UNIVERSIDAD AUTÓNOMA DE NUEVO LEÓN
FACULTAD DE INGENIERÍA MECÁNICA Y ELÉCTRICA

OPTIMAL LOCATION OF CAR WRECK ADJUSTERS

POR

LUIS ALBERTO MALTOS ORTEGA

COMO REQUISITO PARCIAL PARA OBTENER EL GRADO DE MAESTRÍA EN CIENCIAS EN INGENIERÍA DE SISTEMAS

JUNIO, 2016

UNIVERSIDAD AUTÓNOMA DE NUEVO LEÓN
FACULTAD DE INGENIERÍA MECÁNICA Y ELÉCTRICA
SUBDIRECCIÓN DE ESTUDIOS DE POSGRADO

OPTIMAL LOCATION OF CAR WRECK ADJUSTERS

POR

LUIS ALBERTO MALTOS ORTEGA

COMO REQUISITO PARCIAL PARA OBTENER EL GRADO DE MAESTRÍA EN CIENCIAS EN INGENIERÍA DE SISTEMAS

JUNIO, 2016

Universidad Autónoma de Nuevo León

Facultad de Ingeniería Mecánica y Eléctrica

Subdireccion de Estudios de Posgrado

Los miembros del Comité de Tesis recomendamos que la Tesis "Optimal Location of Car Wreck Adjusters", realizada por el alumno Luis Alberto Maltos Ortega, con número de matrícula 1390200, sea aceptada para su defensa como requisito parcial para obtener el grado de Maestría en Ciencias en Ingeniería de Sistemas.

El Comité de Tesis

Dr. Roger Z. Ríos Mercado

Director

Dra. María Angélica Salazar Aguilar

Revisor

Dra. María Guadalupe Villareal Marroquín

Revisor

Vo. Bo.

Dr. Simón Martínez Martínez

Subdirector de Estudios de Posgrado

San Nicolás de los Garza, Nuevo León, Junio 2016

To my wife Alejandra

to my children who are the reason to beat.

Table of contents

Acknowledgments xi

Abstract xii

1. Introduction 1

1.1. Problem Statement 1

1.2. Background 2

1.3. Motivation 2

1.4. Objectives 3

1.5. Organization 3

2. Related Work 4

2.1. Mathematical Programming 4

2.1.1. p-Median Models 5

2.1.2. Covering models 5

2.2. Queueing Models 7

2.2.1. Mean Service Calibration 8

2.2.2. Calibration Process 8

2.3. Simulation Models 9

3. Framework **10**

3.1. Problem Statement 10

3.2. Mathematical Framework 11

3.2.1. Model A 12

3.2.2. Model B 14

3.3. Comparison between models 17

4. Heuristic Procedures **18**

4.1. Proposed Metaheuristic 19

4.2. Description of Components 22

4.2.1. A Diversification Generator Method 22

4.2.2. An Improvement Method 24

4.2.3. A Reference Set Update Method 24

4.2.4. A Subset Generation Method 27

4.2.5. A Solution Combination Method 27

5. Computational Experiments **31**

5.1. Assessing the Models 31

5.2. GRASP Evaluation 41

5.3. Path Relinking 42

5.4. Scatter Search 43

6. Conclusions and Future Work **46**

6.1. Main Conclusions 46

6.2. Future Work 47

A. Approximating the Equilibrium Behavior of MSLS **49**

A.1. Motivation 49

A.2. Model Assumptions, Notation, and Terminology 50

A.3. Approximation Procedure 51

B. Berman Heuristic **53**

B.1. The Stochastic Queue p-Median Problem 53

B.2. Mean Service Time Calibration Process 55

Vita **60**

List of figures

4.1. General scheme of scatter search. 19

5.1. Solution times versus ℓ-to-p ratio (for $p = 7, \ldots, 12$). 39

5.2. Solution times versus ℓ-to-p ratio (for $p = 12, \ldots, 24$). 39

5.3. Objective function value versus ℓ-to-p ratio. 40

5.4. Objective value versus α for first greedy function. 41

List of tables

3.1. Model size. 17

5.1. Model A results for instances with $n = 50$. 32

5.2. Model A results for instances with $n = 100$ and $m = 20$. 33

5.3. Model A results for instances with $n = 100$ and $m = 30$. 34

5.4. Model B results for instances with $n = 50$. 35

5.5. Model B results for instances with $n = 100$ and $m = 20$. 36

5.6. Model B results for instances with $n = 100$ and $m = 30$. 37

5.7. Model B results for instances with $n = 100$ and $m = 30$. 38

5.8. Path relinking improvement over multi-start solution. 43

5.9. Scatter Search 44

List of Algorithms

1. Scatter Search – Initial Phase 20
2. Scatter Search – Second Phase 21
3. Reference Set Update Method 25
4. Reference Set Update Diversity Method 26
5. Subset Generation Method 28
6. Path Relinking Combination Method 29

Acknowledgments

I want to specially thank Professor Roger Ríos for giving me the opportunity to work with him and being my advisor, for guiding me in my academic development, and for the effort and support he gave me. I also would like to thank him for his patience. Without his support this work would have not been possible

Special thanks to my thesis committee, Professor Angélica Salazar and Professor Lupita Villareal, for their help, comments and suggestions made throughout this work, and for helping me complete this work

To all Faculty of the Graduate Program in Systems Engineering in Facultad de Ingeniería Mecánica y Eléctrica (FIME) for their courses so complete and enriching.

To FIME and UANL, for their financial support through waivers of tuition and fees, respectively.

To Consejo Nacional de Ciencia y Tecnología (CONACyT) for their financial support through a Graduate Fellowship and the research project grant CB2011-1-166397, that made both my full time studies and my attendance to professional meetings presenting part of this work possible.

Abstract

Luis Alberto Maltos Ortega.

Candidato para obtener el grado de Maestría en Ciencias en Ingeniería de Sistemas.

Universidad Autónoma de Nuevo León.

Facultad de Ingeniería Mecánica y Eléctrica.

Título del estudio: Optimal Location of Car Wreck Adjusters.

Número de páginas: 59.

Problem Description: When a traffic accident occurs in cities with a large traffic flow the roads surrounding the crash site are affected by traffic congestions. In some countries, such as Mexico, even small accidents are troublesome due to the fact that a claim adjuster from the car insurance company must arrive to the site and document the accident before the vehicle may be moved as required by law. Thus, in this particular setting, the location of adjusters becomes a key factor in providing timely service.

Objectives and method of study: The purpose of this thesis is to

- Provide quantitative tools with scientific support for the optimal location of car wreck adjusters.
- Develop adequate mathematical models for representing some of the important

company concerns.

- Design and develop efficient solution techniques for handling real-world instances of the problem.
- Assess the performance of the proposed techniques based on an appropriate experimental design.

CONTRIBUTIONS: The most important contributions of this work are the following:

- Two integer programming models are introduced with the aim of minimizing average response time. The second model takes into account special features to reduce the number of variables.
- A scatter search heuristic was designed, implemented, and tested on a wide set of instances with very good results.

Firma del asesor: 

Dr. Roger Z. Ríos Mercado

Chapter 1

Introduction

The main idea behind this thesis is to develop mathematical models to improve the service offered by car insurance agents. The goal is to determine, given a set of potential location sites, the best location of the adjusters to perform the service, with the aim to help them arrive to accident sites sooner.

1.1 Problem Statement

Car insurance companies, in countries such as Mexico, face the daily issue of where to locate their insurance agents (adjusters) in such a way that they provide the best possible service to their customers. Typically the quality of service is measured by how quickly the adjusters arrive to the places where an incident has occurred.

This problem can be seen as an emergency location system, in which it is desired to "cover" the territory under study (city intersections) by a fixed number of adjusters so as to reduce the average arrival time to accidents. An issue that makes the problem even more complex is the fact that accidents occur randomly and decisions on where to place the adjusters must be made before accidents occur.

1.2 Background

The problem of locating car insurance agents is a practical problem that, to the best of our knowledge, has not been studied before. This comes from the fact that in other countries, when an accident occurs car drivers are allowed to move their cars from the accident site if this obstructs traffic flow. Unfortunately, in many developing countries, insurance agencies ask their insurees not to move the car until an adjuster arrives.

Nonetheless, there are many related location problems that look at similar problems. For example, the case of ambulance location problems, where emergency services (ambulances) must be located in such a way that ambulances arrive promptly at the site of the accident. A main difference between our problem and the one of an emergency service location is that in those problems time is a matter of life or death, and in the case of adjusters' location is not necessarily so. This resulting model looks then at different objective functions and constraints. In Chapter 2, related problems are discussed.

1.3 Motivation

When a car accident occurs, traffic congestion starts to pile up. This is because customers are not allowed to move their vehicles until the adjuster arrives. The adjuster must determine and record the causes of the accident in order to move the car from the accident area and restore the flow. Needless to say, an early arrival of the adjuster means everything to everyone. Customers wait less, and traffic jams are cleared faster when the adjusters arrive promptly.

Given that accidents occur randomly, the idea of this research is to use stochastic location models to derive efficient policies, based on historical data that can be implemented by car insurance companies.

1.4 Objectives

The main objectives of this research are the following:

- Providing quantitative tools with scientific support for optimal location of car wreck adjusters.
- Developing or using adequate mathematical models for representing some of the important company concerns.
- Designing and developing efficient solution techniques for handling real-world instances of the problem.
- Assessing the performance of the proposed techniques based on an appropriate experimental design.

1.5 Organization

This thesis is organized as follows: Chapter 2 presents a brief literature review of diverse approaches of Emergency Service Systems (ESS), starting with the simple deterministic models and ending with the hypercube and simulation models. The problem statement and two proposed models are presented in Chapter 3. Chapter 4 describes the proposed heuristic and its components. Chapter 5 contains the experimenal work, including a description of the instance database and the experiments for assessing the models and the proposed heuristics. Conclusions, contributions, and directions for future research are highlighted in Chapter 6. Additionally, Appendix A contains a summary of how to approximate the hypercube results, and together with Appendix B completes the explanation of how to estimate the response times of a solution.

Chapter 2

Related Work

Work in emergency vehicle based location has generally involved the use of three approaches: mathematical programming, queueing, and simulation. In this chapter, we review the most relevant works in each of these approaches.

2.1 Mathematical Programming

Location models are classified in two main categories, deterministic and probabilistic. Deterministic models are typically used at the planning stage and ignore stochastic considerations regarding the availability of servers or the demand distribution. Probabilistic models reflect the fact that vehicles operate as servers in a queuing system and are not always available to answer a call; these models allow a more accurate planning of Emergency Service Systems (ESSs) at the strategic level. They were initially developed based on the assumptions that servers are independent and do not cooperate, which is not realistic in practice.

