universität wien

DIPLOMARBEIT

Titel der Diplomarbeit

„Competitive Location Models for Transportation in Disaster Relief Logistics“

Verfasser

Christian Burkart

angestrebter akademischer Grad

Magister (Mag.)

Wien, 2014

Studienkennzahl lt. Studienblatt:	A 057 390
Studienrichtung lt. Studienblatt:	Individuelles Diplomstudium Internationale Entwicklung
Betreuer:	Univ.Prof. Dr. Walter Gutjahr

Diploma Thesis

Competitive Location Models for Transportation in Disaster Relief Logistics

by Christian Burkart

Supervisors:

Univ.Prof. Dr. Walter Gutjahr,

Dr. Pamela Nolz

January 25, 2014

Contents

List of Figures

List of Tables

List of Algorithms

List of Abbreviations

Abbreviation	Meaning
CRED	Centre for Research on the Epidemiology of Disasters
CTP	Covering Tour Problem
DC	Distribution Center
FTL	Full Truckload
GIS	Geographical Information System
MOEA	Multiobjective Evolutionary Algorithm
NGO	Non-Governmental Organization
NSGA II	Nondominated Sorting Genetic Algorithm 2
OD Matrix	Origin-Destination Matrix
OOQ	Optimal Order Quantity
OR	Operations Research
PC	Population Center
SC	Supply Chain
TSP	Traveling Salesman Problem
VRP	Vehicle Routing Problem

Acknowledgments

I am lucky. Lucky in every sense. I was born in a rich European country, the needs that come up in the course of this thesis are completely unknown to my own experience. When I feel hungry, I go to the fridge, not to one of the distribution centers for relief goods that some crazy student placed kilometers from my village in his case study. I live in a livable city, together with my wonderful girlfriend, and I can do for a living what I like to do. But this luck is not, or at least just to a small degree, the result of my own work or abilities. Therefore, I'm very thankful for my current situation. Here, I want to thank my family, my supervisors and my friends. Without their help, this diploma thesis, and especially this kind of diploma thesis would not exist for this type of studies.

First of all, my family did (and does) not only help me wherever they could, but most important, they would give me the freedom to do with my life what my personal interests would lead me to. Studying without pressure is a gift that should not be taken for granted. I want to specifically thank my grandparents Gertrude and Franz, who not only took my physical well-being very serious from the moment I was born but, of the things relevant in life, my grandparents taught me a lot. After this thesis I know why my grandfather enjoyed working major parts of his life on optimizing tours for postal deliveries.

I also want to thank my supervisors, Prof. Walter Gutjahr and Pamela Nolz. They not only integrated me fully into their research project, without even knowing me. But, what is more, they gave me, a student of completely unrelated studies, the opportunity to grow with a challenge. I think I never learned so many valuable things in such a short time, while having fun doing it.

Finally, my backing in every situation: my friends Martina, Vicky (special thanks!), Stephan and Alex. Life would be so boring without you...

Chapter 1

Introduction

1.1 Problem Statement: Research Motivation

The delivery of humanitarian relief goods to people in need after a disaster is a logistical challenge. Very often not only the local stocks of basic supplies such as drinking water, basic food items and shelter are destroyed or unusable, but also the infrastructure networks that function as the logistical basis for delivering the required goods. As Tomasini and van Wassenhove (2009) argue, there are certain critical differences between supply chains of normal businesses such as retailers and disaster response operations. Humanitarian logistics usually suffers from a lack of resources, be it 'human', capital or infrastructure resources. Staff might not be ready at all times required, money, most probably coming through donations, might not be available in the right quantity or at the right time and the infrastructure, as described, might be damaged. Furthermore, required information might not be available at the time of planning, such as information on quality of infrastructure or the quantity of affected people, while quick reaction is required to prevent human suffering to the highest possible extent. (Tomasini and van Wassenhove, 2009, pp.9)

While these factors create a situation that is already hard to cope with from an organizational point of view, the whole supply chain is also a "Politicized Environment" (Tomasini and van Wassenhove, 2009, p.10), this means that from the sources of donations to the process of delivery all steps can have a political motive. As Fink and Redaelli (2009) show, donations are influenced by the size and proximity of beneficiary countries,

their degree of political alignment and, ideally, their status as oil exporting country, since oil exporting countries appear to receive more assistance in cases of disasters.

