ID[YYYZ, 6] := ICAP;
      goto 288;
     end;
    ID[YYYZ, 6] := ICAP;
196:
    I := YYYZ;
    if ID[I, 5] = 0 then
     I := ID[I, 7]
    else
     I := ID[I, 8];
    repeat
     if ROUTDATA[JJKK, 5] = 0 then
      ICAP := ICAP + Q[IPOINT]
     else
      ICAP := ICAP - Q[IPOINT];
     ID[I, 6] := ICAP;
     if IPOINT = ID[I, 1] then
```

```
          IPOINT := ID[I, 2]
        else
          IPOINT := ID[I, 1];
        If ID[YYYZ, 5] = 0 then
          I := ID[I, 7]
        else
          I := ID[I, 8];
       until IPOINT = 1;
288:
    end;
189:
  end;
  procedure NETGEN (var X, Y: VECT100; var NPT1, NPT, newscale: COUNTER);
    var
      I, I, J, NODES, backhaul, NPTB: integer;
      r, w: real;
      yy: array[1..200] of real;
  begin
    i := 0;
    I := 0;
    j := 0;
    NODES := 0;
    backhaul := 0;
    NPTB := 0;
    r := 0;
    w := 0;
    for I := 1 to 100 do
      begin
        YY[I] := 0;
      end;
    i := 1;
    P1 := ROUND(X[I]) * (newscale);
    P2 := ROUND(Y[i]) * newscale;
    PaintCircle(P1, P2, 5);
    Moveto(P1 - 20, P2);
{WriteDraw(I);}
    for I := 2 to NPT do
      begin
        P1 := ROUND(X[I]) * (newscale);
        P2 := ROUND(Y[I]) * newscale;
        drawline(P1 - 3, P2 - 3, P1 + 3, P2 - 3);
        drawline(P1 - 3, P2 + 3, P1 + 3, P2 + 3);
        drawline(P1 - 3, P2 - 3, P1 - 3, P2 + 3);
        drawline(P1 + 3, P2 - 3, P1 + 3, P2 + 3);
        Moveto(P1 - 15, P2);
{WriteDraw(I);}
      end;
    NODES := NPT1;
    for I := NPT + 1 to NODES do
      begin
        P1 := ROUND(X[I]) * (newscale);
        P2 := ROUND(Y[I]) * newscale;
        drawline(P1, P2 - 4, P1 - 4, P2 + 4);
        drawline(P1, P2 - 4, P1 + 4, P2 + 4);
        drawline(P1 - 4, P2 + 4, P1 + 4, P2 + 4);
        Moveto(P1 - 15, P2);
{WriteDraw(I);}
      end;
  end;
begin
  II := 0;
  III := 0;
  III2 := 0;
  III3 := 0;
  III4 := 0;
  T0 := 0;
  T1 := 0;
  T2 := 0;
  T3 := 0;
```

```
T4 := 0;
T5 := 0;
T6 := 0;
ELAPS := 0;
HRS := 0;
MINS := 0;
SECS := 0;
AHEAD := 0;
BEHIND := 0;
NARC1 := 0;
NARC3 := 0;
I := 0;
J := 0;
CMIN1 := 0;
CMIN2 := 0;
CMIN3 := 0;
CMIN4 := 0;
CMIN5 := 0;
CALC := 0;
BACKHAUL := 0;
JPOINT := 0;
IPOINT := 0;
DELIVER := 0;
PENALTY1 := 0;
PERCENT1 := 0;
INFLATION1 := 0;
INCREASE := 0;
PENALTY := 0;
PERCENT := 0;
INFLATION := 0;
REJECTED := 0;
AFTER := 0;
AFTER1 := 0;
NLINKS := 0;
MIDPOINT := 0;
YYYZ := 0;
XOFF := 0;
NM1 := 0;
IPLUS1 := 0;
TEMP1 := 0;
DIST1 := 0;
DIST2 := 0;
TEMP2 := 0;
TEMP3 := 0;
TEMP4 := 0;
TEMP5 := 0;
TEMPX := 0;
TEMPXX := 0;
NODES := 0;
K := 0;
L := 0;
XXXX := 0;
TDIST1 := 0;
TDIST2 := 0;
P0 := 0;
P1 := 0;
P2 := 0;
P3 := 0;
P4 := 0;
TOTTIME1 := 0;
TOTTIME := 0;
JJ := 0;
DIRECTION := 0;
KK := 0;
V := 0;
NTOUR := 0;
III := 0;
TEMP := 0;
ICAP := 0;
```

```
  ICAP1 := 0;
  ICAP2 := 0;
  ICAP5 := 0;
  ISUM := 0;
  JJKK := 0;
  NEWSCALE := 0;
  XXX := 0;
  TOURS := 0;
  YYY := 0;
  CCXX := 0;
  CCXX2 := 0;
  for I := 1 to 50 do
    begin
      for j := 1 to 3 do
        begin
          REJARR[I, J] := 0;
        end;
    end;
  rejecount := 0;
  for i := 1 to 100 do
    begin
      for j := 1 to 11 do
        begin
          ROUTDATA[I, J] := 0;
        end;
    end;
  for i := 1 to 100 do
    begin
      for j := 1 to 5 do
        begin
          CC[I, J] := 0;
        end;
    end;
  BASE(NPT1, NPT, NARC, NTOURS, MAXDROP, NVT, MAXTIME, MILMIN, ID, PTOUR, Q, X, Y, TRCAP);
{WRITELN('NPT1', NPT1, 'NPT', NPT, 'NARC', NARC, 'NTOURS', NTOURS);}
{    goto 301;}
  WRITELN('ENTER DELIVERY AFTER PICKUP PENALTY(PERCENT)');
  READLN(PENALTY);
  WRITELN('ENTER OVERTIME RATE(PERCENT)');
  READLN(PERCENT);
  WRITELN('ENTER SUPPLIER SURCHARGE(PERCENT)');
  READLN(INFLATION);
  PERCENT1 := PERCENT / 100;
  PENALTY1 := PENALTY / 100;
  INFLATION1 := INFLATION / 100;
  NODES := NPT1;
  DELIVER := NPT;
  BACKHAUL := NODES - DELIVER;
  REJARR[1, 1] := 1;
  REJARR[1, 2] := ROUND(X[1]);
  REJARR[1, 3] := ROUND(Y[1]);
  for I := 1 to NODES do
    begin
      for J := 1 to NODES do
        begin
{C[I, J] := ROUND(SQRT(SQR(X[I] - X[J]) + SQR(Y[I] - Y[J])));}
{WRITELN('C[I,J]    ', C[I, J]);}
{WRITELN('C1', C1(X, Y, I, J));}
          if (I = 1) and (J > DELIVER) then
            CC[J, 4] := ROUND((1 + INFLATION1) * C1(X, Y, I, J));
        end;
    end;
  for I := 1 to NTOURS do
    begin
      ROUTDATA[I, 1] := 0;
      ROUTDATA[I, 2] := 0;
      ROUTDATA[I, 3] := -1;
      ROUTDATA[I, 8] := 0;
    end;
```

```
  for J := 1 to NARC do
    begin
      ID[J, 6] := 0;
      if PTOUR[ID[J, 2], 1] > NTOURS then
        goto 112;
      if ID[J, 3] = 0 then
        goto 112;
      I := PTOUR[ID[J, 2], 1];
      while ID[J, 3] = 2 do
        begin
          if ID[J, 2] > DELIVER then
            ROUTDATA[I, 1] := ROUND(C1(X, Y, ID[J, 1], ID[J, 2]) * (1 + INFLATION1)) + ROUTDATA[I, 1]
          else
            ROUTDATA[I, 1] := (2 * C1(X, Y, ID[J, 1], ID[J, 2])) + ROUTDATA[I, 1];
          ROUTDATA[I, 3] := 1;
          goto 112;
        end;
      ROUTDATA[I, 1] := C1(X, Y, ID[J, 1], ID[J, 2]) + ROUTDATA[I, 1];
      ROUTDATA[I, 3] := ROUTDATA[I, 3] + 1;
112:
    end;
  WRITELN('     I   ', 'LENGTH', '  DIRECT', ' #OFNODES', '  TIME');
  for I := 1 to NTOURS do
    begin
{WRITELN('HI');}
      ROUTDATA[I, 6] := ROUND(ROUTDATA[I, 1] / MILMIN);
{WRITELN(I, ROUTDATA[I, 1], ROUTDATA[I, 2], ROUTDATA[I, 3], '  ', ROUTDATA[I, 6]);}
      TDIST1 := TDIST1 + ROUTDATA[I, 1];
      TOTTIME := TOTTIME + ROUTDATA[I, 6];
    end;
{WRITELN('NPT1', NPT1, 'NPT', NPT, 'NARC', NARC, 'NTOURS', NTOURS);}
  WRITELN('TOTAL DISTANCE', TDIST1, '  ', 'TOTAL TIME ', TOTTIME);
  GETTIME(DATETIME);
  MINS := DATETIME.MINUTE * 60;
  SECS := DATETIME.SECOND;
  T0 := MINS + SECS;
  T2 := MINS + SECS;
  for J := 1 to NARC do
    begin
      while ID[J, 3] = 2 do
        begin
          ID[J, 7] := 0;
          ID[J, 8] := 0;
          goto 134;
        end;
      if ID[J, 3] = 0 then
        goto 134;
      if ID[J, 6] = 1 then
        goto 134;
      if ID[J, 1] <> 1 then
        goto 111;
      IPOINT := ID[J, 2];
      YYYZ := PTOUR[ID[J, 2], 1];
      ID[J, 5] := 0;
      ID[J, 4] := 0;
      ID[J, 7] := 0;
      ID[J, 8] := 0;
      CALC := 0;
      NLINKS := ROUTDATA[YYYZ, 3];
      MIDPOINT := ROUND(NLINKS / 2);
116:
      JPOINT := J;
      for I := J + 1 to NARC do
        begin
          if (CALC >= NLINKS) then
            goto 134;
          if ID[I, 3] = 0 then
            goto 114;
          if ID[I, 6] = 1 then
```

```
        goto 114;
      if (IPOINT <> ID[I, 1]) and (IPOINT <> ID[I, 2]) then
        goto 114;
      while IPOINT = ID[JPOINT, 2] do
        begin
          ID[JPOINT, 7] := I;
          ID[I, 8] := JPOINT;
          goto 3333;
        end;
      if ID[AHEAD, 1] <> 1 then
        ID[AHEAD, 7] := I;
      ID[I, 8] := BEHIND;
3333:
      AHEAD := I;
      BEHIND := I;
      CALC := CALC + 1;
      ID[I, 6] := 1;
      if CALC < MIDPOINT then
        ID[I, 4] := CALC
      else
        ID[I, 4] := NLINKS - CALC;
      if CALC < MIDPOINT then
        ID[I, 5] := 0
      else
        ID[I, 5] := 1;
      if IPOINT = ID[I, 1] then
        IPOINT := ID[I, 2]
      else
        IPOINT := ID[I, 1];
      goto 116;
114:
    end;
134:
  end;
111:
{WRITELN('      I    ', '  J    ', 'LNK STAT  ', 'ROUTE', '  #OF DEL ', 'DIREC', '  AHEAD', '  BEHIND');}
 for J := 1 to NARC do
  begin
{if ID[J, 3] <> 0 then}
{WRITELN(ID[J, 1], ID[J, 2], ID[J, 3], PTOUR[ID[J, 2], 1], ID[J, 4], ID[J, 5], ID[J, 7], ID[J, 8]);}
  end;
 for II := 1 to NTOURS do
  begin
   for J := 1 to NARC do
    begin
      if ID[J, 3] = 0 then
        goto 335;
      if II = PTOUR[ID[J, 2], 1] then
        goto 336
      else
        goto 335;
336:
      YYYZ := J;
      while ID[J, 3] = 2 do
        begin
          ID[YYYZ, 6] := Q[ID[YYYZ, 2]];
          if (ID[YYYZ, 2] > DELIVER) then
            ICAP := 0
          else
            ICAP := Q[ID[YYYZ, 2]];
          goto 388;
        end;
      goto 201;
335:
    end;
201:
  ICAP := 0;
  ICAP5 := 0;
  III3 := 0;
```

```
    III := 0;
    III2 := 0;
    III4 := 0;
383:
    while III3 <> 0 do
      begin
        IPOINT := ID[III4, 2];
{WRITELN('IPOINT', IPOINT, 'I', III4, ' ID [ 5 ] ', ID[III4, 5]);}
        if ID[III4, 5] = 0 then
          goto 388;
        ICAP5 := ICAP5 + Q[IPOINT];
        ID[YYYZ, 10] := ICAP5;
{WRITELN('LINK1', I, 'I,J', ID[I, 1], ID[I, 2], 'ID[I,9]', ID[I, 9], 'ID[I,10]', ID[I, 10]);}
        if IPOINT = ID[YYYZ, 1] then
          IPOINT := ID[YYYZ, 2]
        else
          IPOINT := ID[YYYZ, 1];
        goto 296;
      end;
    IPOINT := ID[YYYZ, 2];
{ID[YYYZ, 6] := ICAP;}
296:
    I := YYYZ;
    if ID[I, 5] = 0 then
      I := ID[I, 7]
    else
      I := ID[I, 8];
    repeat
{WRITELN('IPOINT', IPOINT, 'I', I, 'ID[5]', ID[I, 5]);}
      if (III3 <> 0) then
        ICAP5 := ICAP5 + Q[IPOINT]
      else
        ICAP := ICAP + Q[IPOINT];
      if (III3 <> 0) then
        ID[I, 10] := ICAP5
      else
        ID[I, 9] := ICAP;
{ID[I, 6] := ICAP;}
{WRITELN('LINK', I, 'I,J', ID[I, 1], ID[I, 2], 'ID[I,9]', ID[I, 9], 'ID[I,10]', ID[I, 10]);}
      if IPOINT = ID[I, 1] then
        IPOINT := ID[I, 2]
      else
        IPOINT := ID[I, 1];
      III := ID[I, 7];
      III2 := ID[I, 8];
      III4 := I;
      if ID[YYYZ, 5] = 0 then
        I := ID[I, 7]
      else
        I := ID[I, 8];
    until IPOINT = 1;
    if III > 0 then
      YYYZ := III
    else
      YYYZ := III2;
    while (ID[YYYZ, 5] = 1) and (IPOINT = 1) do
      begin
        III3 := 1;
        I := YYYZ;
        goto 383;
      end;
388:
    ROUTDATA[II, 7] := ICAP;
{WRITELN('ROUTE', II, 'LOAD', ROUTDATA[II, 7]);}
  end;
289:
  GETTIME(DATETIME);
  MINS := DATETIME.MINUTE * 60;
  SECS := DATETIME.SECOND;
```

```
  T3 := MINS + SECS;
  If T3 < T2 then
    T3 := T3 + 3600;
  ELAPS := T3 - T2;
{WRITELN('ELAPSED TIME TO SET UP LINKS', ELAPS);}
  WRITELN('      I      ', 'COST', '      LNKID', '   RADIAL SAV', '  ROUTE COST', '   DEL PENALTY');
  CMIN3 := 0;
  for K := DELIVER + 1 to DELIVER + BACKHAUL do
    begin
      CMIN1 := 9999;
      CMIN2 := 9999;
      CMIN3 := 0;
      CMIN4 := 0;
      CMIN5 := 0;
      for I := 9 to 10 do
        begin
          for L := 1 to NARC do
            begin
              If ID[L, 3] = 0 then
                goto 120;
              If ID[L, 3] = 2 then
                goto 120;
              If (ID[L, 3] = 2) and (ID[L, 2] > DELIVER) then
                goto 120;
              CC[K, 1] := ROUND(C1(X, Y, ID[L, 1], K) + C1(X, Y, K, ID[L, 2]) - C1(X, Y, ID[L, 1], ID[L, 2]) - CC[K, 4] + ROUND(PENALTY1
        * ID[L, I]));
              CC[K, 2] := L;
{If CC[K, 1] < 0 then}
{WRITELN(K, CC[K, 1], ID[CC[K, 2], 1], ID[CC[K, 2], 2], CC[K, 4], '      ', ROUND(C1(X,Y,ID[CC[K, 2], 1], K) + C1(X,Y,K, ID[CC[K, 2], 2]) -
       C1(X,Y,ID[CC[K, 2], 1], ID[CC[K, 2], 2])), '      ', ROUND(PENALTY1 * ID[CC[K, 2], I]));}
              If CMIN1 < CC[K, 1] then
                goto 120;
              CMIN1 := CC[K, 1];
              CMIN2 := CC[K, 2];
              CMIN4 := CC[K, 4];
              CMIN5 := I;
120:
            end;
          CC[K, 1] := CMIN1;
          CC[K, 2] := CMIN2;
          CC[K, 3] := K;
          CC[K, 4] := CMIN4;
          CC[K, 5] := CMIN5;
        end;
      WRITELN(K, CC[K, 1], ID[CC[K, 2], 1], ID[CC[K, 2], 2], CC[K, 4], '      ', ROUND(C1(X, Y, ID[CC[K, 2], 1], K) + C1(X, Y, K, ID[CC[K,
        2], 2]) - C1(X, Y, ID[CC[K, 2], 1], ID[CC[K, 2], 2])), '      ', ROUND(PENALTY1 * ID[CC[K, 2], CC[K, 5]]));
    end;
  GETTIME(DATETIME);
  MINS := DATETIME.MINUTE * 60;
  SECS := DATETIME.SECOND;
  T4 := MINS + SECS;
  If T4 < T3 then
    T4 := T4 + 3600;
  ELAPS := T4 - T3;
{  WRITELN('ELAPSED TIME ASSIGN BK TO NEAREST LINK', ELAPS);}
  NM1 := DELIVER + BACKHAUL - 1;
  for I := DELIVER + 1 to NM1 do
    begin
      IPLUS1 := I + 1;
      for J := IPLUS1 to DELIVER + BACKHAUL do
        begin
          If CC[I, 1] <= CC[J, 1] then
            goto 50;
          TEMP1 := CC[I, 1];
          TEMP2 := CC[I, 2];
          TEMP3 := CC[I, 3];
          TEMP4 := CC[I, 4];
          TEMP5 := CC[I, 5];
          CC[I, 1] := CC[J, 1];
```

```
        CC[I, 2] := CC[J, 2];
        CC[I, 3] := CC[J, 3];
        CC[I, 4] := CC[J, 4];
        CC[I, 5] := CC[J, 5];
        CC[J, 1] := TEMP1;
        CC[J, 2] := TEMP2;
        CC[J, 3] := TEMP3;
        CC[J, 4] := TEMP4;
        CC[J, 5] := TEMP5;
50:
      end;
    end;
  WRITELN('     I     ', 'COST', '     LNKID', '     K', '  NO OF DEL', '  REV COST', '  NODE DIRECT');

  for K := DELIVER + 1 to DELIVER + BACKHAUL do
    begin
      WRITELN(K, CC[K, 1], ID[CC[K, 2], 1], ID[CC[K, 2], 2], CC[K, 3], ID[CC[K, 2], 4], '     ', CC[K, 1] - ROUND(PENALTY1 * ID[CC[K, 2],
        CC[K, 5]]), CC[K, 5]);
{If CC[K, 1] < 0 then}
{CC[K, 1] := CC[K, 1] - ROUND(PENALTY1 * ID[CC[K, 2], CC[K, 5]]);}
    end;
  GETTIME(DATETIME);
  MINS := DATETIME.MINUTE * 60;
  SECS := DATETIME.SECOND;
  T5 := MINS + SECS;
  If T5 < T4 then
    T5 := T5 + 3600;
  ELAPS := T5 - T4;
{WRITELN('ELAPSED TIME TO SORT', ELAPS);}
  for I := 1 to NTOURS do
    begin
      ROUTDATA[I, 2] := 2;
      ROUTDATA[I, 4] := 0;
      ROUTDATA[I, 6] := ROUTDATA[I, 3];
      ROUTDATA[I, 10] := 0;
      ROUTDATA[I, 11] := 0;
    end;
  for I := 1 to NARC do
    begin
      ID[I, 6] := 0;
    end;
  DIST2 := 0;
  DIST1 := 0;
  REJECTED := 0;
  AFTER := 0;
  V := 0;
  NARC1 := NARC;
  for K := DELIVER + 1 to DELIVER + BACKHAUL do
    begin
      If ID[CC[K, 2], 3] = 0 then
        begin
          WRITELN('BKHAUL', CC[K, 3], 'INSERT LINK', ID[CC[K, 2], 1], ID[CC[K, 2], 2]);
          goto 144;
        end;
      XXXX := PTOUR[ID[CC[K, 2], 2], 1];
      WRITELN('BKHAUL', CC[K, 3], 'INSERT LINK', ID[CC[K, 2], 1], ID[CC[K, 2], 2]);
      WRITELN('ROUTE ', XXXX, ' ROUTE START ', ROUTDATA[XXXX, 5], ' RDIRECTION ', ROUTDATA[XXXX, 10]);
      while ROUTDATA[XXXX, 10] = 0 do
        begin
          If ROUTDATA[XXXX, 5] > 0 then
            goto 100;
          CAPSET(ID, PTOUR, Q, ROUTDATA, CC, K, NARC, NVT, TRCAP);
          goto 100;
        end;
{WRITELN(' CC[K, 5]', CC[K, 5], 'ROUTDATA[XXXX, 10] ', ROUTDATA[XXXX, 10]);}
      If CC[K, 5] <> ROUTDATA[XXXX, 10] then
        goto 117;
100:
      If ROUTDATA[XXXX, 5] = 0 then
```

```
      CAPSET(ID, PTOUR, Q, ROUTDATA, CC, K, NARC, NVT, TRCAP);
    If ROUTDATA[XXXX, 3] + 1 > MAXDROP then
      goto 133;
    If CC[K, 1] > 0 then
      goto 133;
    ICAP := TRCAP[NVT];
    P0 := CC[K, 2];
{WRITELN('P0', P0, 'ID[1]', ID[P0, 1], 'ID[2]', ID[P0, 2], 'ID[7]', ID[P0, 7], 'ID[8]', ID[P0, 8]);}
{WRITELN('ID[ID[7]]', ID[ID[P0, 7], 1], ID[ID[P0, 7], 2], 'ID[ID[8]]', ID[ID[P0, 8], 1], ID[ID[P0, 7], 2]);}
    NARC := NARC + 1;
    while ID[P0, 8] = 0 do
      begin
        ID[NARC, 1] := ID[P0, 1];
        goto 778;
      end;
{MAY BE BAD ASSUMPTION THAT "1" ALWAYS IN FIRST SPOT}
    while (ID[P0, 1] = 1) and (ID[P0, 7] = 0) do
      begin
        ID[NARC, 1] := ID[P0, 1];
        goto 778;
      end;
    If (ID[P0, 1] = ID[ID[P0, 8], 1]) or (ID[P0, 1] = ID[ID[P0, 8], 2]) then
      ID[NARC, 1] := ID[P0, 1]
    else
      ID[NARC, 1] := ID[P0, 2];
778:
    ID[NARC, 2] := CC[K, 3];
    ID[NARC, 3] := 1;
    ID[NARC, 5] := ID[P0, 5];
    ID[NARC, 9] := ID[P0, 9];
    ID[NARC, 10] := ID[P0, 10];
    If (ID[P0, 5] = 1) and (ID[P0, 1] = 1) and (ID[P0, 8] <> 0) then
      ID[ID[P0, 8], 7] := NARC + 1
    else If (ID[P0, 8] <> 0) then
      ID[ID[P0, 8], 7] := NARC;
    If (ID[P0, 5] = 1) and (ID[P0, 1] = 1) then
      ID[NARC, 7] := ID[P0, 7]
    else
      ID[NARC, 7] := NARC + 1;
    If (ID[P0, 5] = 1) and (ID[P0, 1] = 1) then
      ID[NARC, 8] := NARC + 1
    else
      ID[NARC, 8] := ID[P0, 8];
    If (ID[P0, 3] <> 2) and (ROUTDATA[XXXX, 5] = CC[K, 2]) and (ID[P0, 1] = 1) then
      ROUTDATA[XXXX, 5] := NARC;
{WRITELN('NARC', NARC, 'ID[1],ID[2]', ID[NARC, 1], ID[NARC, 2]);}
    NARC := NARC + 1;
    ID[NARC, 1] := CC[K, 3];
    while ID[P0, 7] = 0 do
      begin
        ID[NARC, 2] := ID[P0, 2];
        goto 777;
      end;
{MAY BE BAD ASSUMPTION}
    while (ID[P0, 1] = 1) and (ID[P0, 8] = 0) do
      begin
        ID[NARC, 2] := ID[P0, 2];
        goto 777;
      end;
    If (ID[P0, 2] = ID[ID[P0, 7], 1]) or (ID[P0, 2] = ID[ID[P0, 7], 2]) then
      ID[NARC, 2] := ID[P0, 2]
    else
      ID[NARC, 2] := ID[P0, 1];
777:
    ID[NARC, 3] := 1;
    ID[NARC, 5] := ID[P0, 5];
    ID[NARC, 9] := ID[P0, 9];
    ID[NARC, 10] := ID[P0, 10];
    If (ID[P0, 5] = 1) and (ID[P0, 1] = 1) and (ID[P0, 7] <> 0) then
```

```
      ID[ID[P0, 7], 8] := NARC + 1
    else if (ID[P0, 7] <> 0) then
      ID[ID[P0, 7], 8] := NARC;
    if (ID[P0, 5] = 1) and (ID[P0, 1] = 1) then
      ID[NARC, 7] := NARC - 1
    else
      ID[NARC, 7] := ID[P0, 7];
    if (ID[P0, 5] = 1) and (ID[P0, 1] = 1) then
      ID[NARC, 8] := ID[P0, 8]
    else
      ID[NARC, 8] := NARC - 1;
    if (ID[P0, 3] <> 2) and (ROUTDATA[XXXX, 5] = CC[K, 2]) and (ID[P0, 2] = 1) then
      ROUTDATA[XXXX, 5] := NARC;
    ID[P0, 3] := ID[P0, 3] - 1;
{WRITELN('NARC', NARC, 'ID[1],ID[2]', ID[NARC, 1], ID[NARC, 2]);}
    CAPCHK(ID, PTOUR, Q, ROUTDATA, CC, K, NARC, NVT, TRCAP, V);
    while V = 1 do
      begin
        ID[P0, 3] := ID[P0, 3] + 1;
        ID[NARC, 1] := 0;
        ID[NARC, 2] := 0;
        ID[NARC, 3] := 0;
        ID[NARC, 5] := 0;
        ID[NARC, 7] := 0;
        ID[NARC, 8] := 0;
        ID[NARC, 9] := 0;
        ID[NARC, 10] := 0;
        if (ROUTDATA[XXXX, 5] = NARC) then
          ROUTDATA[XXXX, 5] := CC[K, 2];
        NARC := NARC - 1;
        ID[NARC, 1] := 0;
        ID[NARC, 2] := 0;
        ID[NARC, 3] := 0;
        ID[NARC, 5] := 0;
        ID[NARC, 7] := 0;
        ID[NARC, 8] := 0;
        ID[NARC, 9] := 0;
        ID[NARC, 10] := 0;
        if (ROUTDATA[XXXX, 5] = NARC) then
          ROUTDATA[XXXX, 5] := CC[K, 2];
        NARC := NARC - 1;
        while (ID[P0, 5] = 1) and (ID[P0, 1] = 1) do
          begin
            ID[ID[P0, 7], 8] := NARC;
            goto 133;
          end;
        while (ID[P0, 5] = 0) and (ID[P0, 1] = 1) do
          begin
            ID[ID[P0, 8], 7] := NARC;
            goto 133;
          end;
        goto 133;
      end;
    WRITELN('CONNECT BACKHAUL', CC[K, 3]);
    ID[P0, 3] := ID[P0, 3] + 1;
    P1 := PTOUR[ID[P0, 2], 1];
    P2 := PTOUR[ID[P0, 2], 2];
    P3 := PTOUR[ID[P0, 2], 3];
    P4 := PTOUR[ID[P0, 2], 4];
    NARC := NARC - 1;
    ID[NARC, 4] := ID[P0, 4];
    ID[NARC, 9] := ID[P0, 9];
    ID[NARC, 10] := ID[P0, 10];
    NARC := NARC + 1;
    ID[NARC, 4] := ID[P0, 4];
    ID[NARC, 5] := ID[P0, 5];
    ID[NARC, 9] := ID[P0, 9];
    ID[NARC, 10] := ID[P0, 10];
    ID[P0, 3] := ID[P0, 3] - 1;
```

```
{DISCONNECT BKHAUL LINK}
{WRITELN(NARC1, NPT1, CC[K, 3], 'ID[1..2]', ID[NARC1 - (NPT1 - CC[K, 3]), 1], ID[NARC1 - (NPT1 - CC[K, 3]), 2]);}
      ID[NARC1 - (NPT1 - CC[K, 3]), 3] := 0;
{WRITELN('ROUTDATA[BKHAUL]', ROUTDATA[PTOUR[ID[NARC1 - (NPT1 - CC[K, 3]), 2], 1], 9]);}
      ROUTDATA[PTOUR[ID[NARC1 - (NPT1 - CC[K, 3]), 2], 1], 9] := 3;
      PTOUR[CC[K, 3], 1] := P1;
      PTOUR[CC[K, 3], 2] := P2;
      PTOUR[CC[K, 3], 3] := P3;
      PTOUR[CC[K, 3], 4] := P4 + 1;
      PTOUR[ID[P0, 1], 4] := P4 + 1;
      PTOUR[ID[P0, 2], 4] := P4 + 1;
      TDIST2 := TDIST2 + ROUND(CC[K, 1]);
      TOTTIME1 := TOTTIME1 + ROUND(CC[K, 1] / MILMIN);
      WRITELN('BKHAUL', CC[K, 3], ' INSERTION SAVINGS ', CC[K, 1], ' LNKID ', ID[CC[K, 2], 1], ID[CC[K, 2], 2]);
      If ROUTDATA[XXXX, 10] = 0 then
        ROUTDATA[P1, 2] := CC[K, 5];
      If ID[P0, CC[K, 5]] > ROUTDATA[XXXX, 11] then
        ROUTDATA[XXXX, 11] := ID[P0, CC[K, 5]];
      If ID[P0, 4] > ROUTDATA[XXXX, 4] then
        ROUTDATA[XXXX, 4] := ID[P0, 4];
      WRITELN('ROUTE', XXXX, 'BKHAUL', CC[K, 3], 'NO OF DEL AFTER PICKUP', ROUTDATA[XXXX, 4]);
      ROUTDATA[XXXX, 3] := ROUTDATA[XXXX, 3] + 1;
      ROUTDATA[XXXX, 8] := ROUTDATA[XXXX, 8] + ROUND((((1 + INFLATION1) * C1(X, Y, 1, CC[K, 3])) + CC[K, 1]) / MILMIN);
{WRITELN('ROUT[,8]', ROUTDATA[XXXX, 8], 'C1(X,Y,1,CC[K,3])', (1 + INFLATION1) * C1(X,Y,1, CC[K, 3]), 'CC[K,1]', CC[K, 1]);}
      goto 118;

117:
      If (CC[K, 5] = 10) then
        CC[K, 1] := CC[K, 1] + ROUND(PENALTY1 * ID[CC[K, 2], 9])
      else
        CC[K, 1] := CC[K, 1] + ROUND(PENALTY1 * ID[CC[K, 2], 10]);
{CMIN1 := CC[K, 1];}
{CMIN2 := CC[K, 2];}
{CMIN3 := CC[K, 3];}
      CMIN1 := 9999;
      CMIN2 := 9999;
      CMIN3 := 0;
      goto 145;
144:
      CMIN1 := 9999;
      CMIN2 := 9999;
      CMIN3 := 0;
      CMIN4 := 0;
      CMIN5 := 0;
145:
{WRITELN('CMIN1', CMIN1, 'CMIN2', CMIN2, 'CMIN5', CMIN5);}
      for J := 9 to 10 do
        begin
          for L := 1 to NARC do
            begin
              If (ID[L, 3] = 2) and (ID[L, 2] > DELIVER) then
                goto 115;
              If ID[L, 3] = 0 then
                goto 115;
              If ID[L, 3] = 2 then
                goto 115;

              TEMPX := PTOUR[ID[L, 2], 1];
              If L = CC[K, 2] then
                goto 115;
{WRITELN('ID[1]', ID[L, 1], 'ID[2]', ID[L, 2], 'L', L, 'TEMPX', TEMPX);}
              If (ROUTDATA[TEMPX, 10] = 0) then
                CC[K, 1] := ROUND(C1(X, Y, ID[L, 1], CC[K, 3]) + C1(X, Y, CC[K, 3], ID[L, 2]) - C1(X, Y, ID[L, 1], ID[L, 2]) - CC[K, 4] +
      ROUND(PENALTY1 * ID[L, J]));
              If (ROUTDATA[TEMPX, 10] <> 0) then
                CC[K, 1] := ROUND(C1(X, Y, ID[L, 1], CC[K, 3]) + C1(X, Y, CC[K, 3], ID[L, 2]) - C1(X, Y, ID[L, 1], ID[L, 2]) - CC[K, 4] +
      ROUND(PENALTY1 * ID[L, ROUTDATA[TEMPX, 10]]));
              If CMIN1 < CC[K, 1] then
                goto 115;
```

```
            CMIN1 := CC[K, 1];
            CMIN2 := L;
            If (ROUTDATA[TEMPX, 10] = 0) then
              CMIN5 := J
            else
              CMIN5 := ROUTDATA[TEMPX, 10];
{WRITELN('L', L, 'J', J, 'CMIN1', CMIN1, 'CMIN2', CMIN2, 'CMIN5', CMIN5);}
115:
          end;
        end;
      CC[K, 1] := CMIN1;
      CC[K, 2] := CMIN2;
      CC[K, 5] := CMIN5;
      TEMPXX := PTOUR[ID[CMIN2, 2], 1];

{if CC[K, 1] < 0 then}
{CC[K, 1] := CC[K, 1] - ROUND(PENALTY1 * ID[CC[K, 2], CC[K, 5]]);}

{WRITELN('NUMBER OF DEL', ID[CMIN2, ROUTDATA[TEMPXX, 10]]);}
{WRITELN('CMIN1', CMIN1, 'CMIN2', CMIN2, 'CMIN5', CMIN5, 'TEMPXX', TEMPXX);}
{WRITELN('ID[CC[K, 2], CC[K, 5]]', ID[CC[K, 2], CC[K, 5]]);}
{WRITELN('TEST CC[K,1]', CC[K, 1] - ROUND(PENALTY1 * ID[CC[K, 2], CC[K, 5]]));}
{CC[K, 1] := CC[K, 1] - ROUND(PENALTY1 * ID[CC[K, 2], CC[K, 5]]);}
      XXXX := TEMPXX;
      WRITELN('REDIRECT', CC[K, 3], 'LINK ID', ID[CC[K, 2], 1], ID[CC[K, 2], 2], 'CC[K,1]', CC[K, 1], 'ROUTE', XXXX);
      If (ID[CMIN2, 5] = ROUTDATA[TEMPXX, 2]) or (ROUTDATA[TEMPXX, 2] = 2) then
        goto 100;
      ID[CMIN2, 4] := ROUTDATA[TEMPXX, 6] - ID[CC[K, 2], 4];
      goto 100;
133:
      REJECTED := REJECTED + 1;
      WRITELN('REJECT BACKHAUL', CC[K, 3]);
      REJARR[REJECTED + 1, 1] := CC[K, 3];
      REJARR[REJECTED + 1, 2] := ROUND(X[CC[K, 3]]);
      REJARR[REJECTED + 1, 3] := ROUND(Y[CC[K, 3]]);
118:
    end;
  GETTIME(DATETIME);
  MINS := DATETIME.MINUTE * 60;
  SECS := DATETIME.SECOND;
  T6 := MINS + SECS;
  If T6 < T5 then
    T6 := T6 + 3600;
  ELAPS := T6 - T5;
{   WRITELN('ELAPSED TIME TO INSERT BKHAULS', ELAPS);}
{     while ID[P0, 7] = 0 do}
  begin
{ID[NARC, 2] := ID[P0, 2];}
{writeln('ID[NARC, 2] ', ID[NARC, 2], 'ID[P0, 2]', ID[P0, 2]);}
{     goto 777;}
  end;
  GETTIME(DATETIME);
  MINS := DATETIME.MINUTE * 60;
  SECS := DATETIME.SECOND;
  T1 := MINS + SECS;
  If T1 < T0 then
    T1 := T1 + 3600;
  ELAPS := T1 - T0;
  NTOUR := NTOURS;
  CAPREV(ID, PTOUR, Q, ROUTDATA, CC, K, NARC, NVT, TRCAP, NTOUR);
{WRITELN('REVISED ARCS');}
{   WRITELN('     I  ',' J  ', 'ROUTE',' #OF DEL ', 'DIREC',' CAP',' AHEAD',' BEHIND');}
  for J := 1 to NARC do
    begin
{if ID[J, 3] <> 0 then}
{WRITELN(ID[J, 1], ID[J, 2], PTOUR[ID[J, 2], 1], ID[J, 4], ID[J, 5], ID[J, 6], ID[J, 7], ID[J, 8]);}
    end;
  AFTER := 0;
  AFTER1 := 0;
```