The location literature is very extense, therefore we focus the discussion in this chapter on models that are typically used in the location of emergency services. As stated before, no previous work on location of car wreck adjusters exists, to the best of my knowledge, however, there are many common elements between our problem

and the problem of location of emergency services.

2.1.1 p-Median Models

The first approaches incorporate median type objectives. The aim of these problems is to minimize the sum of distances between each demand point and its server. Examples of such models are the deterministic p-median model of Hakimi [14], the stochastic p-median and the vector assignment p-median models of Weaver and Church [29, 30], and the capacitated p-median model of Pirkul and Schilling [22]

The strength of these models is that optimal solution procedures that can accommodate practical problems have been developed. Weaknesses include assumptions such as noncooperation between vehicles, the probability of each system state must be known, and the fraction of call served by the closest, second closest, and so on, must be known for each zone.

2.1.2 Covering models

In the location set covering model (LSCM) introduced by Toregas et al. [28], the aim is to minimize the number of ambulances needed to cover all demand points. This model ignores several aspects of real-life problems, the most important probably being that once an ambulance is dispatched, some demand points are no longer covered. However, the authors provide a lower bound on the number of ambulances required to ensure full coverage.

The maximal covering location problem (MCLP) originally proposed by Church and ReVelle [5] is an alternative approach proposed to overcome some of the shortcomings of the LSCM. In the MCLP the objective is to maximize population coverage subject to limited ambulance availability.

A limitation of the deterministic models is that they assume that servers are available when requested, which is not always true in practical situations. Congestion in emergency services, which may cause the unavailability of servers located within the critical distance when a call is placed, leads to the development of a second generation of location covering models focused on additional coverage.

The definition of probabilistic location models for planning these systems is a natural extension of their deterministic equivalents, the location models with covering constraints. The notion of coverage implies the definition of a service distance (time), which is the critical distance (time) beyond which a demand area is considered not covered. A demand area is therefore considered as covered if it is within a predefined critical distance (say D) from at least one of the existing facilities.

The Maximum Expected Covering Location Problem (MEXCLP) defined by Daskin [6], whose objective is to maximize the expected coverage of all demand areas under consideration, assumes that servers operate independently and that all servers have the same busy probability (workload) ρ, allowing that more than one server be situated in any given location. Daskin and Stern [7] assume that travel times are deterministic and coverage is an "all-or-nothing" property.

ReVelle and Hogan [24] propose two variations of the Maximum Availability Location Problem (MALP), which locates p servers in such a way so as to maximize the population that will find a server available within a given reliability value of α. The first model assumes, such as Daskin, that each server has the same busy probability, and predetermines the number of times a demand point needs to be covered. The second model allows busy fractions to be different in the various sections of the region under consideration (but not for each server to be located.)

These models emphasize the importance of additional coverage for the demand areas, given the possibility that in congested systems the first server, possibly the only server in a particular coverage area, might not be available when requested. Gendreau, Laporte, and Semet [9] propose a model with double coverage, using two

radious values r_1, and r_2 ($r_2 > r_1$), to locate p ambulances, such that all the demand must be covered by an ambulance located within r_2 time units, and, a proportion α of the demand must also be within r_1 units of an ambulance, which may or may not be the same ambulance that covers this customer within r_2 time units. Note that a feasible solution may not exist if the parameters r_1, r_2, and α are too restrictive.

2.2 Queueing Models

The hypercube model (denoted Hypercube) and the hypercube approximation (denoted A-Hypercube), developed by Larson [18, 19], are the most well known queueing approaches. for addressing busy-server type of models. These are not optimization models, but descriptive models that permit the analysis of scenarios. Both models estimate system operating characteristics that are used to evaluate a series of objectives. They can evaluate cooperation between vehicles. Some of their limitations are:

- assumptions of an exponentially distributed service time,
- computational difficulties for problems with many vehicles,
- requirement that service time be only vehicle-dependent rather than call location-dependent.

The computational issues present in the Hypercube are addressed in the A-Hypercube. The vehicle busy probabilities are dealt with by solving a system of nonlinear equations whose size depends on the number of vehicles.

Optimization models for locating Emergency Medical Services (EMS) that use Hypercube or A-Hypercube as a function evaluation subroutine include Jarvis' location-allocation problem [16], Berman and Larson's congested median problem [3], Benveniste's location-allocation problem [1], and Berman, Larson and Parkan's

stochastic queue p-median problem [4]. These methods are heuristic local improvement approaches that assume it is possible to locate a vehicle in every zone.

2.2.1 Mean Service Calibration

Call location-dependent service time can be modeled using the Mean Service Calibration method (denoted MSC). As in Jarvis [16] and Halpern [15] where mean service time is sufficient for obtaining accurate estimates of system performance. The major shortcoming of MSC is that either Hypercube or A-Hypercube is evaluated in each iteration; can thus be a computationally expensive approach.

To eliminate the computational inefficiency of the MSC method, Jarvis [17] developed an approximation model for spatially distributed queueing systems (see Appendix A). The model assumes that call service time is call location-dependent, where all vehicles have the same service rate and utilization while service is exponentially distributed.

2.2.2 Calibration Process

In certain EMSs and other emergency systems, travel times may represent a considerable part of service times. In such cases, it may be advisable to adjust the service times by means of a calibration process, which can be performed using a simple iterative procedure such as the one proposed by Berman, Larson, and Parkan [4]. Basically, the procedure consists of verifying if there are significant differences among the input mean service times and the output mean service times (computed by the hypercube model). In this case, the hypercube is solved using the computed mean service times as inputs, until the differences among input and output values are sufficiently small. This procedure is called a calibration process. Note that this procedure takes into account that the mean travel time depends on the location

of the user and the identity of the server. Empirical experiments show that this procedure usually converges in two or three iterations, for a reasonably accurate estimation of the mean service times, although a formal proof of the convergence of the method is apparently not available in the literature.

2.3 Simulation Models

Simulation models can be formulated with great detail, and have been used for evaluating EMS system performance in numerous papers such as Savas [26], Berlin and Liebman [2], and Swoveland et al. [27]. These simulation approaches provide in general a wealth of output measures. Their main drawback is that they are rarely used because of high run times and data collection costs. These models have some questionable assumptions, but successful applications do exist in the literature.

Chapter 3

Framework

3.1 Problem Statement

The location and dispatching policies used by insurance agencies should aim at arriving to accident areas as early as possible, due to several reasons such as:

1. providing a timely service to their customers,
2. helping clear out the accident area, and
3. keeping the workloads of its adjusters as balanced as possible.

The location policies should be optimized for service time but they should also consider cooperation based on adjuster workload. However, for the car insurance agency that assist in explaning operation criteria current policies are empirical, and do not consider cooperation. By neglecting this, adjusters tend to take more time to arrive at the accident site, making the insurance agency less competitive, and generating more traffic congestion.

The financial costs of applying empirical policies instead of optimum policies is difficult to measure. However, the costs includes more use of fuel, more use of vehicles (that implies more maintenance costs), and the opportunity cost of losing

a customer for low service quality. The use of quantitative models may also help in what-if analysis to assess the overall service rate if more adjusters are placed. The use of mathematical models to determine better location policies based on scenarios, and the use of real data to simulate and evaluate new scenarios versus the current policy are two of the main contributions on this work.

The benefit of this framework is that several policies can be evaluated for different scenarios (for instance high level of congestion in rainy days), and the best policy for each scenario can be determined. In summary, the location of adjusters could be improved with the use of mathematical models and simulation.

The problem studied in this thesis consists may be stated as, given a number of adjusters, a set of potential sites for placing them, and a set of demand points, determine where to place the adjusters, so as to minimize the average response time, assuming that calls arrive with a Poisson distribution and with an own arrival rate for each demand point.

3.2 Mathematical Framework

Two mathematical models are proposed, the first model (model A) was created based on the one proposed by Goldberg et al. [13] for which we made some relaxations to obtain a linear model.

The second model (model B) has additional simplifications, considering that it is unlikely to assign an adjuster to a demand point covered previously by more adjusters, omitting allocation variables, and adding constraints to guarantee the correct order of allocation.

3.2.1 Model A

This model is based on the model proposed by Goldberg et al. [13], and it contains assignment variables for all possible orders.

The following assumptions are considered in the model:

- The probability that an adjuster is busy is ρ and it is independent of the state of the system.
- There is a strict ordering of the basis preferred for each zone that does not depend on the current state of the system.
- All calls are answered by an adjuster originating from its base, not in route back to the base.
- The arrival of calls to the system follows a stationary distribution.
- The model is presented using a 0-queue assumption.

Sets and indexes:

- n number of demand points
- m number of potential sites to locate adjusters/facilities
- p number of available adjusters
- i index for demand points; $i \in V = \{1, 2, \ldots, n\}$
- j index for potential sites for adjusters/facilities $j \in W = \{1, 2, \ldots, m\}$
- k index for possible order; $k \in K = \{1, 2 \ldots, p\}$
- $S_{ij} = \{r \in W |$ site r is preferred by proximity before site j for demand point $i\}$

Parameters:

- λ_i arrival rate of calls from demand point i
- ρ is the utilization of each adjuster, the value is between 0 and 1, where 0 means that the server is always idle. To obtain an approximate value of ρ we use the formula proposed by Berman and Larson [3]

$$\rho = \frac{\sum_{i=0}^{n} \lambda_i}{mp} \tag{3.1}$$

- t_{ij} is the expected travel time between demand point i and potential site j.
- h_{ij}^k is the probability that adjuster j serves point i given that it is the k-th preferred. It is calculated using the following formula:

$$h_{ij}^k = (1-\rho)\rho^{k-1} \tag{3.2}$$

Variables:

- $x_j = \begin{cases} 1 & \text{if an adjuster is placed at potential site } j \\ 0 & \text{otherwise.} \end{cases}$
- $y_{ij}^k = \begin{cases} 1 & \text{if the adjuster at site } j \text{ is the } k\text{-th to cover demand point } i \\ 0 & \text{otherwise.} \end{cases}$

Model A:

- Objetive – Minimize the average expected response time:

$$\text{mín} \sum_{j=1}^{m} \sum_{k=1}^{p} \sum_{i=1}^{n} h_{ij}^k t_{ij} y_{ij}^k \tag{3.3}$$

- Subject to location of only p adjusters:

$$\sum_{j \in W} x_j = p \tag{3.4}$$

- Each demand point i is covered by an adjuster on each order k:

$$\sum_{j \in W} y_{ij}^k = 1 \qquad i \in V, k \in K \tag{3.5}$$

- Relationship between variables x and y:

$$y_{ij}^k \leq x_j \qquad i \in V, j \in W, k \in K \tag{3.6}$$

- For each located adjuster, there can only be a maximum of one ordered assignment:

$$\sum_{k=1}^{p} y_{ij}^k \leq x_j \qquad i \in V, j \in W \tag{3.7}$$

- Assign j to cover i in order k only if the assignment of order $k-1$ was made for some $r \in S_{ij}$:

$$y_{ij}^k \leq \sum_{r \in S_{ij}} y_{ir}^{k-1} \qquad i \in V, j \in W, k \in K \setminus \{1\} \tag{3.8}$$

- Nature of decision variables:

$$x_j \in \{0,1\} \qquad j \in W$$
$$y_{ij}^k \in \{0,1\} \qquad i \in V, j \in W, k \in K$$

Observe that we do not need to add a constraint to ensure the counterpart of (3.7) because (3.4) and (3.6) ensure that each adjuster must cover each demand point for some order, therefore if an adjuster located at j does not cover demand point i at order k (indicated by the maximum covering order in S_{ij}) there will be at least one adjuster that does not cover demand point i at any order resulting in an infeasible solution.