This Diploma Thesis tackles most of the aforementioned factors in a very practical way. It presents mathematical problems for determining potentially optimal locations for distribution centers of disaster relief goods and the routing of trucks to deliver these goods. Optimality in this respect is defined by minimizing the unserved population and the costs at the same time, and leads to another important topic: the justice in distribution of disaster relief goods. The presented problem will be examined on basis of various, conflicting definitions and theories of justice. This helps resolving some political questions about humanitarian logistics, especially about fairness of distribution.

Methods for determining distribution locations for distributing relief goods in a timely and efficient manner are a very important component of disaster relief logistics, and, according to Beamon (2004, p.5), pose one of the big challenges in relief chain management. Thus, the motivation for this thesis comes from the creation of a new approach of modeling the position of distribution centers in a way that accommodates not only the above mentioned points, but also tries to respect the habits of the recipient population. The core idea behind this problem is that people requiring assistance will have some degree of freedom in their choice of where (i.e. from which distribution center (DC)) they get their relief goods from. The solution approach to this problem allows for computational solution of complex problems in relatively short time and employs Competitive Location Models as presented in Drezner and Eiselt (2002) which take the beneficiaries' preferences into account.

As time is a crucial factor in disaster relief operations, the requirements for the solution of such problems are of crucial importance. While such problems should enable the relief chain planner to take decisions in a quick manner, the output of such problems should be of practical relevance too. As Daskin (2008, pp.285) describes, usually covering-based models or median-based models are used as location models. While covering-based models try to optimize the choice of which facilities to open based on a coverage distance, which means that demand within a certain distance of a distribution center is considered to be covered, median-based models try to achieve optimal location planning by minimizing average distances between the demand points and the distribution centers. These procedures

usually take cost or distance into account, but do not provide insight concerning the choice of the population on the decision of where to go to. This shortcoming also provides research incentive for this thesis.

1.2 Research Questions

The main questions that this thesis answers are the following:

- *How can emergency relief practitioners use competitive location models for their planning?*
- *What impact do planning algorithms employing competitive location models for distribution center planning have on the fairness of distribution of relief goods?*
- *Which limitations apply for the use of competitive location models in disaster relief?*
- *What crucial assumptions have to be made if competitive location models are employed?*

1.3 Thesis Overview

In this thesis, a combination of different types of sciences is aspired, although the basis is formed by the social sciences. This is due to two facts: First, the social science perspective concerning the management of development projects, as frequently encountered in the studies of International Development at the University of Vienna, or, as in the case of this thesis, the management of disaster relief operations, quickly encounters problems when it alone should form the basis for rational and efficient managerial decision making. Second, if only a mechanistic viewpoint is used in assisting with decision making concerning such projects or relief operations, the output might be optimal from a mathematical point of view, i.e. the costs associated might be minimal, or the efficiency of the project in terms of people reached per Euro spent might be high - but a crucial question remains not only unanswered, but also unasked: Is the solution attained *good*?

Since this term, 'good', meaning just in the sense of distribution, or beneficial for the people affected, is not easy to measure with ordinal numbers, let alone with cardinal

numbers, which impairs the usability in mathematical models, social sciences can help in evaluating the decision making behavior of the aforementioned models.

This necessity of cooperation of sciences is exactly what this thesis tries to show: Here, a situation of disaster is assumed and the hypothetical task is to design the distribution of disaster relief goods, such as food items. Due to the nature of the disaster (a drought), the transportation network remains largely intact and, after a depot, where the arriving goods are stored prior to delivery to smaller distribution centers, has been positioned, the next task is to determine which DCs (of a set of possible DCs) are to be opened. While opening a DC incurs fixed and variable cost, also the routing of the vehicles transporting the goods to the DCs results in variable costs, as well as the provisioning of supplies itself. These costs determine one of the two factors on which decision making is based, while the other factor is the amount of unserved inhabitants. Since inhabitants might be able to choose between more than one DCs in their surroundings, a model for determining demand allocation for every opened DC is presented and subsequently the objective values of cost and unserved demand for each opening pattern of DCs can be calculated. Furthermore, if the next available DC is very far away, some might choose not to go to the DC at all. Since the number of possible opening patterns grows very fast, a heuristic is used to provide the decision maker with a set of non-dominated solutions, from which he/she could choose according to available resources.

The whole approach will be examined according to how just the distribution of relief goods can be established with this approach.e definition of justice is crucial in this case, therefore four different definitions are presented and used in the assessment of the approach. Also, crucial assumptions to ensure the functioning of the approach as well as the requirements in terms of resources are assessed according to their potential practical relevance for disaster relief operations.