```
  for I := 1 to NTOURS do
    begin
      AFTER := AFTER + ROUTDATA[I, 4];
      AFTER1 := AFTER1 + ROUTDATA[I, 11];
      If ROUTDATA[I, 11] > 0 then
        WRITELN('ROUTE', I, 'DEL AFTER', ROUTDATA[I, 4], 'LOAD DEL AFTER', ROUTDATA[I, 11]);
    end;
  WRITELN('TOTAL ELAPSED TIME IS', ELAPS, '  SECONDS');
  WRITELN('NODES', DELIVER + BACKHAUL);
  WRITELN('ARCS', NARC);
  WRITELN('BACKHAUL', BACKHAUL);
  WRITELN('PENALTY(PERCENT)', PENALTY);
  WRITELN('TOTAL DISTANCE IS', (TDIST1 + TDIST2), 'UNITS');
  WRITELN('BACKHAUL INSERTION SAVINGS', -TDIST2);
  INCREASE := -TDIST2 / TDIST1;
  WRITELN('% DECREASE IN DISTANCE', ROUND(INCREASE * 100));
  WRITELN('TOTAL TRAVEL TIME IS', (TOTTIME + TOTTIME1), 'MINUTES');
  WRITELN('BACKHAUL INSERTION TRAVEL TIME SAVINGS', -TOTTIME1);
  INCREASE := 0;
  INCREASE := -TOTTIME1 / TOTTIME;
  WRITELN('% DECREASE IN TRAVEL TIME', ROUND(INCREASE * 100));
  WRITELN('TOTAL # OF REJECTED BACKHAULS', REJECTED);
  WRITELN('TOTAL # OF DELIVERIES AFTER PICKUPS', AFTER);
  WRITELN('TOTAL LOAD OF DELIVERIES AFTER PICKUPS', AFTER1);
  for III := 1 to NTOUR do
    begin
      ICAP := 0;
      ICAP1 := 0;
      ICAP2 := 0;
      JJKK := III;
      YYYZ := ROUTDATA[JJKK, 5];
      If ROUTDATA[III, 9] = 3 then
        goto 188;
      while YYYZ = 0 do
        begin
          for J := 1 to NARC do
            begin
              If ID[J, 3] = 0 then
                goto 197;
              If JJKK = PTOUR[ID[J, 2], 1] then
                goto 195
              else
                goto 197;
195:
              YYYZ := J;
              IPOINT := ID[J, 2];
              while ID[J, 3] = 2 do
                begin
                  ICAP := ICAP + Q[IPOINT];
                  If Q[IPOINT] >= 0 then
                    ICAP1 := ICAP1 + Q[IPOINT]
                  else
                    ICAP2 := ICAP2 - Q[IPOINT];
                  goto 935;
                end;
              goto 1011;
197:
            end;
        end;
1011:
      I := YYYZ;
      IPOINT := ID[I, 2];
      while ID[I, 3] = 2 do
        begin
          ICAP := ICAP + Q[IPOINT];
          If Q[IPOINT] >= 0 then
            ICAP1 := ICAP1 + Q[IPOINT]
          else
            ICAP2 := ICAP2 - Q[IPOINT];
```

```
        goto 935;
      end;
    repeat
      ICAP := ICAP + Q[IPOINT];
      if Q[IPOINT] >= 0 then
        ICAP1 := ICAP1 + Q[IPOINT]
      else
        ICAP2 := ICAP2 - Q[IPOINT];
      if ID[YYYZ, 5] = 0 then
        I := ID[I, 7]
      else
        I := ID[I, 8];
      if IPOINT = ID[I, 1] then
        IPOINT := ID[I, 2]
      else
        IPOINT := ID[I, 1];
    until IPOINT = 1;
935:
    ROUTDATA[III, 1] := ICAP;
    ROUTDATA[III, 2] := ICAP1;
    ROUTDATA[III, 3] := ICAP2;
188:
  end;
 ISUM := NTOURS;
 WRITELN(' ');
{ WRITELN('    ROUTE', ' NO OF DEL', ' NO OF PICKUPS', ' TRIP DURATION', ' OVERTIME');}
 for J := 1 to ISUM do
  begin
   if ROUTDATA[J, 9] = 3 then
     goto 1313;
   CCXX := ROUTDATA[J, 8] - MAXTIME * 60;
   if CCXX <= 0 then
     CCXX := 0;
   CCXX2 := CCXX2 + CCXX;
{  WRITELN(J, '  ', ROUTDATA[J, 2], '  ', ROUTDATA[J, 3], '          ', ROUTDATA[J, 8], '    ', CCXX);}
1313:
  end;
 WRITELN('TOTAL OVERTIME PENALTY(IN DIST UNITS)', ROUND(CCXX2 * MILMIN * PERCENT1));
 XXX := 0;
{WRITELN('REJ NODE  ', 'REJ X COORD ', ' REJ Y COORD ');}
 for I := 1 to REJECTED + 1 do
  begin
{WRITELN(REJARR[I, 1], '   ', REJARR[I, 2], '   ', REJARR[I, 3]);}
  end;
1213:
{writeln('enter node 1', ' enter node 2');}
{readln(I, J);}
{  if i = 0 then}
{  goto 1212;}
{  C1(X,Y,I, J) := ROUND(SQRT(SQR(X[I] - X[J]) + SQR(Y[I] - Y[J])));}
{   writeln('distance for ', I, ' and ', J, ' is ', C1(X,Y,I, J));}
{  goto 1213;}
 I := 0;
1212:
 WRITELN('ENTER NEW GRAPH SCALE FOR SPECIFIC ROUTES(enter 0 for none, or integer)');
 readln(newscale);
 if newscale = 0 then
  goto 301;
38:
 writeln('ENTER ROUTE NUMBER DESIRED(enter 0 for none,enter 100 for all)');
 readln(tours);
 if tours = 0 then
  goto 301;
 if XXX = 0 then
  goto 39;
 writeln('ENTER 0 IF YOU WOULD LIKE PREVIOUS TOUR ERASED');
 writeln('ENTER 1 OTHERWISE');
 readln(YYY);
 if YYY = 1 then
```

```
    goto 40;
39:
  PaintCircle(0, 0, 500);
  InvertCircle(0, 0, 500);
  XXX := 1;
  NETGEN(X, Y, NPT1, NPT, newscale);
40:
  for J := 1 to NARC do
    begin
      if (ID[J, 3] = 0) then
        goto 36;
      if (PTOUR[ID[J, 2], 1] = 0) then
        goto 36;
      P1 := ROUND(X[ID[J, 1]]) * (newscale);
      P2 := ROUND(Y[ID[J, 1]]) * newscale;
      P3 := ROUND(X[ID[J, 2]]) * (newscale);
      P4 := ROUND(Y[ID[J, 2]]) * newscale;
      if (ID[J, 3] = 2) and (ID[J, 2] > DELIVER) then
        goto 36;
      DRAWLINE(P1, P2, P3, P4);
36:
    end;
  if tours = 100 then
    goto 301;
  goto 38;
301:
  cwvrp(REJARR, REJECTED, SAVINGS);
  WRITELN('TOTAL DISTANCE INCL RADIALS IS', (TDIST1 + TDIST2), 'UNITS');
  if SAVINGS > 0 then
    begin
      WRITELN('TOTAL DEDICATED BKHAUL SAVINGS IS', SAVINGS, 'UNITS');
      WRITELN('NEW TOTAL DISTANCE IS', (TDIST1 + TDIST2) - SAVINGS, 'UNITS');
    end;
end.
```

```
unit BASECASE;
interface
 type
  MATRIX1 = array[1..1000, 1..10] of INTEGER;
  MATRIX2 = array[1..100, 1..5] of INTEGER;
  COUNTER = INTEGER;
  VECT100 = array[1..100] of REAL;
  IVECT100 = array[1..100] of INTEGER;
  MATRIX5 = array[1..5] of INTEGER;
  REAL1 = REAL;
 procedure BASE (var NPT1: COUNTER; var NPT: COUNTER; var NARC: COUNTER; var NTOURS: COUNTER; var MAXDROP: COUNTER;
     var NVT: COUNTER; var MAXTIME: COUNTER; var MILMIN: REAL1; var ID: MATRIX1; var PTOUR: MATRIX2; var Q: IVECT100;
     var X: VECT100; var Y: VECT100; var TRCAP: MATRIX5);
implementation
 procedure BASE (var NPT1: COUNTER; var NPT: COUNTER; var NARC: COUNTER; var NTOURS: COUNTER; var MAXDROP: COUNTER;
     var NVT: COUNTER; var MAXTIME: COUNTER; var MILMIN: REAL1; var ID: MATRIX1; var PTOUR: MATRIX2; var Q: IVECT100;
     var X: VECT100; var Y: VECT100; var TRCAP: MATRIX5);
  label
   6767, 1010, 102, 998, 776, 17, 16, 10, 64, 105, 11, 777, 97, 12, 301, 98, 90, 35, 36, 38, 39, 40, 91;
  type
   COUNTER = INTEGER;
   VECTOR = array[1..1000] of real;
   IVECTOR = array[1..1000] of INTEGER;
   IVECT100 = array[1..100] of INTEGER;
   MATRIX3 = array[1..100, 1..8] of INTEGER;
   MATRIX4 = array[1..100, 1..3] of INTEGER;
   MATRIX5 = array[1..5] of INTEGER;
   COUNTER2 = INTEGER;
   REAL2 = REAL;
   DateTimeRec = record
      Year, Month, Day, Hour, Minute, Second, DayOfWeek: integer
     end;
   OUTREC = record
      NO1: INTEGER;
      NO2: INTEGER;
      NO3: INTEGER;
      NO4: INTEGER;
      NO5: INTEGER;
      NO6: INTEGER;
      NO7: INTEGER;
      NO8: REAL
     end;
  var
   DateTime: DateTimeRec;
   TOTTIME, SUM1: EXTENDED;
   SPEED: REAL;
   T0, T1, T2, Elaps2, Elaps, Hrs, Mins, Secs: integer;
   NN, IC, SCALE, BKPTR, tours, XXX, YYY, newscale, NARC2: COUNTER;
   S: VECTOR;
   IX, JX: IVECTOR;
   JO, K, P, LIN, LJN, IPT1, IPT2, ICAP, MCAP, LM, L, M, ITK, HIT, KT, IT, P1, P2, P3, P4: INTEGER;
   TDIST, SUM, DISTAN: REAL;
   D: array[1..100] of real;
   ADJA: array[1..100, 1..100] of integer;
   TRA, TRU: MATRIX5;
   POINT: array[1..100] of INTEGER;
   BX, BY: array[1..100] of INTEGER;
   I, WIDTH, HEIGHT, DIVISOR, II, NT, J, IP1, IA, IB, ISUM: INTEGER;
   TCAP: COUNTER;
 procedure GETTIME (var DATETIME: DATETIMEREC);
 EXTERNAL;
 procedure STHEAP (var NN: COUNTER; var S: VECTOR; var IX, JX: IVECTOR);
  label
   1, 50, 60, 2, 5, 7, 9;
  var
   NH, II, I, IXC, JXC, J: INTEGER;
   DT, COPY: REAL;
 begin
  sh := 0;
```

```
    ii := 0;
    i := 0;
    ixc := 0;
    jxc := 0;
    j := 0;
    dt := 0;
    copy := 0;
    if (NN - 2) < 0 then
      goto 50;
    if (NN - 2) = 0 then
      goto 60;
    if (NN - 2) > 0 then
      goto 1;
1:
    NH := ROUND(NN / 2);
    for I := 1 to NH do
      begin
        II := NH + 1 - I;
        COPY := S[II];
        IXC := IX[II];
        JXC := IXC;
2:
        J := II + II;
        if (J - NN) < 0 then
          goto 5;
        if (J - NN) = 0 then
          goto 7;
        if (J - NN) > 0 then
          goto 9;
5:
        if S[J + 1] > S[J] then
          J := J + 1;
7:
        if S[J] <= COPY then
          goto 9;
        S[II] := S[J];
        IX[II] := IX[J];
        JX[IX[J]] := II;
        II := J;
        goto 2;
9:
        S[II] := COPY;
        IX[II] := IXC;
        JX[JXC] := II;
      end;
    goto 50;
60:
    if S[1] >= S[2] then
      goto 50;
    DT := S[1];
    S[1] := S[NN];
    S[2] := DT;
    IX[1] := 2;
    JX[1] := 2;
    IX[2] := 1;
    JX[2] := 1;
50:
  end;
  procedure OTHEAP (var NN: COUNTER; var S: VECTOR; var IX: IVECTOR; var JX: IVECTOR; IC: COUNTER);

    label
      13, 111, 112, 19;
    var
      II, IXC, JXC, J: INTEGER;
      COPY: REAL;
  begin
    ii := 0;
    ixc := 0;
    jxc := 0;
```

```
    J := 0;
    copy := 0;
    II := IC;
    COPY := S[II];
    IXC := IX[II];
    JXC := IXC;
13:
    J := II + II;
    If (J - NN) < 0 then
      goto 111;
    If (J - NN) = 0 then
      goto 112;
    If (J - NN) > 0 then
      goto 19;
111:
    If S[J + 1] > S[J] then
      J := J + 1;
112:
    If S[J] <= COPY then
      goto 19;
    S[II] := S[J];
    IX[II] := IX[J];
    JX[IX[J]] := II;
    II := J;
    goto 13;
19:
    S[II] := COPY;
    IX[II] := IXC;
    JX[JXC] := II;
  end;
  procedure NETGEN (var X, Y: VECT100; var NPT, SCALE, BKPTR, newscale: COUNTER);
    label
      333, 222, 223;
    var
      i, I, j, NODES, backhaul, NPTB: integer;
      r, w: real;
      yy: array[1..200] of real;
      TEST: INTEGER;
      concentration: integer;
      conc: real;
  begin
    concentration := 0;
    conc := 0;
    i := 0;
    I := 0;
    j := 0;
    nodes := 0;
    backhaul := 0;
    nptb := 0;
    for i := 1 to 100 do
      begin
        yy[i] := 0;
      end;
    I := 0;
    If newscale > 0 then
      goto 333;
{newscale := 1;}
    writeln('ENTER NO OF NODES(<=62)');
    readln(nodes);
    writeln('ENTER SCALE OF GRAPH');
    readln(scale);
    writeln('ENTER PERCENTAGE OF BACKHAUL LOCATIONS(18%)');
    readln(backhaul);
    NPT := NODES;
    NPTB := ROUND(NODES * (backhaul / 100));
    BKPTR := NPT - NPTB + 1;
    TEST := 0;
{WRITELN('USE TEST NODES?(1-YES,0-NO)');}
{READLN(TEST);}
```

```
    while TEST = 1 do
      begin
        X[1] := 30;
        Y[1] := 40;
        X[51] := 8;
        Y[51] := 52;
        X[2] := 5;
        Y[2] := 64;
        X[3] := 17;
        Y[3] := 63;
        X[4] := 27;
        Y[4] := 68;
        X[16] := 31;
        Y[16] := 62;
        X[5] := 37;
        Y[5] := 69;
        X[6] := 43;
        Y[6] := 67;
        X[7] := 52;
        Y[7] := 64;
        X[9] := 62;
        Y[9] := 63;
        X[8] := 63;
        Y[8] := 69;
        X[14] := 16;
        Y[14] := 57;
        X[15] := 25;
        Y[15] := 55;
        X[11] := 42;
        Y[11] := 57;
        X[10] := 57;
        Y[10] := 58;
        X[12] := 37;
        Y[12] := 52;
        X[13] := 21;
        Y[13] := 47;
        X[17] := 30;
        Y[17] := 48;
        X[18] := 38;
        Y[18] := 46;
        X[20] := 49;
        Y[20] := 49;
        X[30] := 58;
        Y[30] := 48;
        X[46] := 7;
        Y[46] := 38;
        X[47] := 12;
        Y[47] := 42;
        X[25] := 32;
        Y[25] := 39;
        X[19] := 42;
        Y[19] := 41;
        X[21] := 52;
        Y[21] := 41;
        X[31] := 62;
        Y[31] := 42;
        X[50] := 17;
        Y[50] := 33;
        X[48] := 25;
        Y[48] := 32;
        X[24] := 31;
        Y[24] := 32;
        X[23] := 40;
        Y[23] := 30;
        X[22] := 45;
        Y[22] := 35;
        X[29] := 56;
        Y[29] := 37;
        X[28] := 52;
```

```
        Y[28] := 33;
        X[32] := 61;
        Y[32] := 33;
        X[45] := 5;
        Y[45] := 25;
        X[49] := 20;
        Y[49] := 26;
        X[26] := 48;
        Y[26] := 28;
        X[33] := 58;
        Y[33] := 27;
        X[38] := 27;
        Y[38] := 23;
        X[39] := 32;
        Y[39] := 22;
        X[27] := 51;
        Y[27] := 21;
        X[44] := 10;
        Y[44] := 17;
        X[43] := 5;
        Y[43] := 6;
        X[42] := 13;
        Y[42] := 13;
        X[41] := 21;
        Y[41] := 10;
        X[40] := 30;
        Y[40] := 15;
        X[37] := 36;
        Y[37] := 16;
        X[36] := 39;
        Y[36] := 10;
        X[35] := 46;
        Y[35] := 10;
        X[34] := 50;
        Y[34] := 15;
        X[52] := 15;
        Y[52] := 51;
        X[53] := 52;
        Y[53] := 55;
        X[54] := 12;
        Y[54] := 25;
        X[55] := 22;
        Y[55] := 17;
        X[56] := 46;
        Y[56] := 24;
        X[57] := 31;
        Y[57] := 11;
        X[58] := 56;
        Y[58] := 5;
        X[59] := 14;
        Y[59] := 27;
        X[60] := 7;
        Y[60] := 44;
        X[61] := 27;
        Y[61] := 50;
        X[62] := 66;
        Y[62] := 60;
        goto 333;
      end;
    writeln('ENTER DELIVERY CONCENTRATION(60%)');
    readln(concentration);
    CONC := (concentration / 100);
{   WRITELN('BKPTR', BKPTR, 'CONC', CONC);}
    repeat
      I := I + 1;
      r := random;
      w := ABS(r / 32767);
      if I >= (BKPTR * 2) then
        goto 222;
```

```
      yy[i] := SCALE * (1 - CONC) + (w * (2 * (SCALE * CONC) - SCALE));
      goto 223;
222:
      yy[i] := (w * SCALE);
223:
{WRITELN('I', I, 'YY[I]', ROUND(YY[I]));}
    until I = nodes * 2;
    DRAWLINE(0, 100, 2, 100);
    DRAWLINE(100, 0, 100, 2);
    DRAWLINE(0, 200, 2, 200);
    DRAWLINE(200, 0, 200, 2);
    DRAWLINE(0, 300, 2, 300);
    DRAWLINE(300, 0, 300, 2);
{BEGIN TO COORD TO ARRAY}
    i := 1;
    j := 1;
    repeat
      X[i] := yy[j];
      i := i + 1;
      j := j + 2
    until j > NODES * 2;
    i := 1;
    j := 2;
    repeat
      Y[i] := yy[j];
      i := i + 1;
      j := j + 2
    until j > NODES * 2;
333:
{newscale := 1;}
    I := 1;
    P1 := ROUND(X[I] * newscale);
    P2 := ROUND(Y[I] * newscale);
{writeln('node', I, P1, P2);}
    PaintCircle(P1, P2, 5);
    Moveto(P1 - 45, P2);
{WriteDraw(I);}
    for I := 2 to (BKPTR - 1) do
      begin
        P1 := ROUND(X[I] * newscale);
        P2 := ROUND(Y[I] * newscale);
{writeln('node', I, P1, P2);}
        drawline(P1 - 3, P2 - 3, P1 + 3, P2 - 3);
        drawline(P1 - 3, P2 + 3, P1 + 3, P2 + 3);
        drawline(P1 - 3, P2 - 3, P1 - 3, P2 + 3);
        drawline(P1 + 3, P2 - 3, P1 + 3, P2 + 3);
        Moveto(P1 - 15, P2);
{WriteDraw(I);}
      end;
    NODES := NPT;
    for I := BKPTR to NODES do
      begin
        P1 := ROUND(X[I] * newscale);
        P2 := ROUND(Y[I] * newscale);
{writeln('node', I, P1, P2);}
        drawline(P1, P2 - 4, P1 - 4, P2 + 4);
        drawline(P1, P2 - 4, P1 + 4, P2 + 4);
        drawline(P1 - 4, P2 + 4, P1 + 4, P2 + 4);
        Moveto(P1 - 15, P2);
{WriteDraw(I);}
      end;
  end;
begin
  I := 0;
  j := 0;
  T0 := 0;
  T1 := 0;
  Elaps := 0;
  Hrs := 0;
```

```
    Mins := 0;
    Secs := 0;
    NN := 0;
    IC := 0;
    NPT := 0;
    SCALE := 0;
    BKPTR := 0;
    tours := 0;
    XXX := 0;
    YYY := 0;
    for i := 1 to 1000 do
      begin
        s[i] := 0;
      end;
    for i := 1 to 1000 do
      begin
        ix[i] := 0;
      end;
    for i := 1 to 1000 do
      begin
        jx[i] := 0;
      end;
    NARC := 0;
    JO := 0;
    K := 0;
    P := 0;
    LIN := 0;
    LJN := 0;
    IPT1 := 0;
    IPT2 := 0;
    ICAP := 0;
    MCAP := 0;
    LM := 0;
    L := 0;
    M := 0;
    ITK := 0;
    HIT := 0;
    KT := 0;
    IT := 0;
    P1 := 0;
    P2 := 0;
    P3 := 0;
    P4 := 0;
    TDIST := 0;
    SUM := 0;
    for i := 1 to 62 do
      begin
        d[i] := 0;
      end;
    for i := 1 to 62 do
      begin
        x[i] := 0;
      end;
    for i := 1 to 62 do
      begin
        y[i] := 0;
      end;
    for i := 1 to 900 do
      begin
        for j := 1 to 10 do
          begin
            id[i, j] := 0;
          end;
      end;
    for i := 1 to 62 do
      begin
        for j := 1 to 62 do
          begin
            adja[i, j] := 0;
```

```
        end;
      end;
    for i := 1 to 5 do
      begin
        tra[i] := 0;
      end;
    for i := 1 to 5 do
      begin
        trcap[i] := 0;
      end;
    for i := 1 to 5 do
      begin
        tru[i] := 0;
      end;
    for i := 1 to 62 do
      begin
        for j := 1 to 5 do
          begin
            plour[i, j] := 0;
          end;
      end;
    for i := 1 to 62 do
      begin
        point[i] := 0;
      end;
    for i := 1 to 62 do
      begin
        q[i] := 0;
      end;
    for i := 1 to 62 do
      begin
        bx[i] := 0;
      end;
    for i := 1 to 62 do
      begin
        by[i] := 0;
      end;
    I := 0;
    WIDTH := 0;
    HEIGHT := 0;
    DIVISOR := 0;
    II := 0;
    NT := 0;
    J := 0;
    IP1 := 0;
    IA := 0;
    IB := 0;
    ISUM := 0;
    DISTAN := 0;
    MAXDROP := 0;
    NVT := 0;
    MAXTIME := 0;
    TCAP := 0;
    TOTTIME := 0;
    MILMIN := 0;
    SUM1 := 0;
    SPEED := 0;
    newscale := 0;
    SUM1 := 0;
    NETGEN(X, Y, NPT, SCALE, BKPTR, newscale);
    NPT1 := NPT;
    NPT := BKPTR - 1;
{WRITELN('ENTER NO OF VEHICLE TYPES(DEFAULT=1)');
(READLN(NVT);}
    NVT := 1;
    for I := 1 to NVT do
      begin
        WRITELN('FOR TRUCK', I, 'SPECIFY THE FOLLOWING PARAMETERS');
        WRITELN('ENTER TRUCK CAPACITY(DEFAULT=20)');
```

```
      READLN(TRCAP[I]);
{WRITELN('ENTER TRUCK AVAILABILITY(DEFAULT=9999)');}
{READLN(TRA[I]);}
      TRA[1] := 1000;
    end;
  WRITELN('ENTER NODE DEMAND REQUIREMENTS(DEFAULT=3)');
  READLN(Q[1]);
  If Q[1] = 999 then
    goto 776;
  for I := 1 to NPT1 do
    begin
      If I <= NPT then
        Q[I] := Q[1]
      else
        Q[I] := -Q[1];
    end;
  goto 6767;
776:
  Q[1] := 0;
  Q[51] := 10;
  Q[2] := 11;
  Q[3] := 19;
  Q[4] := 7;
  Q[5] := 11;
  Q[6] := 14;
  Q[7] := 16;
  Q[8] := 6;
  Q[9] := 17;
  Q[10] := 26;
  Q[11] := 8;
  Q[12] := 7;
  Q[13] := 15;
  Q[14] := 16;
  Q[15] := 17;
  Q[16] := 23;
  Q[17] := 15;
  Q[18] := 12;
  Q[19] := 19;
  Q[20] := 30;
  Q[21] := 15;
  Q[22] := 15;
  Q[23] := 21;
  Q[24] := 29;
  Q[25] := 5;
  Q[26] := 18;
  Q[27] := 5;
  Q[28] := 11;
  Q[29] := 10;
  Q[30] := 6;
  Q[31] := 6;
  Q[32] := 26;
  Q[33] := 19;
  Q[34] := 14;
  Q[35] := 23;
  Q[36] := 10;
  Q[37] := 10;
  Q[38] := 3;
  Q[39] := 9;
  Q[40] := 16;
  Q[41] := 13;
  Q[42] := 9;
  Q[43] := 7;
  Q[44] := 27;
  Q[45] := 23;
  Q[46] := 26;
  Q[47] := 21;
  Q[48] := 25;
  Q[49] := 9;
  Q[50] := 41;
```

```
6767:
{   WRITELN('ENTER DIVISOR,C-W ACCURACY PARM.(DEFAULT=1)');}
{   READLN(DIVISOR);}
{WRITELN('ENTER MAX NUMBER OF DROPS PER ROUTE(DEFAULT=50)');}
{READLN(MAXDROP);}
{WRITELN('ENTER MAX NUMBER OF HOURS PER ROUTE(DEFAULT=8 )');}
{READLN(MAXTIME);}
{WRITELN('ENTER AVERAGE SPEED OF VEHICLES(DEFAULT= 60 MPH)');}
{READLN(SPEED);}
    DIVISOR := 4;
    MAXDROP := 100;
    MAXTIME := 8;
    SPEED := 60;
    MILMIN := SPEED / 60;
{WRITELN('SPEED', SPEED, 'MILMIN', MILMIN);}
{GetTime(DateTime);}
    Mins := DateTime.Minute * 60;
    Secs := DateTime.Second;
    T0 := Mins + Secs;
    WIDTH := SCALE;
    HEIGHT := SCALE;
1010:
    WIDTH := ROUND(WIDTH / DIVISOR);
    HEIGHT := ROUND(HEIGHT / DIVISOR);
    for I := 2 to NPT do
      begin
        POINT[I] := 0;
        BX[I] := ROUND(X[I] / WIDTH + 1);
        BY[I] := ROUND(Y[I] / HEIGHT + 1);
      end;
    NARC := 0;
    NT := NPT - 1;
    I := 1;
    for J := 2 to NPT do
      begin
        NARC := NARC + 1;
        while NARC >= 1000 do
          begin
            DIVISOR := DIVISOR + 2;
            goto 1010
          end;
        ID[NARC, 1] := I;
        ID[NARC, 2] := J;
        DISTAN := SQR(X[I] - X[J]) + SQR(Y[I] - Y[J]);
        D[J] := SQRT(DISTAN);
      end;
{WRITELN('NARC', NARC);}
    II := 0;
    for I := 2 to NT do
      begin
        IP1 := I + 1;
        for J := IP1 to NPT do
          begin
            IA := ROUND(ABS(BX[I] - BX[J]));
            IB := ROUND(ABS(BY[I] - BY[J]));
            if (IA > 1) or (IB > 1) then
              goto 102;
            NARC := NARC + 1;
{if NARC >= 700 then}
{WRITELN('NARC', NARC);}
            while NARC >= 1000 do
              begin
                DIVISOR := DIVISOR + 2;
                goto 1010
              end;
            II := II + 1;
            POINT[I] := POINT[I] + 1;
            POINT[J] := POINT[J] + 1;
            if (POINT[I] > 100) or (POINT[J] > 100) then
```

```
          goto 998;
        ADJA[I, POINT[I]] := II;
        ADJA[J, POINT[J]] := II;
        ID[NARC, 1] := I;
        ID[NARC, 2] := J;
        DISTAN := SQR(X[I] - X[J]) + SQR(Y[I] - Y[J]);
        S[II] := D[I] + D[J] - SQRT(DISTAN);
102:
      end;
    end;
  WRITELN('NARCFIRST', NARC);
{WRITELN('MAX S[]', II);}
  SUM := 0;
  SUM1 := 0;
  for J := 2 to NPT do
    begin
      SUM := SUM + D[J];
      SUM1 := SUM1 + ROUND(D[J] / MILMIN);
    end;
  TDIST := 2 * SUM;
  TOTTIME := 2 * SUM1;
{WRITELN('TOTTIME', TOTTIME);}
  NN := NARC - NPT + 1;
  TRU[1] := NPT - 1;
  if NVT = 1 then
    goto 17;
  for I := 2 to NVT do
    begin
      TRU[I] := 0;
    end;
17:
  for I := 2 to NPT do
    begin
      PTOUR[I, 1] := I;
      PTOUR[I, 2] := 1;
      PTOUR[I, 3] := O[I];
      PTOUR[I, 4] := 1;
      PTOUR[I, 5] := 2 * ROUND(D[I] / MILMIN);
    end;
  for I := 1 to NT do
    begin
      ID[I, 3] := 2;
    end;
  for I := NPT to NARC do
    begin
      ID[I, 3] := 0;
    end;
91:
  for I := 1 to NN do
    begin
      JX[I] := I;
      IX[I] := I;
    end;
  STHEAP(NN, S, IX, JX);
90:
  JO := 0;
  K := 1;
  if S[K] <= 0 then
    goto 16;
  P := IX[K] + NPT - 1;
  LIN := ID[P, 1];
  LJN := ID[P, 2];
  IPT1 := PTOUR[LIN, 1];
  IPT2 := PTOUR[LJN, 1];
  if (ID[LIN - 1, 3] = 0) or (ID[LJN - 1, 3] = 0) then
    goto 12;
  if IPT1 = IPT2 then
    goto 12;
  ICAP := PTOUR[LIN, 3] + PTOUR[LJN, 3];
```

```
      for LM := 1 to NVT do
        begin
          if (TRCAP[LM] >= ICAP) and (TRU[LM] < TRA[LM]) then
            goto 10;
        end;
      goto 12;
  10:
      MCAP := PTOUR[LIN, 4] + PTOUR[LJN, 4];
      if MCAP > MAXDROP then
        goto 12;
      TCAP := PTOUR[LIN, 5] + PTOUR[LJN, 5] - ROUND(S[K] / MILMIN);
      if TCAP > MAXTIME * 60 then
        goto 12;
      TRU[LM] := TRU[LM] + 1;
      TRU[PTOUR[LIN, 2]] := TRU[PTOUR[LIN, 2]] - 1;
      TRU[PTOUR[LJN, 2]] := TRU[PTOUR[LJN, 2]] - 1;
      DISTAN := D[LIN] + D[LJN] - S[K];
      TDIST := TDIST - (D[LIN] + D[LJN] - DISTAN);
      TOTTIME := TOTTIME - ROUND(S[K] / MILMIN);
  {WRITELN('TOTTIME', TOTTIME, 'MILMIN', MILMIN);}
      for L := 2 to NPT do
        begin
          if (PTOUR[L, 1] = IPT1) or (PTOUR[L, 1] = IPT2) then
            goto 64;
        end;
  64:
      for M := 2 to NPT do
        begin
          if (PTOUR[M, 1] <> IPT1) and (PTOUR[M, 1] <> IPT2) then
            goto 105;
          PTOUR[M, 1] := PTOUR[L, 1];
          PTOUR[M, 2] := LM;
          PTOUR[M, 3] := ICAP;
          PTOUR[M, 4] := MCAP;
          PTOUR[M, 5] := TCAP;
  105:
        end;
      ID[P, 3] := 1;
      ID[LIN - 1, 3] := ID[LIN - 1, 3] - 1;
      ID[LJN - 1, 3] := ID[LJN - 1, 3] - 1;
      if ID[LJN - 1, 3] <> 0 then
        goto 11;
      IB := POINT[LJN];
      JO := 1;
      for I := 1 to IB do
        begin
          IC := JX[ADJA[LJN, I]];
          if S[IC] <= 0 then
            goto 97;
          S[IC] := 0;
          OTHEAP(NN, S, IX, JX, IC);
  97:
        end;
  11:
      if ID[LIN - 1, 3] <> 0 then
        goto 12;
      IB := POINT[LIN];
      for I := 1 to IB do
        begin
          IC := JX[ADJA[LIN, I]];
          if S[IC] <= 0 then
            goto 98;
          S[IC] := 0;
          OTHEAP(NN, S, IX, JX, IC);
  98:
        end;
      goto 90;
  12:
      if JO = 1 then
```