3.2.2 MODEL B

This model was developed with the idea that it is unlikely that the farthest adjusters serve demand points on cases where the system does not become congested.

In these cases we can make the assumption that the probability of being served by the ℓ-th adjuster is almost zero, where ℓ is large enough but less than p.

Parameters:

- M is a large integer
- ℓ is the number of allowed adjusters per demand point
- a_{ik} contains the k-th preferred location server regarding the point i.

Variables:

- z_j represents the number of adjusters placed at site j
- $y_{ij}^k = \begin{cases} 1 & \text{if the adjuster at site } j \text{ is the } k\text{-th to cover demand point } i \\ 0 & \text{otherwise.} \end{cases}$

The objective, and constraints (3.4)-(3.7) are practically the same as in model A, with the difference that binary variables x_j from Model A are replaced by integer variables z_j inspired by the results of Berman, Larson, and Parkan [4], and the addition of the following binary variables

- $u_{ij} = \begin{cases} 1 & \text{if the number of adjusters between } i \text{ and } j, \text{ inclusive, is less than } \ell \\ 0 & \text{otherwise.} \end{cases}$
- $v_{ij} = \begin{cases} 1 & \text{if the number of adjusters between } i \text{ and } j, \text{ is less than } \ell - 1 \\ 0 & \text{otherwise.} \end{cases}$

Model B:

- Objective – Minimize the average expected response time:

$$\text{mín} \sum_{j=1}^{m} \sum_{k=1}^{\ell} \sum_{i=1}^{n} h_{ij}^k t_{ij} y_{ij}^k \tag{3.9}$$

- Subject to the location of only p adjusters:

$$\sum_{j \in W} z_j = p \tag{3.10}$$

- Each demand point i is covered by an adjuster in each order from 1 up to ℓ:

$$\sum_{j \in W} y_{ij}^k = 1 \qquad\qquad i \in V, k \in \{1, \ldots, \ell\} \tag{3.11}$$

- Relationship between variables z and y:

$$y_{ij}^k \leq z_j \qquad\qquad i \in V, j \in W, k \in \{1, \ldots, \ell\} \tag{3.12}$$

- These two constraints set the relationship between the z and u variables. If $u = 1$, constraints (3.14) become redundant, and contraints (3.13) guarantee that the number of adjusters between i and j is less or equal than ℓ; otherwise if $u = 0$, constraints (3.13) become redundant, and constraints (3.14) guarantee that the number of adjusters between i and j is more than ℓ.

$$\sum_{r \in S_{ij} \cup \{j\}} z_r + (p - \ell) u_{ij} \leq p \qquad\qquad i \in V, j \in W \tag{3.13}$$

$$\sum_{r \in S_{ij} \cup \{j\}} z_r + M u_{ij} \geq \ell + 1 \qquad\qquad i \in V, j \in W \tag{3.14}$$

- Assign z_j times j to i if $u_{ij} = 1$; otherwise it becomes redundant:

$$\sum_{k=1}^{\ell} y_{ij}^k + M(1 - u_{ij}) \geq z_j \qquad\qquad i \in V, j \in W \tag{3.15}$$

- Similar to constraints (3.13)-(3.14) these constraints set the relationship between the z and v variables.

$$\sum_{r \in S_{ij}} z_r + (p - (\ell - 1)) v_{ij} \leq p \qquad\qquad i \in V, j \in W \tag{3.16}$$

$$\sum_{r \in S_{ij}} z_r + M v_{ij} \geq \ell \qquad\qquad i \in V, j \in W \tag{3.17}$$

- Assign j to i the remaining times to complete ℓ assignments:

$$\sum_{k=1}^{\ell} y_{ij}^k + M(1 - v_{ij} + u_{ij}) \geq \ell - \sum_{r \in Sij} z_r \qquad i \in V, j \in W \tag{3.18}$$

$$\sum_{k=1}^{\ell} y_{ij}^k - M(1 - v_{ij} + u_{ij}) \leq \ell - \sum_{r \in S_{ij}} z_r \qquad i \in V, j \in W \tag{3.19}$$

- Assign j to i only if j is one of the first ℓ adjusters near i:

$$y_{ij}^k \leq u_{ij} + v_{ij} \qquad i \in V, j \in W \tag{3.20}$$

- Nature of decision variables:

$$z_j \in \{0, 1, \ldots, p\} \qquad j \in V$$

$$y_{ij}^k \in \{0, 1\} \qquad i \in V, j \in W, k \in I$$

$$u_{ij}, v_{ij} \in \{0, 1\} \qquad i \in V, j \in W$$

3.3 Comparison between models

As can be seen, model A has a considerable amount of binary variables. This is why model B was introduced; however, due the lack of allocation variables we needed to add more variables and constraints to ensure a similar behavior.

The size of models as function of n, m, p, and ℓ as shown in Table 3.1.

Table 3.1: Model size.

	Model A	Model B
variables	$m(np + 1)$	$m(n(\ell + 2) + 1)$
constraints	$n(2mp + p) + 1$	$n((l + 8)m + 1) + 1$

Since $\ell < p - 2$, it can be seen that model B has fewer binary variables. When $\ell < 2p - 8$ model B has less constraints than model A.

Chapter 4

Heuristic Procedures

The problem addressed in this thesis has several stochastic properties, limiting the applicability of linear models to specific cases where certain assumptions are required. As it was assumed in the previous chapter, each adjuster has the same busy probability. In other words, we deal with a congested system. Congestion happens when a service center (adjuster) is not able to deal, simultaneously, with all the service requests that are made to it. The traditional models that deal with congestion include a capacity constraint, which forces the demand for service, normally constant in time and equal to an average, to be smaller than the maximum capacity of the center all the time. This is a deterministic approach to the problem, not considering the dynamic nature of the congestion. Depending on how the capacity constraint is developed, this means that the solution model produces idle servers, or results in a system that is not able to deal with all the demand [20, 21].

Given the limitation of integer programming models, the use of heuristic methods to solve this type of problems is a very powerful technique. In this chapter, a metaheuristic based on scatter search is proposed for tackling the problem studied in this thesis. Each component is described in detail.

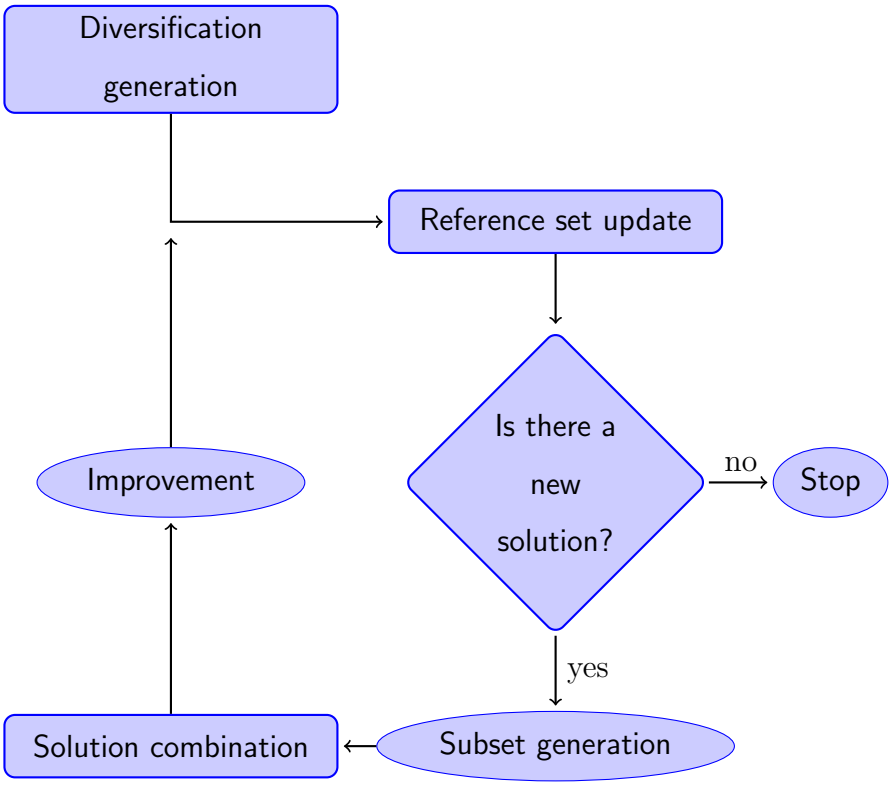

Figure 4.1: General scheme of scatter search.

4.1 Proposed Metaheuristic

Scatter Search (SS) is a population-based metaheuristic whose original ideas are due to Glover [10] as a metaheuristic for integer programming. It is based on diversifying the search through the solution space. It operates on a set of solutions, named the Reference Set, formed by good and diverse solutions of the main population. These solutions are combined with the aim of generating new solutions with better fitness, while maintaining diversity. The general scheme of a SS method is depicted in Figure 4.1.

More specifically, a scatter search metaheuristic with a dynamic reference set and path relinking as a combination method is proposed and implemented. We opted for a dynamic reference set because the number of solutions generated in each combination is very large. To enhance solution quality, two improvement methods

are included, one is based on an idea proposed by Berman, Larson, and Parkan [4], and the other is a local search method proposed in this research.

Algorithm 1 Scatter Search – Initial Phase

Input: An instance of the problem; $MaxIter$:= maximum number of iterations; $MaxIterUnchanged$:= maximum number of iterations with no change in the reference set

Output: $RefSet$:= A reference set of cardinality $2b$

```
procedure SS-PHASE1
    Iteration ← 0
    IterationsUnchanged ← 0
    RefSet ← ∅
    repeat
        Sols ← DiversificationGenerator()
        for S ∈ Sols do
            S ← Improvement(S)          ▷ either of the two methods proposed
        end for
        RefSet.Update(Sols)
        Iteration ← Iteration + 1
        IterationsUnchanged ← IterationsUnchanged + 1
        if HasNewSolutions(RefSet) then
            IterationsUnchanged ← 0
        end if
    until IterationsUnchanged ≥ MaxIterUnchanged or Iteration ≥ MaxIter
    return RefSet
end procedure
```

Our proposed method is comprised of two phases. In the first phase, starting from scratch, an initial population is generated and stored in the Reference Set. This phase is depicted in Algorithm 1. The cardinality of the reference set is $2b$ (for a given b), consisting of the best b solutions by objective function value and the b most diverse solutions. To measure the "distance" between any two solutions, we used the

value of the minimum perfect matching between the corresponding center sets of the two solutions. Clearly, a high value of this minimum perfect matching among centers indicates the solutions are far apart.