The remainder of the thesis is structured as follows: In Chapter 2, an introduction to the world and problems of Humanitarian Logistics is given, with a special focus on the topic of Justice in Humanitarian Logistics by comparing different approaches to (distributive) justice presented by the most influential schools of thought in this area. In Chapter 3, the methods that are used to apply the problem of Chapter 4 to the case study of Chapter 5 are described in detail. Afterwards, in Chapter 4, the problem employed in

this work is presented in large detail, while special focus is laid on the critical assessment of the problem, not only but also in terms of justice. The following Chapter 5 presents the core applied work of this thesis by describing the Case Study concerning a realistic, but hypothetical drought disaster in the Limpopo region of southern Mozambique. In this chapter, the case study is described, the tasks necessary are explained and results are analyzed. To show the model's impact in the case study application, the practical relevance is critically assessed. The last chapter, Chapter 6, sums up the findings of the thesis.

Chapter 2

Humanitarian Logistics

In this chapter, the focus is to 'ground' the Chapters 4, 3 and 5 in the target environment of Humanitarian Logistics. In the beginning, the reader is introduced to the field of Humanitarian Logistics and then the basis is laid to connect to the assessment of the model from the viewpoint of justice in Chapter 5, four conceptions of justice that have diverse meaning in the context of Humanitarian Logistics are presented with focus on distributive justice.

2.1 Introduction to Humanitarian Logistics

Humanitarian Logistics, with the core task to help to alleviate pain by supplying relief goods to people in need after a disaster, forms the environment of the work carried out in the course of this thesis and thus deserves a detailed definition and thorough introduction. It, just like every logistics process in the normal business world, deals with a Supply Chain (SC), which "[...] consists of all parties involved, directly or indirectly, in fulfilling a customer request. The supply chain includes not only the manufacturer and suppliers, but also transporters, warehouses, retailers, and even customers themselves" (Chopra and Meindl, 2010, p.20).

Of course, this definition, taken out of one of the leading textbooks on Supply Chain Management, has to be adapted for the task of Humanitarian Logistics. There are many differences between normal and disaster relief supply chains (Beamon, 2004). First of

all, the customer is not a regular customer who wishes to exchange his money for goods. In this case, we have an affected population of a region or a country and organizations, governments or private donors that pay for the services. Second, the demand patterns are very different. While commercial SCs usually have stable demand and thus replenishment on a regular basis, Humanitarian SCs suffer from unpredictable and sudden high demand situations. This is even more difficult to deal with since, in contrast to commercial SCs, Humanitarian SCs require zero lead times, since fast reaction to a disaster can literally save lives, which is not required in normal commercial SCs. Further on, the distribution network of commercial SCs is already up and working, which is true as well for methods of inventory control, the information systems behind and the performance measurement systems are also established and support management in decision making. In Humanitarian SCs, however, this is not necessarily the case and leaves many unknown factors for planning. The current state of the transport network is mostly unknown, leaving especially the last mile distribution of relief goods in a state of insecurity, which is generally true for all types of information about the affected region. The goals of commercial SCs are rather clear: cost minimization, profit maximization and high quality of products and services, while Humanitarian SCs usually have the more complex goal of preventing or minimizing the loss of human lives and suffering. What adds to the complexity is that not only goods have to be delivered, but also the people organizing the Humanitarian SC.

Another big difference is the special environment of a disaster situation. A disaster can be characterized as

> "[...]a serious disruption of the functioning of society, posing a significant, widespread threat to human life, health, property or the environment, whether caused by accident, nature or human activity, and whether developing suddenly or as a result of a complex, long-term processes." (UN/ISDR 2004, cited in: Kovacs and Spens, 2009, p.508)

van Wassenhove (2006, p.476) provides a categorization of different types of disasters, since the circumstances, under which a disaster emerges, can be diverse, as can be seen in Figure 2.1. There are two mutually exclusive sets of disaster categories. On the one hand, disasters can either happen suddenly, like earthquakes or tsunamis, or they can gradually develop into big disasters, like a famine or a political crisis. On the other hand, disasters

can also be categorized according to the reason of their existence. While some disasters emanate from natural phenomena like, earthquakes, others are result of human action, like war or accidents. These four categories form the 2x2 matrix in Figure 2.1, since natural as well as man-made disasters can possibly develop suddenly or gradually.