```
      goto 90;
    S[K] := 0;
    OTHEAP(NN, S, IX, JX, K);
    goto 90;
16:
{GetTime(DateTime);}
    Mins := DateTime.Minute * 60;
    Secs := DateTime.Second;
    T1 := Mins + Secs;
    If T1 < T0 then
      T1 := T1 + 3600;
    Elaps := T1 - T0;
    I := 1;
    NARC2 := NARC;
    for J := BKPTR to NPT1 do
      begin
        NARC := NARC + 1;
        ID[NARC, 1] := I;
        ID[NARC, 2] := J;
        DISTAN := SQR(X[I] - X[J]) + SQR(Y[I] - Y[J]);
        D[J] := SQRT(DISTAN);
        ID[NARC, 3] := 2;
      end;
    WRITELN('NARC', NARC);
    SUM := 0;
    SUM1 := 0;
{MILMIN := 1;}
{WRITELN('TOTTIME', TOTTIME, 'MILMIN', ROUND(MILMIN));}
    for J := BKPTR to NPT1 do
      begin
        SUM := SUM + D[J];
        SUM1 := SUM1 + ROUND(D[J] / MILMIN);
        ISUM := ISUM + 1;
        PTOUR[J, 1] := J;
        PTOUR[J, 2] := 1;
        PTOUR[J, 3] := Q[J];
        PTOUR[J, 4] := 1;
        PTOUR[J, 5] := ROUND(D[I] / MILMIN);
      end;
    WRITELN('C-W DIST', ROUND(TDIST), 'RADIAL DIST', ROUND(SUM));
    TDIST := TDIST + SUM;
    TOTTIME := TOTTIME + SUM1;
{WRITELN('TOTTIME', TOTTIME);}
    ITK := 1;
    for IT := 1 to NPT1 do
      begin
        HIT := 0;
        for KT := 2 to NPT1 do
          begin
            while PTOUR[KT, 1] = IT do
              begin
                PTOUR[KT, 1] := ITK;
                HIT := 1;
              end;
          end;
        If HIT = 1 then
          ITK := ITK + 1;
      end;
    WRITELN('ELAPSED TIME IS', Elaps, ' SECONDS');
    WRITELN('TOTAL DISTANCE IS', ROUND(TDIST), ' UNITS');
    WRITELN('TOTAL TRAVEL TIME IS', ROUND(TOTTIME), ' MINUTES');
    for I := 1 to NVT do
      begin
        ISUM := ISUM + TRU[I];
      end;
    WRITELN('THERE ARE', ISUM, ' TOURS IN ALL');
    NTOURS := ISUM;
{WRITELN('      ', 'I      ', 'J    ', 'T(I,J) ', 'TRUCKTYPE', '  TOUR', ' DELIVERY ', '#DROPS');}
    for J := 1 to NARC do
```

```
    begin
      if ID[J, 3] = 0 then
        goto 35;
{WRITELN(ID[J, 1], ID[J, 2], ID[J, 3], PTOUR[ID[J, 2], 2], '  ', PTOUR[ID[J, 2], 1], PTOUR[ID[J, 2], 3], PTOUR[ID[J, 2], 4]);}
      P1 := ROUND(X[ID[J, 1]]);
      P2 := ROUND(Y[ID[J, 1]]);
      P3 := ROUND(X[ID[J, 2]]);
      P4 := ROUND(Y[ID[J, 2]]);
      if J <= NARC2 then
        DRAWLINE(P1, P2, P3, P4);
35:
    end;
  WRITELN(' TRUCK TYPE', '  #AVAIL  ', '   CAP    ', '#USED IN ROUTING');
  for M := 1 to NVT do
    begin
      WRITELN(M, '     ', TRA[M], '    ', TRCAP[M], '  ', TRU[M] + (NPT1 - BKPTR + 1));
    end;
  WRITELN('MAX NUMBER OF DROPS', MAXDROP);
  WRITELN('MAX NUMBER OF HOURS PER ROUTE', MAXTIME);
  WRITELN('AVERAGE SPEED OF VEHICLE IS', ROUND(SPEED), ' MPH');
  WRITELN('ENTER NEW GRAPH SCALE FOR SPECIFIC ROUTES(enter 0 for none, or integer)');
  readln(newscale);
  if newscale = 0 then
    goto 301;
38:
  writeln('ENTER ROUTE NUMBER DESIRED(enter 0 for none,enter 100 for all)');
  readln(tours);
  if tours = 0 then
    goto 301;
  if XXX = 0 then
    goto 39;
  writeln('ENTER 0 IF YOU WOULD LIKE PREVIOUS TOUR ERASED');
  writeln('ENTER 1 OTHERWISE');
  readln(YYY);
  if YYY = 1 then
    goto 40;
39:
  PaintCircle(0, 0, 500);
  InvertCircle(0, 0, 500);
  XXX := 1;
  NETGEN(X, Y, NPT1, SCALE, BKPTR, newscale);
  for J := 1 to NARC do
    begin
      if ID[J, 3] = 0 then
        goto 777;
{WRITELN(ID[J, 1], ID[J, 2], ID[J, 3], PTOUR[ID[J, 2], 2], '  ', PTOUR[ID[J, 2], 1], PTOUR[ID[J, 2], 3], PTOUR[ID[J, 2], 4]);}
      P1 := ROUND(X[ID[J, 1]]) * (newscale);
      P2 := ROUND(Y[ID[J, 1]]) * newscale;
      P3 := ROUND(X[ID[J, 2]]) * (newscale);
      P4 := ROUND(Y[ID[J, 2]]) * newscale;
      DRAWLINE(P1, P2, P3, P4);
777:
    end;
  if tours = 100 then
    goto 301;
40:
  for J := 1 to NARC do
    begin
      if (ID[J, 3] = 0) then
        goto 36;
      if (PTOUR[ID[J, 2], 1] <> tours) then
        goto 36;
      P1 := ROUND(X[ID[J, 1]]) * (newscale + 1);
      P2 := ROUND(Y[ID[J, 1]]) * newscale;
      P3 := ROUND(X[ID[J, 2]]) * (newscale + 1);
      P4 := ROUND(Y[ID[J, 2]]) * newscale;
      DRAWLINE(P1, P2, P3, P4);
36:
    end;
```

```
   goto 38;
998:
   WRITELN('POINT(I) FOR SOME I EXCEEDS 50.');
301:
   WRITELN('THE END');
 end;
end.
```

```
unit CWUNIT;
interface
  type
    matrix9 = array[1..50, 1..3] of integer;
  procedure CWVRP (var REJARR: MATRIX9; var REJECOUNT: INTEGER; var SAVINGS: INTEGER);
implementation
  procedure CWVRP (var REJARR: MATRIX9; var REJECOUNT: INTEGER; var SAVINGS: INTEGER);
    label
      7799, 1010, 102, 998, 17, 16, 10, 64, 105, 11, 97, 12, 301, 98, 90, 35, 91, 22, 23, 25, 26;
    type
      MATRIX1 = array[1..500, 1..10] of INTEGER;
      MATRIX2 = array[1..50, 1..5] of INTEGER;
      COUNTER = INTEGER;
      VECTOR = array[1..500] of real;
      IVECTOR = array[1..500] of INTEGER;
      VECT100 = array[1..50] of REAL;
      IVECT100 = array[1..50] of INTEGER;
      MATRIX3 = array[1..50, 1..8] of INTEGER;
      MATRIX4 = array[1..100, 1..3] of INTEGER;
      MATRIX5 = array[1..5] of INTEGER;
      COUNTER2 = INTEGER;
      REAL1 = REAL;
      matrix9 = array[1..50, 1..3] of integer;
      DateTimeRec = record
        Year, Month, Day, Hour, Minute, Second, DayOfWeek: integer
      end;
    var
      DateTime: DateTimeRec;
      TOTTIME, MILMIN, SUM1, SPEED: REAL1;
      T0, T1, T2, Elaps2, Elaps, Hrs, Mins, Secs: integer;
      NN, IC, NPT, NPT1, SCALE, BKPTR, tours, XXX, YYY, newscale, NARC, NTOURS: COUNTER;
      S: VECTOR;
      IX, JX: IVECTOR;
      JO, K, P, LIN, LJN, IPT1, IPT2, ICAP, MCAP, LM, L, M, ITK, HIT, KT, IT, P1, P2, P3, P4: INTEGER;
      TDIST, SUM, DISTAN, DISTAN2, SUMDIST: REAL;
      D: array[1..50] of real;
      X, Y: VECT100;
      ID: MATRIX1;
      ADJA: array[1..50, 1..50] of integer;
      TRA, TRCAP, TRU: MATRIX5;
      PTOUR: MATRIX2;
      POINT: array[1..50] of INTEGER;
      Q: IVECT100;
      BX, BY: array[1..50] of INTEGER;
      I, WIDTH, HEIGHT, DIVISOR, II, NT, J, IP1, IA, IB, ISUM: INTEGER;
      NVT, MAXDROP, MAXTIME, TCAP: COUNTER;
      NONO, IPOINT, IPOINT2, RADIAL, JJ: INTEGER;
      TEMP1, TEMP2, TEMP3, LL, LLL: INTEGER;
    procedure GETTIME (var DATETIME: DATETIMEREC);
    EXTERNAL;
    procedure STHEAP (var NN: COUNTER; var S: VECTOR; var IX, JX: IVECTOR);
      label
        1, 50, 60, 2, 5, 7, 9;
      var
        NH, II, I, IXC, JXC, J: INTEGER;
        DT, COPY: REAL;
    begin
      nh := 0;
      ii := 0;
      i := 0;
      ixc := 0;
      jxc := 0;
      j := 0;
      dt := 0;
      copy := 0;
      if (NN - 2) < 0 then
        goto 50;
      if (NN - 2) = 0 then
```

```
      goto 60;
    if (NN - 2) > 0 then
      goto 1;
1:
    NH := ROUND(NN / 2);
    for I := 1 to NH do
      begin
        II := NH + 1 - I;
        COPY := S[II];
        IXC := IX[II];
        JXC := IXC;
2:
        J := II + II;
        if (J - NN) < 0 then
          goto 5;
        if (J - NN) = 0 then
          goto 7;
        if (J - NN) > 0 then
          goto 9;
5:
        if S[J + 1] > S[J] then
          J := J + 1;
7:
        if S[J] <= COPY then
          goto 9;
        S[II] := S[J];
        IX[II] := IX[J];
        JX[IX[J]] := II;
        II := J;
        goto 2;
9:
        S[II] := COPY;
        IX[II] := IXC;
        JX[JXC] := II;
      end;
    goto 50;
60:
    if S[1] >= S[2] then
      goto 50;
    DT := S[1];
    S[1] := S[NN];
    S[2] := DT;
    IX[1] := 2;
    JX[1] := 2;
    IX[2] := 1;
    JX[2] := 1;
50:
  end;
  procedure OTHEAP (var NN: COUNTER; var S: VECTOR; var IX: IVECTOR; var JX: IVECTOR; IC: COUNTER);

    label
      13, 111, 112, 19;
    var
      II, IXC, JXC, J: INTEGER;
      COPY: REAL;
  begin
    II := 0;
    ixc := 0;
    jxc := 0;
    j := 0;
    copy := 0;
    II := IC;
    COPY := S[II];
    IXC := IX[II];
    JXC := IXC;
13:
    J := II + II;
    if (J - NN) < 0 then
      goto 111;
```

```
      if (J - NN) = 0 then
        goto 112;
      if (J - NN) > 0 then
        goto 19;
111:
      if S[J + 1] > S[J] then
        J := J + 1;
112:
      if S[J] <= COPY then
        goto 19;
      S[II] := S[J];
      IX[II] := IX[J];
      JX[IX[J]] := II;
      II := J;
      goto 13;
19:
      S[II] := COPY;
      IX[II] := IXC;
      JX[JXC] := II;
    end;
    procedure NETGEN (var X, Y: VECT100; var NPT, SCALE, BKPTR, newscale: COUNTER);
      label
        333, 222, 223;
      var
        P1, P2, P3, P4: INTEGER;
        i, l, j, NODES, backhaul, NPTB: integer;
        r, w: real;
        yy: array[1..124] of real;
        TEST: INTEGER;
        concentration: integer;
        conc: real;
    begin
      concentration := 0;
      conc := 0;
      i := 0;
      l := 0;
      j := 0;
      nodes := 0;
      backhaul := 0;
      nptb := 0;
      for i := 1 to 100 do
        begin
          yy[i] := 0;
        end;
      i := 0;
      if newscale > 0 then
        goto 333;
      newscale := 1;
      writeln('ENTER NO OF NODES(<=62)');
      readln(nodes);
      writeln('ENTER SCALE OF GRAPH');
      readln(scale);
      writeln('ENTER PERCENTAGE OF BACKHAUL LOCATIONS(18%)');
      readln(backhaul);
      NPT := NODES;
      NPTB := ROUND(NODES * (backhaul / 100));
      BKPTR := NPT - NPTB + 1;
      writeln('ENTER DELIVERY CONCENTRATION(60%)');
      readln(concentration);
      CONC := (concentration / 100);
{   WRITELN('BKPTR', BKPTR, 'CONC', CONC);}
      repeat
        i := i + 1;
        r := random;
        w := ABS(r / 32767);
        if i >= (BKPTR * 2) then
          goto 222;
        yy[i] := SCALE * (1 - CONC) + (w * (2 * (SCALE * CONC) - SCALE));
        goto 223;
```

```
222:
      yy[i] := (w * SCALE);
223:
{WRITELN('r, i, 'YY[i]', ROUND(YY[i]));}
    until i = nodes * 2;
    DRAWLINE(0, 100, 2, 100);
    DRAWLINE(100, 0, 100, 2);
    DRAWLINE(0, 200, 2, 200);
    DRAWLINE(200, 0, 200, 2);
    DRAWLINE(0, 300, 2, 300);
    DRAWLINE(300, 0, 300, 2);
{BEGIN TO COORD TO ARRAY}
    i := 1;
    j := 1;
    repeat
      X[i] := yy[j];
      i := i + 1;
      j := j + 2
    until j > NODES * 2;
    i := 1;
    j := 2;
    repeat
      Y[i] := yy[j];
      i := i + 1;
      j := j + 2
    until j > NODES * 2;
333:
    newscale := 2;
    i := 1;
    P1 := ROUND(X[i] * newscale);
    P2 := ROUND(Y[i] * newscale);
{writeln('node', i, P1, P2);}
    PaintCircle(P1, P2, 5);
    Moveto(P1 - 45, P2);
    WriteDraw(i);
    for i := 2 to (BKPTR - 1) do
      begin
        P1 := ROUND(X[i] * newscale);
        P2 := ROUND(Y[i] * newscale);
{writeln('node', i, P1, P2);}
        drawline(P1 - 3, P2 - 3, P1 + 3, P2 - 3);
        drawline(P1 - 3, P2 + 3, P1 + 3, P2 + 3);
        drawline(P1 - 3, P2 - 3, P1 - 3, P2 + 3);
        drawline(P1 + 3, P2 - 3, P1 + 3, P2 + 3);
        Moveto(P1 - 15, P2);
        WriteDraw(i);
      end;
    NODES := NPT;
    for i := BKPTR to NODES do
      begin
        P1 := ROUND(X[i] * newscale);
        P2 := ROUND(Y[i] * newscale);
{writeln('node', i, P1, P2);}
        drawline(P1, P2 - 4, P1 - 4, P2 + 4);
        drawline(P1, P2 - 4, P1 + 4, P2 + 4);
        drawline(P1 - 4, P2 + 4, P1 + 4, P2 + 4);
        Moveto(P1 - 15, P2);
        WriteDraw(i);
      end;
  end;
begin
  i := 0;
  j := 0;
  T0 := 0;
  T1 := 0;
  Elaps := 0;
  Hrs := 0;
  Mins := 0;
  Secs := 0;
```

```
  NN := 0;
  IC := 0;
  NPT := 0;
  SCALE := 0;
  BKPTR := 0;
  tours := 0;
  XXX := 0;
  YYY := 0;
  for i := 1 to 500 do
    begin
      s[i] := 0;
    end;
  for i := 1 to 500 do
    begin
      ix[i] := 0;
    end;
  for i := 1 to 500 do
    begin
      jx[i] := 0;
    end;
  NARC := 0;
  JO := 0;
  K := 0;
  P := 0;
  LIN := 0;
  LJN := 0;
  IPT1 := 0;
  IPT2 := 0;
  ICAP := 0;
  MCAP := 0;
  LM := 0;
  L := 0;
  M := 0;
  ITK := 0;
  HIT := 0;
  KT := 0;
  IT := 0;
  P1 := 0;
  P2 := 0;
  P3 := 0;
  P4 := 0;
  TDIST := 0;
  SUM := 0;
  for i := 1 to 50 do
    begin
      d[i] := 0;
    end;
  for i := 1 to 50 do
    begin
      x[i] := 0;
    end;
  for i := 1 to 50 do
    begin
      y[i] := 0;
    end;
  for i := 1 to 500 do
    begin
      for j := 1 to 10 do
        begin
          id[i, j] := 0;
        end;
    end;
  for i := 1 to 50 do
    begin
      for j := 1 to 50 do
        begin
          adj[i, j] := 0;
        end;
    end;
```

```
    for I := 1 to 5 do
      begin
        tra[I] := 0;
      end;
    for I := 1 to 5 do
      begin
        trcap[I] := 0;
      end;
    for I := 1 to 5 do
      begin
        tru[I] := 0;
      end;
    for I := 1 to 50 do
      begin
        for J := 1 to 5 do
          begin
            ptour[I, J] := 0;
          end;
      end;
    for I := 1 to 50 do
      begin
        point[I] := 0;
      end;
    for I := 1 to 50 do
      begin
        q[I] := 0;
      end;
    for I := 1 to 50 do
      begin
        bx[I] := 0;
      end;
    for I := 1 to 50 do
      begin
        by[I] := 0;
      end;
    I := 0;
    WIDTH := 0;
    HEIGHT := 0;
    DIVISOR := 0;
    II := 0;
    NT := 0;
    J := 0;
    IP1 := 0;
    IA := 0;
    IB := 0;
    ISUM := 0;
    DISTAN := 0;
    MAXDROP := 0;
    NVT := 0;
    MAXTIME := 0;
    TCAP := 0;
    TOTTIME := 0;
    MILMIN := 0;
    SUM1 := 0;
    SPEED := 0;
    newscale := 0;
{WRITELN('REJ NODE  ', 'REJ X COORD ', ' REJ Y COORD ');}
    NPT := REJECOUNT + 1;
    NEWSCALE := 1;
    for I := 1 to REJECOUNT + 1 do
      begin
{WRITELN(REJARR[I, 1], '    ', REJARR[I, 2], '    ', REJARR[I, 3]);}
        X[I] := REJARR[I, 2];
        Y[I] := REJARR[I, 3];
      end;
    SUMDIST := 0;
    DISTAN := 0;
    DISTAN2 := 0;
    for I := 2 to REJECOUNT + 1 do
```

```
      begin
        DISTAN := SQR(X[I] - X[1]) + SQR(Y[I] - Y[1]);
        DISTAN2 := SQRT(DISTAN);
        WRITELN('RADIAL DISTANCE FOR  ', REJARR[I, 1], ' IS ', ROUND(DISTAN2));
        SUMDIST := SUMDIST + DISTAN2;
      end;
{NETGEN(X, Y, NPT, SCALE, BKPTR, newscale);}
    NPT1 := NPT;
{WRITELN('ENTER NO OF VEHICLE TYPES(DEFAULT=1)');}
{READLN(NVT);}
    NVT := 1;
    for I := 1 to NVT do
      begin
        WRITELN('FOR TRUCK', I, 'SPECIFY THE FOLLOWING PARAMETERS');
        WRITELN('ENTER TRUCK CAPACITY(DEFAULT=20)');
        READLN(TRCAP[I]);
{WRITELN('ENTER TRUCK AVAILABILITY(DEFAULT=9999)');}
{READLN(TRA[I]);}
        TRA[1] := 1000;
      end;
    WRITELN('ENTER NODE DEMAND REQUIREMENTS(DEFAULT=3)');
    READLN(Q[1]);
    for I := 1 to NPT1 do
      begin
        if I <= NPT then
          Q[I] := Q[1]
        else
          Q[I] := -Q[1];
      end;
{   WRITELN('ENTER DIVISOR,C-W ACCURACY PARM.(DEFAULT=1)');}
{   READLN(DIVISOR);}
{WRITELN('ENTER MAX NUMBER OF DROPS PER ROUTE(DEFAULT=50)');}
{READLN(MAXDROP);}
{WRITELN('ENTER MAX NUMBER OF HOURS PER ROUTE(DEFAULT=8 )');}
{READLN(MAXTIME);}
{WRITELN('ENTER AVERAGE SPEED OF VEHICLES(DEFAULT= 60 MPH)');}
{READLN(SPEED);}
    DIVISOR := 1;
    MAXDROP := 100;
    MAXTIME := 8;
    SPEED := 60;
    MILMIN := SPEED / 60;
    WIDTH := SCALE;
    HEIGHT := SCALE;
1010:
    WIDTH := ROUND(WIDTH / DIVISOR);
    HEIGHT := ROUND(HEIGHT / DIVISOR);
    for I := 2 to NPT do
      begin
        POINT[I] := 0;
        BX[I] := ROUND(X[I] / WIDTH + 1);
        BY[I] := ROUND(Y[I] / HEIGHT + 1);
      end;
    NARC := 0;
    NT := NPT - 1;
    I := 1;
    for J := 2 to NPT do
      begin
        NARC := NARC + 1;
        while NARC >= 1000 do
          begin
            DIVISOR := DIVISOR + 2;
            goto 1010
          end;
        ID[NARC, 1] := I;
        ID[NARC, 2] := J;
        DISTAN := SQR(X[I] - X[J]) + SQR(Y[I] - Y[J]);
        D[J] := SQRT(DISTAN);
      end;
```

```
   II := 0;
   for I := 2 to NT do
     begin
       IP1 := I + 1;
       for J := IP1 to NPT do
         begin
           IA := ROUND(ABS(BX[I] - BX[J]));
           IB := ROUND(ABS(BY[I] - BY[J]));
           if (IA > 1) or (IB > 1) then
             goto 102;
           NARC := NARC + 1;
           while NARC >= 1000 do
             begin
               DIVISOR := DIVISOR + 2;
               goto 1010
             end;
           II := II + 1;
           POINT[I] := POINT[I] + 1;
           POINT[J] := POINT[J] + 1;
           if (POINT[I] > 50) or (POINT[J] > 50) then
             goto 998;
           ADJA[I, POINT[I]] := II;
           ADJA[J, POINT[J]] := II;
           ID[NARC, 1] := I;
           ID[NARC, 2] := J;
           DISTAN := SQR(X[I] - X[J]) + SQR(Y[I] - Y[J]);
           S[II] := D[I] + D[J] - SQRT(DISTAN);
102:
         end;
     end;
   SUM := 0;
   SUM1 := 0;
   for J := 2 to NPT do
     begin
       SUM := SUM + D[J];
       SUM1 := SUM1 + ROUND(D[J] / MILMIN);
     end;
   TDIST := 2 * SUM;
   TOTTIME := 2 * SUM1;
   NN := NARC - NPT + 1;
   TRU[1] := NPT - 1;
   if NVT = 1 then
     goto 17;
   for I := 2 to NVT do
     begin
       TRU[I] := 0;
     end;
17:
   for I := 2 to NPT do
     begin
       PTOUR[I, 1] := I;
       PTOUR[I, 2] := 1;
       PTOUR[I, 3] := Q[I];
       PTOUR[I, 4] := 1;
       PTOUR[I, 5] := 2 * ROUND(D[I] / MILMIN);
     end;
   for I := 1 to NT do
     begin
       ID[I, 3] := 2;
     end;
   for I := NPT to NARC do
     begin
       ID[I, 3] := 0;
     end;
91:
   for I := 1 to NN do
     begin
       JX[I] := I;
       IX[I] := I;
```

```
      end;
    STHEAP(NN, S, IX, JX);
90:
    JO := 0;
    K := 1;
    if S[K] <= 0 then
      goto 16;
    P := IX[K] + NPT - 1;
    LIN := ID[P, 1];
    LJN := ID[P, 2];
    IPT1 := PTOUR[LIN, 1];
    IPT2 := PTOUR[LJN, 1];
    if (ID[LIN - 1, 3] = 0) or (ID[LJN - 1, 3] = 0) then
      goto 12;
    if IPT1 = IPT2 then
      goto 12;
    ICAP := PTOUR[LIN, 3] + PTOUR[LJN, 3];
    for LM := 1 to NVT do
      begin
        if (TRCAP[LM] >= ICAP) and (TRU[LM] < TRA[LM]) then
          goto 10;
      end;
    goto 12;
10:
    MCAP := PTOUR[LIN, 4] + PTOUR[LJN, 4];
    if MCAP > MAXDROP then
      goto 12;
    TCAP := PTOUR[LIN, 5] + PTOUR[LJN, 5] - ROUND(S[K] / MILMIN);
    if TCAP > MAXTIME * 60 then
      goto 12;
    TRU[LM] := TRU[LM] + 1;
    TRU[PTOUR[LIN, 2]] := TRU[PTOUR[LIN, 2]] - 1;
    TRU[PTOUR[LJN, 2]] := TRU[PTOUR[LJN, 2]] - 1;
    DISTAN := D[LIN] + D[LJN] - S[K];
    TDIST := TDIST - (D[LIN] + D[LJN] - DISTAN);
    TOTTIME := TOTTIME - ROUND(S[K] / MILMIN);
    for L := 2 to NPT do
      begin
        if (PTOUR[L, 1] = IPT1) or (PTOUR[L, 1] = IPT2) then
          goto 64;
      end;
64:
    for M := 2 to NPT do
      begin
        if (PTOUR[M, 1] <> IPT1) and (PTOUR[M, 1] <> IPT2) then
          goto 105;
        PTOUR[M, 1] := PTOUR[L, 1];
        PTOUR[M, 2] := LM;
        PTOUR[M, 3] := ICAP;
        PTOUR[M, 4] := MCAP;
        PTOUR[M, 5] := TCAP;
105:
      end;
    ID[P, 3] := 1;
    ID[LIN - 1, 3] := ID[LIN - 1, 3] - 1;
    ID[LJN - 1, 3] := ID[LJN - 1, 3] - 1;
    if ID[LJN - 1, 3] <> 0 then
      goto 11;
    IB := POINT[LJN];
    JO := 1;
    for I := 1 to IB do
      begin
        IC := JX[ADJA[LJN, I]];
        if S[IC] <= 0 then
          goto 97;
        S[IC] := 0;
        OTHEAP(NN, S, IX, JX, IC);
97:
      end;
```

```
11:
   if ID[LIN - 1, 3] <> 0 then
     goto 12;
   IB := POINT[LIN];
   for I := 1 to IB do
     begin
       IC := JX[ADJA[LIN, I]];
       if S[IC] <= 0 then
         goto 98;
       S[IC] := 0;
       OTHEAP(NN, S, IX, JX, IC);
98:
     end;
   goto 90;
12:
   if JO = 1 then
     goto 90;
   S[K] := 0;
   OTHEAP(NN, S, IX, JX, K);
   goto 90;
16:
   TDIST := TDIST;
   TOTTIME := TOTTIME;
   ITK := 1;
   for IT := 1 to NPT1 do
     begin
       HIT := 0;
       for KT := 2 to NPT1 do
         begin
           while PTOUR[KT, 1] = IT do
             begin
               PTOUR[KT, 1] := ITK;
               HIT := 1;
             end;
         end;
       if HIT = 1 then
         ITK := ITK + 1;
     end;
   WRITELN('TOTAL RADIAL DIST FOR REJECTED IS', ROUND(SUMDIST), ' UNITS');
   WRITELN('TOTAL DISTANCE FOR DEDICATED BK ROUTES IS', ROUND(TDIST), ' UNITS');
   SAVINGS := ROUND(SUMDIST - TDIST);
   WRITELN('TOTAL SAVINGS IS', ROUND(SUMDIST - TDIST), ' UNITS');
   WRITELN('TOTAL TRAVEL TIME IS', ROUND(TOTTIME), ' MINUTES');
   for I := 1 to NVT do
     begin
       ISUM := ISUM + TRU[I];
     end;
   WRITELN('THERE ARE', ISUM, ' TOURS IN ALL');
   NTOURS := ISUM;
   RADIAL := 0;
   for JJ := 1 to ISUM do
     begin
       WRITELN('ROUTE', JJ);
       NONO := 99999;
       IPOINT := 0;
       IPOINT2 := 0;
       for II := 1 to 2 do
         begin
           for I := 1 to NARC do
             begin
               if PTOUR[ID[I, 2], 4] < 3 then
                 goto 22;
               if ID[I, 3] <> 1 then
                 goto 22;
               if PTOUR[ID[I, 2], 4] < 3 then
                 goto 22;
               if PTOUR[ID[I, 2], 1] <> JJ then
                 goto 22;
               if ID[I, 1] <> 1 then
```

```
            goto 22;
          if NONO = I then
            goto 22;
          IPOINT := ID[I, 2];
          NONO := I;
          WRITELN('NONO', NONO, 'IPOINT', IPOINT, 'REJARR[IPOINT, 1]', REJARR[IPOINT, 1]);
          for J := 1 to NARC do
            begin
              if ID[J, 3] <> 1 then
                goto 23;
              if PTOUR[ID[J, 2], 4] < 4 then
                goto 23;
              if PTOUR[ID[J, 2], 1] <> JJ then
                goto 23;
              if NONO = J then
                goto 23;
              if (ID[J, 1] <> IPOINT) and (ID[J, 2] <> IPOINT) then
                goto 23;
              if ID[J, 1] = IPOINT then
                IPOINT2 := ID[J, 2];
              if ID[J, 2] = IPOINT then
                IPOINT2 := ID[J, 1];
              WRITELN('IPOINT2', IPOINT2, 'REJARR[IPOINT2, 1]', REJARR[IPOINT2, 1]);
              TEMP1 := ROUND(SQRT(SQR(X[IPOINT] - X[IPOINT2]) + SQR(Y[IPOINT] - Y[IPOINT2])));
              WRITELN('IPOINT TO IPOINT2 ', TEMP1);
              TEMP2 := ROUND(SQRT(SQR(X[1] - X[IPOINT2]) + SQR(Y[1] - Y[IPOINT2])));
              WRITELN('IPOINT2 TO DEPOT', TEMP2);
              TEMP3 := ROUND(SQRT(SQR(X[1] - X[IPOINT]) + SQR(Y[1] - Y[IPOINT])) * (RADIAL / 100));
              WRITELN('IPOINT TO DEPOT RADIAL SURCHARGE', TEMP3);
              if (TEMP1) < (TEMP2 + TEMP3) then
                goto 25;
              WRITELN('DISCONNECT', REJARR[IPOINT, 1], ' ', REJARR[IPOINT2, 1], 'DIST ', ROUND(SQRT(SQR(X[IPOINT] -
      X[IPOINT2]) + SQR(Y[IPOINT] - Y[IPOINT2]))));
              WRITELN('CONNECT', REJARR[IPOINT2, 1], ' TO DEPOT   DIST', ROUND(SQRT(SQR(X[1] - X[IPOINT2]) + SQR(Y[1] -
      Y[IPOINT2])) * (1 + (RADIAL / 100))));

              goto 26;

{DISCONNECT IPOINT TO IPOINT2 LINK AND CONNECT IPOINT2 TO 1 LINK}

{ CONNECT IPOINT2 TO DEPOT LINK }

              NARC := NARC + 1;
              ID[NARC, 1] := 1;
              ID[NARC, 2] := IPOINT2;
              ID[NARC, 3] := 1;
              for M := 2 to NPT do
                begin
                  if (PTOUR[M, 1] <> PTOUR[IPOINT2, 1]) then
                    goto 7799;
                  PTOUR[M, 3] := ICAP - Q[IPOINT2];
                  PTOUR[M, 4] := MCAP - 1;
7799:
                end;

{REDEFINE IPOINT TO DEPOT LINK}
              ID[LL, 3] := 2;
              PTOUR[IPOINT, 1] := ISUM + 1;
              PTOUR[IPOINT, 2] := 1;
              PTOUR[M, 3] := Q[IPOINT];
              PTOUR[M, 4] := 1;
              PTOUR[M, 5] := 0;

{DISCONNECT IPOINT TO IPOINT2 LINK}
              ID[LLL, 3] := 0;

{SOLUTION UPDATES}
              ISUM := ISUM + 1;
```

```
                TDIST := TDIST + TEMP2 + TEMP3 - TEMP1;
                goto 26;

23:

              end;
22:
            end;
25:
          WRITELN(' LEAVE', REJARR[IPOINT, 1], ' TO ', REJARR[IPOINT2, 1], ' CONNECTED ', ROUND(SQRT(SQR(X[IPOINT] -
        X[IPOINT2]) + SQR(Y[IPOINT] - Y[IPOINT2]))));
26:
        end;

      end;
    newscale := 2;
    WRITELN('      ', 'I      ', 'J    ', 'T(I,J) ', 'TRUCKTYPE', '  TOUR', ' DELIVERY ', '#DROPS', '  DISTANCE');
    for J := 1 to NARC do
      begin
        if ID[J, 3] = 0 then
          goto 35;
        WRITELN(REJARR[ID[J, 1], 1], REJARR[ID[J, 2], 1], ID[J, 3], PTOUR[ID[J, 2], 2], '  ', PTOUR[ID[J, 2], 1], PTOUR[ID[J, 2], 3],
         PTOUR[ID[J, 2], 4], ROUND(SQRT(SQR(X[ID[J, 1]] - X[ID[J, 2]]) + SQR(Y[ID[J, 1]] - Y[ID[J, 2]]))));
        P1 := ROUND(X[ID[J, 1]]) * (newscale);
        P2 := ROUND(Y[ID[J, 1]]) * newscale;
        P3 := ROUND(X[ID[J, 2]]) * (newscale);
        P4 := ROUND(Y[ID[J, 2]]) * newscale;
        if ROUND(SUMDIST - TDIST) > 0 then
          DRAWLINE(P1, P2, P3, P4);
35:
      end;
    WRITELN('  TRUCK TYPE', '  AVAIL  ', '   CAP    ', '#USED IN ROUTING');
    for M := 1 to NVT do
      begin
        WRITELN(M, '     ', TRA[M], '   ', TRCAP[M], '   ', TRU[M] + (NPT1 - BKPTR + 1));
      end;
    WRITELN('MAX NUMBER OF DROPS', MAXDROP);
    WRITELN('MAX NUMBER OF HOURS PER ROUTE', MAXTIME);
    WRITELN('AVERAGE SPEED OF VEHICLE IS', ROUND(SPEED), ' MPH');

    goto 301;
998:
    WRITELN('POINT(I) FOR SOME I EXCEEDS 50.');
301:
    WRITELN('THE END');
  end;
end.
```

APPENDIX E GP and GP+A Code

Options	File (by build order)	Size
	Runtime.lib	18256
	Interface.lib	8104
D N V R	basic arcopt unit	11008
D N V R	BASECASEUNIT(B)	22514
D N V R	CWUNIT(B)3	24290
D N V R	BCKINSXX(B)3	27932