Once the initial reference set has been obtained, the second phase consists of the actual application of the SS methodology as depicted in Algorithm 2. Each pair of solutions is combined using a path relinking procedure. Because the Reference Set is dynamic some pairs are not combined. This is due the fact that some solutions are removed from the set before trying to combine them.

Algorithm 2 Scatter Search – Second Phase

Input: $RefSet$:= A reference set of cardinality $2b$

Output: A feasible solution for the problem

```
procedure SS-Phase2(RefSet)
    repeat
        G_X ← SubsetGenerator(RefSet)
        for all X ∈ G_X do
            if X ⊂ RefSet then      ▷ Some element of X may have been removed
                C_X = Combination(X)
                for all Sol ∈ C_X do
                    Sol ← Improvement(Sol)
                end for
                RefSet.Update(C_X)
            end if
        end for
    until NoNewSolutions(RefSet)
    return RefSet.TopSolution()
end procedure
```

In the following subsection, each componens of the SS is described in detail.

4.2 Description of Components

The components of the scatter search metaheuristic [11] consists of the following problem-specific methods:

- A Diversification Generator method
- An Improvement method
- A Solution Combination method
- A Reference Set Update method
- A Subset Generation method

We described below each of the components.

4.2.1 A Diversification Generator Method

This component is used to generate a collection of diverse trial solutions. In this research we develop a Greedy Randomized Adaptive Search Procedure (GRASP) as a diversification generator. GRASP [8] is a multi-start procedure that combines the power of greedy heuristics and randomization. GRASP typically consists of two phases: a greedy randomized construction phase and an improvement phase. In this case, our proposed GRASP consists only of the construction phase because the improvement phase is applied in other parts of the scatter search algorithm.

We chose GRASP because in the proposed problem any allocation of sites for adjusters is a feasible solution. Thus it was decided to give some intelligence to this simple allocation instead of choosing random points. In a given iteration of GRASP, we have a partial solution X (set of adjuster location sites) where some adjusters have already been located. The algorithm has now to decide where to place the

next adjuster. To this end, a greedy function is evaluated for each potential site, and, following the GRASP philosophy, a restricted candidate list (RCL) is formed with all the candidate sites whose greedy function value falls within $\alpha\,\%$ of the best possible value. The parameter α is known as the GRASP quality function threshold parameter. Then an element from the RCL is randomly chosen and the adjuster is located at this site. The procedure iteratively proceeds until p adjusters have been located.

Three greedy functions were evaluated for the constructive algorithm The first function, was the p-means metric consisting on the sum of the distances between each demand point and its nearest adjuster.

$$f_1(r) = \sum_{j \in X} \sum_{i \in R_1(j, X \cup \{r\})} t_{ij} + \sum_{i \in R_1(r, X)} t_{ir} \tag{4.1}$$

where $R_k(j \in W, Y \subset W) = \{i \in V \mid |S_{ij} \cap Y| = k-1\}$, and can be described as the subset of demand points for which the server in site j, would be the k-th in subset Y.

For the second function, we compute the distances from each demand point to their ℓ nearest adjusters, weighted by the *idle* probability of being allocated in the k-th position.

$$f_2(r) = \sum_{k=1}^{\ell} \sum_{j \in X} \sum_{i \in R_k(j, X \cup \{r\})} h_{ij}^k t_{ij} + \sum_{k=1}^{\ell} \sum_{i \in R_k(r, X)} h_{ir}^k t_{ir} \tag{4.2}$$

The third function uses the Mean Service Time (MST) calibration method proposed by Jarvis [17], to obtain more accuracy values of the current mean response time. This is done by iterativelys electing each potential site, placing a facility there and then running the MST calibration algorithm (described in Appendix B) for computing the MST of the given solution. We do this for each possible value and compute the greedy function from the found values.

$$\begin{aligned} f_3(r) = {} & \text{value of MST after the calibration method has been applied} \\ & \text{to solution } X \cup \{r\}. \end{aligned} \tag{4.3}$$

4.2.2 An Improvement Method

The main role of the improvement method is to attempt to improve the quality of a given feasible solution. Two improvement methods are developed. The first is an adaptation of the method proposed by Berman, Larson, and Parkan [4] (see Appendix B for mode details) and the second is a local search procedure described below.

Proposed Local Search: The move considered is to relocate an agent from its current position to a different position. The entire neighborhood consist of all possible moves that can take place from a given solution. However, due to the high computational cost of assessing the entire neighborhood, a reduced neighborhood is considered. Therefore, instead of considering all possible moves, the procedure focuses on relocating the adjuster with the smallest workload to a place around or near the adjuster with the largest workload.

4.2.3 A Reference Set Update Method

A two-tier Reference Set is designed to maintain both a pool of good quality solutions and a pool of diverse solutions. Each tier has size b, where b is typically between 20 and 40. In addition, since the number of generated solutions is relatively large, the Reference Set is dynamically updated.

The Reference Set is updated by trying to incorporate new solutions generated by the combination method. Because the Reference Set is divided in two parts, a new trial solution is first tested by the quality criterion. If this trial solution is better than any of the solutions in the quality tier subset of the Reference Set, the new solution enters the Reference Set and the worst solution is removed from it. These steps are shown in Algorithm 3.

If the new solution does not enter the Reference Set by quality or if a solution

Algorithm 3 Reference Set Update Method

Input: *SolutionSet*: set of solutions to be tried for the Reference Set.

```
procedure RefSet.Update(SolutionSet)
    Bin ← PublicBin()
    for Sol ∈ SolutionSet do
        if Sol.Quality > RefSet.LowestQuality then
            DegradedSol ← RefSet.LowestQualitySol()
            RefSet.Remove(DegragedSol)
            RefSet.InsertByQuality(Sol)
            Bin.Add(DegradedSol)
        else
            Bin.Add(Sol)
        end if
    end for
    RefSet.UpdateDiversity( )
end procedure
```

is removed from the quality-tier subset, then this solution is tested based on the diversity criterion. If this trial solution has a better diversity value than any of the solutions in the diversity-tier subset, it enters the subset and the worst solution (from the diversity standpoint) is removed from the subset. Note that every time the quality-tier subset is re-evaluated since the value of the diversity-based solutions depends on the quality-set subset. This is shown in Algorithm 4.

Algorithm 4 Reference Set Update Diversity Method

```
procedure RefSet.UpdateDiversity( )
    Bin ← PublicBin()
    Bin ← Bin ∪ RefSet.DiversitySols()
    RefSet.Remove(RefSet.DiversitySols())
    for Sol ∈ Bin do
        Sol.DiversityValue ← MinCostPerfectMatching(RefSet, Sol)
    end for
    repeat
        DiverseSol ← MostDiverse(Bin)
        Bin.Remove(DiverseSol)
        RefSet.InsertByDiversity(DiverseSol)
        for Sol ∈ Bin do
            Sol.UpdateDiversityValue(DiverseSol)
        end for
    until Count(RefSet.DiversitySols()) = b or Count(Bin) = 0
end procedure
```

4.2.4 A Subset Generation Method

The subset generation method operates on the Reference Set and its main role is to create subsets of solutions to be combined later by the Combination method. In this particular case, we only consider subsets of size two, that is, pairs of solutions, because our Reference Set is dynamic. To identify the new solutions, we label the solutions as new and old at the start of each iteration. We combine first the new solutions between them, next the new solutions with the old. The new solutions that enter the Reference Set as a combined solution are not part of the generated subsets until the next iteration. Each solution generated that did not enter the Reference Set, displaced by a better solution, or actually a member of the diverse part of the Refence Set, is evaluated for membership in the Reference Set as a diverse solution.

4.2.5 A Solution Combination Method

The main goal of the combination method, in general, is to transform a given subset of solutions produced by the subset generation method into one or more combined solutions. In this particular case, the proposed combination method takes two differently solutions, say X^1 and X^2 as input and apply a path relinking procedure between these two solutions to generate a path of different solutions between X^1 and X^2. Path relinking [12] has been very successful in many combinatorial optimization problems including some location related problems such as the p-median problem [23] and commercial districting [25], to name a few.

Since each solution is in fact driven or represented by the location of the adjusters (or servers), to do the path relinking between any two solutions we must first find a match between servers form one solution to servers of the other solution. According to these, we test the following three different strategies for finding this matching.

Algorithm 5 Subset Generation Method

Input: *RefSet*: Reference set for chosing the solutions to be combined.

Output: *Set*: Set of the solutions to be combined.

```
procedure SubsetGeneration(RefSet)
    NewSols ← RefSet.SolutionsSince(NowTime)
    for all (Sol_x, Sol_y) ∈ NewSols do
        if Sol_x ∈ RefSet and Sol_y ∈ RefSet then
            CombinedSols ← Combination(Sol_x, Sol_y)
            RefSet.Update(CombinedSols)
        end if
    end for
    RefSet.UpdateDiversity()
    if RefSet.NumberOfOldSols(NowTime) > 0 then
        OldSols ← RefSet.SolutionsUntil(NowTime)
        for all Sol_x ∈ NewSols do
            for all Sol_y ∈ OldSols do
                if Sol_x ∈ RefSet and Sol_y ∈ RefSet then
                    CombinedSols ← Combination(Sol_x, Sol_y)
                    RefSet.Update(CombinedSols)
                end if
            end for
            RefSet.UpdateDiversity()
        end for
    end if
    return RefSet
end procedure
```

- Perfect matching: Pair the servers by the corresponding perfect matching problem, that is, by minimizing the distance between paired servers.
- Workload matching: The servers in each solution are sorted according to their workloads (from highest to lowest), and then match them according to these sorted lists (e.g., server with highest workload in solution 1 with server with highest load in solution 2 and so on).
- Random matching: Pair the serveres in a totally random fashion.

Algorithm 6 Path Relinking Combination Method

Input: Sol_x, Sol_y: A pair of solutions to be combined.