	Natural	**Man-made**
Sudden-onset	Earthquake Hurricane Tornadoes	Terrorist Attack Coup d'Etat Chemical leak
Slow-onset	Famine Drought Poverty	Political Crisis Refugee Crisis

Figure 2.1: Categorization of Disasters
(van Wassenhove (2006, p.476))

These different categories result in different environments for humanitarian logistics operations, which adds to the complexity of the field. Thus, the definition of Humanitarian Logistics tends to be more complex than that of a regular Logistics:

> Humanitarian Logistics is defined as the process of planning, implementing and controlling the efficient, cost-effective flow and storage of goods and materials, as well as related information, from the point of origin to the point of consumption for the purpose of alleviating the suffering of vulnerable people. The function encompasses a range of activities, including preparedness, planning, procurement, transport, warehousing, tracking and tracing, and customs clearance. (Thomas and Kopczak, 2005, p.2)

Now, every term specific to Humanitarian Logistics has been explained in great detail, except for one: The term 'Humanitarian' itself. Obviously, the whole raison d'être for Humanitarian Logistics lies in its motivation: Providing assistance to people in need. Humanitarianism is founded on three core values: Humanity, Neutrality and Impartiality. The value humanity means that human suffering should be reduced wherever possible, while neutrality means that, for example in a conflict, no-one's side should be taken and additionally, help should be given impartially, which means to all sides, without discrimination or priority except for the most urgent needs (Tomasini and van Wassenhove,

2009, pp. 20ff.). Following these three principles in relief work, organizations can try to open up the possibility for action even if the disaster takes place in a complex situation of political insurgency, since these values can create acceptance on all sides of conflicts. Thus, these values are, like in Figure 2.2, very often depicted as a triangle, opening up the humanitarian space (Tomasini and van Wassenhove, 2009, p.27).

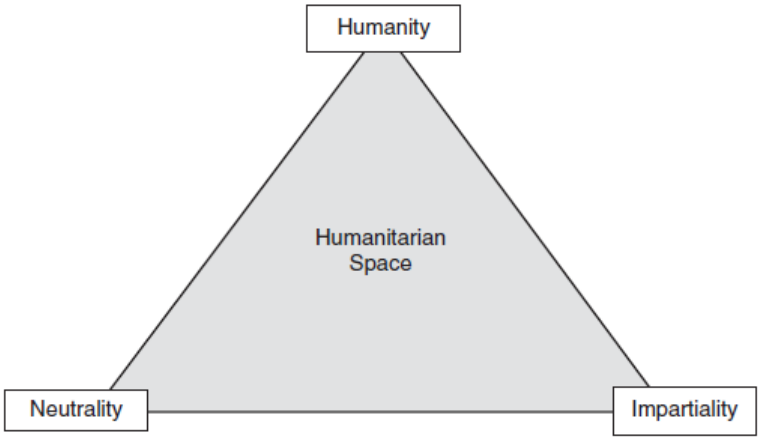


Figure 2.2: Triangle of Humanitarian Space
(Tomasini and van Wassenhove (2009, p.27))

The close study of Humanitarian Logistics is especially worthwhile, since both commercial as well as humanitarian SCs can profit from potential learning effects. There are many differences that make humanitarian SCs a very special case. Such circumstances occur also in regular logistics, so best practices from Humanitarian Logistics can be very valuable, which is also true vice versa.

One special obstacle in management of Humanitarian Logistic processes is the coordination, both vertical and horizontal. A wide variety of different actors from different countries, speaking different languages, have to coordinate their actions, while every actor has to please its donors to make sure that financial resources keep flowing or even start to flow in the time necessary. Unfortunately, the organizations compete for the same funding sources, which is an incentive not to cooperate too closely, especially for big NGOs. The lack of information leaves most parts of a relief operation in uncertainty, from the political situation to security, infrastructure as well as funding and especially severity of the disaster. Not only funds can be unpredictable, but also the availability of skilled workforce and the nature of donations. Sometimes, donations are made in provision of supplies without coordination if the supplies are needed at all. And of course the coordination results in

cost before it can create gains in efficiency (Balcik et al., 2010, pp.23f.).

Kovacs and Spens (2009, p.519) provide a more detailed table about challenges faced by humanitarian logistics sorted by stakeholder, which can be seen in Figure 2.3 and is based on experience from a case study on Ghana.