```
program BCKINSERT;
 uses
  BASECASE, CWUNIT, ARCOPT;
 label
  7478, 35, 6666, 2424, 2323, 1011, 1313, 383, 195, 197, 935, 777, 778, 296, 289, 188, 301, 38, 36, 39, 40, 100, 111, 112,
      114, 116, 50, 115, 120, 117, 118, 133, 134, 144, 145, 3333, 335, 336, 388, 201, 1212, 1213, 6767, 6777;
 type
  MATRIX3 = array[1..100, 1..11] of INTEGER;
  MATRIX4 = array[1..100, 1..6] of INTEGER;
  MATRIX10 = array[1..10, 1..10] of INTEGER;
  DATETIMEREC = record
    YEAR, MONTH, DAY, HOUR, MINUTE, SECOND, DAYOFWEEK: INTEGER;
   end;
 var
  ARCSAVINGS: INTEGER;
  RX, RY: ARRN11;
  ROUTELENGTH, NPTT, COUNTT, TROUTE: COUNTER3;
  RROUTE: array[1..50, 1..3] of INTEGER;
  NPT1: COUNTER;
  NPT: COUNTER;
  NARC: COUNTER;
  NTOURS: COUNTER;
  MAXDROP: COUNTER;
  NVT: COUNTER;
  MAXTIME: COUNTER;
  MILMIN: REAL1;
  ID: MATRIX1;
  PTOUR: MATRIX2;
  Q: IVECT100;
  X: VECT100;
  Y: VECT100;
  TRCAP: MATRIX5;
  SCALE: COUNTER;
  CONC: REAL1;
  REJARR: matrix9;
  C: MATRIX10;
  rejcount: INTEGER;
  II, IIII, III2, III3, III4: INTEGER;
  DATETIME: DATETIMEREC;
  T0, T1, T2, T3, T4, T5, T6, ELAPS, HRS, MINS, SECS, AHEAD, BEHIND: INTEGER;
  NARC1, NARC3: COUNTER;
  ROUTDATA: MATRIX3;
  CC: MATRIX4;
  V, NTOUR, III: COUNTER;
  I, J, CMIN1, CMIN2, CMIN3, CMIN4, CMIN5, CALC, BACKHAUL, JPOINT, IPOINT, DELIVER: INTEGER;
  PENALTY1, PERCENT1, INFLATION1, INCREASE: REAL;
  PENALTY, PERCENT, INFLATION: INTEGER;
  REJECTED, AFTER, AFTER1, NLINKS, MIDPOINT, YYYZ, XOFF, FLAG: INTEGER;
  NM1, IPLUS1, TEMP1, DIST1, DIST2, TEMP2, TEMP3, TEMP4, TEMP5, TEMPX, TEMPXX, NODES, K, L, XXXX, TDIST1, TDIST2, P0, P1,
      P2, P3, P4: INTEGER;
  TOTTIME1, TOTTIME, SAVINGS, THRESHOLD: INTEGER;
  JJ, DIRECTION, KK: INTEGER;
  TEMP, ICAP, ICAP1, ICAP2, ICAP5, ISUM, JJKK: INTEGER;
  NEWSCALE, XXX, TOURS, YYY: COUNTER;
  CCXX, CCXX2: INTEGER;
function C1 (X: VECT100; Y: VECT100; I: INTEGER; J: INTEGER): INTEGER;
 var
  CC1: INTEGER;
begin
  CC1 := ROUND(SQRT(SQR(X[I] - X[J]) + SQR(Y[I] - Y[J])));
{WRITELN('CC1 ', CC1);}
  C1 := CC1;
end;
procedure GETTIME (var DATETIME: DATETIMEREC);
EXTERNAL;
procedure CAPSET (var ID: MATRIX1; var PTOUR: MATRIX2; var Q: IVECT100; var ROUTDATA: MATRIX3; var CC: MATRIX4; var
      K: COUNTER; var NARC, NVT: COUNTER; var TRCAP: MATRIX5);
 label
  236, 235, 288;
```

```
  var
    ICAP, I, J, JJ, IPOINT, YYYZ, DIRECTION, KK, II: INTEGER;
 begin
  ICAP := 0;
  I := 0;
  J := 0;
  JJ := 0;
  IPOINT := 0;
  YYYZ := 0;
  DIRECTION := 0;
  KK := 0;
  II := 0;
  for J := 1 to NARC do
   begin
    JJ := CC[K, 2];
    if ((ID[J, 3] = 2) and (J = JJ)) then
     goto 236;
    if ID[J, 1] <> 1 then
     goto 288;
    if ((ROUTDATA[PTOUR[ID[JJ, 2], 1], 7] = ID[J, CC[K, 5]]) and (PTOUR[ID[JJ, 2], 1] = PTOUR[ID[J, 2], 1])) then
     goto 236
    else
     goto 235;
235:
   end;
236:
  YYYZ := PTOUR[ID[J, 2], 1];
  DIRECTION := J;
  ROUTDATA[YYYZ, 5] := DIRECTION;
  ROUTDATA[YYYZ, 10] := CC[K, 5];
{WRITELN('ROUTE', YYYZ, 'ROUTDATA[5]', ROUTDATA[YYYZ, 5], 'R[10]', ROUTDATA[YYYZ, 10]);}
288:
 end;
{CAPACITY CHECK SUBROUTINE}
 procedure CAPCHK (var ID: MATRIX1; var PTOUR: MATRIX2; var Q: IVECT100; var ROUTDATA: MATRIX3; var CC: MATRIX4; var
       K: COUNTER; var NARC, NVT: COUNTER; var TRCAP: MATRIX5; var V: COUNTER);
  label
   934;
  var
   ICAP, I, J, JJ, IPOINT, YYYZ, DIRECTION, KK, JJKK: INTEGER;
 begin
  ICAP := 0;
  I := 0;
  J := 0;
  JJ := 0;
  IPOINT := 0;
  YYYZ := 0;
  DIRECTION := 0;
  KK := 0;
  JJKK := 0;
  V := 0;
  JJ := CC[K, 2];
  JJKK := PTOUR[ID[JJ, 2], 1];
  YYYZ := ROUTDATA[JJKK, 5];
  ICAP := ROUTDATA[JJKK, 7];
  IPOINT := ID[YYYZ, 2];
  I := YYYZ;
{WRITELN('CC[K,2]', JJ, ' PTOUR[ID[JJ, 2], 1]', JJKK, 'ROUTDATA[JJKK, 5]', YYYZ);}
{WRITELN('ROUTDATA [ JJKK , 7 ]', ICAP, 'ID[YYYZ, 2]', IPOINT, 'YYYZ', I);}
  repeat
   ICAP := ICAP - Q[IPOINT];
   if ID[YYYZ, 5] = 0 then
    I := ID[I, 7]
   else
    I := ID[I, 8];
   while ICAP > TRCAP[NVT] do
    begin
     V := 1;
     goto 934;
```

```
    end;
   if IPOINT = ID[I, 1] then
    IPOINT := ID[I, 2]
   else
    IPOINT := ID[I, 1];
{WRITELN('IPOINT', IPOINT);}
  until IPOINT = 1;
934:
 end;
 procedure CAPREV (var ID: MATRIX1; var PTOUR: MATRIX2; var Q: IVECT100; var ROUTDATA: MATRIX3; var CC: MATRIX4; var
      K: COUNTER; var NARC, NVT: COUNTER; var TRCAP: MATRIX5; var NTOUR: COUNTER);
  label
   196, 199, 235, 236, 288, 101;
  var
   ICAP, I, J, JJ, IPOINT, YYYZ, DIRECTION, KK, JJKK, II: INTEGER;
 begin
  ICAP := 0;
  I := 0;
  J := 0;
  JJ := 0;
  IPOINT := 0;
  YYYZ := 0;
  DIRECTION := 0;
  KK := 0;
  JJKK := 0;
  II := 0;
  for II := 1 to NTOUR do
   begin
    JJKK := II;
    YYYZ := ROUTDATA[JJKK, 5];
    if ROUTDATA[JJKK, 9] = 3 then
     goto 288;
    while YYYZ = 0 do
     begin
      for J := 1 to NARC do
       begin
        if ID[J, 3] = 0 then
         goto 235;
        if JJKK = PTOUR[ID[J, 2], 1] then
         goto 236
        else
         goto 235;
236:
        YYYZ := J;
        while ID[J, 3] = 2 do
         begin
          ID[YYYZ, 6] := Q[ID[YYYZ, 2]];
          goto 288;
         end;
        goto 101;
235:
       end;
     end;
101:
    ICAP := ROUTDATA[JJKK, 7];
    IPOINT := ID[YYYZ, 2];
    while ID[YYYZ, 3] = 2 do
     begin
      ID[YYYZ, 6] := ICAP;
      goto 288;
     end;
    ID[YYYZ, 6] := ICAP;
196:
    I := YYYZ;
    if ID[I, 5] = 0 then
     I := ID[I, 7]
    else
     I := ID[I, 8];
    repeat
```

```
        If ROUTDATA[JJKK, 5] = 0 then
          ICAP := ICAP + Q[IPOINT]
        else
          ICAP := ICAP - Q[IPOINT];
        ID[I, 6] := ICAP;
        If IPOINT = ID[I, 1] then
          IPOINT := ID[I, 2]
        else
          IPOINT := ID[I, 1];
        If ID[YYYZ, 5] = 0 then
          I := ID[I, 7]
        else
          I := ID[I, 8];
      until IPOINT = 1;
  288:
    end;
  189:
  end;
  procedure NETGEN (var X, Y: VECT100; var NPT1, NPT, newscale: COUNTER);
   var
    i, I, j, NODES, backhaul, NPTB: integer;
    r, w: real;
    yy: array[1..200] of real;
  begin
   I := 0;
   I := 0;
   j := 0;
   NODES := 0;
   backhaul := 0;
   NPTB := 0;
   r := 0;
   w := 0;
   for I := 1 to 100 do
    begin
     YY[I] := 0;
    end;
   I := 1;
   P1 := ROUND(X[I]) * (newscale);
   P2 := ROUND(Y[I]) * newscale;
   PaintCircle(P1, P2, 5);
   Moveto(P1 - 20, P2);
 {WriteDraw(I);}
   for I := 2 to NPT do
    begin
     P1 := ROUND(X[I]) * (newscale);
     P2 := ROUND(Y[I]) * newscale;
     drawline(P1 - 3, P2 - 3, P1 + 3, P2 - 3);
     drawline(P1 - 3, P2 + 3, P1 + 3, P2 + 3);
     drawline(P1 - 3, P2 - 3, P1 - 3, P2 + 3);
     drawline(P1 + 3, P2 - 3, P1 + 3, P2 + 3);
     Moveto(P1 - 15, P2);
{WriteDraw(I);}
    end;
  NODES := NPT1;
  for I := NPT + 1 to NODES do
   begin
    P1 := ROUND(X[I]) * (newscale);
    P2 := ROUND(Y[I]) * newscale;
    drawline(P1, P2 - 4, P1 - 4, P2 + 4);
    drawline(P1, P2 - 4, P1 + 4, P2 + 4);
    drawline(P1 - 4, P2 + 4, P1 + 4, P2 + 4);
    Moveto(P1 - 15, P2);
{WriteDraw(I);}
   end;
 end;
begin
 II := 0;
 IIII := 0;
 III2 := 0;
```

```
 III3 := 0;
 III4 := 0;
 T0 := 0;
 T1 := 0;
 T2 := 0;
 T3 := 0;
 T4 := 0;
 T5 := 0;
 T6 := 0;
 ELAPS := 0;
 HRS := 0;
 MINS := 0;
 SECS := 0;
 AHEAD := 0;
 BEHIND := 0;
 NPT1 := 0;
 NPT := 0;
 NARC := 0;
 NARC1 := 0;
 NARC3 := 0;
 NTOURS := 0;
 MAXDROP := 0;
 NVT := 0;
 MAXTIME := 0;
 NARC1 := 0;
 NARC3 := 0;
 I := 0;
 J := 0;
 CMIN1 := 0;
 CMIN2 := 0;
 CMIN3 := 0;
 CMIN4 := 0;
 CMIN5 := 0;
 CALC := 0;
 BACKHAUL := 0;
 JPOINT := 0;
 IPOINT := 0;
 DELIVER := 0;
 PENALTY1 := 0;
 PERCENT1 := 0;
 INFLATION1 := 0;
 INCREASE := 0;
 PENALTY := 0;
 PERCENT := 0;
 INFLATION := 0;
 REJECTED := 0;
 AFTER := 0;
 AFTER1 := 0;
 NLINKS := 0;
 MIDPOINT := 0;
 YYYZ := 0;
 XOFF := 0;
 NM1 := 0;
 IPLUS1 := 0;
 TEMP1 := 0;
 DIST1 := 0;
 DIST2 := 0;
 TEMP2 := 0;
 TEMP3 := 0;
 TEMP4 := 0;
 TEMP5 := 0;
 TEMPX := 0;
 TEMPXX := 0;
 NODES := 0;
 K := 0;
 L := 0;
 XXXX := 0;
 TDIST1 := 0;
 TDIST2 := 0;
```

```
 P0 := 0;
 P1 := 0;
 P2 := 0;
 P3 := 0;
 P4 := 0;
 TOTTIME1 := 0;
 TOTTIME := 0;
 JJ := 0;
 DIRECTION := 0;
 KK := 0;
 V := 0;
 NTOUR := 0;
 III := 0;
 TEMP := 0;
 ICAP := 0;
 ICAP1 := 0;
 ICAP2 := 0;
 ICAP5 := 0;
 ISUM := 0;
 JJKK := 0;
 NEWSCALE := 0;
 XXX := 0;
 TOURS := 0;
 YYY := 0;
 CCXX := 0;
 CCXX2 := 0;
 MILMIN := 0;
 for I := 1 to 50 do
  begin
   for J := 1 to 8 do
    begin
     REJARR[I, J] := 0;
    end;
  end;
 rejecount := 0;
 for I := 1 to 100 do
  begin
   for J := 1 to 11 do
    begin
     ROUTDATA[I, J] := 0;
    end;
  end;
 for I := 1 to 100 do
  begin
   for J := 1 to 5 do
    begin
     CC[I, J] := 0;
    end;
  end;
 BASE(NPT1, NPT, NARC, NTOURS, MAXDROP, NVT, MAXTIME, MILMIN, ID, PTOUR, Q, X, Y, TRCAP, SCALE, CONC);
{WRITELN('NPT1', NPT1, 'NPT', NPT, 'NARC', NARC, 'NTOURS', NTOURS);}
{goto 301;}
 writeln('MIXED ROUTE PARAMETERS');
 WRITELN('ENTER DELIVERY AFTER PICKUP PENALTY(PERCENT');
 READLN(PENALTY);
 WRITELN('ENTER OVERTIME RATE(PERCENT)');
 READLN(PERCENT);
 WRITELN('ENTER SUPPLIER SURCHARGE(PERCENT)');
 READLN(INFLATION);
 PERCENT1 := PERCENT / 100;
 PENALTY1 := PENALTY / 100;
 INFLATION1 := INFLATION / 100;
 NODES := NPT1;
 DELIVER := NPT;
 BACKHAUL := NODES - DELIVER;
 REJARR[1, 1] := 1;
 REJARR[1, 2] := ROUND(X[1]);
 REJARR[1, 3] := ROUND(Y[1]);
 REJARR[1, 4] := 0;
```

```
  REJARR[1, 5] := ROUND(X[1]);
  REJARR[1, 6] := ROUND(Y[1]);
  REJECTED := 1;
{WRITELN('SCALE', SCALE, 'CONC', ROUND(CONC * 100));}
{WRITELN('LOWER ', ROUND(SCALE * (1 - CONC)), 'UPPER ', ROUND((SCALE * (1 - CONC)) + (2 * SCALE * CONC) - SCALE));}
 for I := DELIVER + 1 to DELIVER + BACKHAUL do
  begin
   REJECTED := REJECTED + 1;
   REJARR[REJECTED, 1] := I;
   REJARR[REJECTED, 4] := 0;
   REJARR[REJECTED, 5] := ROUND(X[I]);
   REJARR[REJECTED, 6] := ROUND(Y[I]);
{WRITELN('I ', I, ' X[I] ', ROUND(X[I]), ' Y[I] ', ROUND(Y[I]));}
   If ((X[I] >= SCALE * (1 - CONC)) and (X[I] <= SCALE * (1 - CONC) + (2 * (SCALE * CONC) - SCALE))) and ((Y[I] >= SCALE * (1 -
     CONC)) and (Y[I] <= SCALE * (1 - CONC) + (2 * (SCALE * CONC) - SCALE))) then
    begin
     REJARR[REJECTED, 2] := ROUND(X[1]);
     REJARR[REJECTED, 3] := ROUND(Y[1]);
     WRITELN(' NODE ', I, ' IS WITHIN CONCENTRATION AREA');
     goto 2424;
    end;
   REJARR[REJECTED, 2] := ROUND(X[I]);
   REJARR[REJECTED, 3] := ROUND(Y[I]);
2424:
  end;
 for I := 1 to REJECTED do
  begin
{WRITELN(REJARR[I, 1], '  ', REJARR[I, 2], '  ', REJARR[I, 3]);}
  end;
 REJECTED := REJECTED - 1;
 WRITELN('NO OF BACKHAULS', REJECTED);
 FLAG := 0;
 cwvrp(REJARR, REJECTED, SAVINGS, FLAG, THRESHOLD);
 for I := 1 to REJECTED + 1 do
  begin
{WRITELN(REJARR[I, 1], '  ', REJARR[I, 2], '  ', REJARR[I, 3], '  ', REJARR[I, 4]);}
   If REJARR[I, 4] = 1 then
    begin
     If (REJARR[I, 2] = ROUND(X[1])) and (REJARR[I, 3] = ROUND(Y[1])) then
      goto 2323;
     X[REJARR[I, 1]] := X[1];
     Y[REJARR[I, 1]] := Y[1];
{WRITELN('NODE ', REJARR[I, 1], ' IS IN DEDICATED BACKHAUL ROUTE');}
2323:
{WRITELN(REJARR[I, 1], '  ', REJARR[I, 2], '  ', REJARR[I, 3]);}
{WRITELN(REJARR[I, 1], '  ', ROUND(X[REJARR[I, 1]]), '  ', ROUND(Y[REJARR[I, 1]]));}
    end;
  end;
 for I := 1 to REJECTED + 1 do
  begin
   If REJARR[I, 7] <> 0 then
    WRITELN('IF INSERTED IN DED BK ROUTE NODE ', REJARR[I, 1], ' WOULD CHANGE DIST BY ', REJARR[I, 7]);
  end;
 REJECTED := 0;
 for I := 1 to NODES do
  begin
   for J := 1 to NODES do
    begin
{C[I, J] := ROUND(SQRT(SQR(X[I] - X[J]) + SQR(Y[I] - Y[J])));}
{WRITELN('C[I,J]', C[I, J]);}
{WRITELN('C1', C1(X, Y, I, J));}
     If (I = 1) and (J > DELIVER) then
      CC[J, 4] := ROUND((1 + INFLATION1) * C1(X, Y, I, J));
    end;
  end;
 for I := 1 to NTOURS do
  begin
   ROUTDATA[I, 1] := 0;
   ROUTDATA[I, 2] := 0;
```

```
    ROUTDATA[I, 3] := -1;
    ROUTDATA[I, 8] := 0;
   end;
  for J := 1 to NARC do
   begin
    ID[J, 6] := 0;
    if PTOUR[ID[J, 2], 1] > NTOURS then
     goto 112;
    if ID[J, 3] = 0 then
     goto 112;
    I := PTOUR[ID[J, 2], 1];
    while ID[J, 3] = 2 do
     begin
      if ID[J, 2] > DELIVER then
       ROUTDATA[I, 1] := ROUND(C1(X, Y, ID[J, 1], ID[J, 2]) * (1 + INFLATION1)) + ROUTDATA[I, 1]
      else
       ROUTDATA[I, 1] := (2 * C1(X, Y, ID[J, 1], ID[J, 2])) + ROUTDATA[I, 1];
      ROUTDATA[I, 3] := 1;
      goto 112;
     end;
    ROUTDATA[I, 1] := C1(X, Y, ID[J, 1], ID[J, 2]) + ROUTDATA[I, 1];
    ROUTDATA[I, 3] := ROUTDATA[I, 3] + 1;
112:
   end;
{  WRITELN('     I  ', 'LENGTH', '  DIRECT', ' #OFNODES', '  TIME');}
  for I := 1 to NTOURS do
   begin
{WRITELN('HI');}
    ROUTDATA[I, 8] := ROUND(ROUTDATA[I, 1] / MILMIN);
{WRITELN(I, ROUTDATA[I, 1], ROUTDATA[I, 2], ROUTDATA[I, 3], '  ', ROUTDATA[I, 8], '  ', TDIST1);}
    TDIST1 := TDIST1 + ROUTDATA[I, 1];
    TOTTIME := TOTTIME + ROUTDATA[I, 8];
   end;
{WRITELN('NPT1', NPT1, 'NPT', NPT, 'NARC', NARC, 'NTOURS', NTOURS);}
 WRITELN('TOTAL DISTANCE', TDIST1, '  ', 'TOTAL TIME ', TOTTIME);
 GETTIME(DATETIME);
 MINS := DATETIME.MINUTE * 60;
 SECS := DATETIME.SECOND;
 T0 := MINS + SECS;
 T2 := MINS + SECS;
 for J := 1 to NARC do
  begin
   while ID[J, 3] = 2 do
    begin
     ID[J, 7] := 0;
     ID[J, 8] := 0;
     goto 134;
    end;
   if ID[J, 3] = 0 then
    goto 134;
   if ID[J, 6] = 1 then
    goto 134;
   if ID[J, 1] <> 1 then
    goto 111;
   IPOINT := ID[J, 2];
   YYYZ := PTOUR[ID[J, 2], 1];
   ID[J, 5] := 0;
   ID[J, 4] := 0;
   ID[J, 7] := 0;
   ID[J, 8] := 0;
   CALC := 0;
   NLINKS := ROUTDATA[YYYZ, 3];
   MIDPOINT := ROUND(NLINKS / 2);
116:
   JPOINT := J;
   for I := J + 1 to NARC do
    begin
     if (CALC >= NLINKS) then
      goto 134;
```

```
      If ID[I, 3] = 0 then
       goto 114;
      If ID[I, 6] = 1 then
       goto 114;
      If (IPOINT <> ID[I, 1]) and (IPOINT <> ID[I, 2]) then
       goto 114;
      while IPOINT = ID[JPOINT, 2] do
       begin
        ID[JPOINT, 7] := I;
        ID[I, 8] := JPOINT;
        goto 3333;
       end;
      If ID[AHEAD, 1] <> 1 then
       ID[AHEAD, 7] := I;
      ID[I, 8] := BEHIND;
3333:
      AHEAD := I;
      BEHIND := I;
      CALC := CALC + 1;
      ID[I, 6] := 1;
      If CALC < MIDPOINT then
       ID[I, 4] := CALC
      else
       ID[I, 4] := NLINKS - CALC;
      If CALC < MIDPOINT then
       ID[I, 5] := 0
      else
       ID[I, 5] := 1;
      If IPOINT = ID[I, 1] then
       IPOINT := ID[I, 2]
      else
       IPOINT := ID[I, 1];
      goto 116;
114:
    end;
134:
  end;
111:
{WRITELN('     I   ', '  J   ', 'LNK STAT ', 'ROUTE', '  #OF DEL ', 'DIREC', '  AHEAD', '  BEHIND');}
 for J := 1 to NARC do
  begin
{If ID[J, 3] <> 0 then}
{WRITELN(ID[J, 1], ID[J, 2], ID[J, 3], PTOUR[ID[J, 2], 1], ID[J, 4], ID[J, 5], ID[J, 7], ID[J, 8]);}
  end;
 for II := 1 to NTOURS do
  begin
   for J := 1 to NARC do
    begin
     If ID[J, 3] = 0 then
      goto 335;
     If II = PTOUR[ID[J, 2], 1] then
      goto 336
     else
      goto 335;
336:
     YYYZ := J;
     while ID[J, 3] = 2 do
      begin
       ID[YYYZ, 6] := Q[ID[YYYZ, 2]];
       If (ID[YYYZ, 2] > DELIVER) then
        ICAP := 0
       else
        ICAP := Q[ID[YYYZ, 2]];
       goto 388;
      end;
     goto 201;
335:
    end;
201:
```

```
    ICAP := 0;
    ICAP5 := 0;
    III3 := 0;
    IIII := 0;
    III2 := 0;
    III4 := 0;
383:
    while III3 <> 0 do
     begin
      IPOINT := ID[III4, 2];
{WRITELN('IPOINT', IPOINT, 'I', III4, ' ID [ 5 ] ', ID[III4, 5]);}
      if ID[III4, 5] = 0 then
       goto 388;
      ICAP5 := ICAP5 + Q[IPOINT];
      ID[YYYZ, 10] := ICAP5;
{WRITELN('LINK1', I, 'I,J', ID[I, 1], ID[I, 2], 'ID[I,9]', ID[I, 9], 'ID[I,10]', ID[I, 10]);}
      if IPOINT = ID[YYYZ, 1] then
       IPOINT := ID[YYYZ, 2]
      else
       IPOINT := ID[YYYZ, 1];
      goto 296;
     end;
    IPOINT := ID[YYYZ, 2];
{ID[YYYZ, 6] := ICAP;}
296:
    I := YYYZ;
    if ID[I, 5] = 0 then
     I := ID[I, 7]
    else
     I := ID[I, 8];
    repeat
{WRITELN('IPOINT', IPOINT, 'I', I, 'ID[5]', ID[I, 5]);}
     if (III3 <> 0) then
      ICAP5 := ICAP5 + Q[IPOINT]
     else
      ICAP := ICAP + Q[IPOINT];
     if (III3 <> 0) then
      ID[I, 10] := ICAP5
     else
      ID[I, 9] := ICAP;
{ID[I, 6] := ICAP;}
{WRITELN('LINK', I, 'I,J', ID[I, 1], ID[I, 2], 'ID[I,9]', ID[I, 9], 'ID[I,10]', ID[I, 10]);}
     if IPOINT = ID[I, 1] then
      IPOINT := ID[I, 2]
     else
      IPOINT := ID[I, 1];
     IIII := ID[I, 7];
     III2 := ID[I, 8];
     III4 := I;
     if ID[YYYZ, 5] = 0 then
      I := ID[I, 7]
     else
      I := ID[I, 8];
    until IPOINT = 1;
    if III > 0 then
     YYYZ := IIII
    else
     YYYZ := III2;
    while (ID[YYYZ, 5] = 1) and (IPOINT = 1) do
     begin
      III3 := 1;
      I := YYYZ;
      goto 383;
     end;
388:
    ROUTDATA[II, 7] := ICAP;
{WRITELN('ROUTE', II, 'LOAD', ROUTDATA[II, 7]);}
   end;
289:
```

```
 GETTIME(DATETIME);
 MINS := DATETIME.MINUTE * 60;
 SECS := DATETIME.SECOND;
 T3 := MINS + SECS;
 If T3 < T2 then
  T3 := T3 + 3600;
 ELAPS := T3 - T2;
{WRITELN('ELAPSED TIME TO SET UP LINKS', ELAPS);}
 WRITELN('      I      ', 'COST', '      LNKID', '   RADIAL SAV', '  ROUTE COST', '   DEL PENALTY');
 CMIN3 := 0;
 for K := DELIVER + 1 to DELIVER + BACKHAUL do
  begin
   CMIN1 := 9999;
   CMIN2 := 9999;
   CMIN3 := 0;
   CMIN4 := 0;
   CMIN5 := 0;
   for I := 9 to 10 do
    begin
     for L := 1 to NARC do
      begin
       If ID[L, 3] = 0 then
        goto 120;
       If ID[L, 3] = 2 then
        goto 120;
       If (ID[L, 3] = 2) and (ID[L, 2] > DELIVER) then
        goto 120;
       CC[K, 1] := ROUND(C1(X, Y, ID[L, 1], K) + C1(X, Y, K, ID[L, 2]) - C1(X, Y, ID[L, 1], ID[L, 2]) - CC[K, 4] + ROUND(PENALTY1 *
      ID[L, I]));
       CC[K, 2] := L;
{    If CC[K, 1] < 0 then}
{    WRITELN(K, CC[K, 1], ID[CC[K, 2], 1], ID[CC[K, 2], 2], CC[K, 4], '      ', ROUND(C1(X, Y, ID[CC[K, 2], 1], K)+ C1(X, Y, K, ID[CC[K, 2],
      2]) - C1(X, Y, ID[CC[K, 2], 1], ID[CC[K, 2], 2])), '      ',ROUND (PENALTY1 * ID[CC[K, 2], I]));}
       If CMIN1 < CC[K, 1] then
        goto 120;
       CMIN1 := CC[K, 1];
       CMIN2 := CC[K, 2];
       CMIN4 := CC[K, 4];
       CMIN5 := I;
120:
      end;
     CC[K, 1] := CMIN1;
     CC[K, 2] := CMIN2;
     CC[K, 3] := K;
     CC[K, 4] := CMIN4;
     CC[K, 5] := CMIN5;
    end;
   WRITELN(K, CC[K, 1], ID[CC[K, 2], 1], ID[CC[K, 2], 2], CC[K, 4], '      ', ROUND(C1(X, Y, ID[CC[K, 2], 1], K) + C1(X, Y, K, ID[CC[K, 2],
     2]) - C1(X, Y, ID[CC[K, 2], 1], ID[CC[K, 2], 2])), '      ', ROUND(PENALTY1 * ID[CC[K, 2], CC[K, 5]]));
  end;
{  WRITELN('      K      ', 'COST', '      LNKID', '      ROUTE      ', '#OF DEL   ', '  DIREC ');}
 CMIN3 := 0;
 for K := DELIVER + 1 to DELIVER + BACKHAUL do
  begin
{WRITELN(K, CC[K, 1], ID[CC[K, 2], 1], ID[CC[K, 2], 2], PTOUR[ID[CC[K, 2], 2], 1], '      ', ID[CC[K, 2], 4], '      ', ID[CC[K, 2], 5]);}
  end;

 GETTIME(DATETIME);
 MINS := DATETIME.MINUTE * 60;
 SECS := DATETIME.SECOND;
 T4 := MINS + SECS;
 If T4 < T3 then
  T4 := T4 + 3600;
 ELAPS := T4 - T3;
{  WRITELN('ELAPSED TIME ASSIGN BK TO NEAREST LINK', ELAPS);}
 NM1 := DELIVER + BACKHAUL - 1;
 for I := DELIVER + 1 to NM1 do
  begin
   IPLUS1 := I + 1;
```

```
    for J := IPLUS1 to DELIVER + BACKHAUL do
     begin
      if CC[I, 1] <= CC[J, 1] then
       goto 50;
      TEMP1 := CC[I, 1];
      TEMP2 := CC[I, 2];
      TEMP3 := CC[I, 3];
      TEMP4 := CC[I, 4];
      TEMP5 := CC[I, 5];
      CC[I, 1] := CC[J, 1];
      CC[I, 2] := CC[J, 2];
      CC[I, 3] := CC[J, 3];
      CC[I, 4] := CC[J, 4];
      CC[I, 5] := CC[J, 5];
      CC[J, 1] := TEMP1;
      CC[J, 2] := TEMP2;
      CC[J, 3] := TEMP3;
      CC[J, 4] := TEMP4;
      CC[J, 5] := TEMP5;
50:
     end;
   end;
{   for I := 1 to 30 do}
 begin
{if REJARR[I, 7] <> 0 then}
{WRITELN('IF INSERTED IN DED BK ROUTE NODE ', REJARR[I, 1], ' WOULD CHANGE DIST BY ', REJARR[I, 7]);}
 end;
 for K := DELIVER + 1 to DELIVER + BACKHAUL do
  begin
   for I := 1 to 50 do
    begin
     if REJARR[I, 1] = 0 then
      goto 6767;
     if (CC[K, 3] <> REJARR[I, 1]) then
      goto 6767;
     CC[K, 6] := I;
     REJARR[I, 8] := CC[K, 1];
     if (CC[K, 1] >= 0) and (REJARR[I, 7] >= 0) then
      goto 6767;
     if (CC[K, 1] > REJARR[I, 7]) then
      begin
       TDIST1 := TDIST1 - CC[K, 4];
       TOTTIME := TOTTIME - ROUND(CC[K, 4] / MILMIN);
      end;
     if (CC[K, 1] > REJARR[I, 7]) then
      CC[K, 1] := 0
     else
      REJARR[I, 7] := 0;
     if (CC[K, 1] > REJARR[I, 7]) then
      begin
       X[CC[K, 3]] := X[1];
       Y[CC[K, 3]] := Y[1];
      end;
6777:
{WRITELN('CC[K,3]', CC[K, 3], 'CC[K,1]', CC[K, 1], 'REJARR[I,1]', REJARR[I, 1], 'REJARR[I,7]', REJARR[I, 7]);}
{WRITELN('X[]', ROUND(X[CC[K, 3]]), 'Y', ROUND(Y[CC[K, 3]]), 'CC[K,6]', CC[K, 6]);}
      WRITELN(' ');
6767:
    end;
  end;
 WRITELN('TOTAL DISTANCE', TDIST1, ' ', 'TOTAL TIME ', TOTTIME);

{   WRITELN('     I     ', 'COST', '     LNKID', '          K', '   NO OF DEL', 'REV COST');}

 for K := DELIVER + 1 to DELIVER + BACKHAUL do
  begin
{WRITELN(K, CC[K, 1], ID[CC[K, 2], 1], ID[CC[K, 2], 2], CC[K, 3], ID[CC[K, 2], 4], CC[K, 1] - ROUND(PENALTY1 * ID[CC[K, 2], CC[K,
      5]]));}
{if CC[K, 1] < 0 then}
```

```
{begin}
{CC[K, 1] := CC[K, 1] - ROUND(PENALTY1 * ID[CC[K, 2], CC[K, 5]]);}
{end;}
   end;
  FLAG := 1;
  cavrp(REJARR, REJECTED, SAVINGS, FLAG, THRESHOLD);
  for K := DELIVER + 1 to DELIVER + BACKHAUL do
   begin
    for I := 1 to 50 do
     begin
      if REJARR[I, 1] = 0 then
       goto 6666;
      if (CC[K, 3] <> REJARR[I, 1]) then
       goto 6666;
      if REJARR[I, 4] <> 2 then
       goto 6666;
      if (CC[K, 1] > REJARR[I, 7]) then
       begin
        TDIST1 := TDIST1 - CC[K, 4];
        TOTTIME := TOTTIME - ROUND(CC[K, 4] / MILMIN);
       end;
      if (CC[K, 1] > REJARR[I, 7]) then
       CC[K, 1] := 0
      else
       REJARR[I, 7] := 0;
      if (CC[K, 1] > REJARR[I, 7]) then
       begin
        X[CC[K, 3]] := X[1];
        Y[CC[K, 3]] := Y[1];
       end;
{WRITELN('CC[K,3]', CC[K, 3], 'CC[K,1]', CC[K, 1], 'REJARR[I,1]', REJARR[I, 1], 'REJARR[I,7]', REJARR[I, 7]);}
{WRITELN('X[]', ROUND(X[CC[K, 3]]), 'Y', ROUND(Y[CC[K, 3]]), 'CC[K,6]', CC[K, 6]);}
6666:
     end;
   end;
  WRITELN('TOTAL DISTANCE', TDIST1, ' ', 'TOTAL TIME ', TOTTIME);

{WRITELN('     I     ', 'COST', '     LNKID', '     K', '  NO OF DEL', 'REV COST');}

  for K := DELIVER + 1 to DELIVER + BACKHAUL do
   begin
{WRITELN(K, CC[K, 1], ID[CC[K, 2], 1], ID[CC[K, 2], 2], CC[K, 3], ID[CC[K, 2], 4], CC[K, 1] - ROUND(PENALTY1 * ID[CC[K, 2], CC[K,
       5]]));}
   end;

  GETTIME(DATETIME);
  MINS := DATETIME.MINUTE * 60;
  SECS := DATETIME.SECOND;
  T5 := MINS + SECS;
  if T5 < T4 then
   T5 := T5 + 3600;
  ELAPS := T5 - T4;
{WRITELN('ELAPSED TIME TO SORT', ELAPS);}
  for I := 1 to NTOURS do
   begin
    ROUTDATA[I, 2] := 2;
    ROUTDATA[I, 4] := 0;
    ROUTDATA[I, 6] := ROUTDATA[I, 3];
    ROUTDATA[I, 10] := 0;
    ROUTDATA[I, 11] := 0;
   end;
  for I := 1 to NARC do
   begin
    ID[I, 6] := 0;
   end;
  DIST2 := 0;
  DIST1 := 0;
  REJECTED := 0;
  AFTER := 0;
```