Output: *CombinedSols*: A trajectory of solutions generated in the path relinking between Sol_x and Sol_y.

```
procedure COMBINATION(Sol_x, Sol_y)
    CombinedSols ← EmptyList()
    match ← Matching(Sol_x, Sol_y)                ▷ perfect,workload,random
    order ← ProcessOrder(Sol_x, match, Sol_y)     ▷ nearest/farthest first,random
    for i ← 1, p do
        j ← order[i]
        if Sol_x.ServerLocation(j)! = Sol_y.ServerLocation(match[j]) then
            Sol_x.SetServerLocation(j, Sol_y.ServerLocation(match[j]))
            CombinedSols.insert(Sol_x)
        end if
    end for
    return CombinedSols
end procedure
```

Once the server matching is found, we proceed to generate the path from one solution to the other by exchanging one server at a time. Each move generates one new solution in the path. There are different criteria for choosing the order of the exchanges. The following three strategies are tested.

- Nearest First: Perform the exchange from nearest to farthest, that is, start exchanging the servers whose paired distance is the lowest.
- Farthest First: Perform the exchange from farthest to nearest, that is, exchanging the servers whose paired distance is the highest.
- Random: Perform the exchange in random order of servers.

The combination method is depicted in in Algorithm 6.

Chapter 5

Computational Experiments

In this chapter the computational experimentation is described. The experiments are carried out to validate the mathematical models, fine-tune the algorithmic parameters of the scatter search, and assess the individual components of the proposed heuristic.

These tests were evaluated in a Lanix Spine BW Processor Intel Xenon, CPU E5-2867W at 3.10 GHz under Ubuntu 14.04.3 LTS operating system. The models were solved with CPLEX 12.6 callable libraries, coded in C++. The g++ compiler version 7.2 from GNU was used.

5.1 Assessing the Models

Different random instances were generated to validate the proposed formulations. In each of the experiments, the size of the instances testes is specified. Unless otherwise noted, in all the experiments we use $\mu = 72$ and $\lambda = 0.006$.

The results when solving model A for the instances with $n = 50$ are shown in Table 5.1. The first three columns show the size of the instance solved in terms of n, m, and p. The last two columns indicate the results obtained. Time is shown in seconds. The relative optimality gap is 0.0 when an optimal solution was found,

Table 5.1: Model A results for instances with $n = 50$.

n	m	p	time	gap (%)
50	10	5	2.0	0.0
50	10	6	2.9	0.0
50	10	7	1.5	0.0
50	10	8	0.8	0.0
50	20	7	62.2	0.0
50	20	8	113.7	0.0
50	20	9	140.2	0.0
50	20	10	187.4	0.0
50	20	11	209.4	0.0
50	20	12	191.8	0.0
50	20	13	109.2	0.0
50	20	14	28.3	0.0
50	20	15	54.1	0.0
50	30	9	1200.0	2.6
50	30	11	1200.0	1.9

and strictly greater than 0.0 when the algorithm stops by time limit. As we can see, all instances with $m < 20$ were optimally solved in less than four minutes. The two instances with $m = 30$ were not solved in the alloted time of 20 minutes.

The results when solving model A for instances with $n = 100$ are shown in Tables 5.2 and 5.3. As we can see, all instances with $m = 20$ were optimally solved in less than thirteen minutes. However, once we jump to $m = 30$ the problem becomes more difficult finding optimal solutions in only two out of eighteen instances.

We also attempted to solve 10 instances with $n = 150$ and $m = 30$ (for $p = 5, 7, 9, \ldots, 23$) but no optimal solutions were found in a time limit of 20 minutes. The worst relative optimality gap was 5.1 %.

Table 5.2: Model A results for instances with $n = 100$ and $m = 20$.

n	m	p	time	gap (%)
100	20	5	51.0	0.0
100	20	6	150.4	0.0
100	20	7	450.2	0.0
100	20	8	503.4	0.0
100	20	9	623.7	0.0
100	20	10	861.6	0.0
100	20	11	744.6	0.0
100	20	12	620.0	0.0
100	20	13	559.3	0.0
100	20	14	393.6	0.0
100	20	15	287.7	0.0
100	20	16	161.9	0.0
100	20	17	21.7	0.0
100	20	18	7.2	0.0

Table 5.3: Model A results for instances with $n = 100$ and $m = 30$.

n	m	p	time	gap (%)
100	30	7	3600.0	1.0
100	30	8	3600.0	1.8
100	30	9	3600.0	1.6
100	30	10	3600.0	1.9
100	30	11	3600.0	1.3
100	30	12	3600.0	1.3
100	30	13	3600.0	1.1
100	30	14	3600.0	0.8
100	30	15	3600.0	1.8
100	30	16	3600.0	1.1
100	30	17	3600.0	0.7
100	30	18	3600.0	0.5
100	30	19	3600.0	0.4
100	30	20	3600.0	0.2
100	30	21	3600.0	0.3
100	30	22	3600.0	0.2
100	30	23	1443.7	0.0
100	30	24	744.4	0.0

In the following experiments we solve Model B under different combinations of n, m, p, and ℓ varying from 1 to p.

Table 5.4: Model B results for instances with $n = 50$.

n	m	p	ℓ	NI	OPT	AvGap (%)	AvTime
50	20	7	$1, \ldots, 7$	7	7	0.0	993.0
50	20	8	$1, \ldots, 8$	8	8	0.0	1124.1
50	20	9	$1, \ldots, 9$	9	8	0.2	1552.9
50	20	10	$1, \ldots, 10$	10	7	0.1	2187.0
50	20	11	$1, \ldots, 11$	11	6	0.1	2166.5
50	20	12	$1, \ldots, 12$	12	7	0.5	2137.5
50	20	13	$1, \ldots, 13$	13	6	1.2	2360.3
50	20	14	$1, \ldots, 14$	14	6	4.5	2428.1
50	20	15	$1, \ldots, 15$	15	6	3.4	2486.9
50	30	9	$1, \ldots, 9$	9	2	1.8	2975.1
50	30	10	$1, \ldots, 10$	10	2	6.1	3012.7

The average results when solving model B for the instances with $n = 50$ are shown in Table 5.4. In each row, the first three columns show the size of the instance solved in terms of n, m, and p. The fourth column represents the different values of ℓ tried which is consistent with the fifth column indicating the total number of instances solved per row. The last three columns indicate the results obtained in terms of total number of instances solved optimally, the average relative optimality gap and average running time for that specific group of instances. The time limit was set to 3600.0 seconds (1 hour). It was observed that the model was easier to solve for small values of ℓ. As the rate ℓ/p gets closer to 1, the model becomes more difficult. Overall, 65 out of 118 instances were optimally solved. The average relative gaps are relatively good, finding near optimal solutions in almost all instances tested.

The results when solving model B for the instances with $n = 100$ are shown in Tables 5.5 and 5.6. As we can see, all instances with $m = 20$ were optimally solved in less than 9.0 second on average. However, once we jump to $m = 30$ the problem

becomes more difficult. Nevertheless 259 optimal solutions were found out of 304 instances. The non-optimal solutions observed relatively low optimality gaps.

Table 5.5: Model B results for instances with $n = 100$ and $m = 20$.

n	m	p	ℓ	NI	OPT	AvGap (%)	AvTime
100	20	5	$1, \ldots, 5$	5	5	0.0	3.7
100	20	6	$1, \ldots, 6$	6	6	0.0	3.8
100	20	7	$1, \ldots, 7$	7	7	0.0	3.9
100	20	8	$1, \ldots, 8$	8	8	0.0	3.7
100	20	9	$1, \ldots, 9$	9	9	0.0	4.5
100	20	10	$1, \ldots, 10$	10	10	0.0	5.1
100	20	11	$1, \ldots, 11$	11	11	0.0	5.3
100	20	12	$1, \ldots, 12$	12	12	0.0	5.6
100	20	13	$1, \ldots, 13$	13	13	0.0	5.8
100	20	14	$1, \ldots, 14$	14	14	0.0	6.6
100	20	15	$1, \ldots, 15$	15	15	0.0	7.0
100	20	16	$1, \ldots, 16$	16	16	0.0	7.7
100	20	17	$1, \ldots, 17$	17	17	0.0	8.2
100	20	18	$1, \ldots, 18$	18	18	0.0	9.0

Table 5.6: Model B results for instances with $n = 100$ and $m = 30$.

n	m	p	ℓ	NI	OPT	AvGap (%)	AvTime
100	30	7	$1, \ldots, 7$	7	7	0.0	6.8
100	30	8	$1, \ldots, 8$	8	8	0.0	6.8
100	30	9	$1, \ldots, 9$	9	9	0.0	8.3
100	30	10	$1, \ldots, 10$	10	10	0.0	8.9
100	30	11	$1, \ldots, 11$	11	11	0.0	9.1
100	30	12	$1, \ldots, 12$	12	12	0.0	10.7
100	30	13	$1, \ldots, 13$	13	13	0.0	10.8
100	30	14	$1, \ldots, 14$	14	14	0.0	12.1
100	30	15	$1, \ldots, 15$	15	15	0.0	13.0
100	30	16	$1, \ldots, 16$	16	16	0.0	13.6
100	30	17	$1, \ldots, 17$	17	17	0.0	14.9
100	30	18	$1, \ldots, 18$	18	18	0.0	52.9
100	30	19	$1, \ldots, 19$	19	17	0.3	628.4
100	30	20	$1, \ldots, 20$	20	16	0.4	733.6
100	30	21	$1, \ldots, 21$	21	16	0.6	1032.8
100	30	22	$1, \ldots, 22$	22	15	0.7	1155.9
100	30	23	$1, \ldots, 23$	23	16	0.7	1291.6
100	30	24	$1, \ldots, 24$	24	14	0.9	1607.2
100	30	25	$1, \ldots, 25$	25	15	1.1	1580.8

The results when solving model B for the instances with $n = 150$ are shown in Table 5.7. Time limit was set to 1200.0 seconds. As we can see, these instances were even harder to solve as only a very few were optimally solved.

Table 5.7: Model B results for instances with $n = 100$ and $m = 30$.

n	m	p	ℓ	NI	OPT	AvGap (%)	AvTime
150	30	7	$1, 3, 5, 7$	4	0	12.4	1200.0
150	30	8	$1, 3, 5, 7$	4	0	24.4	1200.0
150	30	9	$1, 3, \ldots, p$	5	0	21.6	1200.0
150	30	10	$1, 3, \ldots, p$	5	0	23.5	1200.0
150	30	11	$1, 3, \ldots, p$	6	1	36.6	1181.1
150	30	12	$1, 3, \ldots, p$	6	1	41.3	1108.6
150	30	13	$1, 3, \ldots, p$	7	1	46.7	1164.2
150	30	14	$1, 3, \ldots, p$	7	1	49.2	1045.3
150	30	15	$1, 3, \ldots, p$	8	1	54.8	1072.9
150	30	16	$1, 3, \ldots, p$	8	1	56.3	1054.7

In our next experiment, an assessment of both solution quality and running time as a function of the ratio ℓ/p is carried out. To this end, we select the instance with $n = 100$ and $m = 30$ and solve model B for different values of $p = 7, 8, \ldots, 24$. For each fixed value of p, we solved p instances by fixing $\ell = 1, \ldots, p$.