Source of challenge	Input/output environment	Competitive environment	Regulatory environment	Internal stakeholders
Perspective				
International humanitarian organization perspective	Delays Lack of funding Inappropriate donations Limits in use of funding Aid dependency	Lack of coordination	Dependence on government declaring state of emergency Lack of transport infrastructure Duties Lack of early warning systems Lack of governance	Lack of trained logisticians Lack of vehicles Lack of warehouse
Governmental organization perspective	Lack of supplies Lack of equipment Lack of funding Inappropriate donations Lack of access to water Difficulties in enforcing standards	Lack of communication Lack of knowledge of humanitarian organizations	Absence of legislation Lack of funding Streets have no name Security problems Difficulties in enforcing standards	Lack of trained logisticians Brain drain Lack of vehicles Lack of supplies Lack of equipment Lack of ICT

Figure 2.3: Challenges categorized by stakeholder (Kovacs and Spens (2009, p.519))

Although Humanitarian Logistics is a task related to preparing for disaster relief in the sense of prepositioning of goods and subsequently reacting to the disaster in the most effective way, it should not be forgotten that disaster management does not only consist of reaction, but as well acting before and after. Tomasini and van Wassenhove (2009, p.44) define full disaster management as starting in the mitigation phase, thus laying the framework to facilitate resilience (e.g through laws) far before a disaster strikes. The second phase is the preparedness phase, which requires actual prepositioning and preparation for a possible disaster, while the third phase, the response phase, contains the reaction to the disaster itself. After a disaster struck and it has been reacted to, the rehabilitation phase should help return the communities to normality again.

In the next section, different aspects of justice are presented, all suited for analysis in the context of Humanitarian Logistics.

2.2 Aspects of Justice for Humanitarian Logistics

A question closely related to the core problem of this thesis, the Location Problem, is the question of distributive justice in the allocation of disaster relief goods. The decision to open or close a distribution center as well as the mode of allocating the demand for the goods to the opened distribution centers, implicitly expresses a certain set of moral ideas on distributive justice. If the distribution of relief goods aims at providing everybody affected with the exact same amount of relief goods, even if that means that the amount will be small due to limited financial resources and higher delivery costs, the ideas of justice employed are quite different from a situation in which the aim is to provide a maximum amount of goods with the available resources. In order to assess the ideas on justice expressed by the Competitive Location Problem and the according model from the Chapters 4 and 5, an overview of prominent ideas on justice is needed, since every person will have their own ideas on what is just in which situation, and most likely it will not be the same idea for different situations. The philosopher John Stuart Mill expressed this idea in very elegant form:

> Not only have different nations and individuals different notions of justice, but in the mind of one and the same individual, justice is not some one rule, principle, or maxim, but many, which do not always coincide in their dictates, and in choosing between which, he is guided either by some extraneous standard, or by his own personal predilections. (Mill, 2007 [1861], p.42).

This question also reached the attention of practitioners in Humanitarian Logistics, and questions asked concerning justice in the distribution of disaster relief goods come from a wide range of different view points. The targeting of disaster relief goods at the people with the greatest needs is discussed and the feasibility of such policies and the justification are sources of controversy (Jaspars and Shoham, 1999). Also the distribution of assistance between different countries in need, basically a 'targeting' approach based on different reasons and on a different geographical scale, and the possible reasons found the attention of researchers (Fink and Redaelli, 2009).

Other questions relate to justice in a fundamental way: What incentives are provided when development aid or disaster aid is provided or can be expected to be provided?

(Raschky and Schwindt, 2009a) (Raschky and Schwindt, 2009b)

In the following four sections, four different schools of thought concerning distributive justice are presented by their core ideas: Utilitarianism, Libertarianism, Egalitarian Liberalism and Socialism. Those four schools are predestined for such a comparison, since they contributed very different points of view to the topic of distributive justice. Even today, the different opinions can be seen in politics and the programmes of different parties show clear influences of those ideas. Clearly, some antagonistic ideas are included: Socialists are considered to be on the opposite end of the spectrum of the trade off 'taxation/welfare state' compared to the Libertarians. And Rawls' Egalitarian Liberalism shows that there is enough space for a position in between the extremes. The Utilitarians have been chosen due to the fact that their ideas are predestined to be included in mathematical models, although their position has been considered requiring improvement by all of the other three schools of thought. However, one cannot say that one of the schools is more 'right' than the others, since this is a matter of opinion.