```
  V := 0;
  NARC1 := NARC;
  for K := DELIVER + 1 to DELIVER + BACKHAUL do
   begin
    if (X[CC[K, 3]] = X[1]) and (Y[CC[K, 3]] = Y[1]) then
     goto 133;
    if ID[CC[K, 2], 3] = 0 then
     goto 144;
    XXXX := PTOUR[ID[CC[K, 2], 2], 1];
    WRITELN('BKHAUL', CC[K, 3], 'INSERT LINK', ID[CC[K, 2], 1], ID[CC[K, 2], 2], 'ROUTE', XXXX, 'ROUTE START', ROUTDATA[XXXX, 5],
       'RDIRECTION', ROUTDATA[XXXX, 10]);
    while ROUTDATA[XXXX, 10] = 0 do
     begin
      if ROUTDATA[XXXX, 5] > 0 then
       goto 100;
      CAPSET(ID, PTOUR, Q, ROUTDATA, CC, K, NARC, NVT, TRCAP);
      goto 100;
     end;
    if CC[K, 5] <> ROUTDATA[XXXX, 10] then
     goto 117;
100:
    if ROUTDATA[XXXX, 5] = 0 then
     CAPSET(ID, PTOUR, Q, ROUTDATA, CC, K, NARC, NVT, TRCAP);
    if ROUTDATA[XXXX, 3] + 1 > MAXDROP then
     goto 133;
{WRITELN('CC[K,3]', CC[K, 3], 'CC[K,1]', CC[K, 1], 'CC[K,6]', CC[K, 6], 'REJARR[7]', REJARR[CC[K, 6], 7]);}
    if (CC[K, 1] >= 0) or (REJARR[CC[K, 6], 7] < 0) then
     goto 133;
    ICAP := TRCAP[NVT];
    P0 := CC[K, 2];
{WRITELN('P0', P0, 'ID[1]', ID[P0, 1], 'ID[2]', ID[P0, 2], 'ID[7]', ID[P0, 7], 'ID[8]', ID[P0, 8]);}
{WRITELN('ID[ID[7]]', ID[ID[P0, 7], 1], ID[ID[P0, 7], 2], 'ID[ID[8]]', ID[ID[P0, 8], 1], ID[ID[P0, 7], 2]);}
    NARC := NARC + 1;
    while ID[P0, 8] = 0 do
     begin
      ID[NARC, 1] := ID[P0, 1];
      goto 778;
     end;
{MAY BE BAD ASSUMPTION THAT "1" ALWAYS IN FIRST SPOT}
    while (ID[P0, 1] = 1) and (ID[P0, 7] = 0) do
     begin
      ID[NARC, 1] := ID[P0, 1];
      goto 778;
     end;
    if (ID[P0, 1] = ID[ID[P0, 8], 1]) or (ID[P0, 1] = ID[ID[P0, 8], 2]) then
     ID[NARC, 1] := ID[P0, 1]
    else
     ID[NARC, 1] := ID[P0, 2];
778:
    ID[NARC, 2] := CC[K, 3];
    ID[NARC, 3] := 1;
    ID[NARC, 5] := ID[P0, 5];
    ID[NARC, 9] := ID[P0, 9];
    ID[NARC, 10] := ID[P0, 10];
    if (ID[P0, 5] = 1) and (ID[P0, 1] = 1) and (ID[P0, 8] <> 0) then
     ID[ID[P0, 8], 7] := NARC + 1
    else if (ID[P0, 8] <> 0) then
     ID[ID[P0, 8], 7] := NARC;
    if (ID[P0, 5] = 1) and (ID[P0, 1] = 1) then
     ID[NARC, 7] := ID[P0, 7]
    else
     ID[NARC, 7] := NARC + 1;
    if (ID[P0, 5] = 1) and (ID[P0, 1] = 1) then
     ID[NARC, 8] := NARC + 1
    else
     ID[NARC, 8] := ID[P0, 8];
    if (ID[P0, 3] <> 2) and (ROUTDATA[XXXX, 5] = CC[K, 2]) and (ID[P0, 1] = 1) then
     ROUTDATA[XXXX, 5] := NARC;
{WRITELN('NARC', NARC, 'ID[1],ID[2]', ID[NARC, 1], ID[NARC, 2]);}
```

```
    NARC := NARC + 1;
    ID[NARC, 1] := CC[K, 3];
    while ID[P0, 7] = 0 do
     begin
      ID[NARC, 2] := ID[P0, 2];
      goto 777;
     end;
{MAY BE BAD ASSUMPTION}
    while (ID[P0, 1] = 1) and (ID[P0, 8] = 0) do
     begin
      ID[NARC, 2] := ID[P0, 2];
      goto 777;
     end;
    if (ID[P0, 2] = ID[ID[P0, 7], 1]) or (ID[P0, 2] = ID[ID[P0, 7], 2]) then
     ID[NARC, 2] := ID[P0, 2]
    else
     ID[NARC, 2] := ID[P0, 1];
777:
    ID[NARC, 3] := 1;
    ID[NARC, 5] := ID[P0, 5];
    ID[NARC, 9] := ID[P0, 9];
    ID[NARC, 10] := ID[P0, 10];
    if (ID[P0, 5] = 1) and (ID[P0, 1] = 1) and (ID[P0, 7] <> 0) then
     ID[ID[P0, 7], 8] := NARC + 1
    else if (ID[P0, 7] <> 0) then
     ID[ID[P0, 7], 8] := NARC;
    if (ID[P0, 5] = 1) and (ID[P0, 1] = 1) then
     ID[NARC, 7] := NARC - 1
    else
     ID[NARC, 7] := ID[P0, 7];
    if (ID[P0, 5] = 1) and (ID[P0, 1] = 1) then
     ID[NARC, 8] := ID[P0, 8]
    else
     ID[NARC, 8] := NARC - 1;
    if (ID[P0, 3] <> 2) and (ROUTDATA[XXXX, 5] = CC[K, 2]) and (ID[P0, 2] = 1) then
     ROUTDATA[XXXX, 5] := NARC;
    ID[P0, 3] := ID[P0, 3] - 1;
{WRITELN('NARC', NARC, 'ID[1],ID[2]', ID[NARC, 1], ID[NARC, 2]);}
    CAPCHK(ID, PTOUR, Q, ROUTDATA, CC, K, NARC, NVT, TRCAP, V);
    while V = 1 do
     begin
      ID[P0, 3] := ID[P0, 3] + 1;
      ID[NARC, 1] := 0;
      ID[NARC, 2] := 0;
      ID[NARC, 3] := 0;
      ID[NARC, 5] := 0;
      ID[NARC, 7] := 0;
      ID[NARC, 8] := 0;
      ID[NARC, 9] := 0;
      ID[NARC, 10] := 0;
      if (ROUTDATA[XXXX, 5] = NARC) then
       ROUTDATA[XXXX, 5] := CC[K, 2];
      NARC := NARC - 1;
      ID[NARC, 1] := 0;
      ID[NARC, 2] := 0;
      ID[NARC, 3] := 0;
      ID[NARC, 5] := 0;
      ID[NARC, 7] := 0;
      ID[NARC, 8] := 0;
      ID[NARC, 9] := 0;
      ID[NARC, 10] := 0;
      if (ROUTDATA[XXXX, 5] = NARC) then
       ROUTDATA[XXXX, 5] := CC[K, 2];
      NARC := NARC - 1;
      while (ID[P0, 5] = 1) and (ID[P0, 1] = 1) do
       begin
        ID[ID[P0, 7], 8] := NARC;
        goto 133;
       end;
```

```
        while (ID[P0, 5] = 0) and (ID[P0, 1] = 1) do
          begin
           ID[ID[P0, 8], 7] := NARC;
           goto 133;
          end;
        goto 133;
       end;
      WRITELN('CONNECT BACKHAUL', CC[K, 3]);
      ID[P0, 3] := ID[P0, 3] + 1;
      P1 := PTOUR[ID[P0, 2], 1];
      P2 := PTOUR[ID[P0, 2], 2];
      P3 := PTOUR[ID[P0, 2], 3];
      P4 := PTOUR[ID[P0, 2], 4];
      NARC := NARC - 1;
      ID[NARC, 4] := ID[P0, 4];
      ID[NARC, 9] := ID[P0, 9];
      ID[NARC, 10] := ID[P0, 10];
      NARC := NARC + 1;
      ID[NARC, 4] := ID[P0, 4];
      ID[NARC, 5] := ID[P0, 5];
      ID[NARC, 9] := ID[P0, 9];
      ID[NARC, 10] := ID[P0, 10];
      ID[P0, 3] := ID[P0, 3] - 1;
  {DISCONNECT BKHAUL LINK}
  {WRITELN(NARC1, NPT1, CC[K, 3], 'ID[1..2]', ID[NARC1 - (NPT1 - CC[K, 3]), 1], ID[NARC1 - (NPT1 - CC[K, 3]), 2]);}
      ID[NARC1 - (NPT1 - CC[K, 3]), 3] := 0;
  {WRITELN('ROUTDATA[BKHAUL]', ROUTDATA[PTOUR[ID[NARC1 - (NPT1 - CC[K, 3]), 2], 1], 9]);}
      ROUTDATA[PTOUR[ID[NARC1 - (NPT1 - CC[K, 3]), 2], 1], 9] := 3;
      PTOUR[CC[K, 3], 1] := P1;
      PTOUR[CC[K, 3], 2] := P2;
      PTOUR[CC[K, 3], 3] := P3;
      PTOUR[CC[K, 3], 4] := P4 + 1;
      PTOUR[ID[P0, 1], 4] := P4 + 1;
      PTOUR[ID[P0, 2], 4] := P4 + 1;
      TDIST2 := TDIST2 + ROUND(CC[K, 1]);
      TOTTIME1 := TOTTIME1 + ROUND(CC[K, 1] / MILMIN);
      WRITELN('BKHAUL', CC[K, 3], ' INSERTION SAVINGS ', CC[K, 1], ' LNKID ', ID[CC[K, 2], 1], ID[CC[K, 2], 2]);
      if ROUTDATA[XXXX, 10] = 0 then
       ROUTDATA[P1, 2] := CC[K, 5];
      if ID[P0, CC[K, 5]] > ROUTDATA[XXXX, 11] then
       ROUTDATA[XXXX, 11] := ID[P0, CC[K, 5]];
      if ID[P0, 4] > ROUTDATA[XXXX, 4] then
       ROUTDATA[XXXX, 4] := ID[P0, 4];
      WRITELN('ROUTE', XXXX, 'BKHAUL', CC[K, 3], 'NO OF DEL AFTER PICKUP', ROUTDATA[XXXX, 4]);
      ROUTDATA[XXXX, 3] := ROUTDATA[XXXX, 3] + 1;
      ROUTDATA[XXXX, 8] := ROUTDATA[XXXX, 8] + ROUND((((1 + INFLATION1) * C1(X, Y, 1, CC[K, 3])) + CC[K, 1]) / MILMIN);
  {    WRITELN('ROUT[,8]', ROUTDATA[XXXX, 8], 'C1(X, Y, 1,CC[K,3])', (1 + INFLATION1) * C1(X, Y, 1, CC[K, 3]), 'CC[K,1]', CC[K, 1]);}
      goto 118;

  117:

  {WRITELN('CC[K,1]', CC[K, 1], 'CC[K, 5] ', CC[K, 5]);}
  {WRITELN('ID[CC[K, 2], 9]', ID[CC[K, 2], 9], 'ID[CC[K, 2], 10]', ID[CC[K, 2], 10]);}
      if (CC[K, 5] = 10) then
       CC[K, 1] := CC[K, 1] + ROUND(PENALTY1 * ID[CC[K, 2], 9])
      else
       CC[K, 1] := CC[K, 1] + ROUND(PENALTY1 * ID[CC[K, 2], 10]);
  {WRITELN('CC[K,1]', CC[K, 1]);}
  {CMIN1 := CC[K, 1];}
  {CMIN2 := CC[K, 2];}
  {CMIN3 := CC[K, 3];}
      CMIN1 := 9999;
      CMIN2 := 9999;
      CMIN3 := 0;
      goto 145;
  144:
      CMIN1 := 9999;
      CMIN2 := 9999;
      CMIN3 := 0;
```

```
    CMIN4 := 0;
    CMIN5 := 0;
145:
    for J := 9 to 10 do
     begin
      for L := 1 to NARC do
       begin
        if (ID[L, 3] = 2) and (ID[L, 2] > DELIVER) then
         goto 115;
        if ID[L, 3] = 0 then
         goto 115;
        if ID[L, 3] = 2 then
         goto 115;
        TEMPX := PTOUR[ID[L, 2], 1];
        if L = CC[K, 2] then
         goto 115;
{WRITELN('ID[1]', ID[L, 1], 'ID[2]', ID[L, 2], 'L', L, 'TEMPX', TEMPX);}
        if (ROUTDATA[TEMPX, 10] = 0) then
         CC[K, 1] := ROUND(C1(X, Y, ID[L, 1], CC[K, 3]) + C1(X, Y, CC[K, 3], ID[L, 2]) - C1(X, Y, ID[L, 1], ID[L, 2]) - CC[K, 4] +
       ROUND(PENALTY1 * ID[L, J]));
        if (ROUTDATA[TEMPX, 10] <> 0) then
         CC[K, 1] := ROUND(C1(X, Y, ID[L, 1], CC[K, 3]) + C1(X, Y, CC[K, 3], ID[L, 2]) - C1(X, Y, ID[L, 1], ID[L, 2]) - CC[K, 4] +
       ROUND(PENALTY1 * ID[L, ROUTDATA[TEMPX, 10]]));
        if CMIN1 < CC[K, 1] then
         goto 115;
        CMIN1 := CC[K, 1];
        CMIN2 := L;
        if (ROUTDATA[TEMPX, 10] = 0) then
         CMIN5 := J
        else
         CMIN5 := ROUTDATA[TEMPX, 10];
{WRITELN('L', L, 'J', J, 'CMIN1', CMIN1, 'CMIN2', CMIN2, 'CMIN5', CMIN5);}
115:
       end;
     end;
    CC[K, 1] := CMIN1;
    CC[K, 2] := CMIN2;
    CC[K, 5] := CMIN5;
{WRITELN('CMIN1', CMIN1, 'CMIN2', CMIN2, 'CMIN5', CMIN5);}
    TEMPXX := PTOUR[ID[CMIN2, 2], 1];
{WRITELN('CC[K,3]', CC[K, 3], 'CC[K,1]', CC[K, 1]);}

{if CC[K, 1] < 0 then}
{CC[K, 1] := CC[K, 1] - ROUND(PENALTY1 * ID[CC[K, 2], CC[K, 5]]);}

{CC[K, 1] := CC[K, 1] - ROUND(PENALTY1 * ID[CC[K, 2], CC[K, 5]]);}
{WRITELN('CC[K,3]', CC[K, 3], 'CC[K,1]', CC[K, 1]);}
    XXXX := TEMPXX;
    WRITELN('REDIRECT', CC[K, 3], 'LINK ID', ID[CC[K, 2], 1], ID[CC[K, 2], 2], 'CC[K,1]', CC[K, 1], 'ROUTE', XXXX);
    if (ID[CMIN2, 5] = ROUTDATA[TEMPXX, 2]) or (ROUTDATA[TEMPXX, 2] = 2) then
     goto 100;
    ID[CMIN2, 4] := ROUTDATA[TEMPXX, 6] - ID[CC[K, 2], 4];
    goto 100;
133:
    REJECTED := REJECTED + 1;
{WRITELN('REJECT BACKHAUL', CC[K, 3]);}
{REJARR[REJECTED + 1, 1] := CC[K, 3];}
{REJARR[REJECTED + 1, 2] := ROUND(X[CC[K, 3]]);}
{REJARR[REJECTED + 1, 3] := ROUND(Y[CC[K, 3]]);}
118:
  end;
 GETTIME(DATETIME);
 MINS := DATETIME.MINUTE * 60;
 SECS := DATETIME.SECOND;
 T6 := MINS + SECS;
 if T6 < T5 then
  T6 := T6 + 3600;
 ELAPS := T6 - T5;
{  WRITELN('ELAPSED TIME TO INSERT BKHAULS', ELAPS);}
```

```
{    while ID[P0, 7] = 0 do}
 begin
{ID[NARC, 2] := ID[P0, 2];}
{writeln('ID[NARC, 2] ', ID[NARC, 2], 'ID[P0, 2]', ID[P0, 2]);}
{    goto 777;}
 end;
 GETTIME(DATETIME);
 MINS := DATETIME.MINUTE * 60;
 SECS := DATETIME.SECOND;
 T1 := MINS + SECS;
 If T1 < T0 then
  T1 := T1 + 3600;
 ELAPS := T1 - T0;
 NTOUR := NTOURS;
{   CAPREV(ID, PTOUR, Q, ROUTDATA, CC, K, NARC, NVT, TRCAP, NTOUR);}
{WRITELN('REVISED ARCS');}
{   WRITELN('      I   ', ' J  ', 'ROUTE', '  #OF DEL ', 'DIREC', '  CAP', '   AHEAD', '  BEHIND');}
 for J := 1 to NARC do
  begin
{    if ID[J, 3] <> 0 then}
{     WRITELN(ID[J, 1], ID[J, 2], PTOUR[ID[J, 2], 1], ID[J, 4], ID[J, 5], ID[J, 6], ID[J, 7], ID[J, 8]);}
  end;
 AFTER := 0;
 AFTER1 := 0;
 for I := 1 to NTOURS do
  begin
   AFTER := AFTER + ROUTDATA[I, 4];
   AFTER1 := AFTER1 + ROUTDATA[I, 11];
   If ROUTDATA[I, 11] > 0 then
    WRITELN('ROUTE', I, 'DELIVERY STOPS AFTER PICKUP', ROUTDATA[I, 4], 'DELIVERY LOAD AFTER PICKUP ', ROUTDATA[I, 11]);
  end;
 WRITELN('TOTAL ELAPSED TIME IS', ELAPS, '  SECONDS');
 WRITELN('NODES', DELIVER + BACKHAUL);
 WRITELN('ARCS', NARC);
 WRITELN('BACKHAUL', BACKHAUL);
 WRITELN('PENALTY(PERCENT)', PENALTY);
 WRITELN('TOTAL DISTANCE IS', (TDIST1 + TDIST2), 'UNITS');
 WRITELN('BACKHAUL INSERTION SAVINGS', -TDIST2);
 INCREASE := -TDIST2 / TDIST1;
 WRITELN('% DECREASE IN DISTANCE', ROUND(INCREASE * 100));
{WRITELN('TOTAL TRAVEL TIME IS', (TOTTIME + TOTTIME1), 'MINUTES');}
{WRITELN('BACKHAUL INSERTION TRAVEL TIME SAVINGS', -TOTTIME1);}
 INCREASE := 0;
 INCREASE := -TOTTIME1 / TOTTIME;
{WRITELN('% DECREASE IN TRAVEL TIME', ROUND(INCREASE * 100));}
 WRITELN('TOTAL # OF REJECTED BACKHAULS', REJECTED);
 WRITELN('TOTAL # OF DELIVERIES AFTER PICKUPS', AFTER);
 WRITELN('TOTAL LOAD OF DELIVERIES AFTER PICKUPS', AFTER1);
{    goto 301;}
 for III := 1 to NTOUR do
  begin
   ICAP := 0;
   ICAP1 := 0;
   ICAP2 := 0;
   JJKK := III;
   YYYZ := ROUTDATA[JJKK, 5];
   If ROUTDATA[III, 9] = 3 then
    goto 188;
   while YYYZ = 0 do
    begin
     for J := 1 to NARC do
      begin
       If ID[J, 3] = 0 then
        goto 197;
       If JJKK = PTOUR[ID[J, 2], 1] then
        goto 195
       else
        goto 197;
195:
```

```
          YYYZ := J;
          IPOINT := ID[J, 2];
          while ID[J, 3] = 2 do
           begin
            ICAP := ICAP + Q[IPOINT];
            if Q[IPOINT] >= 0 then
             ICAP1 := ICAP1 + Q[IPOINT]
            else
             ICAP2 := ICAP2 - Q[IPOINT];
            goto 935;
           end;
          goto 1011;
 197:
        end;
      end;
1011:
    I := YYYZ;
    IPOINT := ID[I, 2];
    while ID[I, 3] = 2 do
     begin
      ICAP := ICAP + Q[IPOINT];
      if Q[IPOINT] >= 0 then
       ICAP1 := ICAP1 + Q[IPOINT]
      else
       ICAP2 := ICAP2 - Q[IPOINT];
      goto 935;
     end;
    repeat
     ICAP := ICAP + Q[IPOINT];
     if Q[IPOINT] >= 0 then
      ICAP1 := ICAP1 + Q[IPOINT]
     else
      ICAP2 := ICAP2 - Q[IPOINT];
     if ID[YYYZ, 5] = 0 then
      I := ID[I, 7]
     else
      I := ID[I, 8];
     if IPOINT = ID[I, 1] then
      IPOINT := ID[I, 2]
     else
      IPOINT := ID[I, 1];
    until IPOINT = 1;
935:
    ROUTDATA[III, 1] := ICAP;
    ROUTDATA[III, 2] := ICAP1;
    ROUTDATA[III, 3] := ICAP2;
188:
  end;
 ISUM := NTOURS;
 WRITELN('NTOURS', NTOURS);
 WRITELN('    ROUTE', ' NO OF DEL', ' NO OF PICKUPS', ' TRIP DURATION', ' OVERTIME');
 for J := 1 to ISUM - BACKHAUL do
  begin
   if ROUTDATA[J, 9] = 3 then
    goto 1313;
   CCXX := ROUTDATA[J, 8] - MAXTIME * 60;
   if CCXX <= 0 then
    CCXX := 0;
   CCXX2 := CCXX2 + CCXX;
   WRITELN(J, '  ', ROUTDATA[J, 2], '  ', ROUTDATA[J, 3], '          ', ROUTDATA[J, 8], '    ', CCXX);
1313:
  end;
 for I := 1 to 50 do
  begin
   RROUTE[I, 1] := 0;
   RROUTE[I, 2] := 0;
   RROUTE[I, 3] := 0;
  end;
NTOURS := ISUM;
```

```
 for J := 1 to NARC do
  begin
   if ID[J, 3] = 0 then
    goto 35;
   RROUTE[PTOUR[ID[J, 2], 1], 1] := RROUTE[PTOUR[ID[J, 2], 1], 1] + ROUND(SQRT(SQR(X[ID[J, 1]] - X[ID[J, 2]]) + SQR(Y[ID[J, 1]] -
       Y[ID[J, 2]]))):
   RROUTE[PTOUR[ID[J, 2], 1], 2] := PTOUR[ID[J, 2], 4];
35:
  end;
 TROUTE := 0;
 ARCSAVINGS := 0;
 for II := 1 to ISUM do
  begin
   if (ROUND(RROUTE[II, 1]) = 0) then
    goto 7478;
   if (rroute[II, 2] < 2) then
    goto 7478;
{WRITELN(' NO OF STOPS', RROUTE[II, 2]);}
   TROUTE := II;
   RROUTE[II, 2] := 0;
   RX[1] := X[1];
   RY[1] := Y[1];
{WRITELN('RX', ROUND(X[1]), 'RY', ROUND(Y[1]));}
   COUNTT := 0;
   for I := 1 to NPT1 do
    begin
     if TROUTE = PTOUR[I, 1] then
      begin
       RROUTE[II, 2] := RROUTE[II, 2] + 1;
       COUNTT := COUNTT + 1;
{WRITELN('COUNT', COUNTT, 'NODE', I, 'PTOUR[I,1]', PTOUR[I, 1], 'RX', ROUND(X[I]), 'RY', ROUND(Y[I]));}
       RX[COUNTT + 1] := X[I];
       RY[COUNTT + 1] := Y[I];
      end;
    end;
   WRITELN(' NO OF STOPS', RROUTE[II, 2], 'ROUTELNGTH', RROUTE[II, 1]);
   NPTT := RROUTE[II, 2] + 1;
   ROUTELENGTH := 0;
   POSTARC(RX, RY, NPTT, ROUTELENGTH);
   WRITELN('CW ROUTE', RROUTE[II, 1], 'ARC ROUTE', ROUTELENGTH);
   if (RROUTE[II, 1] > ROUTELENGTH) and (ROUTDATA[II, 3] = 0) then
    begin
     WRITELN('ROUTE SAVINGS ', RROUTE[II, 1] - ROUTELENGTH);
     ARCSAVINGS := ARCSAVINGS + RROUTE[II, 1] - ROUTELENGTH;
    end;
7478:
  end;
 WRITELN('ARCSAVINGS ON MIXED ROUTES', ARCSAVINGS);
{WRITELN('TOTAL OVERTIME PENALTY(IN DIST UNITS)', ROUND(CCXX2 * MILMIN * PERCENT1));}
 XXX := 1;
{WRITELN('REJ NODE  ', 'REJ X COORD ', ' REJ Y COORD ');}
 for I := 1 to REJECTED + 1 do
  begin
{WRITELN(REJARR[I, 1], '   ', REJARR[I, 2], '   ', REJARR[I, 3]);}
  end;
1213:
{writeln('enter node 1', ' enter node 2');}
{readln(I, J);}
{   if I = 0 then}
{   goto 1212;}
{   C1(X, Y, I, J) := ROUND(SQRT(SQR(X[I] - X[J]) + SQR(Y[I] - Y[J])));}
{   writeln('distance for ', I, ' and ', J, ' is ', C1(X, Y, I, J));}
{   goto 1213;}
 I := 0;
1212:
{WRITELN('ENTER NEW GRAPH SCALE FOR SPECIFIC ROUTES(enter 0 for none, or integer)');}
{readln(newscale);}
 NEWSCALE := 1;
 if newscale = 0 then
```

```
  goto 301;
38:
{writeln('ENTER ROUTE NUMBER DESIRED(enter 0 for none,enter 100 for all)');}
{readln(tours);}
 TOURS := 100;
 if tours = 0 then
  goto 301;
 if XXX = 0 then
  goto 39;
{writeln('ENTER 0 IF YOU WOULD LIKE PREVIOUS TOUR ERASED');}
{   writeln('ENTER 1 OTHERWISE');}
{   readln(YYY);}
 YYY := 1;
 if YYY = 1 then
  goto 40;
39:
 PaintCircle(0, 0, 500);
 InvertCircle(0, 0, 500);
 XXX := 1;
 NETGEN(X, Y, NPT1, NPT, newscale);
40:
 newscale := 2;
 for J := 1 to NARC do
  begin
   if (ID[J, 3] = 0) then
    goto 36;
   if (PTOUR[ID[J, 2], 1] = 0) then
    goto 36;
   P1 := ROUND(X[ID[J, 1]]) * (newscale);
   P2 := ROUND(Y[ID[J, 1]]) * newscale;
   P3 := ROUND(X[ID[J, 2]]) * (newscale);
   P4 := ROUND(Y[ID[J, 2]]) * newscale;
   if (ID[J, 3] = 2) and (ID[J, 2] > DELIVER) then
    goto 36;
   DRAWLINE(P1, P2, P3, P4);
36:
  end;
 if tours = 100 then
  goto 301;
 goto 38;
301:
 WRITELN('TOTAL DISTANCE INCL MIXED ROUTES', (TDIST1 + TDIST2) - ARCSAVINGS, 'UNITS');
 WRITELN('TOTAL DEDICATED BKHAUL ROUTES ARE', SAVINGS, 'UNITS');
 WRITELN('NEW TOTAL DISTANCE IS', (TDIST1 + TDIST2) + SAVINGS - ARCSAVINGS, 'UNITS');
end.
```

```
unit BASECASE;
interface
 uses
  ARCOPT;
 type
  MATRIX1 = array[1..1000, 1..10] of INTEGER;
  MATRIX2 = array[1..100, 1..5] of INTEGER;
  COUNTER = INTEGER;
  VECT100 = array[1..100] of REAL;
  IVECT100 = array[1..100] of INTEGER;
  MATRIX5 = array[1..5] of INTEGER;
  REAL1 = REAL;
 procedure BASE (var NPT1: COUNTER; var NPT: COUNTER; var NARC: COUNTER; var NTOURS: COUNTER; var MAXDROP: COUNTER;
       var NVT: COUNTER; var MAXTIME: COUNTER; var MILMIN: REAL1; var ID: MATRIX1; var PTOUR: MATRIX2; var Q: IVECT100;
       var X: VECT100; var Y: VECT100; var TRCAP: MATRIX5; var SCALE: COUNTER; var CONC: REAL1);
implementation
 procedure BASE (var NPT1: COUNTER; var NPT: COUNTER; var NARC: COUNTER; var NTOURS: COUNTER; var MAXDROP: COUNTER;
       var NVT: COUNTER; var MAXTIME: COUNTER; var MILMIN: REAL1; var ID: MATRIX1; var PTOUR: MATRIX2; var Q: IVECT100;
       var X: VECT100; var Y: VECT100; var TRCAP: MATRIX5; var SCALE: COUNTER; var CONC: REAL1);
  label
   7478, 6767, 1010, 102, 998, 776, 17, 16, 10, 64, 105, 11, 777, 97, 12, 301, 98, 90, 35, 36, 38, 39, 40, 91, 355;
  type
   COUNTER = INTEGER;
   VECTOR = array[1..1000] of real;
   IVECTOR = array[1..1000] of INTEGER;
   IVECT100 = array[1..100] of INTEGER;
   MATRIX3 = array[1..100, 1..8] of INTEGER;
   MATRIX4 = array[1..100, 1..3] of INTEGER;
   MATRIX5 = array[1..5] of INTEGER;
   COUNTER2 = INTEGER;
   REAL2 = REAL;
   DateTimeRec = record
     Year, Month, Day, Hour, Minute, Second, DayOfWeek: integer
    end;
   OUTREC = record
     NO1: INTEGER;
     NO2: INTEGER;
     NO3: INTEGER;
     NO4: INTEGER;
     NO5: INTEGER;
     NO6: INTEGER;
     NO7: INTEGER;
     NO8: REAL
    end;
  var
   DateTime: DateTimeRec;
   TOTTIME, SUM1: EXTENDED;
   SPEED: REAL;
   T0, T1, T2, Elaps2, Elaps, Hrs, Mins, Secs, CUM: integer;
   NN, IC, BKPTR, tours, XXX, YYY, newscale, NARC2: COUNTER;
   S: VECTOR;
   IX, JX: IVECTOR;
   JO, K, P, LIN, LJN, IPT1, IPT2, ICAP, MCAP, LM, L, M, ITK, HIT, KT, IT, P1, P2, P3, P4: INTEGER;
   TDIST, SUM, DISTAN: REAL;
   D: array[1..100] of real;
   RROUTE: array[1..50, 1..3] of INTEGER;
   ADJA: array[1..100, 1..100] of integer;
   TRA, TRU: MATRIX5;
   POINT: array[1..100] of INTEGER;
   BX, BY: array[1..100] of INTEGER;
   I, WIDTH, HEIGHT, DIVISOR, II, NT, J, IP1, IA, IB, ISUM: INTEGER;
   TCAP: COUNTER;
   RX, RY: ARRN11;
   ROUTELENGTH, NPTT, COUNTT, TROUTE: COUNTER3;
procedure GETTIME (var DATETIME: DATETIMEREC);
EXTERNAL;
procedure STHEAP (var NN: COUNTER; var S: VECTOR; var IX, JX: IVECTOR);
 label
  1, 50, 60, 2, 5, 7, 9;
```

```
    var
      NH, II, I, IXC, JXC, J: INTEGER;
      DT, COPY: REAL;
    begin
     nh := 0;
     ii := 0;
     i := 0;
     ixc := 0;
     jxc := 0;
     j := 0;
     dt := 0;
     copy := 0;
     if (NN - 2) < 0 then
      goto 50;
     if (NN - 2) = 0 then
      goto 60;
     if (NN - 2) > 0 then
      goto 1;
1:
     NH := ROUND(NN / 2);
     for I := 1 to NH do
      begin
       II := NH + 1 - I;
       COPY := S[II];
       IXC := IX[II];
       JXC := IXC;
2:
       J := II + II;
       if (J - NN) < 0 then
        goto 5;
       if (J - NN) = 0 then
        goto 7;
       if (J - NN) > 0 then
        goto 9;
5:
       if S[J + 1] > S[J] then
        J := J + 1;
7:
       if S[J] <= COPY then
        goto 9;
       S[II] := S[J];
       IX[II] := IX[J];
       JX[IX[J]] := II;
       II := J;
       goto 2;
9:
       S[II] := COPY;
       IX[II] := IXC;
       JX[JXC] := II;
      end;
     goto 50;
60:
     if S[1] >= S[2] then
      goto 50;
     DT := S[1];
     S[1] := S[NN];
     S[2] := DT;
     IX[1] := 2;
     JX[1] := 2;
     IX[2] := 1;
     JX[2] := 1;
50:
    end;
    procedure OTHEAP (var NN: COUNTER; var S: VECTOR; var IX: IVECTOR; var JX: IVECTOR; IC: COUNTER);

    label
     13, 111, 112, 19;
    var
     II, IXC, JXC, J: INTEGER;
```

```
    COPY: REAL;
  begin
   ii := 0;
   ixc := 0;
   jxc := 0;
   j := 0;
   copy := 0;
   II := IC;
   COPY := S[II];
   IXC := IX[II];
   JXC := IXC;
13:
   J := II + II;
   If (J - NN) < 0 then
    goto 111;
   If (J - NN) = 0 then
    goto 112;
   If (J - NN) > 0 then
    goto 19;
111:
   If S[J + 1] > S[J] then
    J := J + 1;
112:
   If S[J] <= COPY then
    goto 19;
   S[II] := S[J];
   IX[II] := IX[J];
   JX[IX[J]] := II;
   II := J;
   goto 13;
19:
   S[II] := COPY;
   IX[II] := IXC;
   JX[JXC] := II;
  end;
  procedure NETGEN (var X, Y: VECT100; var NPT, SCALE, BKPTR, newscale: COUNTER);
   label
    333, 222, 223;
   var
    i, I, j, NODES, backhaul, NPTB: integer;
    r, w: real;
    yy: array[1..200] of real;
    TEST: INTEGER;
    concentration: integer;
  begin
   concentration := 0;
   conc := 0;
   i := 0;
   I := 0;
   j := 0;
   nodes := 0;
   backhaul := 0;
   nptb := 0;
   for i := 1 to 200 do
    begin
     yy[i] := 0;
    end;
   i := 0;
   If newscale > 0 then
    goto 333;
   newscale := 1;
   writeln('ENTER NO OF NODES(<=62)');
   readln(nodes);
   writeln('ENTER SCALE OF GRAPH');
{READLN(scale);}
   scale := 250;
   writeln('SCALE', SCALE);
   writeln('ENTER PERCENTAGE OF BACKHAUL LOCATIONS(18%)');
{readln(backhaul);}
```

```
    BACKHAUL := 20;
    WRITELN('BACKHAUL', BACKHAUL);
    NPT := NODES;
    NPTB := ROUND(NODES * (backhaul / 100));
    BKPTR := NPT - NPTB + 1;
    TEST := 0;
{WRITELN('USE TEST NODES?(1-YES,0-NO)');}
{READLN(TEST);}
    while TEST = 1 do
     begin
      X[1] := 30;
      Y[1] := 40;
      X[51] := 8;
      Y[51] := 52;
      X[2] := 5;
      Y[2] := 64;
      X[3] := 17;
      Y[3] := 63;
      X[4] := 27;
      Y[4] := 68;
      X[16] := 31;
      Y[16] := 62;
      X[5] := 37;
      Y[5] := 69;
      X[6] := 43;
      Y[6] := 67;
      X[7] := 52;
      Y[7] := 64;
      X[9] := 62;
      Y[9] := 63;
      X[8] := 63;
      Y[8] := 69;
      X[14] := 16;
      Y[14] := 57;
      X[15] := 25;
      Y[15] := 55;
      X[11] := 42;
      Y[11] := 57;
      X[10] := 57;
      Y[10] := 58;
      X[12] := 37;
      Y[12] := 52;
      X[13] := 21;
      Y[13] := 47;
      X[17] := 30;
      Y[17] := 48;
      X[18] := 38;
      Y[18] := 46;
      X[20] := 49;
      Y[20] := 49;
      X[30] := 58;
      Y[30] := 48;
      X[46] := 7;
      Y[46] := 38;
      X[47] := 12;
      Y[47] := 42;
      X[25] := 32;
      Y[25] := 39;
      X[19] := 42;
      Y[19] := 41;
      X[21] := 52;
      Y[21] := 41;
      X[31] := 62;
      Y[31] := 42;
      X[50] := 17;
      Y[50] := 33;
      X[48] := 25;
      Y[48] := 32;
      X[24] := 31;
```