Figures 5.1 and 5.2 displays the results in terms of the running times. The first figure correspond to value of $p = 7, \ldots, 12$ and the second corresponds to the remaining values of p. In this second figure, note that the right-hand side plot is a larger zoom of the left-hand side plot by scaling the vertical axis. Figure 5.3 displays the results in terms of the objective function value.

As can bee seen, when the value of ℓ is close to p model resolution tends to take longer and in some cases optimality is not achieved. However, it can also be seen that optimal solutions do not change much when the ratio is between 0.2 and 0.6.

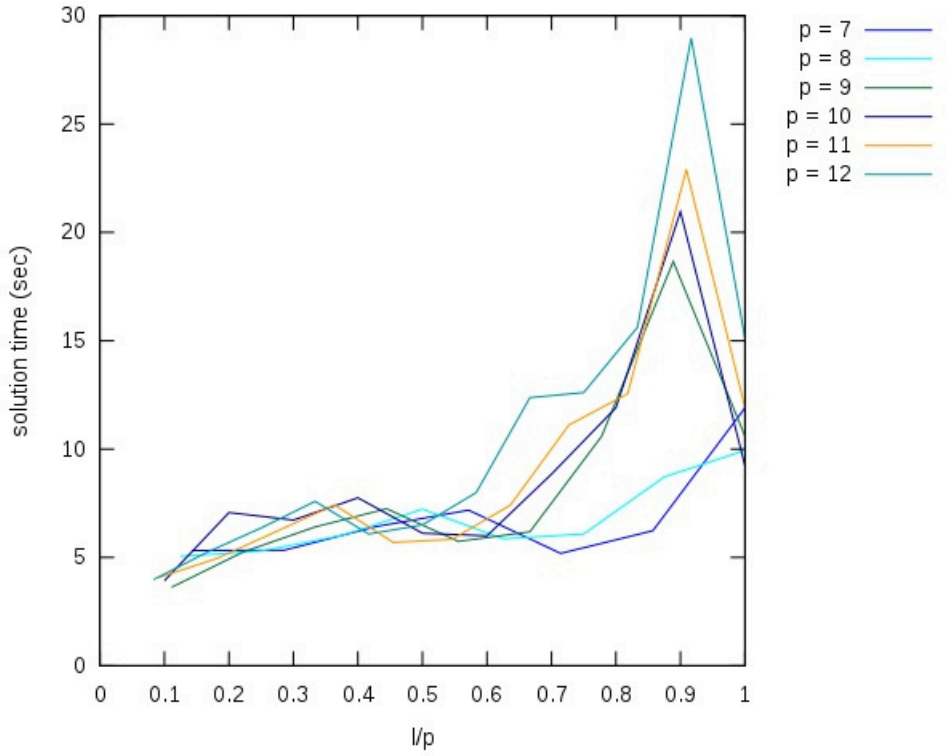

Figure 5.1: Solution times versus ℓ-to-p ratio (for $p = 7, \ldots, 12$).

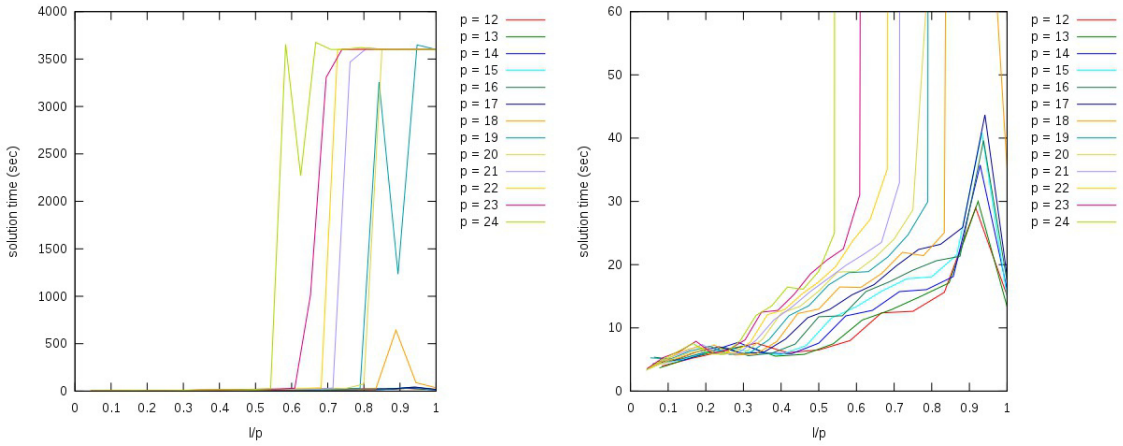

Figure 5.2: Solution times versus ℓ-to-p ratio (for $p = 12, \ldots, 24$).

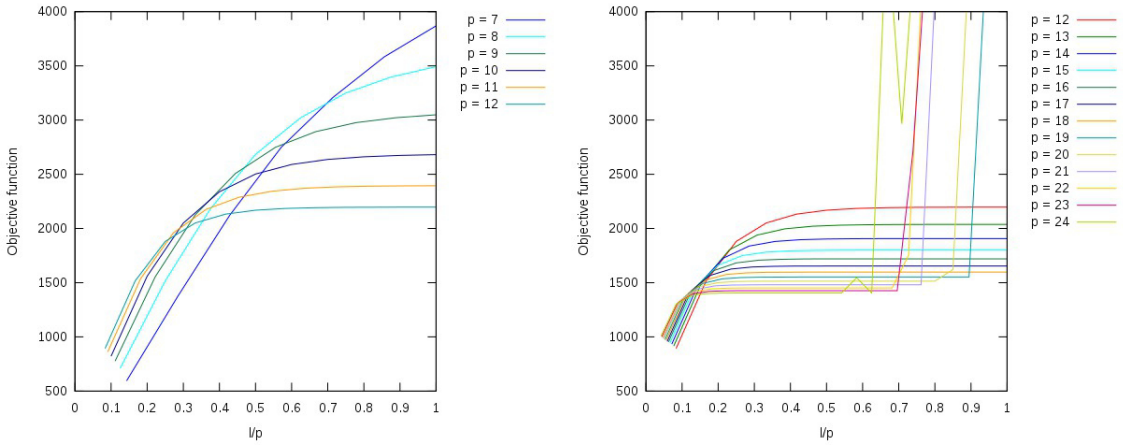

Figure 5.3: Objective function value versus ℓ-to-p ratio.

5.2 GRASP Evaluation

The purpose of this experiment is to assess the behavior of the GRASP heuristic, in particular, as a function of the GRASP threshold quality parameter α. To this end we ran the heuristic consisting of the construction phase only, that is ignoring the local search phase.

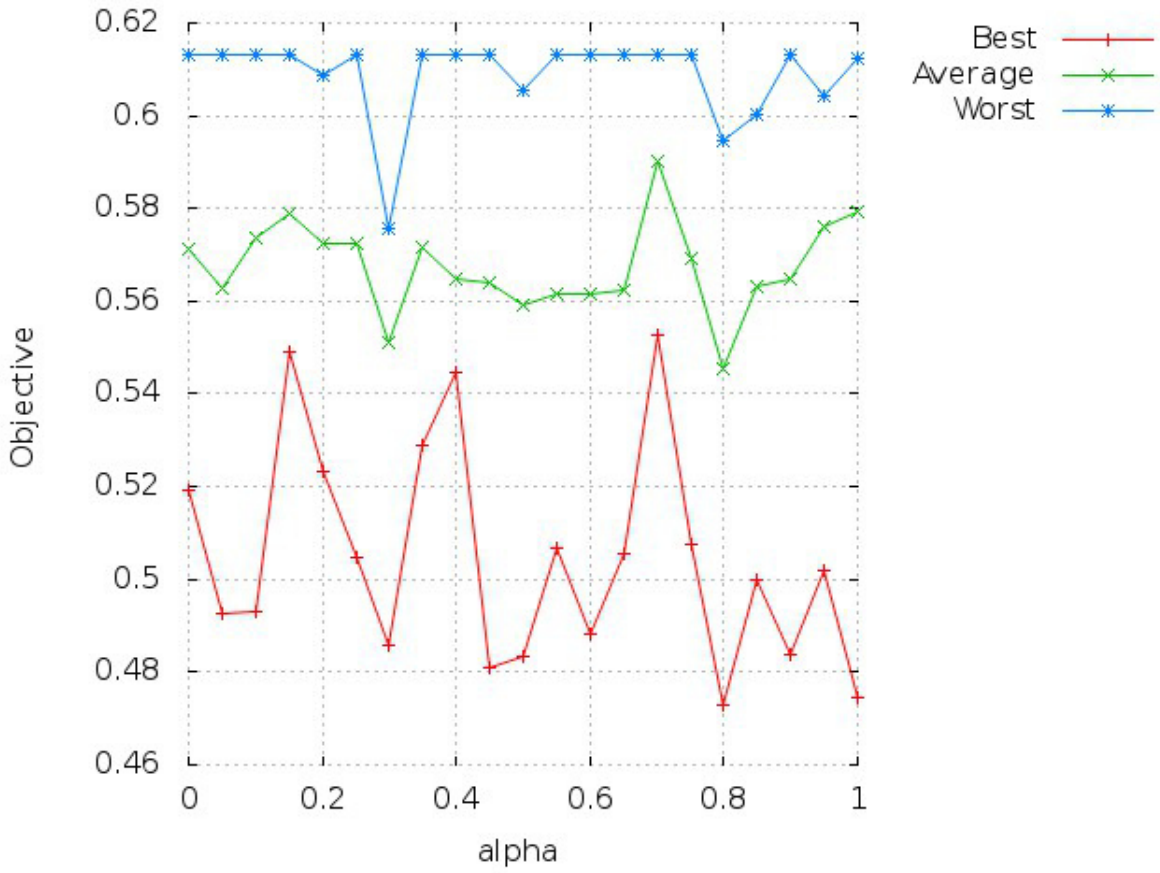

Figure 5.4: Objective value versus α for first greedy function.

We present the results obtained when the heuristic was applied to an instance of size $n = 30$, $m = 30$ and $p = 5$ with different α values (from 0 to 1 in steps of 0.05) and iteration limit set to 100. Figure 5.2 displays the results. In this case, we used the first greedy function (vertical axis). For each particualr value of α, we plot the best, average, and worst values obtained over the 100 iterations. Clearly, the solution delivered by GRASP at the end of the 100 iterations is the best solution

found (red line).

The most interesting observation is that the variation of α does not have a significant effect on solution quality. In other words, a random solution is as good as a solution found by other smaller values of α. Similar results were observed for the other two greedy functions. This is not a common behavior observed by GRASP. Typically, when an appropriate greedy function is assessed one can observe that best solutions are obtained for small values of α. In our particular case, as we can learn from the figure, this unusual behavior may be attributed to the high interaction among servers making the proposed greedy function be not too representative of the final cost in the objective function. As a conclusion, we decide to use $\alpha = 1$ for the remaining experiments.