The problem solved in this thesis, as described in Chapter 4, will be analyzed according to if the different schools would consider the solving procedure to be in accordance with a just distribution of disaster relief or not.

2.2.1 Utilitarianism

Utilitarian logic for the decision about what is morally right follows the "principle of utility" (Bentham, 2007 [1789], p.10), which favors increasing the balance of sum of pleasures minus sum of pain. For utilitarians, nature provides human being with these "[...] two sovereign masters, pain and pleasure." (Bentham, 2007 [1789], p.9), according to which our actions should be examined. Pain and pleasure should not be taken too literally, since they are umbrella terms for things like good or bad, profit or loss or whatever denomination those antagonists are prone to adopt (Bentham, 2007 [1789], p.14). This 'accounting of pain and pleasures' is not designed to stop at the individual, but should take the utility of the whole society into consideration. Mill (2007 [1861], p.19) describes the "utilitarian standard" as "[...] not the agent's own greatest happiness, but the greatest amount of happiness altogether;" and quickly discloses the question behind this principle: How can different pleasures or pains be evaluated to allow for calculation and offsetting each other?

Bentham (2007 [1789], p.13) gives a very structured answer to this crucial and controversial question. The value of a certain pleasure or pain is determined by its intensity and duration, by the degree of certainty that it will occur and how close or remote the pleasure or pain will be. Furthermore, the value will be a different one if it can be foreseen that the realization of the specific sensation will result in more of the same sensation in the future (the sensations fecundity) or by the opposite sensations (the purity of a pleasure or pain). This means that a pleasure will be worth more, if it initiates following pleasure, while its value is decreased if pain will follow. Last but not least, the amount of people affected determines the value alongside the other factors.

Another implication from the summing up of pains and pleasures and promoting the maximization of total pleasure in a society is that this can be accomplished via sacrificing the pleasure of certain members of society for the 'greater good'. This troublesome and controversial notion in utilitarian logic is acknowledged by Mill (2007 [1861], p.22):

> The utilitarian morality does recognize in human beings the power of sacrificing their own greatest good for the good of others. It only refuses to admit that the sacrifice is itself a good. A sacrifice which does not increase, or tend to increase, the sum total of happiness, it considers as wasted.

Mill further explains, how utilitarian morality perceives and itself defines justice. He admits that ideas of justice might be different for different people and situations, but a few concepts are recurrent: Injustice as violation of legal rights (where it has to be said that the rights themselves might be bad, like apartheid legislation) or breaking somebodies faith, justice as receiving what one deserves in an impartial way, everybody being treated equally. To Mill, all those opinions on justice are plausible in their own right, while for him, justice, just like utility, depends on the opinions of the members of society, thus not an absolute standard Mill (2007 [1861], pp. 35ff.). Therefore, the definition of justice for a certain society can be considered as "[...] something which is not only right to do, and wrong not to do, but which some individual person can claim from *us* as his moral right."(p.39 Mill, 2007 [1861], emphasis C.B.). Mill binds this moral right to the idea that society wants to punish anybody inflicting injustice in some form, which establishes a, what he calls, "perfect obligation", while an "imperfect obligation" would come without any right assigned to it, like generosity.

For sure the possibility to assign values to sentiments and facilitate comparison is very alluring, as will be shown in Section 4.2.

2.2.2 Libertarianism

Libertarians believe that the highest good is the personal liberty of an individual, and that nobody has the right to interfere with this liberty as long as it itself does not infringe the liberty of others. This results in the principle of self-ownership, as described in Nozick (2007, p.70). The special value of personal liberty comes from one of its properties that have a simplifying effect on questions of justice from a libertarian point of view: Everybody is responsible for his own happiness in life.

Equality, as a positive value, from a libertarian perspective, can only be the equality of opportunities, which means that everybody can possibly achieve what he wants if he or she works hard enough for it. Equality in outcomes, as promoted in socialism (see further down), which tries to alter the distribution of wealth or goods in a desired way thus necessarily interferes with the liberty of some members of society, since redistribution necessarily has to take from somebody, possibly against his will, and gives it to somebody else (Friedman and Friedman, 2007 [1980], pp.51ff.). The answer to the question if a certain distribution is just or not depends only on three conditions, which have nothing to do with the question of fairness. But fairness is not a criterion for justice, since, as Friedman and Friedman (2007 [1980], p.54) state: "Life is not fair.", while correcting this would first require to define 'fairness' in a universally valid way. This would be necessary to avoid the criticism of arbitrariness, since not only wealth, but also other things and abilities are distributed unequally, thus potentially unfair (Friedman and Friedman, 2007 [1980], p.53).