```
        Y[24] := 32;
        X[23] := 40;
        Y[23] := 30;
        X[22] := 45;
        Y[22] := 35;
        X[29] := 56;
        Y[29] := 37;
        X[28] := 52;
        Y[28] := 33;
        X[32] := 61;
        Y[32] := 33;
        X[45] := 5;
        Y[45] := 25;
        X[49] := 20;
        Y[49] := 26;
        X[26] := 48;
        Y[26] := 28;
        X[33] := 58;
        Y[33] := 27;
        X[38] := 27;
        Y[38] := 23;
        X[39] := 32;
        Y[39] := 22;
        X[27] := 51;
        Y[27] := 21;
        X[44] := 10;
        Y[44] := 17;
        X[43] := 5;
        Y[43] := 6;
        X[42] := 13;
        Y[42] := 13;
        X[41] := 21;
        Y[41] := 10;
        X[40] := 30;
        Y[40] := 15;
        X[37] := 36;
        Y[37] := 16;
        X[36] := 39;
        Y[36] := 10;
        X[35] := 46;
        Y[35] := 10;
        X[34] := 59;
        Y[34] := 15;
        X[52] := 15;
        Y[52] := 51;
        X[53] := 52;
        Y[53] := 55;
        X[54] := 12;
        Y[54] := 25;
        X[55] := 22;
        Y[55] := 17;
        X[56] := 46;
        Y[56] := 24;
        X[57] := 31;
        Y[57] := 11;
        X[58] := 56;
        Y[58] := 5;
        X[59] := 14;
        Y[59] := 27;
        X[60] := 7;
        Y[60] := 44;
        X[61] := 27;
        Y[61] := 50;
        X[62] := 66;
        Y[62] := 60;
        goto 333;
      end;
    writeln('ENTER DELIVERY CONCENTRATION(60%)');
{readln(concentration);}
```

```
    CONCENTRATION := 75;
    WRITELN('CONCENTRATION', CONCENTRATION);
    CONC := (concentration / 100);
{   WRITELN('BKPTR', BKPTR, 'CONC', CONC);}
    repeat
     i := i + 1;
     r := random;
     w := ABS(r / 32767);
     if i >= (BKPTR * 2) then
      goto 222;
     yy[i] := SCALE * (1 - CONC) + (w * (2 * (SCALE * CONC) - SCALE));
     goto 223;
222:
     yy[i] := (w * SCALE);
223:
{WRITELN('r', i, 'YY[i]', ROUND(YY[i]));}
    until i = nodes * 2;
    DRAWLINE(0, 100, 2, 100);
    DRAWLINE(100, 0, 100, 2);
    DRAWLINE(0, 200, 2, 200);
    DRAWLINE(200, 0, 200, 2);
    DRAWLINE(0, 300, 2, 300);
    DRAWLINE(300, 0, 300, 2);
{BEGIN TO COORD TO ARRAY}
    i := 1;
    j := 1;
    repeat
     X[i] := yy[j];
     i := i + 1;
     j := j + 2
    until j > NODES * 2;
    i := 1;
    j := 2;
    repeat
     Y[i] := yy[j];
     i := i + 1;
     j := j + 2
    until j > NODES * 2;
333:
    newscale := 2;
    i := 1;
    P1 := ROUND(X[i] * newscale);
    P2 := ROUND(Y[i] * newscale);
{writeln('node', i, P1, P2);}
    PaintCircle(P1, P2, 5);
    Moveto(P1 - 45, P2);
{WriteDraw(i);}
    for i := 2 to (BKPTR - 1) do
     begin
      P1 := ROUND(X[i] * newscale);
      P2 := ROUND(Y[i] * newscale);
{writeln('node', i, P1, P2);}
      drawline(P1 - 3, P2 - 3, P1 + 3, P2 - 3);
      drawline(P1 - 3, P2 + 3, P1 + 3, P2 + 3);
      drawline(P1 - 3, P2 - 3, P1 - 3, P2 + 3);
      drawline(P1 + 3, P2 - 3, P1 + 3, P2 + 3);
      Moveto(P1 - 15, P2);
{WriteDraw(i);}
    end;
   NODES := NPT;
   for i := BKPTR to NODES do
    begin
     P1 := ROUND(X[i] * newscale);
     P2 := ROUND(Y[i] * newscale);
{writeln('node', i, P1, P2);}
     drawline(P1, P2 - 4, P1 - 4, P2 + 4);
     drawline(P1, P2 - 4, P1 + 4, P2 + 4);
     drawline(P1 - 4, P2 + 4, P1 + 4, P2 + 4);
     Moveto(P1 - 15, P2);
```

```
{WriteDraw(I);}
      end;
    end;
   begin
    I := 0;
    J := 0;
    T0 := 0;
    T1 := 0;
    Elaps := 0;
    Hrs := 0;
    Mins := 0;
    Secs := 0;
    NN := 0;
    IC := 0;
    NPT := 0;
    SCALE := 0;
    BKPTR := 0;
    Iours := 0;
    XXX := 0;
    YYY := 0;
    for I := 1 to 1000 do
     begin
      s[I] := 0;
     end;
    for I := 1 to 1000 do
     begin
      ix[I] := 0;
     end;
    for I := 1 to 1000 do
     begin
      jx[I] := 0;
     end;
    NARC := 0;
    JO := 0;
    K := 0;
    P := 0;
    LIN := 0;
    LJN := 0;
    IPT1 := 0;
    IPT2 := 0;
    ICAP := 0;
    MCAP := 0;
    LM := 0;
    L := 0;
    M := 0;
    ITK := 0;
    HIT := 0;
    KT := 0;
    IT := 0;
    P1 := 0;
    P2 := 0;
    P3 := 0;
    P4 := 0;
   TDIST := 0;
   SUM := 0;
   for I := 1 to 100 do
    begin
     d[I] := 0;
    end;
  for I := 1 to 100 do
   begin
    x[I] := 0;
   end;
  for I := 1 to 100 do
   begin
    y[I] := 0;
   end;
  for I := 1 to 1000 do
   begin
```

```
      for j := 1 to 10 do
       begin
        id[i, jj] := 0;
       end;
    end;
  for i := 1 to 100 do
   begin
    for j := 1 to 100 do
     begin
      adja[i, j] := 0;
     end;
   end;
  for i := 1 to 5 do
   begin
    tra[i] := 0;
   end;
  for i := 1 to 5 do
   begin
    trcap[i] := 0;
   end;
  for i := 1 to 5 do
   begin
    tru[i] := 0;
   end;
  for i := 1 to 100 do
   begin
    for j := 1 to 5 do
     begin
      ptour[i, j] := 0;
     end;
   end;
  for i := 1 to 100 do
   begin
    point[i] := 0;
   end;
  for i := 1 to 100 do
   begin
    q[i] := 0;
   end;
  for i := 1 to 100 do
   begin
    bx[i] := 0;
   end;
  for i := 1 to 100 do
   begin
    by[i] := 0;
   end;
  I := 0;
  WIDTH := 0;
  HEIGHT := 0;
  DIVISOR := 0;
  II := 0;
  NT := 0;
  J := 0;
  IP1 := 0;
  IA := 0;
  IB := 0;
  ISUM := 0;
  DISTAN := 0;
  MAXDROP := 0;
  NVT := 0;
  MAXTIME := 0;
  TCAP := 0;
  TOTTIME := 0;
  MILMIN := 0;
  SUM1 := 0;
  SPEED := 0;
  newscale := 0;
  SUM1 := 0;
```

```
    NETGEN(X, Y, NPT, SCALE, BKPTR, newscale);
    NPT1 := NPT;
    NPT := BKPTR - 1;
 {WRITELN('ENTER NO OF VEHICLE TYPES(DEFAULT=1)');}
 {READLN(NVT);}
    WRITELN('BASECASE CW PARAMETERS');
    NVT := 1;
    for I := 1 to NVT do
     begin
      WRITELN('FOR TRUCK', I, 'SPECIFY THE FOLLOWING PARAMETERS');
      WRITELN('ENTER TRUCK CAPACITY(DEFAULT=20)');
      READLN(TRCAP[I]);
{WRITELN('ENTER TRUCK AVAILABILITY(DEFAULT=9999)');}
{READLN(TRA[I]);}
      TRA[1] := 1000;
     end;
   WRITELN('ENTER NODE DEMAND REQUIREMENTS(DEFAULT=3)');
{READLN(Q[1]);}
   Q[1] := 3;
   WRITELN('NODE DEMAND', Q[1]);
   if Q[1] = 999 then
    goto 776;
   for I := 1 to NPT1 do
    begin
     if I <= NPT then
      Q[I] := Q[1]
     else
      Q[I] := -Q[1];
    end;
   goto 6767;
776:
   Q[1] := 0;
   Q[51] := 10;
   Q[2] := 11;
   Q[3] := 19;
   Q[4] := 7;
   Q[5] := 11;
   Q[6] := 14;
   Q[7] := 16;
   Q[8] := 6;
   Q[9] := 17;
   Q[10] := 28;
   Q[11] := 8;
   Q[12] := 7;
   Q[13] := 15;
   Q[14] := 16;
   Q[15] := 17;
   Q[16] := 23;
   Q[17] := 15;
   Q[18] := 12;
   Q[19] := 19;
   Q[20] := 30;
   Q[21] := 15;
   Q[22] := 15;
   Q[23] := 21;
   Q[24] := 29;
   Q[25] := 5;
   Q[26] := 18;
   Q[27] := 6;
   Q[28] := 11;
   Q[29] := 10;
   Q[30] := 6;
   Q[31] := 8;
   Q[32] := 26;
   Q[33] := 19;
   Q[34] := 14;
   Q[35] := 23;
   Q[36] := 10;
   Q[37] := 10;
```

```
    Q[38] := 3;
    Q[39] := 9;
    Q[40] := 16;
    Q[41] := 13;
    Q[42] := 9;
    Q[43] := 7;
    Q[44] := 27;
    Q[45] := 23;
    Q[46] := 28;
    Q[47] := 21;
    Q[48] := 25;
    Q[49] := 9;
    Q[50] := 41;
6767:
{   WRITELN('ENTER DIVISOR,C-W ACCURACY PARM,(DEFAULT=1)');}
{   READLN(DIVISOR);}
{WRITELN('ENTER MAX NUMBER OF DROPS PER ROUTE(DEFAULT=50)');}
{READLN(MAXDROP);}
{WRITELN('ENTER MAX NUMBER OF HOURS PER ROUTE(DEFAULT=8 )');}
{READLN(MAXTIME);}
{WRITELN('ENTER AVERAGE SPEED OF VEHICLES(DEFAULT= 60 MPH)');}
{READLN(SPEED);}
  DIVISOR := 4;
  MAXDROP := 100;
  MAXTIME := 8;
  SPEED := 60;
{   WRITELN('DIVISOR', DIVISOR, 'MAXDROP', MAXDROP, 'MAXTIME', MAXTIME, 'SPEED', SPEED);}
  MILMIN := SPEED / 60;
{WRITELN('SPEED', SPEED, 'MILMIN', MILMIN);}
{GetTime(DateTime);}
  Mins := DateTime.Minute * 60;
  Secs := DateTime.Second;
  T0 := Mins + Secs;
  WIDTH := SCALE;
  HEIGHT := SCALE;
1010:
  WIDTH := ROUND(WIDTH / DIVISOR);
  HEIGHT := ROUND(HEIGHT / DIVISOR);
  for I := 2 to NPT do
   begin
    POINT[I] := 0;
    BX[I] := ROUND(X[I] / WIDTH + 1);
    BY[I] := ROUND(Y[I] / HEIGHT + 1);
   end;
  NARC := 0;
  NT := NPT - 1;
  I := 1;
  for J := 2 to NPT do
   begin
    NARC := NARC + 1;
    while NARC >= 1000 do
     begin
      DIVISOR := DIVISOR + 2;
      goto 1010
     end;
    ID[NARC, 1] := I;
    ID[NARC, 2] := J;
    DISTAN := SQR(X[I] - X[J]) + SQR(Y[I] - Y[J]);
    D[J] := SQRT(DISTAN);
   end;
{WRITELN('NARC', NARC);}
  M := 0;
  for I := 2 to NT do
   begin
    IP1 := I + 1;
    for J := IP1 to NPT do
     begin
      IA := ROUND(ABS(BX[I] - BX[J]));
      IB := ROUND(ABS(BY[I] - BY[J]));
```

```
        If (IA > 1) or (IB > 1) then
         goto 102;
        NARC := NARC + 1;
{If NARC >= 700 then}
{WRITELN('NARC', NARC);}
        while NARC >= 1000 do
         begin
          DIVISOR := DIVISOR + 2;
          goto 1010
         end;
        II := II + 1;
        POINT[I] := POINT[I] + 1;
        POINT[J] := POINT[J] + 1;
        If (POINT[I] > 100) or (POINT[J] > 100) then
         goto 998;
        ADJA[I, POINT[I]] := II;
        ADJA[J, POINT[J]] := II;
        ID[NARC, 1] := I;
        ID[NARC, 2] := J;
        DISTAN := SQR(X[I] - X[J]) + SQR(Y[I] - Y[J]);
        S[II] := D[I] + D[J] - SQRT(DISTAN);
102:
      end;
    end;
{WRITELN('NARC', NARC);}
{WRITELN('MAX S[]', II);}
  SUM := 0;
  SUM1 := 0;
  for J := 2 to NPT do
   begin
    SUM := SUM + D[J];
    SUM1 := SUM1 + ROUND(D[J] / MILMIN);
   end;
  TDIST := 2 * SUM;
  WRITELN('INITIAL CW SOLUTION', ROUND(TDIST));
  TOTTIME := 2 * SUM1;
{WRITELN('TOTTIME', TOTTIME);}
  NN := NARC - NPT + 1;
  TRU[1] := NPT - 1;
  If NVT = 1 then
   goto 17;
  for I := 2 to NVT do
   begin
    TRU[I] := 0;
   end;
17:
  for I := 2 to NPT do
   begin
    PTOUR[I, 1] := I;
    PTOUR[I, 2] := 1;
    PTOUR[I, 3] := Q[I];
    PTOUR[I, 4] := 1;
    PTOUR[I, 5] := 2 * ROUND(D[I] / MILMIN);
   end;
  for I := 1 to NT do
   begin
    ID[I, 3] := 2;
   end;
  for I := NPT to NARC do
   begin
    ID[I, 3] := 0;
   end;
91:
  for I := 1 to NN do
   begin
    JX[I] := I;
    IX[I] := I;
   end;
  STHEAP(NN, S, IX, JX);
```

```
 90:
    JO := 0;
    K := 1;
    If S[K] <= 0 then
     goto 16;
    P := IX[K] + NPT - 1;
    LIN := ID[P, 1];
    LJN := ID[P, 2];
    IPT1 := PTOUR[LIN, 1];
    IPT2 := PTOUR[LJN, 1];
    If (ID[LIN - 1, 3] = 0) or (ID[LJN - 1, 3] = 0) then
     goto 12;
    If IPT1 = IPT2 then
     goto 12;
    ICAP := PTOUR[LIN, 3] + PTOUR[LJN, 3];
    for LM := 1 to NVT do
     begin
      If (TRCAP[LM] >= ICAP) and (TRU[LM] < TRA[LM]) then
       goto 10;
     end;
    goto 12;
 10:
    MCAP := PTOUR[LIN, 4] + PTOUR[LJN, 4];
    If MCAP > MAXDROP then
     goto 12;
    TCAP := PTOUR[LIN, 5] + PTOUR[LJN, 5] - ROUND(S[K] / MILMIN);
    If TCAP > MAXTIME * 60 then
     goto 12;
    TRU[LM] := TRU[LM] + 1;
    TRU[PTOUR[LIN, 2]] := TRU[PTOUR[LIN, 2]] - 1;
    TRU[PTOUR[LJN, 2]] := TRU[PTOUR[LJN, 2]] - 1;
    DISTAN := D[LIN] + D[LJN] - S[K];
    TDIST := TDIST - (D[LIN] + D[LJN] - DISTAN);
{WRITELN('TDIST', ROUND(TDIST));}
    TOTTIME := TOTTIME - ROUND(S[K] / MILMIN);
{WRITELN('TOTTIME', TOTTIME, 'MILMIN', MILMIN);}
    for L := 2 to NPT do
     begin
      If (PTOUR[L, 1] = IPT1) or (PTOUR[L, 1] = IPT2) then
       goto 64;
     end;
 64:
    for M := 2 to NPT do
     begin
      If (PTOUR[M, 1] <> IPT1) and (PTOUR[M, 1] <> IPT2) then
       goto 105;
      PTOUR[M, 1] := PTOUR[L, 1];
      PTOUR[M, 2] := LM;
      PTOUR[M, 3] := ICAP;
      PTOUR[M, 4] := MCAP;
      PTOUR[M, 5] := TCAP;
105:
     end;
   ID[P, 3] := 1;
   ID[LIN - 1, 3] := ID[LIN - 1, 3] - 1;
   ID[LJN - 1, 3] := ID[LJN - 1, 3] - 1;
   If ID[LJN - 1, 3] <> 0 then
    goto 11;
   IB := POINT[LJN];
   JO := 1;
   for I := 1 to IB do
    begin
     IC := JX[ADJA[LJN, I]];
     If S[IC] <= 0 then
      goto 97;
     S[IC] := 0;
     OTHEAP(NN, S, IX, JX, IC);
97:
    end;
```

```
  11:
    If ID[LIN - 1, 3] <> 0 then
     goto 12;
    IB := POINT[LIN];
    for I := 1 to IB do
     begin
      IC := JX[ADJA[LIN, I]];
      If S[IC] <= 0 then
       goto 98;
      S[IC] := 0;
      OTHEAP(NN, S, IX, JX, IC);
  98:
     end;
    goto 90;
  12:
    If JO = 1 then
     goto 90;
    S[K] := 0;
    OTHEAP(NN, S, IX, JX, K);
    goto 90;
  16:
  {GetTime(DateTime);}
    Mins := DateTime.Minute * 60;
    Secs := DateTime.Second;
    T1 := Mins + Secs;
    If T1 < T0 then
     T1 := T1 + 3600;
    Elaps := T1 - T0;
    I := 1;
    NARC2 := NARC;
    for J := BKPTR to NPT1 do
     begin
      NARC := NARC + 1;
      ID[NARC, 1] := I;
      ID[NARC, 2] := J;
      DISTAN := SQR(X[I] - X[J]) + SQR(Y[I] - Y[J]);
      D[J] := SQRT(DISTAN);
  {WRITELN('I ', I, ' J ', J, ' D[J] ', ROUND(D[J]));}
      ID[NARC, 3] := 2;
     end;
    WRITELN('NARC', NARC);
    SUM := 0;
    SUM1 := 0;
  {MILMIN := 1;}
  {WRITELN('TOTTIME', TOTTIME, 'MILMIN', ROUND(MILMIN));}
    for J := BKPTR to NPT1 do
     begin
      SUM := SUM + D[J];
      SUM1 := SUM1 + ROUND(D[J] / MILMIN);
      ISUM := ISUM + 1;
      PTOUR[J, 1] := J;
      PTOUR[J, 2] := 1;
      PTOUR[J, 3] := Q[J];
      PTOUR[J, 4] := 1;
      PTOUR[J, 5] := ROUND(D[I] / MILMIN);
     end;
    WRITELN('C-W DIST', ROUND(TDIST), 'RADIAL DIST', ROUND(SUM));
    TDIST := TDIST + SUM;
    TOTTIME := TOTTIME + SUM1;
  {WRITELN('TOTTIME', TOTTIME);}
    ITK := 1;
    for IT := 1 to NPT1 do
     begin
      HIT := 0;
      for KT := 2 to NPT1 do
       begin
        while PTOUR[KT, 1] = IT do
         begin
          PTOUR[KT, 1] := ITK;
```

```
          HIT := 1;
        end;
      end;
     if HIT = 1 then
      ITK := ITK + 1;
    end;
{WRITELN('ELAPSED TIME IS', Elaps, ' SECONDS');}
  WRITELN('TOTAL DISTANCE IS', ROUND(TDIST), ' UNITS');
{WRITELN('TOTAL TRAVEL TIME IS', ROUND(TOTTIME), ' MINUTES');}
  for I := 1 to NVT do
   begin
    ISUM := ISUM + TRU[I];
   end;
  WRITELN('THERE ARE', ISUM, ' TOURS IN ALL');
  for I := 1 to 50 do
   begin
    RROUTE[I, 1] := 0;
    RROUTE[I, 2] := 0;
    RROUTE[I, 3] := 0;
   end;
  NTOURS := ISUM;
{WRITELN('      ', 'I      ', 'J    ', 'T(I,J)  ', 'TRUCKTYPE', '   TOUR', ' DELIVERY ', '#DROPS');}
  for J := 1 to NARC do
   begin
    if ID[J, 3] = 0 then
     goto 35;
{WRITELN(ID[J, 1], ID[J, 2], ID[J, 3], PTOUR[ID[J, 2], 2], '  ', PTOUR[ID[J, 2], 1], PTOUR[ID[J, 2], 3], PTOUR[ID[J, 2], 4]);}
    P1 := ROUND(X[ID[J, 1]]);
    P2 := ROUND(Y[ID[J, 1]]);
    P3 := ROUND(X[ID[J, 2]]);
    P4 := ROUND(Y[ID[J, 2]]);
    RROUTE[PTOUR[ID[J, 2], 1], 1] := RROUTE[PTOUR[ID[J, 2], 1], 1] + ROUND(SQRT(SQR(X[ID[J, 1]] - X[ID[J, 2]]) + SQR(Y[ID[J, 1]] -
      Y[ID[J, 2]])));
    RROUTE[PTOUR[ID[J, 2], 1], 2] := PTOUR[ID[J, 2], 4];
{If J <= NARC2 then}
{DRAWLINE(P1, P2, P3, P4);}
35:
   end;
  CUM := 0;
  for I := 1 to ISUM do
   begin
    CUM := CUM + RROUTE[I, 1];
{WRITELN('ROUTE', I, ' DISTANCE ', RROUTE[I, 1], 'NO OF STOP', RROUTE[I, 2], 'CUM', CUM);}
   end;
  TROUTE := 0;
  for II := 1 to ISUM do
   begin
    if (ROUND(RROUTE[II, 1]) = 0) then
     goto 7478;
    if (route[II, 2] < 2) then
     goto 7478;
{WRITELN(' NO OF STOPS', RROUTE[II, 2]);}
    TROUTE := II;
    RROUTE[II, 2] := 0;
    RX[1] := X[1];
    RY[1] := Y[1];
{WRITELN('RX', ROUND(X[1]), 'RY', ROUND(Y[1]));}
    COUNTT := 0;
    for J := 1 to NARC do
     begin
      if ID[J, 3] = 0 then
       goto 355;
      if II <> PTOUR[ID[J, 2], 1] then
       goto 355;
{WRITELN(' I ', ID[J, 1], ' J ', ID[J, 2], 'STATUS ', ID[J, 3], ' ROUTE ', PTOUR[ID[J, 2], 1], 'LINK DIST ', ROUND(SQRT(SQR(X[ID[J, 1]] -
      X[ID[J, 2]]) + SQR(Y[ID[J, 1]] - Y[ID[J, 2]]))));}
355:
     end;
    for I := 1 to NPT1 do
```

```
      begin
       if TROUTE = PTOUR[I, 1] then
        begin
         RROUTE[II, 2] := RROUTE[II, 2] + 1;
         COUNTT := COUNTT + 1;
{WRITELN('COUNT', COUNTT, 'NODE', I, 'PTOUR[I,1]', PTOUR[I, 1], 'RX', ROUND(X[I]), 'RY', ROUND(Y[I]));}
         RX[COUNTT + 1] := X[I];
         RY[COUNTT + 1] := Y[I];
        end;
      end;
     WRITELN(' NO OF STOPS', RROUTE[II, 2], 'ROUTELNGTH', RROUTE[II, 1]);
     NPTT := RROUTE[II, 2] + 1;
     ROUTELENGTH := 0;
{POSTARC(RX, RY, NPTT, ROUTELENGTH);}
7478:
    end;


{   WRITELN('  TRUCK TYPE', '  #AVAIL  ', '   CAP    ', '#USED IN ROUTING');}
   for M := 1 to NVT do
    begin
{   WRITELN(M, '    ', TRA[M], '   ', TRCAP[M], '  ', TRU[M] + (NPT1 - BKPTR + 1));}
    end;
   WRITELN('DIVISOR', DIVISOR);
   WRITELN('MAX NUMBER OF DROPS', MAXDROP);
   WRITELN('MAX NUMBER OF HOURS PER ROUTE', MAXTIME);
   WRITELN('AVERAGE SPEED OF VEHICLE IS', ROUND(SPEED), ' MPH');
{WRITELN('ENTER NEW GRAPH SCALE FOR SPECIFIC ROUTES(enter 0 for none, or integer)');}
{readln(newscale);}
   newscale := 0;
   if newscale = 0 then
    goto 301;
38:
   writeln('ENTER ROUTE NUMBER DESIRED(enter 0 for none,enter 100 for all)');
   readln(tours);
   if tours = 0 then
    goto 301;
   if XXX = 0 then
    goto 39;
   writeln('ENTER 0 IF YOU WOULD LIKE PREVIOUS TOUR ERASED');
   writeln('ENTER 1 OTHERWISE');
   readln(YYY);
   if YYY = 1 then
    goto 40;
39:
   PaintCircle(0, 0, 500);
   InvertCircle(0, 0, 500);
   XXX := 1;
   NETGEN(X, Y, NPT1, SCALE, BKPTR, newscale);
   for J := 1 to NARC do
    begin
     if ID[J, 3] = 0 then
      goto 777;
{WRITELN(ID[J, 1], ID[J, 2], ID[J, 3], PTOUR[ID[J, 2], 2], '  ', PTOUR[ID[J, 2], 1], PTOUR[ID[J, 2], 3], PTOUR[ID[J, 2], 4]);}
     P1 := ROUND(X[ID[J, 1]]) * (newscale);
     P2 := ROUND(Y[ID[J, 1]]) * newscale;
     P3 := ROUND(X[ID[J, 2]]) * (newscale);
     P4 := ROUND(Y[ID[J, 2]]) * newscale;
     DRAWLINE(P1, P2, P3, P4);
777:
    end;
   if tours = 100 then
    goto 301;
40:
   for J := 1 to NARC do
    begin
     if (ID[J, 3] = 0) then
      goto 36;
     if (PTOUR[ID[J, 2], 1] <> tours) then
```

```
      goto 36;
    P1 := ROUND(X[ID[J, 1]]) * (newscale + 1);
    P2 := ROUND(Y[ID[J, 1]]) * newscale;
    P3 := ROUND(X[ID[J, 2]]) * (newscale + 1);
    P4 := ROUND(Y[ID[J, 2]]) * newscale;
    DRAWLINE(P1, P2, P3, P4);
36:
   end;
  goto 38;
998:
  WRITELN('POINT(I) FOR SOME I EXCEEDS 50.');
301:
  WRITELN('THE END OF BASECASE');
 end;
end.
```

```
unit CWUNIT;
interface
 uses
  ARCOPT;
 type
  matrix9 = array[1..50, 1..8] of integer;
 procedure CWVRP (var REJARR: MATRIX9; var REJECOUNT: INTEGER; var SAVINGS: INTEGER; var FLAG: INTEGER; var THRESHOLD:
      INTEGER);
implementation
 procedure CWVRP (var REJARR: MATRIX9; var REJECOUNT: INTEGER; var SAVINGS: INTEGER; var FLAG: INTEGER; var THRESHOLD:
      INTEGER);
  label
   7478, 2727, 2728, 1201, 1220, 3575, 1249, 3666, 1200, 1244, 7799, 22, 23, 25, 26, 27, 77, 777, 778, 124, 123, 122, 120,
     1010, 102, 998, 17, 16, 10, 64, 105, 11, 97, 12, 301, 98, 90, 35, 91, 444, 355, 4444, 4445, 3555;
  type
   MATRIX1 = array[1..500, 1..10] of INTEGER;
   MATRIX2 = array[1..100, 1..5] of INTEGER;
   COUNTER = INTEGER;
   VECTOR = array[1..500] of real;
   IVECTOR = array[1..500] of INTEGER;
   VECT100 = array[1..100] of REAL;
   IVECT100 = array[1..100] of INTEGER;
   MATRIX3 = array[1..100, 1..8] of INTEGER;
   MATRIX4 = array[1..100, 1..3] of INTEGER;
   MATRIX5 = array[1..5] of INTEGER;
   COUNTER2 = INTEGER;
   REAL1 = REAL;
{matrix9 = array[1..50, 1..3] of integer;}
   DateTimeRec = record
     Year, Month, Day, Hour, Minute, Second, DayOfWeek: integer
    end;
  var
   ARCSAVINGS: INTEGER;
   RX, RY: ARRN11;
   DateTime: DateTimeRec;
   TOTTIME, MILMIN, SUM1, SPEED, SUPPLIER2: REAL1;
   T0, T1, T2, Elaps2, Elaps, Hrs, Mins, Secs, SUPPLIER, YY, III, INFLATION, TROUTE: integer;
   NN, IC, NPT, NPT1, SCALE, BKPTR, tours, XXX, YYY, newscale, NARC, NTOURS: COUNTER;
   S: VECTOR;
   IX, JX: IVECTOR;
   JO, K, P, LIN, LJN, IPT1, IPT2, ICAP, MCAP, LM, L, M, ITK, HIT, KT, IT, P1, P2, P3, P4: INTEGER;
   TDIST, SUM, DISTAN, DISTAN2, SUMDIST: REAL;
   DEDROUTE: array[1..50] of INTEGER;
   D: array[1..50] of real;
   X, Y: VECT100;
   ID: MATRIX1;
   ADJA: array[1..50, 1..50] of integer;
   RROUTE: array[1..20, 1..3] of INTEGER;
   TRA, TRCAP, TRU: MATRIX5;
   PTOUR: MATRIX2;
   POINT: array[1..100] of INTEGER;
   Q: IVECT100;
   BX, BY: array[1..100] of INTEGER;
   I, WIDTH, HEIGHT, DIVISOR, II, NT, J, IP1, IA, IB, ISUM, MIN, MINPOINT, MINTHRE: INTEGER;
   NVT, MAXDROP, MAXTIME, TCAP: COUNTER;
{C1: array[1..62, 1..62] of INTEGER;}
   CC1: array[1..124, 1..5] of INTEGER;
   CMIN1, CMIN2, CMIN3, CMIN4, CMIN5: INTEGER;
   NODES, rejected, NARC2, NPT2: INTEGER;
   INFLATION1: REAL;
   NONO, IPOINT, IPOINT2, RADIAL, JJ: INTEGER;
   TEMP1, TEMP2, TEMP3, LL, LLL, MMM: INTEGER;
   ROUTELENGTH, NPTT, COUNTT: COUNTER3;
  function C1 (X: VECT100; Y: VECT100; I: INTEGER; J: INTEGER): INTEGER;
   var
    CCC1: INTEGER;
  begin
   CCC1 := ROUND(SQRT(SQR(X[I] - X[J]) + SQR(Y[I] - Y[J])));
```

```
{WRITELN('CCC1 ', CCC1);}
    C1 := CCC1;
  end;
  procedure GETTIME (var DATETIME: DATETIMEREC);
  EXTERNAL;
  procedure STHEAP (var NN: COUNTER; var S: VECTOR; var IX, JX: IVECTOR);
   label
    1, 50, 60, 2, 5, 7, 9;
   var
    NH, II, I, IXC, JXC, J: INTEGER;
    DT, COPY: REAL;
  begin
   nh := 0;
   ii := 0;
   i := 0;
   ixc := 0;
   jxc := 0;
   j := 0;
   dt := 0;
   copy := 0;
   if (NN - 2) < 0 then
    goto 50;
   if (NN - 2) = 0 then
    goto 60;
   if (NN - 2) > 0 then
    goto 1;
1:
   NH := ROUND(NN / 2);
   for I := 1 to NH do
    begin
     II := NH + 1 - I;
     COPY := S[II];
     IXC := IX[II];
     JXC := IXC;
2:
     J := II + II;
     if (J - NN) < 0 then
      goto 5;
     if (J - NN) = 0 then
      goto 7;
     if (J - NN) > 0 then
      goto 9;
5:
     if S[J + 1] > S[J] then
      J := J + 1;
7:
     if S[J] <= COPY then
      goto 9;
     S[II] := S[J];
     IX[II] := IX[J];
     JX[IX[J]] := II;
     II := J;
     goto 2;
9:
     S[II] := COPY;
     IX[II] := IXC;
     JX[JXC] := II;
    end;
   goto 50;
60:
   if S[1] >= S[2] then
    goto 50;
   DT := S[1];
   S[1] := S[NN];
   S[2] := DT;
   IX[1] := 2;
   JX[1] := 2;
   IX[2] := 1;
   JX[2] := 1;
```

```
50:
  end;
  procedure OTHEAP (var NN: COUNTER; var S: VECTOR; var IX: IVECTOR; var JX: IVECTOR; IC: COUNTER);

   label
    13, 111, 112, 19;
   var
    II, IXC, JXC, J: INTEGER;
    COPY: REAL;
  begin
   ii := 0;
   ixc := 0;
   jxc := 0;
   j := 0;
   copy := 0;
   II := IC;
   COPY := S[II];
   IXC := IX[II];
   JXC := IXC;
13:
   J := II + II;
   if (J - NN) < 0 then
    goto 111;
   if (J - NN) = 0 then
    goto 112;
   if (J - NN) > 0 then
    goto 19;
111:
   if S[J + 1] > S[J] then
    J := J + 1;
112:
   if S[J] <= COPY then
    goto 19;
   S[II] := S[J];
   IX[II] := IX[J];
   JX[IX[J]] := II;
   II := J;
   goto 13;
19:
   S[II] := COPY;
   IX[II] := IXC;
   JX[JXC] := II;
  end;
 begin
  if FLAG = 1 then
   goto 4444;
  THRESHOLD := 0;
  i := 0;
  j := 0;
  T0 := 0;
  T1 := 0;
  Elaps := 0;
  Hrs := 0;
  Mins := 0;
  Secs := 0;
  NN := 0;
  IC := 0;
  NPT := 0;
  SCALE := 0;
  BKPTR := 0;
  tours := 0;
  XXX := 0;
  YYY := 0;
  for i := 1 to 500 do
   begin
    s[i] := 0;
   end;
  for i := 1 to 500 do
   begin
```

```
    lx[I] := 0;
   end;
  for i := 1 to 500 do
   begin
    jx[i] := 0;
   end;
  NARC := 0;
  JO := 0;
  K := 0;
  P := 0;
  LIN := 0;
  LJN := 0;
  IPT1 := 0;
  IPT2 := 0;
  ICAP := 0;
  MCAP := 0;
  LM := 0;
  L := 0;
  M := 0;
  ITK := 0;
  HIT := 0;
  KT := 0;
  IT := 0;
  P1 := 0;
  P2 := 0;
  P3 := 0;
  P4 := 0;
  TDIST := 0;
  SUM := 0;
  for i := 1 to 50 do
   begin
    d[i] := 0;
   end;
  for i := 1 to 50 do
   begin
    x[i] := 0;
   end;
  for i := 1 to 50 do
   begin
    y[i] := 0;
   end;
  for i := 1 to 500 do
   begin
    for j := 1 to 10 do
     begin
      id[i, j] := 0;
     end;
   end;
  for i := 1 to 50 do
   begin
    for j := 1 to 50 do
     begin
      adja[i, j] := 0;
{C1[I, J] := 0;}
     end;
   end;
  for i := 1 to 5 do
   begin
    tra[i] := 0;
   end;
  for i := 1 to 5 do
   begin
    trcap[i] := 0;
   end;
  for i := 1 to 5 do
   begin
    tru[i] := 0;
   end;
  for i := 1 to 50 do
```