5.3 Path Relinking

As mentioned before, the path relinking (PR) algorithm consists of two key strategies: the matching method and the processing method.

This experiment aims at assessing the effect of each of these strategies in the solution qualty within the path relinking component. The experiment is carried out as follows. We first apply the randomized multi-start method with 100 iterations to build en elite set of size 10. The best solution obtained is stored. Then we apply the path relinking to each pair in the elite set and report the best solution found by the path relinking. That is, the PR is not tested within the scatter search metaheuristic. We used the mean service time (MST) objective function (Appendix B) as fitness function.

Several instances of different sizes are tested. Results are shown in Table 5.8 for the different values of m and n tested and the different PR strategies. For each $m \times n$ combination, for different values of p are tested ($p = 7, 10, 15, 20$), and for each of this, five replicates are solved. Therefore we have 180 instances in total. The

arrival rate (λ) and the service rate (μ) remain constant. The value shown in each cell indicates the average relative improvement obtained by the PR over the best solution found by the multi-start heuristic, thus the higher the value the better the improvement by the PR. The first column indicates the matching strategy and the second column indicates the processing ordering (FF for farthest first, NF for nearest first, and R for random order).

Table 5.8: Path relinking improvement over multi-start solution.

		$n = 50$			$n = 100$			$n = 150$		
match	order	$m = 30$	50	75	30	50	75	30	50	75
perfect	FF	5.18	10.99	15.94	**6.26**	10.11	11.55	5.14	7.93	**10.56**
	NF	5.22	11.30	14.43	5.61	9.15	10.70	5.26	6.73	10.01
	R	5.58	9.45	12.08	5.23	7.61	10.65	4.02	6.13	8.14
random	FF	5.46	9.96	12.04	4.63	8.14	10.72	3.33	6.59	8.60
	NF	**6.90**	**11.36**	**16.02**	6.13	**10.16**	**12.95**	**5.82**	**9.43**	10.42
	R	5.20	9.73	13.97	4.68	7.70	10.24	4.24	7.18	9.20
workload	FF	5.06	7.31	9.68	4.43	5.73	7.13	4.59	5.86	7.01
	NF	5.17	10.93	13.67	4.87	9.43	11.02	5.04	7.96	9.07
	R	4.58	7.81	10.11	3.34	6.39	6.89	3.48	6.16	6.61

The results clearly indicate a considerale benefit of the PR method. As we can observed the combination that consistently offers better results is the random matching with nearest first ordering. This may be due to the fact that greater diversity is generated by having a random matching. For the remaining experiments the PR is applied under these strategies, that is, random matching with nearest first ordering.

5.4 Scatter Search

Finally, in our last experiment, we evaluate the scatter search metaheuristic proposed in this thesis. We run the same set of instances as in the previous experiment applying the scatter search metaheuristic with no improvement phase. Since

part of the objective function evaluation (MST) involves solving the Hypercube model, we are also interested in investigating how much effort is spend in this evaluation.

Table 5.9 shows the results of the experiment for the diffrent values of m, n, and p tested. Recall that we have 5 replicates for each combination, thus the results shown in the table represent average statistics. The improvement columns (IMP) indicate the average relative improvement on solution quality of the scatter search with respect to the mult-start heuristic solution. The total average running time and average runing time spent on evaluating the MST are measured in seconds

Table 5.9: Scatter search evaluation.

		$n = 50$			$n = 100$		
m	p	IMP (%)	Time	MST	IMP (%)	Time	MST
30	7	6.97	1.64	1.55	6.70	4.33	4.10
	10	9.62	5.20	4.57	5.74	9.97	9.65
	15	6.95	8.55	6.49	3.38	14.90	13.46
	20	3.67	13.79	8.80	2.80	28.30	24.01
50	7	9.76	2.47	2.41	7.68	4.23	4.46
	10	12.20	7.09	6.15	8.09	11.01	10.59
	15	9.75	11.35	8.51	9.55	25.72	22.36
	20	10.63	30.22	19.38	9.51	61.22	49.41
75	7	11.64	2.43	2.37	7.70	4.55	4.76
	10	14.50	6.16	5.35	11.46	10.17	9.82
	15	18.26	19.44	14.08	11.95	36.70	31.19
	20	13.59	44.19	28.05			
Averages		10.63	12.71	8.98	7.69	19.17	16.73

As can be seen, the scatter search delivers significant improvements (up to 18.26 %) to the solution found by the multi-start method in relatively small computing times. Recall that models A and B are pactically intractable for large size instances by conventional branch-and-bound methods. It can also be observed that time spent on solving the MST is about the same order of that of the scatter search.

In any event, running times are relatively small.

Chapter 6

Conclusions and Future Work

6.1 Main Conclusions

The following conclusions are drwan from the present work:

- Two different models (Model A and Model B) for the car wreck adjuster location are proposed. The models can be seen as linear approximations to the Stochastic Queue p-Median Problem.
- We observed that Model A is very difficult to solve for the medium-size instances of around $m = 30$ or larger.
- For small enough values of ℓ, Model B can be solved relatively quickly for instances of medium size.
- The solutions obtained from Model B for small values of ℓ approximates well the solutions of Model A, as well as self solutions for large values of ℓ
- A scatter search metaheuristic is proposed for the problem under study. Within the heuristic, several components are developed and evaluated.
- The multi-start randomized construction algorithm was assessed within a GRASP framework. The method was tested for different values of the quality threshold

parameter. No significant effect was observed on this parameter. Therefore, it was concluded that a totally randomized algorithm is sufficient.

- A path relinking (PR) method was proposed and implemented as a combination method within the scatter search. This PR uses several strategies for deciding the trajetory between two given solutions. These strategies were evaluated finding that the random matching with nearest first ordering as the best among all.

- The scatter search was evaluated on a large set of instances finding very good results with respect to the solution found by the multi-start heuristic.

6.2 Future Work

There are several areas of opportunity for further work in this area. First, the empirical tests were done for a relatively small number of instances. Generating more instances for each given size would allow a more sound study with a corresponding statistical analysis.

Another task is to test the solution methodology in real-world problems. This wolud of course involve a considerable amount of work particulary when considering the potential accident sites in large cities.

From the heuristic perspective, we must point out that the proposed scatter search was evaluated with no improvement method. Therefore, it is evident that a natural extension to this work is the computational implementation of the improvement method discussed in Section 4.2.2 and a complete empirical evaluation of these procedures within the scatter search algorithm.

Advanced strategies such as memory adaptive programming for exploring different local search neighborhoods may be worthwhile investigating as well.

The methods developed in this thesis provide a recommendation as to where the adjusters should be located. Since the problem is subject to the random occurance of the accidents, performance indicators based on simulation models must be calculated and compared with current industry policies.

Appendix A

Approximating the Equilibrium Behavior of MSLS

In Jarvis [17] a procedure is given for approximating the equilibrium behavior of multi-server loss systems having distinguishable servers and multiple customers types under light to moderate traffic intensity.

A.1 Motivation

In an emergency service such as fire or police, the servers are fire fighting units or patrol cars and the customers are calls for service. The simple Erlang loss system is inadequate in two aspects for a detailed system analysis.

- one often wishes to preserve the identity of service units (distinguishable servers).
- because of the geographic nature of these systems, the service time depend on both the server and the customer at least through the travel time between the pair.

A.2 Model Assumptions, Notation, and Terminology

Consider a system in which:

- Exactly one server is assigned to each customer unless all servers are busy, in which case the customer is irrevocably lost from the system.
- Servers are assigned to customers according to a fixed preference assignment rule.
- No preemption of service is allowed.
- Assignments are made immediately upon customer arrival.

with the following parameters:

- $N :=$ the distinguishable number of servers,
- $C :=$ types of customers,
- Customers of type m, arrive according to a Poisson process with rate λ_m,
- $\lambda :=$ total arrival rate,
- $a_{mk} :=$ the k-th preferred server for customers of type m,
- τ_{im} the expected service time for server i and customer of type m.

The performance measures for the system include:

- $\rho_i :=$ the workload of server i,
- $f_{im} :=$ the probability a random customer of type m is assigned to server i,
- $P_N :=$ the probability all servers are busy.

A.3 Approximation Procedure

The procedure described below is based on that given by Larson [19], for approximating performance measures for the Hypercube model assuming exponential service times. Larson developed an approximation for f_{im} as

$$f_{im} \simeq Q(N, p, k-1)(1-\rho_i)\prod_{l=1}^{k-1}\rho_{a_{ml}} \tag{A.1}$$

where

$$Q(N,p,k) = \sum_{j=k}^{N-1}\frac{(N-j)(N^j)(\rho^{j-k})P_0(N-k-1)!}{(j-k)!(1-P_N)^k N!(1-\rho(1-P_N))} \text{for} k = 0,1,\ldots,N-1 \tag{A.2}$$

Let B_i denote the event that server i is busy; and let B_{im} denote the event that server i is busy serving a customer of type m, then

$$\rho_i = Pr\left[B_i\right] = \sum_{m=1}^{C} Pr\left[B_{im}\right] = \sum_{m=1}^{C}\lambda_m f_{im}\tau_{im} \tag{A.3}$$

combine equations (A.1) and (A.3) and solve for ρ_i to obtain the approximation iteration

$$\rho_i(\text{new}) = \frac{V_i}{(1+V_i)} \tag{A.4}$$

where V_i is given by

$$V_i = \sum_{k=1}^{N}\sum_{m:a_{mk}=i}\lambda_m\tau_{im}Q(N,\rho,k-1)\prod_{l=1}^{k-1}\rho_{a_{ml}} \tag{A.5}$$

When there is a common mean service time, the estimates for ρ_i can be normalized using

$$\sum_{i=1}^{N}\rho_i = N\rho(1-P_N) \tag{A.6}$$

In the generalized procedure, τ can be approximated at the end of each iteration by

$$\tau = \sum_{m=1}^{C}\left(\frac{\lambda_m}{\lambda}\right)\sum_{i=1}^{N}\frac{\tau_{im}f_{im}}{(1-P_N)} \tag{A.7}$$

Approximation Algorithm

Given:

$$\lambda_m, \tau_{im}, a_{mk} \text{ for } m = 1, \ldots, C; i = 1, \ldots, N; k = 1, \ldots, N$$

Initialize:

$$\rho_i = \sum_{m:a_{m1}=i} \lambda_m \tau_{im};\ \tau = \sum_{m=1}^{C} (\lambda_m/\lambda)\tau_{a_{m1},m}$$

Iteration:

(1) Compute $Q(N, \rho, k)$ for $k = 1, \ldots, N-1$ where $\rho = \lambda\tau/N$ using equation (A.2).