The three conditions (cf. Nozick, 2007, p.61) necessary for a just distribution are

- If the goods a person possesses have not been in the possession of somebody else before, then the principle of justice in acquisition has to hold, which defines what goods can be taken as property and how.
- If goods have been transferred from somebody else's possession into own possession, then the principle of justice in transfer has to hold, which makes sure that the

transfer resulted either from voluntary exchange of goods or services or they have been a gift. I.e. nobody has been forced to transfer goods.

- Finally, all goods held by anybody belong to either of the previous two categories

This definition of justice is very different from the utilitarian definition in two ways (Nozick, 2007, pp.62): First, it is a historical principle of justice, since it takes only into account if a distributive situation developed in a just way or not. Therefore, it does not depend on the end result as the utilitarian definition suggests. Second, it is not a patterned distribution, since no chosen pattern is considered as ideal for the distribution. A pattern always reflects the preferences of the person deciding on it, thus a pattern could be chosen very arbitrarily, while in the just distribution according to libertarians, only the entitlement to ones possessions counts.

Since "[... t]he minimal state is the most extensive state that can be justified" (Nozick, 2007, p.60) and every redistributive action results in basically assigning rights over people (in terms of their tax money, which translates into their own possession) since it was not given under the principle of justice in transfer, every act of redistribution, even in case of urgent need, is to be considered unjust, unless the means are given as a gift.

2.2.3 Egalitarian Liberalism

John Rawls' conception of justice plays the central role in his reasoning, where every person profits from an inviolability that is based on this concept of justice. While, in the case of the libertarians, their definition of justice was also in the core of their reasoning, they made it dependent on their core value of liberty, which was not to be restricted in any way. Rawls, on the other side, has a more differentiated view on his core value: Although he also states that violating the freedom guaranteed by justice cannot be undone by any net gain of the social greater good (contradicting the utilitarians), he found a way to take the opinion of the members of society into the process of defining what is just by establishing a social contract between all members.

Unfortunately, this can only work on a hypothetical basis, but this gedankenexperiment has a clear advantage: Assuming that every member of the society can stand for his ideas and has equal power in the decision making is the reason why Rawls calls his principle

"justice as fairness" (Rawls, 2007 [1971], pp.203ff.). The idea behind it is that the rational members of society have to choose their standard of justice without any knowledge of their position in the society after the decision. This means that people choose their standard of justice without any knowledge of crucial information such as their place in the hierarchy, the distribution of the natural assets as well as their own abilities, which corresponds to a fair initial situation where the decision of all could affect all in its most advantageous or disadvantageous way possible. This decision making without knowledge in a fair initial situation became known as choosing behind the "veil of ignorance" (Rawls, 2007 [1971], pp.204f.).

The result is a situation of immense insecurity for everyone. Not knowing if one would be poor or rich, intelligent or less gifted provides the basis for two decision principles: On the one hand, the members of society will tend to distribute the rights and duties equally across society, and on the other hand, inequalities in wealth and authority might be accepted, but only on the condition of compensating benefits for the rest of society while everybody can possibly reach the positions tied to and profiting from the inequality (Rawls, 2007 [1971], p.214).

2.2.4 Socialism

As Marx and Engels (2012 [1848]) show, their political idea of socialism is founded on a very specific understanding of how the current societal order has been formed and how it will develop in the future. They do not so much define what is just as they reason and explain what they think will happen with our society as a whole in the future. Borrowing from Mill, who stated that what is just is always defined by society, here a different definition of justice has to be expected from a society that eventually developed as they predicted.