```
    begin
     for j := 1 to 5 do
      begin
       ptour[i, j] := 0;
      end;
    end;
   for i := 1 to 50 do
    begin
     point[i] := 0;
    end;
   for i := 1 to 50 do
    begin
     q[i] := 0;
    end;
   for i := 1 to 50 do
    begin
     bx[i] := 0;
    end;
   for i := 1 to 50 do
    begin
     by[i] := 0;
    end;
   i := 0;
   for i := 1 to 124 do
    begin
     for J := 1 to 5 do
      begin
       CC1[i, J] := 0;
      end;
    end;
   WIDTH := 0;
   HEIGHT := 0;
   DIVISOR := 0;
   II := 0;
   NT := 0;
   J := 0;
   IP1 := 0;
   IA := 0;
   IB := 0;
   ISUM := 0;
   DISTAN := 0;
   MAXDROP := 0;
   NVT := 0;
   MAXTIME := 0;
   TCAP := 0;
   TOTTIME := 0;
   MILMIN := 0;
   SUM1 := 0;
   SPEED := 0;
   newscale := 0;
   WRITELN('DEDICATED BACKHAUL ROUTES');
{WRITELN('REJ NODE  ', 'REJ X COORD ', ' REJ Y COORD ');}
   NPT := REJECOUNT + 1;
   NEWSCALE := 1;
   rejected := 0;
   for i := 1 to REJECOUNT + 1 do
    begin
{WRITELN(REJARR[i, 1], '   ', REJARR[i, 2], '   ', REJARR[i, 3]);}
     X[i] := REJARR[i, 2];
     Y[i] := REJARR[i, 3];
    end;
   SUMDIST := 0;
   DISTAN := 0;
   DISTAN2 := 0;
   for i := 2 to REJECOUNT + 1 do
    begin
     DISTAN := SQR(X[i] - X[1]) + SQR(Y[i] - Y[1]);
     DISTAN2 := SQRT(DISTAN);
{WRITELN('RADIAL DISTANCE FOR  ', REJARR[i, 1], ' IS ', ROUND(DISTAN2));}
```

```
      SUMDIST := SUMDIST + DISTAN2;
    end;
  NPT1 := NPT;
{WRITELN('ENTER NO OF VEHICLE TYPES(DEFAULT=1)');}
{READLN(NVT);}
  NVT := 1;
  for I := 1 to NVT do
   begin
    WRITELN('FOR TRUCK', I, 'SPECIFY THE FOLLOWING PARAMETERS');
    WRITELN('ENTER TRUCK CAPACITY(DEFAULT=20)');
    READLN(TRCAP[I]);
{WRITELN('ENTER TRUCK AVAILABILITY(DEFAULT=9999)');}
{READLN(TRA[I]);}
    TRA[1] := 1000;
   end;
  WRITELN('ENTER NODE DEMAND REQUIREMENTS(DEFAULT=3)');
{READLN(Q[1]);}
  Q[1] := 3;
  WRITELN('NODE DEMAND', Q[1]);
  for I := 1 to NPT1 do
   begin
    If I <= NPT then
     Q[I] := Q[1]
    else
     Q[I] := -Q[1];
   end;
{  WRITELN('ENTER DIVISOR,C-W ACCURACY PARM.(DEFAULT=1)');}
{  READLN(DIVISOR);}
{WRITELN('ENTER MAX NUMBER OF DROPS PER ROUTE(DEFAULT=50)');}
{READLN(MAXDROP);}
{WRITELN('ENTER MAX NUMBER OF HOURS PER ROUTE(DEFAULT=8 )');}
{READLN(MAXTIME);}
{WRITELN('ENTER AVERAGE SPEED OF VEHICLES(DEFAULT= 60 MPH)');}
{READLN(SPEED);}
  DIVISOR := 1;
  MAXDROP := 100;
  MAXTIME := 8;
  SPEED := 60;
  MILMIN := SPEED / 60;
  WIDTH := SCALE;
  HEIGHT := SCALE;
1010:
  WIDTH := ROUND(WIDTH / DIVISOR);
  HEIGHT := ROUND(HEIGHT / DIVISOR);
  for I := 2 to NPT do
   begin
    POINT[I] := 0;
    BX[I] := ROUND(X[I] / WIDTH + 1);
    BY[I] := ROUND(Y[I] / HEIGHT + 1);
   end;
  NARC := 0;
  NT := NPT - 1;
  I := 1;
  for J := 2 to NPT do
   begin
    NARC := NARC + 1;
    while NARC >= 1000 do
     begin
      DIVISOR := DIVISOR + 2;
      goto 1010
     end;
    ID[NARC, 1] := I;
    ID[NARC, 2] := J;
    DISTAN := SQR(X[I] - X[J]) + SQR(Y[I] - Y[J]);
    If REJARR[I, 2] <> REJARR[I, 5] then
     D[J] := -1
    else
     D[J] := SQRT(DISTAN);
   end;
```

```
  II := 0;
  for I := 2 to NT do
   begin
    IP1 := I + 1;
    for J := IP1 to NPT do
     begin
      IA := ROUND(ABS(BX[I] - BX[J]));
      IB := ROUND(ABS(BY[I] - BY[J]));
      if (IA > 1) or (IB > 1) then
       goto 102;
      NARC := NARC + 1;
      while NARC >= 1000 do
       begin
        DIVISOR := DIVISOR + 2;
        goto 1010
       end;
      II := II + 1;
      POINT[I] := POINT[I] + 1;
      POINT[J] := POINT[J] + 1;
      if (POINT[I] > 50) or (POINT[J] > 50) then
       goto 998;
      ADJA[I, POINT[I]] := II;
      ADJA[J, POINT[J]] := II;
      ID[NARC, 1] := I;
      ID[NARC, 2] := J;
      DISTAN := SQR(X[I] - X[J]) + SQR(Y[I] - Y[J]);
      S[II] := D[I] + D[J] - SQRT(DISTAN);
102:
     end;
   end;
  SUM := 0;
  SUM1 := 0;
  for J := 2 to NPT do
   begin
    SUM := SUM + D[J];
    SUM1 := SUM1 + ROUND(D[J] / MILMIN);
   end;
  TDIST := 2 * SUM;
  TOTTIME := 2 * SUM1;
  NN := NARC - NPT + 1;
  TRU[1] := NPT - 1;
  WRITELN('DEDICATED BACKHAUL NARCS ', NARC);
  if NVT = 1 then
   goto 17;
  for I := 2 to NVT do
   begin
    TRU[I] := 0;
   end;
17:
  for I := 2 to NPT do
   begin
    PTOUR[I, 1] := I;
    PTOUR[I, 2] := 1;
    PTOUR[I, 3] := Q[I];
    PTOUR[I, 4] := 1;
    PTOUR[I, 5] := 2 * ROUND(D[I] / MILMIN);
   end;
  for I := 1 to NT do
   begin
    ID[I, 3] := 2;
   end;
  for I := NPT to NARC do
   begin
    ID[I, 3] := 0;
   end;
91:
  for I := 1 to NN do
   begin
    JX[I] := I;
```

```
    IX[I] := I;
   end;
  STHEAP(NN, S, IX, JX);
90:
  JO := 0;
  K := 1;
  If S[K] <= 0 then
   goto 16;
  P := IX[K] + NPT - 1;
  LIN := ID[P, 1];
  LJN := ID[P, 2];
  IPT1 := PTOUR[LIN, 1];
  IPT2 := PTOUR[LJN, 1];
  If (ID[LIN - 1, 3] = 0) or (ID[LJN - 1, 3] = 0) then
   goto 12;
  If IPT1 = IPT2 then
   goto 12;
  ICAP := PTOUR[LIN, 3] + PTOUR[LJN, 3];
  for LM := 1 to NVT do
   begin
    If (TRCAP[LM] >= ICAP) and (TRU[LM] < TRA[LM]) then
     goto 10;
   end;
  goto 12;
10:
  MCAP := PTOUR[LIN, 4] + PTOUR[LJN, 4];
  If MCAP > MAXDROP then
   goto 12;
  TCAP := PTOUR[LIN, 5] + PTOUR[LJN, 5] - ROUND(S[K] / MILMIN);
  If TCAP > MAXTIME * 60 then
   goto 12;
  TRU[LM] := TRU[LM] + 1;
  TRU[PTOUR[LIN, 2]] := TRU[PTOUR[LIN, 2]] - 1;
  TRU[PTOUR[LJN, 2]] := TRU[PTOUR[LJN, 2]] - 1;
  DISTAN := D[LIN] + D[LJN] - S[K];
  TDIST := TDIST - (D[LIN] + D[LJN] - DISTAN);
  TOTTIME := TOTTIME - ROUND(S[K] / MILMIN);
  for L := 2 to NPT do
   begin
    If (PTOUR[L, 1] = IPT1) or (PTOUR[L, 1] = IPT2) then
     goto 64;
   end;
64:
  for M := 2 to NPT do
   begin
    If (PTOUR[M, 1] <> IPT1) and (PTOUR[M, 1] <> IPT2) then
     goto 105;
    PTOUR[M, 1] := PTOUR[L, 1];
    PTOUR[M, 2] := LM;
    PTOUR[M, 3] := ICAP;
    PTOUR[M, 4] := MCAP;
    PTOUR[M, 5] := TCAP;
105:
   end;
  ID[P, 3] := 1;
  ID[LIN - 1, 3] := ID[LIN - 1, 3] - 1;
  ID[LJN - 1, 3] := ID[LJN - 1, 3] - 1;
  If ID[LJN - 1, 3] <> 0 then
   goto 11;
  IB := POINT[LJN];
  JO := 1;
  for I := 1 to IB do
   begin
    IC := JX[ADJA[LJN, I]];
    If S[IC] <= 0 then
     goto 97;
    S[IC] := 0;
    OTHEAP(NN, S, IX, JX, IC);
97:
```

```
    end;
11:
  if ID[LIN - 1, 3] <> 0 then
    goto 12;
  IB := POINT[LIN];
  for I := 1 to IB do
    begin
      IC := JX[ADJA[LIN, I]];
      if S[IC] <= 0 then
        goto 98;
      S[IC] := 0;
      OTHEAP(NN, S, IX, JX, IC);
98:
    end;
  goto 90;
12:
  if JO = 1 then
    goto 90;
  S[K] := 0;
  OTHEAP(NN, S, IX, JX, K);
  goto 90;
16:
  TDIST := TDIST;
  TOTTIME := TOTTIME;
  ITK := 1;
  for IT := 1 to NPT1 do
    begin
      HIT := 0;
      for KT := 2 to NPT1 do
        begin
          while PTOUR[KT, 1] = IT do
            begin
              PTOUR[KT, 1] := ITK;
              HIT := 1;
            end;
        end;
      if HIT = 1 then
        ITK := ITK + 1;
    end;
  WRITELN('TOTAL RADIAL DIST FOR BACKHAULS IS', ROUND(SUMDIST), ' UNITS');
  WRITELN('TOTAL DISTANCE FOR DEDICATED BK ROUTES IS', ROUND(TDIST), ' UNITS');
  SAVINGS := ROUND(SUMDIST - TDIST);
  WRITELN('TOTAL SAVINGS IS', ROUND(SUMDIST - TDIST), ' UNITS');
{WRITELN('TOTAL TRAVEL TIME IS', ROUND(TOTTIME), ' MINUTES');}
  for I := 1 to NVT do
    begin
      ISUM := ISUM + TRU[I];
    end;
  WRITELN('THERE ARE', ISUM, ' TOURS IN ALL');
  NTOURS := ISUM;
  for I := 1 to 20 do
    begin
      RROUTE[I, 1] := 0;
      RROUTE[I, 2] := 0;
      RROUTE[I, 3] := 0;
    end;

{POST OPTIMIZER}
  RADIAL := 0;
  TEMP1 := 0;
  TEMP2 := 0;
  TEMP3 := 0;
{WRITELN('TOURS WITH LESS THAN 4 NODES  NOT CONSIDERED FOR POST-OPT');}
  for JJ := 1 to ISUM do
    begin
{WRITELN('ROUTE', JJ);}
      NONO := 99999;
      IPOINT := 0;
      IPOINT2 := 0;
```

```
    for II := 1 to 2 do
     begin
      IPOINT := 0;
      IPOINT2 := 0;
      for I := 1 to NARC do
       begin
{WRITELN('ID[I, 1]  ', ID[I, 1], '  ID[I, 2]  ', ID[I, 2], '  ID[I, 3]  ', ID[I, 3]);}
        if (ID[I, 3] = 2) and (PTOUR[ID[I, 2], 1] = JJ) then
         begin
          IPOINT := ID[I, 1];
          IPOINT2 := ID[I, 2];
          goto 27;
         end;
        if ID[I, 3] = 0 then
         begin
          goto 22;
         end;
        if PTOUR[ID[I, 2], 4] < 4 then
         goto 22;
        if PTOUR[ID[I, 2], 1] <> JJ then
         goto 22;
        if ID[I, 1] <> 1 then
         goto 22;
        if NONO = I then
         goto 22;
        IPOINT := ID[I, 2];
        NONO := I;
{WRITELN('NONO', NONO, 'IPOINT', IPOINT, 'REJARR[IPOINT, 1]', REJARR[IPOINT, 1]);}
        for J := 1 to NARC do
         begin
          if ID[J, 3] <> 1 then
           goto 23;
          if PTOUR[ID[J, 2], 4] < 4 then
           goto 23;
          if PTOUR[ID[J, 2], 1] <> JJ then
           goto 23;
          if NONO = J then
           goto 23;
          if (ID[J, 1] <> IPOINT) and (ID[J, 2] <> IPOINT) then
           goto 23;
          if ID[J, 1] = IPOINT then
           IPOINT2 := ID[J, 2];
          if ID[J, 2] = IPOINT then
           IPOINT2 := ID[J, 1];
{WRITELN('IPOINT2', IPOINT2, 'REJARR[IPOINT2, 1]', REJARR[IPOINT2, 1]);}
          TEMP1 := ROUND(SQRT(SQR(X[IPOINT] - X[IPOINT2]) + SQR(Y[IPOINT] - Y[IPOINT2])));
{WRITELN('IPOINT TO IPOINT2 ', TEMP1);}
          TEMP2 := ROUND(SQRT(SQR(X[1] - X[IPOINT2]) + SQR(Y[1] - Y[IPOINT2])));
{WRITELN('IPOINT2 TO DEPOT', TEMP2);}
          TEMP3 := ROUND(SQRT(SQR(X[1] - X[IPOINT]) + SQR(Y[1] - Y[IPOINT])) * (RADIAL / 100));
{WRITELN('IPOINT TO DEPOT RADIAL SURCHARGE', TEMP3);}
          if (TEMP1) < (TEMP2 + TEMP3) then
           goto 25;
{WRITELN('DISCONNECT', REJARR[IPOINT, 1], ' ', REJARR[IPOINT2, 1], 'DIST ', ROUND(SQRT(SQR(X[IPOINT] - X[IPOINT2]) +
      SQR(Y[IPOINT] - Y[IPOINT2]))));}
{WRITELN('CONNECT', REJARR[IPOINT2, 1], ' TO DEPOT    DIST', ROUND(SQRT(SQR(X[1] - X[IPOINT2]) + SQR(Y[1] - Y[IPOINT2])) * (1 +
      (RADIAL / 100))));}
          goto 26;
{ CONNECT IPOINT2 TO DEPOT LINK }
          NARC := NARC + 1;
          ID[NARC, 1] := 1;
          ID[NARC, 2] := IPOINT2;
          ID[NARC, 3] := 1;
          for M := 2 to NPT2 do
           begin
            if (PTOUR[M, 1] <> PTOUR[IPOINT2, 1]) then
             goto 7799;
            PTOUR[M, 3] := ICAP - Q[IPOINT2];
            PTOUR[M, 4] := MCAP - 1;
```

```
7799:
            end;

{REDEFINE IPOINT TO DEPOT LINK}
            ID[LL, 3] := 2;
            PTOUR[IPOINT, 1] := ISUM + 1;
            PTOUR[IPOINT, 2] := 1;
            PTOUR[M, 3] := Q[IPOINT];
            PTOUR[M, 4] := 1;
            PTOUR[M, 5] := 0;

{DISCONNECT IPOINT TO IPOINT2 LINK}
            ID[LLL, 3] := 0;

{SOLUTION UPDATES}
            ISUM := ISUM + 1;
            TDIST := TDIST + TEMP2 + TEMP3 - TEMP1;
            goto 26;

23:

          end;
22:
        end;
25:
       begin
{WRITELN(' LEAVE', REJARR[IPOINT, 1], ' TO ', REJARR[IPOINT2, 1], ' CONNECTED ', ROUND(SQRT(SQR(X[IPOINT] - X[IPOINT2]) +
       SQR(Y[IPOINT] - Y[IPOINT2]))));}
       end;
26:
     end;
27:
{if (ID[I, 3] = 2) then}
{WRITELN(' LEAVE(2) ', REJARR[IPOINT, 1], ' TO ', REJARR[IPOINT2, 1], ' CONNECTED ', ROUND(SQRT(SQR(X[IPOINT] - X[IPOINT2]) +
       SQR(Y[IPOINT] - Y[IPOINT2]))));}
   end;


  WRITELN('      ', 'I      ', 'J    ', 'T(I,J)  ', 'TRUCKTYPE', '  TOUR', ' DELIVERY ', '#DROPS', '  DISTANCE');
  for J := 1 to NARC do
   begin
    if ID[J, 3] = 0 then
     goto 35;
    WRITELN(REJARR[ID[J, 1], 1], REJARR[ID[J, 2], 1], ID[J, 3], PTOUR[ID[J, 2], 2], '  ', PTOUR[ID[J, 2], 1], PTOUR[ID[J, 2], 3],
      PTOUR[ID[J, 2], 4], ROUND(SQRT(SQR(X[ID[J, 1]] - X[ID[J, 2]]) + SQR(Y[ID[J, 1]] - Y[ID[J, 2]]))));
    P1 := ROUND(X[ID[J, 1]]);
    P2 := ROUND(Y[ID[J, 1]]);
    P3 := ROUND(X[ID[J, 2]]);
    P4 := ROUND(Y[ID[J, 2]]);
    RROUTE[PTOUR[ID[J, 2], 1], 1] := RROUTE[PTOUR[ID[J, 2], 1], 1] + ROUND(SQRT(SQR(X[ID[J, 1]] - X[ID[J, 2]]) + SQR(Y[ID[J, 1]] -
      Y[ID[J, 2]])));
    RROUTE[PTOUR[ID[J, 2], 1], 2] := PTOUR[ID[J, 2], 4];
{if ROUND(SUMDIST - TDIST) > 0 then}
{DRAWLINE(P1, P2, P3, P4);}
35:
   end;


  for I := 1 to NPT do
   begin
    RROUTE[PTOUR[I, 1], 3] := RROUTE[PTOUR[I, 1], 3] + ROUND(SQRT(SQR(X[I] - X[1]) + SQR(Y[I] - Y[1])));
{WRITELN('ROUTE', PTOUR[I, 1], 'NODE', I, 'RADIAL DIST', ROUND(SQRT(SQR(X[I] - X[1]) + SQR(Y[I] - Y[1]))));}
{WRITELN('CUM ROUT', RROUTE[PTOUR[I, 1], 3]);}
   end;
  for I := 1 to ISUM do
   begin
{WRITELN('ROUTE', I, 'RADIALS DIST ALTERNATIVE', RROUTE[I, 3]);}
```

```
    end;

  MIN := 0;
  MINPOINT := 0;
  for I := 1 to ISUM do
   begin
{WRITELN('ROUTE', I, ' DISTANCE ', RROUTE[I, 1], 'NO OF STOP', RROUTE[I, 2]);}
    if (ROUND(RROUTE[I, 2]) > 1) and (ROUND(RROUTE[I, 1]) <> 0) and (RROUTE[I, 2] > MIN) then
     begin
      MIN := RROUTE[I, 2];
      MINPOINT := I;
     end;
   end;
{WRITELN('MIN', MIN, 'MINPOINT', MINPOINT);}
  SAVINGS := 0;
  YY := 0;
  WRITELN('ENTER DEDICATED ROUTE ACCEPTANCE PARM(%)');
{READLN(MINTHRE);}
  MINTHRE := 10;
  WRITELN('ACCEPTANCE PARM IS ', MINTHRE);
  III := 0;
  for I := 1 to 50 do
   begin
    DEDROUTE[I] := 0;
   end;
  WRITELN('ROUTE', I, 'NARCS', NARC);
  for I := 1 to ISUM do
   begin
{WRITELN('ROUTE', I, ' DISTANCE ', RROUTE[I, 1], 'NO OF STOP', RROUTE[I, 2]);}
{WRITELN('MINTHRE', ROUND(((1 - (MINTHRE / 100)) * MIN)));}
    THRESHOLD := ROUND(((1 - (MINTHRE / 100)) * MIN));
    if (ROUND(RROUTE[I, 1]) = 0) then
     goto 444;
    if (route[I, 2] < 2) then
     goto 444;
    if (RROUTE[I, 2] < ROUND(((1 - (MINTHRE / 100)) * MIN))) then
     begin
      RROUTE[I, 1] := 0;
      goto 444;
     end;
    WRITELN('ROUTE', I, 'RADIAL DIST', RROUTE[I, 3], 'DEDIC ROUTE', RROUTE[I, 2]);
    if (RROUTE[I, 3] < RROUTE[I, 1]) then
     goto 444;
{WRITELN('ROUTE', I, 'NARCS', NARC);}
    II := 0;
    III := III + 1;
    DEDROUTE[III] := I;
{WRITELN('DEDROUTE[III]', I, 'III', III);}
    for J := 1 to NARC do
     begin
      if ID[J, 3] = 0 then
       goto 355;
      if I <> PTOUR[ID[J, 2], 1] then
       goto 355;
      WRITELN('NARC', J, 'I', REJARR[ID[J, 1], 1], 'J', REJARR[ID[J, 2], 1]);
      for II := 1 to REJECOUNT + 1 do
       begin
        if (REJARR[II, 1] = REJARR[ID[J, 1], 1]) or (REJARR[II, 1] = REJARR[ID[J, 2], 1]) then
         REJARR[II, 4] := 1;
       end;
      P1 := ROUND(X[ID[J, 1]]);
      P2 := ROUND(Y[ID[J, 1]]);
      P3 := ROUND(X[ID[J, 2]]);
      P4 := ROUND(Y[ID[J, 2]]);
{DRAWLINE(P1, P2, P3, P4);}
355:
     end;
    SAVINGS := SAVINGS + RROUTE[I, 1];
    WRITELN('SAVINGS', SAVINGS);
```

```
444:
    end;
   NARC2 := NARC;
{   WRITELN(' TRUCK TYPE', '  #AVAIL  ', '  CAP    ', '#USED IN ROUTING');}
   for M := 1 to NVT do
     begin
{WRITELN(M, '    ', TRA[M], '   ', TRCAP[M], '  ', TRU[M] + (NPT1 - BKPTR + 1));}
     end;
{WRITELN('DIVISOR', DIVISOR);}
{WRITELN('MAX NUMBER OF DROPS', MAXDROP);}
{WRITELN('MAX NUMBER OF HOURS PER ROUTE', MAXTIME);}
{WRITELN('AVERAGE SPEED OF VEHICLE IS', ROUND(SPEED), ' MPH');}
   INFLATION1 := 0;
   for I := 1 to REJECOUNT + 1 do
     begin
{WRITELN(REJARR[I, 1], '   ', REJARR[I, 5], '   ', REJARR[I, 6]);}
      X[I] := REJARR[I, 5];
      Y[I] := REJARR[I, 6];
     end;
   WRITELN('ENTER RADIAL (SUPPLIER) SURCHARGE');
   READLN(INFLATION);
   INFLATION1 := INFLATION / 100;
   for I := 1 to NPT do
    begin
     for J := 1 to NPT do
       begin
{C1[I, J] := ROUND(SQRT(SQR(X[I] - X[J]) + SQR(Y[I] - Y[J])));}
        if (I = 1) then
          begin
           CC1[J, 4] := ROUND((1 + INFLATION1) * C1(X, Y, I, J));
{WRITELN('1  ', I, 'J', J, '  C1[I,J]    ', C1(X,Y,I, J);}
          end;
       end;
    end;

{WRITELN('NODES', NODES);}

{ASSUMES MAXIMUM OF FOUR DEDICATED BK ROUTES HAVE BEEN TAKEN}
   WRITELN('RE-ENTER NUMBER OF NODES');
   READLN(NODES);
{WRITELN('     I     ', 'COST', '      LNKID', '   RADIAL SAV', '  ROUTE COST');}
   CMIN3 := 0;
   NPT2 := NPT;
   for K := 2 to NPT do
    begin
     CMIN1 := 9999;
     CMIN2 := 9999;
     CMIN3 := 0;
     CMIN4 := 0;
     CMIN5 := 0;
{WRITELN('K', K);}
{WRITELN('REJARR[K,5]', REJARR[K, 5]);}
{WRITELN('REJARR[K,2]', REJARR[K, 2]);}
     if REJARR[K, 5] <> REJARR[K, 2] then
      goto 123;
{WRITELN('DEDROUT[1]', DEDROUTE[1]);}
{WRITELN('DEDROUT[2]', DEDROUTE[2]);}
{WRITELN('DEDROUT[3]', DEDROUTE[3]);}
{WRITELN('DEDROUT[4]', DEDROUTE[4]);}

     if DEDROUTE[1] = PTOUR[K, 1] then
      goto 122;
     if (DEDROUTE[2] <> 0) and (DEDROUTE[2] = PTOUR[K, 1]) then
      goto 122;
     if (DEDROUTE[3] <> 0) and (DEDROUTE[3] = PTOUR[K, 1]) then
      goto 122;
     if (DEDROUTE[4] <> 0) and (DEDROUTE[4] = PTOUR[K, 1]) then
      goto 122;
```

```
123:
      for L := 1 to NARC do
       begin
        if ID[L, 3] = 0 then
         goto 120;
{if (ID[L, 3] = 2) then}
{goto 120;}
        if (ID[L, 1] = K) or (ID[L, 2] = K) then
         goto 120;
        if DEDROUTE[1] = PTOUR[ID[L, 2], 1] then
         goto 124;
        if (DEDROUTE[2] <> 0) and (DEDROUTE[2] = PTOUR[ID[L, 2], 1]) then
         goto 124;
        if (DEDROUTE[3] <> 0) and (DEDROUTE[3] = PTOUR[ID[L, 2], 1]) then
         goto 124;
        if (DEDROUTE[4] <> 0) and (DEDROUTE[4] = PTOUR[ID[L, 2], 1]) then
         goto 124;
        goto 120;
124:
        CC1[K, 1] := ROUND(C1(X, Y, ID[L, 1], K) + C1(X, Y, K, ID[L, 2]) - C1(X, Y, ID[L, 1], ID[L, 2]) - CC1[K, 4]);
        CC1[K, 2] := L;
{WRITELN(K, CC1[K, 1], ID[CC1[K, 2], 1], ID[CC1[K, 2], 2], CC1[K, 4], '      ', ROUND(C1(X,Y,ID[CC1[K, 2], 1], K)+ C1(X,Y,K, ID[CC1[K,
        2], 2]) - C1(X,Y,ID[CC1[K, 2], 1], ID[CC1[K, 2], 2])));}
        if CMIN1 < CC1[K, 1] then
         goto 120;
        CMIN1 := CC1[K, 1];
        CMIN2 := CC1[K, 2];
        CMIN4 := CC1[K, 4];
120:
      end;
     CC1[K, 1] := CMIN1;
     CC1[K, 2] := CMIN2;
     CC1[K, 3] := K;
     CC1[K, 4] := CMIN4;
     REJARR[K, 7] := CMIN1;
{WRITELN(NODES - NPT + K, CC1[K, 1], REJARR[ID[CC1[K, 2], 1], 1], REJARR[ID[CC1[K, 2], 2], 1], CC1[K, 4], '      ', ROUND(C1(X, Y,
       ID[CC1[K, 2], 1], K) + C1(X, Y, K, ID[CC1[K, 2], 2]) - C1(X, Y, ID[CC1[K, 2], 1], ID[CC1[K, 2], 2])));}
122:
   end;
  for L := 1 to NARC do
   begin
    if DEDROUTE[1] = PTOUR[ID[L, 2], 1] then
     goto 1249;
    if (DEDROUTE[2] <> 0) and (DEDROUTE[2] = PTOUR[ID[L, 2], 1]) then
     goto 1249;
    if (DEDROUTE[3] <> 0) and (DEDROUTE[3] = PTOUR[ID[L, 2], 1]) then
     goto 1249;
    if (DEDROUTE[4] <> 0) and (DEDROUTE[4] = PTOUR[ID[L, 2], 1]) then
     goto 1249;
    ID[L, 3] := 0;
1249:
   end;
{WRITELN('      ', 'I      ', 'J    ', 'T(I,J)  ', 'TRUCKTYPE', '  TOUR', ' DELIVERY ', '#DROPS', '  DISTANCE');}
  for J := 1 to NARC do
   begin
    if ID[J, 3] = 0 then
     goto 3575;
{WRITELN(REJARR[ID[J, 1], 1], REJARR[ID[J, 2], 1], ID[J, 3], PTOUR[ID[J, 2], 2], '  ', PTOUR[ID[J, 2], 1], PTOUR[ID[J, 2], 3],
      PTOUR[ID[J, 2], 4], ROUND(SQRT(SQR(X[ID[J, 1]] - X[ID[J, 2]]) + SQR(Y[ID[J, 1]] - Y[ID[J, 2]]))));}
3575:
   end;

{WRITELN('THRESHOLD', THRESHOLD);}
  goto 301;
4444:
{WRITELN('THRESHOLD', THRESHOLD);}
{WRITELN('TEST');}

{ADD NODE DEDICATED BACKHAUL ROUTE}
```

```
{WRITELN('NPT', NPT2);}
  for K := 2 to NPT2 do
   begin
{WRITELN('K ', K, ' CC1[K,1] ', CC1[K, 1], ' CC1[K,2] ', CC1[K, 2], ' CC1[K,3] ', CC1[K, 3], ' CC1[K,4] ', CC1[K, 4]);}
   end;
  for I := 1 to NARC2 do
   begin
{WRITELN('I ', I, ' ID[1]  ', ID[I, 1], '  ID[,2]  ', ID[I, 2], '  ID [ 3 ] ', ID[I, 3]);}
   end;
  for I := 1 to NPT2 do
   begin
{WRITELN('PTOUR[I,1]', PTOUR[I, 1]);}
   end;
  MMM := 0;
2727:
  ICAP := 0;
  MCAP := 0;
  TCAP := 0;
  for K := 2 to NPT2 do
   begin
{WRITELN('K', NODES - NPT2 + K, 'REJARR[K,4]', REJARR[K, 4], 'REJARR[K,7]', REJARR[K, 7], '  SHOW 1');}
    if (MMM = 1) and (REJARR[K, 4] = 1) then
     goto 778;
    if REJARR[K, 8] < REJARR[K, 7] then
     begin
      if MMM = 1 then
       REJARR[K, 4] := 2;
      goto 778;
     end;
{WRITELN('REJARR[7]', REJARR[K, 7], ' REJARR [ 8 ] ', REJARR[K, 8]);}
    if (REJARR[K, 7] < 0) then
     begin
      if (K > 2) and (ID[CC1[K, 2], 3] = 0) then
       begin
        CMIN1 := 9999;
        CMIN2 := 9999;
        CMIN3 := 0;
        CMIN4 := 0;
        CMIN5 := 0;
{WRITELN('REHOME  K ', NODES - NPT2 + CC1[K, 3], '  ORIGINAL K  TOUR  ASSIGNMENT  ', PTOUR[ID[CC1[K, 2], 2], 1]);}
        for L := 1 to NARC2 do
         begin
          if ID[L, 3] = 0 then
           goto 1200;
          if PTOUR[ID[CC1[K, 2], 2], 1] = PTOUR[ID[L, 2], 1] then
           goto 1244;
          goto 1200;
1244:
{WRITELN('ARC TOUR  ', PTOUR[ID[L, 2], 1]);}
          CC1[K, 1] := ROUND(C1(X, Y, ID[L, 1], K) + C1(X, Y, K, ID[L, 2]) - C1(X, Y, ID[L, 1], ID[L, 2]) - CC1[K, 4]);
          CC1[K, 2] := L;
{WRITELN('K', K, 'CC1[K, 1]', CC1[K, 1], 'LINK ID  ', L, '  ORIG', NODES - NPT2 + ID[CC1[K, 2], 1], 'DEST', NODES - NPT2 + ID[CC1[K, 2],
2], 'RADIAL COST ', CC1[K, 4], ' ROUTE COST ', ROUND(C1(X, Y, ID[CC1[K, 2], 1], K) + C1(X, Y, K, ID[CC1[K, 2], 2]));}
          if CMIN1 < CC1[K, 1] then
           goto 1200;
          CMIN1 := CC1[K, 1];
          CMIN2 := CC1[K, 2];
          CMIN4 := CC1[K, 4];
1200:
         end;
        CC1[K, 1] := CMIN1;
        CC1[K, 2] := CMIN2;
        CC1[K, 3] := K;
        CC1[K, 4] := CMIN4;
        REJARR[K, 7] := CMIN1;
{WRITELN(NODES - NPT2 + K, CC1[K, 1], REJARR[ID[CC1[K, 2], 1], 1], REJARR[ID[CC1[K, 2], 2], 1], CC1[K, 4], '    ', ROUND(C1(X, Y,
ID[CC1[K, 2], 1], K) + C1(X, Y, K, ID[CC1[K, 2], 2]) - C1(X, Y, ID[CC1[K, 2], 1], ID[CC1[K, 2], 2])));}
       end;
      ICAP := PTOUR[CC1[K, 3], 3] + PTOUR[ID[CC1[K, 2], 2], 3];
```

```
    begin
     CMIN1 := 9999;
     CMIN2 := 9999;
     CMIN3 := 0;
     CMIN4 := 0;
     CMIN5 := 0;
     if REJARR[K, 4] <> 0 then
      goto 1220;
     for L := 1 to NARC2 do
      begin
       if ID[L, 3] = 0 then
        goto 1201;
       CC1[K, 1] := ROUND(C1(X, Y, ID[L, 1], K) + C1(X, Y, K, ID[L, 2]) - C1(X, Y, ID[L, 1], ID[L, 2]) - CC1[K, 4]);
       CC1[K, 2] := L;
{WRITELN(K, CC1[K, 1], ID[CC1[K, 2], 1], ID[CC1[K, 2], 2], CC1[K, 4], '      ', ROUND(C1(X,Y,ID[CC1[K, 2], 1], K)+ C1(X,Y,K, ID[CC1[K,
       2], 2]) - C1(X,Y,ID[CC1[K, 2], 1], ID[CC1[K, 2], 2])));}
       if CMIN1 < CC1[K, 1] then
        goto 1201;
       CMIN1 := CC1[K, 1];
       CMIN2 := CC1[K, 2];
       CMIN4 := CC1[K, 4];
1201:
      end;
     CC1[K, 1] := CMIN1;
     CC1[K, 2] := CMIN2;
     CC1[K, 3] := K;
     CC1[K, 4] := CMIN4;
     REJARR[K, 7] := CMIN1;
{WRITELN(NODES - NPT2 + K, CC1[K, 1], REJARR[ID[CC1[K, 2], 1], 1], REJARR[ID[CC1[K, 2], 2], 1], CC1[K, 4], '      ', ROUND(C1(X, Y,
      ID[CC1[K, 2], 1], K) + C1(X, Y, K, ID[CC1[K, 2], 2]) - C1(X, Y, ID[CC1[K, 2], 1], ID[CC1[K, 2], 2])));}
1220:
    end;
   MMM := 1;
   goto 2727;
2728:

  for I := 1 to ISUM do
   begin
    if (ROUND(RROUTE[I, 1]) = 0) then
     goto 4445;
    if (rroute[I, 2] < 2) then
     goto 4445;
{WRITELN(' NO OF STOPS', RROUTE[I, 2]);}
{WRITELN('((1 - (MINTHRE) / 100) * MIN) ', ROUND((1 - (MINTHRE) / 100) * MIN));}
{WRITELN('THRESHOLD', THRESHOLD);}
    if RROUTE[I, 2] < (THRESHOLD) then
     goto 4445;
    if (RROUTE[I, 3] < RROUTE[I, 1]) then
     goto 4445;
    WRITELN('ROUTE', I, 'NARCS', NARC2);
    newscale := 2;
    for J := 1 to NARC2 do
     begin
      if (ID[J, 3] = 0) or (ID[J, 3] = 2) then
       goto 3555;
{WRITELN('NARC', J, 'F', REJARR[ID[J, 1], 1], 'J', REJARR[ID[J, 2], 1]);}
      if ((I = PTOUR[ID[J, 2], 1]) and (I = PTOUR[ID[J, 1], 1]) or ((I = PTOUR[ID[J, 2], 1]) and (ID[J, 1] = 1)) then
       goto 3666;
      goto 3555;
3666:
      WRITELN('NARC', J, 'F', REJARR[ID[J, 1], 1], 'J', REJARR[ID[J, 2], 1], 'ROUTE', PTOUR[ID[J, 2], 1]);
      P1 := ROUND(X[ID[J, 1]]) * (newscale);
      P2 := ROUND(Y[ID[J, 1]]) * newscale;
      P3 := ROUND(X[ID[J, 2]]) * (newscale);
      P4 := ROUND(Y[ID[J, 2]]) * newscale;
      DRAWLINE(P1, P2, P3, P4);
3555:
     end;
4445:
```