(2) For $i = 1, \ldots, N$, the new ρ_i is $V_i/(1+V_i)$, where V_i is given by equation (A.5).

(3) Stop if max change in ρ_i is less than convergence criterion.

(4) Else compute P_N by equation (A.6), τ by equation (A.7), and f_{im} by equation (A.1).

(5) Return to step 1.

No analytic bounds on the accuracy or convergence properties of the approximation procedure have been developed to the best of my knowledge.

In regards to convergence properties, the numerical iteration has proved to be very stable and converges in a small number of iterations under relatively stringent conditions, with 4 to 6 iterations being typical for 10-server systems.

In comparing the accuracy of this approximation to results from the exact Hypercube model, Larson has found errors in server workloads to be less than 1 to 2 percent.

Appendix B

Berman Heuristic

The stochastic queue p-median (SQM) problem is studied by berman, Larson, and Parkan [4]. This problem is very similar to our case since it consists of locating p mobile service units on a network in the presence of queueing-like congestion. The main difference is that they allow the location of the servers in all the network including edges.

Nevertheless, the heuristics are quite applicable to our case. They present two similar heuristics for the problem. Here we use and described heuristic 2 of that paper. The main idea is as follows. Given the location of p servers, heuristic 2 uses the hypercube model to provide for each one of the servers information on the fraction of dispatches to each possible call for service. This information is then used to improve the location of each of the servers by solving the corresponding SQM. Such process continues until no further improvement is possible.

The**Mean Service Time Calibration Process** is used for evaluating the solutions.

B.1 The Stochastic Queue p-Median Problem

To define the problem the following parameters are defined:

- $G(N, L)$:= the transportation network,
- N := the set of demand centers, with $|N| = n$,
- L := the set of all transportation arteries, the links or edges,
- h_j := the fraction of service calls associated with each node j,
- $d(X, Y)$:= the shortest path distance between any two points $X, Y \in G$,
- p := number of response units,
- $\bar{X}$:= the home locations of the service units while available,
- λ := mean rate per unit of time within service calls generated by a Poisson distribution.

Given the arrival of a call for service, exactly one of the servers is dispatched to it assuming that at least one server is available.

The service time for any service unit i is the sum of two components:

- The non-travel time component, which is the sum of on-scene and off-scene service time.
- Travel time component, which is the sum of travel time to the location of the call and travel time back to the home location.

The mean service time for a service unit located at $\bar{X}^i$ is denoted $S(\bar{X}^i)$,

$$S(\bar{X}^i) = \sum_{j=1}^{n} h_j^i \left(\bar{W}_{ij} + \frac{\beta_i}{v_i} d(\bar{X}^i, j) \right) \; i = 1, \ldots, p \tag{B.1}$$

- $\bar{W}_{ij}$ is the mean of the *non-travel time* component W_{ij}, that is, the on-site service time.
- v_i is the *travel speed* of unit i to the scene of the call which is assumed constant

- β_i is a constant that allows different travel speeds to and from the scene of the call
- h_j^i is the probability that server i is dispatched to node j given that server i is dispatched to a call for service.

Whenever a call for service arrives while at least one of the servers is free at its home location, the closest available server to the call will be dispatched. Calls that find all servers busy enter a queue. The queue discipline is assumed to be First-Come-First-Served. The expected response time to a random call denoted by $\bar{T}_R(X)$ is the sum of two components

$$\bar{T}_R(\bar{X}) = \bar{W}_q(\bar{X}) + \bar{t}(\bar{X})$$

- $\bar{W}_q(\bar{X})$ is the expected *waiting time* in the queue
- $\bar{t}(\bar{X})$ is the expected *travel time* to the call.

The objective is to find a set of p locations $\bar{X}^*$ on the network such that

$$\bar{T}_R(\bar{X}^*) \leq \bar{T}_R(\bar{X}) \; \forall \bar{X} \in G$$

$\bar{X}^*$ is called *the stochastic queue p-median.*

B.2 Mean Service Time Calibration Process

In this emergency system, travel times may represent a considerable part of service times. It may be advisable to adjust the service times by means of a calibration process, which can be performed using a simple iterative procedure.

The procedure consists of verifying if there are significant differences among the input mean service times and the output mean service times (computed by the hypercube model). In this case, the hypercube is solved using the computed mean

service times as inputs, until the differences among input and output values are sufficiently small.

The mean service time calibration method

Step 0. The mean service time of unit i, $1/\mu^i = 1/\mu^i_{NT}$, $i = 1, \ldots, p$ ($1/\mu^i_{NT} = \sum_{j=1}^{n} h_j \bar{W}_{ij}$ is the mean of non-travel time component of the service time).
Step 1. Run the Hypercube Model (using μ^i) to obtain f_{ij}, $i = 1, \ldots, p$, $j = 1, \ldots, n$.
Step 2. $1/\hat{\mu}^i = \sum_{k=1}^{n} h^i_k (\bar{W}_{ik} + (\beta_i/v_i) d_{ik})$ where $h^i_k = f_{ik} / \sum_{j=1}^{n} f_{ij}$
Step 3. If $|1/\hat{\mu}^i - 1/\mu^i| > \epsilon$ for at least one i, $i = 1, \ldots, p$, set $1/\mu^i \equiv 1/\hat{\mu}^i$ and go back to *Step 1.* Otherwise stop.

Bibliography

[1] R. Benveniste. Solving the combined zoning and location problem for several emergency units. *Journal of the Operational Research Society*, 36(5):433–450, 1985.

[2] G. N. Berlin and J. C. Liebman. Mathematical analysis of emergency ambulance location. *Socio-Economic Planning Sciences*, 8(6):323–328, 1974.

[3] O. Berman and R. C. Larson. The median problem with congestion. *Computers & Operations Research*, 9(2):119–126, 1982.

[4] O. Berman, R. C. Larson, and C. Parkan. The stochastic queue p-median problem. *Transportation Science*, 21(3):207–216, 1987.

[5] R. Church and C. ReVelle. The maximal covering location problem. *Papers in Regional Science*, 32(1):101–118, 1974.

[6] M. S. Daskin. A maximum expected covering location model: Formulation, properties and heuristic solution. *Transportation Science*, 17(1):48–70, 1983.

[7] M. S. Daskin and E. H. Stern. A hierarchical objective set covering model for emergency medical service vehicle deployment. *Transportation Science*, 15(2):137–152, 1981.

[8] T. A. Feo and M. G. C. Resende. Greedy randomized adaptive search procedures. *Journal of Global Optimization*, 6(2):109–133, 1995.

[9] M. Gendreau, G. Laporte, and F. Semet. Solving an ambulance location model by tabu search. *Location Science*, 5(2):75–88, 1997.

[10] F. Glover. Heuristics for integer programming using surrogate constraints. *Decision Sciences*, 8(1):156–166, 1977.

[11] F. Glover. A template for scatter search and path relinking. In J.-K. Hao, E. Lutton, E. Ronald, M. Schoenauer, and D. Snyers, editors, *Artificial Evolution*, volume 1363 of *Lecture Notes in Computer Science*, pages 1–51. Springer, Berlin, Germany, 2005.

[12] F. Glover, M. Laguna, and R. Martí. Fundamentals of scatter search and path relinking. *Control and Cybernetics*, 29(3):653–684, 2000.

[13] J. Goldberg, R. Dietrich, J. M. Chen, M. G. Mitwasi, T. Valenzuela, and E. Criss. Validating and applying a model for locating emergency medical vehicles in Tuczon, AZ. *European Journal of Operational Research*, 49(3):308–324, 1990.

[14] S. L. Hakimi. Optimum locations of switching centers and the absolute centers and medians of a graph. *Operations Research*, 12(3):450–459, 1964.

[15] J. Halpern. The accuracy of estimates for the performance criteria in certain emergency service queueing systems. *Transportation Science*, 11(3):223–242, 1977.

[16] J. P. Jarvis. *Optimization in Stochastic Service Systems with Distinguishable Servers*. PhD thesis, Massachusetts Institute of Technology, Cambridge, USA, 1975.

[17] J. P. Jarvis. Approximating the equilibrium behavior of multi-server loss systems. *Management Science*, 31(2):235–239, 1985.

[18] R. C. Larson. A hypercube queuing model for facility location and redistricting in urban emergency services. *Computers & Operations Research*, 1(1):67–95, 1974.

[19] R. C. Larson. Approximating the performance of urban emergency service systems. *Operations Research*, 23(5):845–868, 1975.

[20] V. Marianov and D. Serra. Probabilistic maximal covering location-allocation models for congested systems. *Journal of Regional Science*, 38(3):401–424, 1998.

[21] V. Marianov and D. Serra. Hierarchical location-allocation models for congested system. *European Journal of Operational Research*, 135(1):195–208, 2001.

[22] H. Pirkul and D. A. Schilling. The siting of emergency service facilities with workload capacities and backup service. *Management Science*, 34(7):896–908, 1988.

[23] M. G. C. Resende and R. F. Werneck. A hybrid heuristic for the p-median problem. *Journal of Heuristics*, 10(1):59–88, 2004.

[24] C. ReVelle and K. Hogan. The maximum availability location problem. *Transportation Science*, 23(3):192–200, 1989.

[25] R. Z. Ríos-Mercado and H. J. Escalante. GRASP with path relinking for commercial districting. *Expert Systems with Applications*, 44:102–113, 2016.

[26] E. S. Savas. Simulation and cost-effectiveness analysis of New York's emergency ambulance service. *Management Science*, 15(12):B608–B627, 1969.

[27] C. Swoveland, D. Uyeno, I. Vertinsky, and R. Vickson. A simulation-based methodology for optimization of ambulance service policies. *Socio-Economic Planning Sciences*, 7(6):697–703, 1973.

[28] C. Toregas, R. Swain, C. ReVelle, and L. Bergman. The location of emergency service facilities. *Operations Research*, 19(6):1363–1373, 1971.

[29] J. R. Weaver and R. L. Church. Computational procedures for location problems on stochastic networks. *Transportation Science*, 17(2):168–180, 1983.

[30] J. R. Weaver and R. L. Church. A median location model with nonclosest facility service. *Transportation Science*, 19(1):58–74, 1985.

Vita

Luis was born in San Nicolás de los Garza, Nuevo León, México on November 25, 1989, the fourth of five sons of Luis Alberto Maltos Muzquiz and Ruth Ortega Pecina. He received a bachelor degree in mathematics from Universidad Autónoma de Nuevo León (UANL) in 2011. After that, he worked as a consultant, until 2014, analyzing mathematical models and solution methods for multiobjective problems related to industrial, commercial and service sectors. At the beginning of the same year he began to study a Master of Science in Systems Engineering at the Graduate Program in Systems Engineering at UANL.