```
{WRITELN('TRCAP', TRCAP[1], 'ICAP', ICAP);}
      if (TRCAP[1] < ICAP) then
       goto 777;
      MCAP := PTOUR[CC1[K, 3], 4] + PTOUR[ID[CC1[K, 2], 2], 4];
{WRITELN('MCAP', MCAP, 'MAXDROP', MAXDROP);}
      if MCAP > MAXDROP then
       goto 777;
      PTOUR[CC1[K, 3], 1] := PTOUR[ID[CC1[K, 2], 2], 1];
      for M := 2 to NPT2 do
       begin
        if (PTOUR[M, 1] <> PTOUR[ID[CC1[K, 2], 2], 1]) then
         goto 77;
{WRITELN('NODE', NODES - NPT2 + M, ' IN TOUR ', PTOUR[ID[CC1[K, 2], 2], 1]);}
        PTOUR[M, 1] := PTOUR[ID[CC1[K, 2], 2], 1];
        PTOUR[M, 2] := 1;
        PTOUR[M, 3] := ICAP;
        PTOUR[M, 4] := MCAP;
        PTOUR[M, 5] := 0;
77:
       end;
      NARC2 := NARC2 + 1;
      ID[NARC2, 1] := ID[CC1[K, 2], 1];
      ID[NARC2, 2] := CC1[K, 3];
      ID[NARC2, 3] := 1;
      NARC2 := NARC2 + 1;
      ID[NARC2, 1] := CC1[K, 3];
      ID[NARC2, 2] := ID[CC1[K, 2], 2];
      ID[NARC2, 3] := 1;
      ID[CC1[K, 2], 3] := 0;
      if MMM = 0 then
       REJARR[K, 4] := 1
      else
       REJARR[K, 4] := 2;
      WRITELN('INSERT NODE', NODES - NPT2 + CC1[K, 3], ' INTO TOUR ', PTOUR[CC1[K, 3], 1]);
{WRITELN('K', NODES - NPT2 + CC1[K, 3], 'REJARR[K,4]', REJARR[K, 4], 'REJARR[K,7]', REJARR[K, 7], ' SHOW Z');}
{writeln('NARC2-1 ', NARC2 - 1, ' ID[1] ', ID[NARC2 - 1, 1], ' ID[2] ', ID[NARC2 - 1, 2], ' ID[3] ', ID[NARC2 - 1, 3]);}
{writeln('NARC2 ', NARC2, ' ID[1] ', ID[NARC2, 1], ' ID[2] ', ID[NARC2, 2], ' ID[3] ', ID[NARC2, 3]);}
{PTOUR[CC1[K, 3], 1] := PTOUR[ID[CC1[K, 2], 2], 1];}
{WRITELN('K', CC1[K, 3], 'PTOUR[K,1]', PTOUR[CC1[K, 3], 1]);}
{WRITELN('K ROUTE COST', ROUND(C1(X, Y, ID[CC1[K, 2], 1], K) + C1(X, Y, K, ID[CC1[K, 2], 2]) - C1(X, Y, ID[CC1[K, 2], 1], ID[CC1[K,
      2], 2])));}
      RROUTE[PTOUR[CC1[K, 3], 1], 1] := RROUTE[PTOUR[CC1[K, 3], 1], 1] + ROUND(C1(X, Y, ID[CC1[K, 2], 1], K) + C1(X, Y, K,
      ID[CC1[K, 2], 2]) - C1(X, Y, ID[CC1[K, 2], 1], ID[CC1[K, 2], 2]));
      RROUTE[PTOUR[CC1[K, 3], 1], 3] := RROUTE[PTOUR[CC1[K, 3], 1], 3] + ROUND(C1(X, Y, 1, K));

      if MINPOINT = PTOUR[CC1[K, 3], 1] then
       MIN := RROUTE[PTOUR[CC1[K, 3], 1], 1];

      SAVINGS := SAVINGS + ROUND(C1(X, Y, ID[CC1[K, 2], 1], K) + C1(X, Y, K, ID[CC1[K, 2], 2]) - C1(X, Y, ID[CC1[K, 2], 1],
      ID[CC1[K, 2], 2]));
{WRITELN('SAVINGS', SAVINGS, 'ROUTE LENGHT', RROUTE[PTOUR[CC1[K, 3], 1], 1]);}
{WRITELN('RADIAL DISTANCE ALTERNATIVE UPDATE', RROUTE[PTOUR[CC1[K, 3], 1], 3]);}

     end;
    goto 778;
777:
     WRITELN('NODE REJECTED IN DEDICATED BKHAUL ROUTE');
778:
   end;
  if MMM = 1 then
   goto 2728;
  for K := 2 to NPT2 do
   begin
{WRITELN('K', NODES - NPT2 + K : 3, 'REJ[1]', REJARR[K, 1], 'REJ[4]', REJARR[K, 4] : 2, 'REJ[5]', REJARR[K, 5] : 4, 'REJ[6]',
      REJARR[K, 6] : 4, 'REJ[7]', REJARR[K, 7], 'REJ[8]', REJARR[K, 8]);}
   end;
{WRITELN('     I    ', 'COST', '     LNKID', '   RADIAL SAV', ' ROUTE COST');}
  CMIN3 := 0;
  for K := 2 to NPT2 do
```

```
    end;
  TROUTE := 0;
  ARCSAVINGS := 0;
  for II := 1 to ISUM do
   begin
    if (ROUND(RROUTE[II, 1]) = 0) then
     goto 7478;
    if (rroute[II, 2] < 2) then
     goto 7478;
{WRITELN(' NO OF STOPS', RROUTE[II, 2]);}
{WRITELN('THRESHOLD', THRESHOLD);}
    if RROUTE[II, 2] < (THRESHOLD) then
     goto 7478
    else
     TROUTE := II;
    RROUTE[II, 2] := 0;
    RX[1] := X[1];
    RY[1] := Y[1];
{WRITELN('RX', ROUND(X[1]), 'RY', ROUND(Y[1]));}
    COUNTT := 0;
    for I := 1 to NPT2 do
     begin
      if TROUTE = PTOUR[I, 1] then
       begin
{WRITELN('NODE', NODES - NPT2 + I, 'PTOUR[I,1]', PTOUR[I, 1], 'RX', ROUND(X[I]), 'RY', ROUND(Y[I]));}
        RROUTE[II, 2] := RROUTE[II, 2] + 1;
        COUNTT := COUNTT + 1;
        RX[COUNTT + 1] := X[I];
        RY[COUNTT + 1] := Y[I];
       end;
     end;
    WRITELN(' NO OF STOPS', RROUTE[II, 2], 'ROUTELNGTH', RROUTE[II, 1]);
    NPTT := RROUTE[II, 2] + 1;
    ROUTELENGTH := 0;
    POSTARC(RX, RY, NPTT, ROUTELENGTH);
    WRITELN('CW ROUTE', RROUTE[II, 1], 'ARC ROUTE', ROUTELENGTH);
    if (RROUTE[II, 1] > ROUTELENGTH) then
     begin
      WRITELN('ROUTE SAVINGS ', RROUTE[II, 1] - ROUTELENGTH);
      ARCSAVINGS := ARCSAVINGS + RROUTE[II, 1] - ROUTELENGTH;
     end;
7478:
   end;
  WRITELN('SAVINGS', SAVINGS, 'ARCSAVINGS', ARCSAVINGS);
  SAVINGS := SAVINGS - ARCSAVINGS;
  goto 301;
998:
  WRITELN('POINT(I) FOR SOME I EXCEEDS 50.');
301:
  WRITELN('THE END');
 end;
end.
```

```
unit ARCOPT;
interface
 type
  ARRN11 = array[1..50] of REAL;
  COUNTER3 = INTEGER;
 procedure POSTARC (var RX: ARRN11; var RY: ARRN11; var NPTT: COUNTER3; var ROUTELENGHT: COUNTER3);
implementation
 procedure POSTARC (var RX: ARRN11; var RY: ARRN11; var NPTT: COUNTER3; var ROUTELENGHT: COUNTER3);
  label
   9999, 7777;
  type
   ARRNN = array[1..100, 1..100] of INTEGER;
   ARRN = array[1..100] of INTEGER;
   ARRN6 = array[1..100, 1..2] of integer;
   ARRN3 = array[1..101] of integer;
   ARRN5 = array[1..4, 1..3] of INTEGER;
   ARRN7 = array[1..100] of LONGINT;
   COUNTER = INTEGER;
   ARRN4 = array[1..100, 1..3] of integer;
  var
   ROUTELENGTH: INTEGER;
   Ticks, ticks2: Longint;          { keeps track of time }
   COST: ARRN5;
   LIST: ARRN6;
   IROUTE: ARRN;
   MAP, IBEG: ARRN3;
   ROUTE: ARRN;
   ILIST: ARRN4;
   Dist: ARRNN;
   X, Y: ARRN7;
   APRM: INTEGER;
   INITSOL, MINN, MINN2, TIMM2, 999, KLCAP: INTEGER;
   TWEIGHT, arcsol: longint;
   I, J, P1, P2, P3, P4, PARM, ELAPS1, ELAPS2, ELAPS3, ELAPS4: INTEGER;
   MAXTIME, msttime, ZZZZ: integer;
   I1, I2, J1, J2, K1, K2, KSW, N, NTT, IP11, NPT, SCALE, K, II, III, M, zzz, MM, XXX, XXXX: counter;
{generate NODES random numbers }
{between 0 and SCALE}
  procedure NETGEN (var NPT, SCALE, K: COUNTER; var dist: ARRNN; var X, Y: ARRN7; var msttime, xxx, ZZZ: counter);
   label
    30, 1000, 3333, 3334;
   var
    m: longint;
    n: longint;
    nt: integer;
    ip1: integer;
    U: array[1..100] of integer;
    endpt: array[1..2, 1..100] of integer;
    ww: array[1..100] of integer;
    Dmin: real;
    imin: integer;
    d1: integer;
    a: array[1..2000] of integer;
    b: array[1..2000] of integer;
    sstd, SSTD2: integer;
    INTERIM, INTERIM1, INTERIM2: double;
    i, l, j, NODES, p1, p2, idx, ii: integer;
    r, wr: real;
    yy: array[1..200] of real;
   begin
    if xxx = 1 then
     goto 3334;
    for i := 1 to 100 do
     begin
{x[i] := 0;}
{y[i] := 0;}
      route[i] := 0;
     end;
    for i := 1 to 100 do
```

```
    begin
     for J := 1 to 100 do
      begin
       DIST[I, J] := 0;
      end;
    end;
   wr := 0;
   r := 0;
   I := 0;
   l := 0;
   j := 0;
   m := 0;
   n := 0;
   nt := 0;
   ip1 := 0;
   nodes := 0;
   scale := 0;
   interim := 0;
   interim1 := 0;
   interim2 := 0;
   dmin := 0;
   imin := 0;
   d1 := 0;
   for i := 1 to 2000 do
    begin
     a[i] := 0;
     b[i] := 0;
    end;
   ii := 0;
   sstd := 0;
   sstd2 := 0;
{ENTER NODES AND SCALE}
{SCALE := 200;}
   nodes := npt;
   I := 0;
3334:
   sstd := 0;
   nodes := npt;
   nt := nodes - 1;
   m := 0;
   n := 0;
   for m := 1 to nt do
    begin
     ip1 := m + 1;
     for n := ip1 to nodes do
      begin
       INTERIM1 := sqr(x[m] - x[n]);
       INTERIM2 := sqr(Y[m] - Y[n]);
       INTERIM := sqrt(INTERIM1 + INTERIM2);
       Dist[m, n] := round(INTERIM);
       Dist[n, m] := Dist[m, n];
      end;
    end;
   Ticks := TickCount;                    { *** start timing *** }
   kk := k;
   for ii := 1 to kk do
    begin
     I := 0;
     for i := 2 to nodes do
      begin
       U[i] := 1;
       ww[i] := Dist[1, i];
      end;
30:
     Dmin := 1E35;
     for i := 2 to nodes do
      begin
       while (ww[i] <> 0) and (ww[i] < Dmin) do
        begin
```

```
          Dmin := ww[i];
          Imin := i;
        end;
      end;
    I := I + 1;
    endpt[1, I] := Imin;
    endpt[2, I] := U[Imin];
    If (I = (nodes - 1)) then
      goto 1000;
    ww[Imin] := 0;
    for I := 2 to nodes do
      begin
       If (ww[I] <> 0) then
        d1 := Dist[i, Imin];
       If d1 = 0 then
        d1 := 1;
       while (ww[i] > d1) do
        begin
         U[i] := Imin;
         ww[i] := d1;
        end;
      end;
    goto 30;
1000:
{calculate MST lenght and draw}
{resulting network}
      for I := 1 to (nodes - 1) do
       begin
        a[I + ((ii - 1) * (nodes - 1))] := endpt[1, I];
        b[I + ((ii - 1) * (nodes - 1))] := endpt[2, I];
   { IF II = 1 THEN WRITELN(a ', endpt[1, I],' b ',endpt[2, i], 'dist ',Dist[endpt[1, I],endpt[2, I]]);}
        sstd := sstd + Dist[a[I + ((ii - 1) * (nodes - 1))], b[I + ((ii - 1) * (nodes - 1))]] := 500;
        Dist[a[I + ((ii - 1) * (nodes - 1))], b[I + ((ii - 1) * (nodes - 1))]] := 500;
        Dist[b[I + ((ii - 1) * (nodes - 1))], a[I + ((ii - 1) * (nodes - 1))]] := 500;
       end;
      SSTD2 := sstd;
      for I := 1 to nodes - 1 do
       begin
        endpt[1, I] := 0;
        endpt[2, I] := 0;
        sstd := 0;
       end;
{If II = 1 then}
{WRITELN('NODES', NODES : 3, 'MST', SSTD2 : 5);}
    end;
   nt := nodes - 1;
   m := 0;
   n := 0;
   for m := 1 to nt do
    begin
     ip1 := m + 1;
     for n := ip1 to nodes do
      begin
       dist[m, n] := 255;
       DIST[N, M] := 255;
      end;
    end;
   for n := 1 to (k * (nodes - 1)) do
    begin
     INTERIM1 := sqr(x[a[n]] - x[b[n]]);
     INTERIM2 := sqr(Y[a[n]] - Y[b[n]]);
     INTERIM := sqrt(INTERIM1 + INTERIM2);
     Dist[a[n], b[n]] := round(INTERIM);
     Dist[b[n], a[n]] := Dist[a[n], b[n]];
    end;
3333:
   Ticks := ROUND(TickCount - Ticks) / 6.0);          { *** stop timing *** }
   MSTTIME := Ticks;
  end;
```

09/05/90 13:16 basic arcopt unit

```
procedure AREARR (var MAP: ARRN3; var ILIST: ARRN4; var LIST: ARRN6; var COST: ARRN5; var IROUTE: ARRN; var I1, I2,
    J1, J2, K1, K2, KSW, LISTNO, NUMPTS: COUNTER);
 label
  150, 160, 200, 350, 360, 400, 1111;
 var
  I, J, L: INTEGER;
  ILSAV, CSTSAV, JLSAV: INTEGER;
begin
 i := 0;
 ILSav := 0;
 cstsav := 0;
 l := 0;
 j := 0;
 jlsav := 0;
{ WRITELN( 'I1',I1,'I2',I2,'J1',J1,'J2',J2,'K1',K1,'K2', K2,'KSW', KSW,'LISTNO',LISTNO,'NUMPTS',NUMPTS );}
  ILIST[I1, 2] := I2;
  ILIST[I1, 3] := COST[KSW, 1];
  if (KSW = 2) or (KSW = 3) then
   goto 200;
  I := I2;
  ILSAV := ILIST[I, 1];
  CSTSAV := ILIST[ILSAV, 3];
  ILIST[I, 1] := I1;
  goto 160;
150:
  ILSAV := ILIST[I, 1];
  CSTSAV := ILIST[ILSAV, 3];
  ILIST[I, 1] := ILIST[I, 2];
160:
  ILIST[I, 2] := ILSAV;
  ILIST[I, 3] := CSTSAV;
  if I = J1 then
   goto 200;
  I := ILSAV;
  goto 150;
200:
  ILIST[J1, 2] := J2;
  ILIST[J1, 3] := COST[KSW, 2];
  ILIST[I2, 1] := I1;
  if (KSW = 2) or (KSW = 4) then
   goto 400;
  J := J2;
  JLSAV := ILIST[J, 1];
  CSTSAV := ILIST[JLSAV, 3];
  ILIST[J, 1] := J1;
  goto 360;
350:
  JLSAV := ILIST[J, 1];
  CSTSAV := ILIST[JLSAV, 3];
  ILIST[J, 1] := ILIST[J, 2];
360:
  ILIST[J, 2] := JLSAV;
  ILIST[J, 3] := CSTSAV;
  if J = K1 then
   goto 400;
  J := JLSAV;
  goto 350;
400:
  ILIST[K1, 2] := K2;
  ILIST[K1, 3] := COST[KSW, 3];
  ILIST[J2, 1] := J1;
  ILIST[K2, 1] := K1;
  I := IROUTE[1];
  MAP[I] := 1;
  for L := 2 to NUMPTS do
   begin
    IROUTE[L] := ILIST[I, 2];
    I := IROUTE[L];
    MAP[I] := L;
```

```
    end;
   if LISTNO = -1 then
    goto 1111;
   LISTNO := LISTNO + 1;
   LIST[LISTNO, 1] := K1;
   LIST[LISTNO, 2] := K2;
   LISTNO := LISTNO + 1;
   LIST[LISTNO, 1] := J1;
   LIST[LISTNO, 2] := J2;
   LISTNO := LISTNO + 1;
   LIST[LISTNO, 1] := I1;
   LIST[LISTNO, 2] := I2;
1111:
  end;
  procedure ARCOPT (var NPT, SCALE: COUNTER; var dist: ARRNN; var ILIST: ARRN4; var MAP, IBEG: ARRN3; var LIST: ARRN6;
      var COST: ARRN5; var IROUTE: ARRN; var I1, I2, J1, J2, K1, K2, KSW: COUNTER);
   label
    20, 30, 40, 100, 300, 400, 500, 600, 700, 701, 702, 704, 706, 707, 709, 710, 720, 730, 740, 750, 800, 1000, 3333;
   type
    ARRN2 = array[1..5575] of integer;
   var
    ADJ, C: ARRN2;
    BIGINT, NUMPTS, I, J: INTEGER;
    LISTNO, SAVMAX, DIFF, NM, NSW: INTEGER;
    II, JJ, L, IB1, IE1, IB2, IE2, IN1, IN2, LP, N1, N2, VVV: INTEGER;
    LADD, IS1, IS2: INTEGER;
    MAXPTS, K, COUNT, LAST: INTEGER;
    TEMP: LONGINT;
  begin
   MAXPTS := 300;
   BIGINT := 255;
   COUNT := 0;
   LAST := 0;
   I := 0;
   J := 0;
   J := NPT;
   TEMP := 0;
   for I := 1 to NPT do
    begin
     IROUTE[I] := I;
     ILIST[J, 2] := I;
     ILIST[I, 1] := J;
     ILIST[I, 3] := 0;
     J := I;
    end;
   for I := 1 to NPT do
    begin
     for J := 1 to NPT do
      begin
       if I = J then
        goto 20;
       if (dist[I, J] >= BIGINT) then
        goto 20;
       COUNT := COUNT + 1;
       C[COUNT] := dist[I, J];
       ADJ[COUNT] := J;
       if (COUNT > 75 * MAXPTS) then
        goto 30;
       if LAST <> I then
        begin
         LAST := I;
         IBEG[I] := COUNT;
        end;
20:
      end;
    end;
   IBEG[NPT + 1] := COUNT + 1;
   J := NPT;
   for K := 1 to NPT do
```

09/05/90 13:16 basic arcopt unit

```
    begin
     I := ILIST[J, 2];
     ILIST[J, 3] := dist[J, I];
     TEMP := TEMP + ILIST[J, 3];
{WRITELN('I', I, 'J', J, 'DIST', DIST[J, I], 'TEMP', TEMP);}
     J := I;
    end;
   goto 40;
30:
   WRITELN('TOO MANY ARCS');
40:
   WRITELN('NUMBER OF ACCEPTABLE ARCS', COUNT, 'INIT SOL', TEMP);
   bigint := 0;
   numpts := 0;
   i := 0;
   savmax := 0;
   diff := 0;
   nm := 0;
   new := 0;
   ii := 0;
   jj := 0;
   L := 0;
   IB1 := 0;
   IE1 := 0;
   IE2 := 0;
   IN1 := 0;
   IN2 := 0;
   LP := 0;
   N1 := 0;
   N2 := 0;
   VVV := 0;
   LADD := 0;
   IS1 := 0;
   IS2 := 0;
   VVV := 0;
   BIGINT := 255;
   NUMPTS := NPT;
   NM := NUMPTS - 1;
   KSW := 5;
   SAVMAX := 0;
   II := NUMPTS;
   for I := 1 to NUMPTS do
    begin
     MAP[II] := I;
     II := IROUTE[I];
     JJ := ILIST[II, 2];
     LIST[I, 1] := II;
     LIST[I, 2] := JJ;
    end;
   LISTNO := NUMPTS;
100:
   if LISTNO = 0 then
    goto 3333;
   I1 := LIST[LISTNO, 1];
   I2 := ILIST[I1, 2];
   if (I2 = LIST[LISTNO, 2]) then
    goto 300;
   I1 := LIST[LISTNO, 2];
   I2 := ILIST[I1, 2];
   if (I2 = LIST[LISTNO, 1]) then
    goto 300;
   WRITELN('LIST EMPTY');
300:
   LISTNO := LISTNO - 1;
   for L := 1 to NUMPTS do
    begin
     if (MAP[L] < MAP[I1]) then
      MAP[L] := MAP[L] + NUMPTS;
    end;
```

```
    IB1 := IBEG[I1];
    IE1 := IBEG[I1 + 1] - 1;
    IB2 := IBEG[I2];
    IE2 := IBEG[I2 + 1] - 1;
    for IS1 := IB1 to IE1 do
     begin
      IN1 := ADJ[IS1];
      if IN1 = I2 then
       goto 1000;
      for IS2 := IB2 to IE2 do
       begin
        IN2 := ADJ[IS2];
{VVV := VVV + 1;}
        if IN2 = I1 then
         goto 800;
        J1 := IN1;
        K1 := IN2;
        J2 := ILIST[J1, 2];
        if (MAP[K1] < MAP[J2]) then
         goto 400;
        if J2 = I1 then
         goto 400;
        K2 := ILIST[K1, 2];
        if (dist[J2, K2] = BIGINT) then
         goto 400;
        DIFF := ILIST[I1, 3] + ILIST[J1, 3] + ILIST[K1, 3] - (C[IS1] + C[IS2] + dist[J2, K2]);
        if DIFF <= SAVMAX then
         goto 400;
        SAVMAX := DIFF;
        KSW := 1;
        COST[KSW, 1] := C[IS1];
        COST[KSW, 2] := C[IS2];
        COST[KSW, 3] := dist[J2, K2];
        goto 700;
400:
        J2 := IN1;
        K1 := IN2;
        J1 := ILIST[J2, 1];
        if (MAP[K1] < MAP[J2]) then
         goto 500;
        K2 := ILIST[K1, 2];
        if (dist[J1, K2] = BIGINT) then
         goto 500;
        if J1 = K2 then
         goto 500;
        DIFF := ILIST[I1, 3] + ILIST[J1, 3] + ILIST[K1, 3] - (C[IS1] + C[IS2] + dist[J1, K2]);
        if DIFF <= SAVMAX then
         goto 500;
        SAVMAX := DIFF;
        KSW := 2;
        COST[KSW, 1] := C[IS1];
        COST[KSW, 2] := C[IS2];
        COST[KSW, 3] := dist[J1, K2];
        goto 700;
500:
        J2 := IN1;
        K2 := IN2;
        J1 := ILIST[J2, 1];
        K1 := ILIST[K2, 1];
        if (MAP[K1] < MAP[J2]) then
         goto 600;
        if (dist[J1, K1] = BIGINT) then
         goto 600;
        if J1 = K1 then
         goto 600;
        DIFF := ILIST[I1, 3] + ILIST[J1, 3] + ILIST[K1, 3] - (C[IS1] + C[IS2] + dist[J1, K1]);
        if DIFF <= SAVMAX then
         goto 600;
        SAVMAX := DIFF;
```

```
        KSW := 3;
        COST[KSW, 1] := C[IS1];
        COST[KSW, 2] := dist[J1, K1];
        COST[KSW, 3] := C[IS2];
        goto 700;
600:
        K1 := IN1;
        J2 := IN2;
        J1 := ILIST[J2, 1];
        K2 := ILIST[K1, 2];
        if (MAP[K1] < MAP[J2]) then
         goto 700;
        if (dist[J1, K2] = BIGINT) then
         goto 700;
        if J1 = K2 then
         goto 700;
        DIFF := ILIST[I1, 3] + ILIST[J1, 3] + ILIST[K1, 3] - (C[IS1] + C[IS2] + dist[J1, K2]);
        if DIFF <= SAVMAX then
         goto 700;
        SAVMAX := DIFF;
        KSW := 4;
        COST[KSW, 1] := C[IS1];
        COST[KSW, 2] := C[IS2];
        COST[KSW, 3] := dist[J1, K2];
700:
        if KSW = 5 then
         goto 709;
        if LISTNO = 0 then
         goto 709;
        NSW := 1;
        N1 := I1;
        N2 := I2;
701:
        LADD := 0;
        for L := 1 to LISTNO do
         begin
          LP := L - LADD;
          if (LIST[L, 1] = N1) and (LIST[L, 2] = N2) then
           goto 702;
          if (LIST[L, 2] = N1) and (LIST[L, 1] = N2) then
           goto 702;
          LIST[LP, 1] := LIST[L, 1];
          LIST[LP, 2] := LIST[L, 2];
          goto 704;
702:
          LADD := 1;
704:
         end;
        LISTNO := LISTNO - LADD;
        if NSW = 1 then
         goto 706;
        if NSW = 2 then
         goto 707;
        if NSW = 3 then
         goto 709;
706:
        NSW := 2;
        N1 := J1;
        N2 := J2;
        goto 701;
707:
        NSW := 3;
        N1 := K1;
        N2 := K2;
        goto 701;
709:
        if KSW = 1 then
         goto 710;
        if KSW = 2 then
```

```
        goto 720;
       If KSW = 3 then
        goto 730;
       If KSW = 4 then
        goto 740;
       If KSW = 5 then
        goto 800;
710:
       AREARR(MAP, ILIST, LIST, COST, IROUTE, I1, J1, I2, K1, J2, K2, KSW, LISTNO, NUMPTS);
       goto 750;
720:
       AREARR(MAP, ILIST, LIST, COST, IROUTE, I1, J2, K1, I2, J1, K2, KSW, LISTNO, NUMPTS);
       goto 750;
730:
       AREARR(MAP, ILIST, LIST, COST, IROUTE, I1, J2, K1, J1, I2, K2, KSW, LISTNO, NUMPTS);
       goto 750;
740:
       AREARR(MAP, ILIST, LIST, COST, IROUTE, I1, K1, J2, I2, J1, K2, KSW, LISTNO, NUMPTS);
750:
       SAVMAX := 0;
       KSW := 5;
       goto 100;
800:
      end;
1000:
    end;
3333:
   If LISTNO > 0 then
    goto 100;
  end;
 begin
9999:
{WRITELN('NPT', NPTT, 'ROUTELENGHT', ROUTELENGHT);}
  for I := 1 to NPTT do
   begin
{WRITELN('X', ROUND(RX[I]), 'Y', ROUND(RY[I]));}
    X[I] := ROUND(RX[I]);
    Y[I] := ROUND(RY[I]);
   end;
  K := 0;
  NPT := NPTT;
  for GGG := 1 to 1 do
   begin
    If GGG > 1 then
     NPT := NPT + 10;
    for zzz := 1 to 1 do
     begin
      xxx := 0;
      KLCAP := 8;
      for K := KLCAP to 8 do
       begin
        If k > KLCAP then
         XXX := 1;
        for I := 1 to 100 do
         begin
          for J := 1 to 100 do
           begin
            dist[I, J] := 0;
           end;
         end;
        aprm := 0;
        TWEIGHT := 0;
        I := 0;
        J := 0;
        P1 := 0;
        P2 := 0;
        P3 := 0;
        P4 := 0;
        PARM := 0;
```

```
        ELAPS1 := 0;
        ELAPS2 := 0;
        ELAPS3 := 0;
        I1 := 0;
        I2 := 0;
        J1 := 0;
        J2 := 0;
        K1 := 0;
        K2 := 0;
        KSW := 0;
        N := 0;
        SCALE := 0;
        II := 0;
        III := 0;
        M := 0;
        MM := 0;
        arcsol := 0;
        minn2 := 0;
        timm2 := 0;
        minn := 0;
        msttime := 0;
  {GENERATE NETWORK}
        ticks2 := TickCount;
        NETGEN(NPT, SCALE, K, dist, X, Y, msttime, xxx, ZZZ);
        Ticks := TickCount;
        ARCOPT(NPT, SCALE, dist, ILIST, MAP, IBEG, LIST, COST, IROUTE, I1, I2, J1, J2, K1, K2, KSW);
        for I := 1 to NPT do
         begin
          if dist[I, ILIST[I, 2]] = 255 then
           WRITELN('***INFEASIBLE LINK**');
          if dist[I, ILIST[I, 2]] = 255 then
           dist[I, ILIST[I, 2]] := ROUND(sqrt((sqr(x[I] - x[ILIST[I, 2]])) + (sqr(Y[I] - Y[ILIST[I, 2]]))));
          TWEIGHT := TWEIGHT + dist[I, ILIST[I, 2]];
{writeln('NODE 1', I, 'NODE 2', ILIST[I, 2], '  DIST  ', dist[I, ILIST[I, 2]]);}
         end;
        elaps1 := ROUND((TickCount - Ticks) / 6.0);
7777:
        WRITELN('ARC', TWEIGHT : 5, ' TIME ', ROUND((ELAPS1 + msttime) / 10) : 5, 'K', K : 4);
        arcsol := tweight;
        ROUTELENGHT := TWEIGHT;
        TWEIGHT := 0;
        MAXTIME := 0;
       end;
     end;
   end;
 end;
end.
```

REFERENCES

1. Bartholdi, J. and Platzman, l., "An O(N log N) Planar Traveling Salesman Heuristic Based on Spacefilling Curves," **Operations Research Letters**,1, 121-125 (1982).

2. Casco, D., Wasil, E., and Golden, B., "Vehicle Routing with Backhauls : Models, Algorithms, and Case Studies," in **Vehicle Routing: Methods and Studies**, Elsevier Science Publishers B.V. (North-Holland), 127-147 (1988)

3. Clarke, G. and Wright, J., "Scheduling of Vehicles from a Central Depot to a Number of Delivery Points", **Operations Research, 12**, 568-581 (1964).

4. Deif, I. and Bodin, L., "Extension of the Clarke and Wright Algorithm for Solving the Vehicle Routing Problem with Backhauling," in **Proceedings of Babson Conference on Software Uses in Transportation and Logistics Management** (A. E. Kidder, editor), Babson Park, MA,75-96 (1984).

5. Gillett, B., and Miller, L., "A Heuristic Algorithm for the Vehicle Dispatch Problem," **Operations Research, 22**, 711-718 (1976).

6. Goetschalckx, M. and Horsley, C., "The Vehicle Routing Problem with Backhauls," **Working Paper**, Material Handling Research Center, Georgia Institute of Technology (1986).

7. Golden, B., Assad, A., Levy, L., and Gheysens, F., "The Fleet Size and Mix Vehicle Routing Problem," **Computers and Operations Research, 11(1)**, 49-66 (1984).

8. Golden, B., Baker, E., Alfaro, J., and Schaffer, J., "The Vehicle Routing Problem with Backhauling: Two Approaches," in **Proceedings of the Twenty-First Annual Meeting of S. E. TIMS** (R. D. Hammesfahr, editor), Myrtle Beach, SC, 90-92 (1985).

9. Golden, B. , Bodin, L. , Assad, A., and Ball,M., "Routing and Scheduling of Vehicles and Crews: The State of the Art," **Computers & Operations Research, 10**, 63-211 (1983).

10. Golden, B., Bodin, L., Assad, A., and Ball, M., "Routing and Scheduling of Vehicles and Crews: The State of the Art," **Computers & Operations Research, 10**, 168-180 (1983).

11. Golden, B., Skiscim, C., "Using Simulated Annealing to Solve Routing and Location Problems," **Naval Research Logistics Quarterly, 33**, 261-279 (1986).

12. Jacobs-Blecha, C. and Goetschalckx, M., " The Vehicle Routing Problem With Backhauls," **European Journal of Operational Research, 42**, 39-51 (1989).

13. Kalantari, B. ,Hill, A., and Arora, S., "An Algorithm for the TSP with Pickup and Delivery Customers ," **European Journal of Operations Research, 22**, 377-386 (1985).

14. Kearney, A. T. ,Inc. ,"Measuring and Improving Productivity in Physical Distribution," **A report prepared for the National Council of Physical Distribution Management, NCPDM**, Oak Brook, IL.(1984).

15. Rosenkrantz, D., Sterns, R.,and Lewis, P., "An Analysis of Several Heuristics for the Traveling Salesman Problem," **SIAM Journal on Computing, 6**, 563-581 (1977).

16. Stewart, W., "Accelerated Branch Exchange Heuristic for Symmetric Traveling Salesman Problems," **Networks, 17**, 423-437 (1987).

17. Stott, K. and Douglas,B., " A Model-Based Decision Support System for Planning and Scheduling Ocean-Borne Transportation," **Interfaces, 11(4)**, 1-10 (1981).

18. Syslo, M., Narsingh, D. ,Kowalik, J., "Discrete Optimization Algorithms with Pascal Programs," **Prentice-Hall**, New Jersey (1983).

19. Yano, C., Chan, T., Richter, L., Cutler, T., Murty, K., and McGettigan, D., "Vehicle Routing at Quality Stores," **Interfaces, 17(2)**, 52-63 (1987).