In Marx's and Engel's view, the capitalist production system will divide the society as a whole into two antagonistic categories, into the proletariat and the bourgeoisie, the former, the oppressed, consisting of the labor force necessary for sustaining the capitalist system of growth, while the latter, the oppressor, provides the means to work, the capital, and receives the added value. The proletariat will eventually revolt against this situation, in which the value they added, flows to the owners of the capital, characterizing the term

of "class struggle" (Marx and Engels, 2012 [1848], p.251). For Marx and Engels, in the aftermath of the takeover of the communists, society will change:

> In place of the old bourgeois society, with its classes and class antagonisms, we shall have an association, in which the free development of each is the condition for the free development of all. (Marx and Engels, 2012 [1848], p.258)

This egalitarian thinking has an enormous effect on distributive justice: While the individual (property) rights are curtailed by heavy taxation of the wealthy, reducing the possibility to inherit values to the following generations and generally confiscation of property through the state, the redistribution of the so acquired goods and values follows a different principle of distributive justice than the collection. The association as the basis of economic activity determines the individual working times and the contribution required by the individuals, and distributes the produced wealth according to the individual contribution. For those who are not able to contribute, a collective fund (e.g. the state) has to provide support as well as operate all infrastructure, be it health care related, public traffic, education or something else (Berger, 2004, pp.60). In the "Critique of the Gotha Programme", Marx wrote: "From each according to his ability, to each according to his needs!" (Marx, 1875), which sums up the conception of distributive justice employed.

While this conception of distributive justice suggests wide assistance to people in need, another factor of Marx' and Engels' reasoning might suggest the denial of help in special situations of man-made disasters. The authors show their despise for the "conservative socialists" which consist of the "[...]economists, philanthropists, humanitarians, improvers of the condition of the working class, organisers of charity, [...]" (Marx and Engels, 2012 [1848], p.263), thus everybody ameliorating the situation of class struggle to prevent the communist takeover. This, of course, has no influence on natural disasters, but demonstrates a moral dilemma that socialism faces in its reasoning.

Finally, Table 2.1 summarizes the findings of this section on justice.

Concluding, one can summarize the ideas of the different schools in a brief way. While Utilitarians define justice through a accounting system of pains and pleasures and favor a definition by moral right, Libertarians define a principle which cannot be subordinated to anything and also defines justice: the personal freedom. A just distribution is only achieved if the acquisition of a good is just. Egalitarian Liberalism qualified this absolute

	Utilitarian	**Libertarian**	**Egalitarian Liberalism**	**Socialism**
Main Thinkers	J. Bentham, J.S. Mill	R. Nozick, F.A. Hayek, M. Friedman	J. Rawls	K. Marx F. Engels
Leading Principle	Utility (Sum of pain & pleasure)	Personal liberty	Justice as Fairness	Class struggle
Definition of distributive Justice	not absolute, moral right to it	Acquisition and transfer	Social contract	egalitarian, needs-based
Should State provide disaster relief	only if net gain in utility	no	yes	yes

Table 2.1: Four Schools of Thought in Justice and their core ideas (Source: own table)

statement by introducing a notion of democracy into justice, projecting the single human being into society as a whole by forming a social contract. The same society point of view is used by Socialists, but from a very different angle. Their Historical Materialism describes the environment in which people live, the society including the material standards that people have at their disposition, as given through a historic development process formed by class struggle that turns society one day into its socialist form. Justice, therefore, is, what would be considered just in the socialist society.

Concerning the employment of disaster relief, just Rawls' Egalitarian Liberalism and the Socialists would support it without concerns, while Libertarians cannot support it if financed by taxes, while Utilitarians would only support it, if taking away money through taxation and financing disaster relief would increase the total level of utility.

Chapter 3

The Methods

In this chapter, the methods used to solve the problem described in Chapter 4 are described in greater detail. These methods are established concepts and have proven to be very useful for various application possibilities, while the problem solution presented in the next chapter is, to the best knowledge of the author, a new approach on location planning in this field.

3.1 Multiobjective Optimization

In Humanitarian Logistics, decision makers can and most likely will face different objectives on which they base their decisions, as was the case in the problem of Chapter 4. Due to limited financial resources it might be advisable to keep the costs of operations to a minimum. At the same time it will be required to provide as many people as possible with assistance. Other objectives are as well possible, but since the objectives of this thesis coincide with the just mentioned two, the methods for multiobjective optimization shall be examined using this illustrative example.

In the case of cost and served inhabitants, the problem is that the more people can be served, the higher the costs tend to be, which shows that the goals in this case (i.e. minimizing cost, maximizing served inhabitants) are conflicting. For easier depiction in the course of this section, the goal of 'maximizing the number of served inhabitants' can be easily changed into 'minimizing the number of unserved inhabitants' without loss of

any information, assuming that the total population in the region is known.

The problem can now be defined like it was presented in Marler and Arora (2004, p.370):

$$\begin{aligned} &Minimize\ \boldsymbol{F(x)} = [F_1(x), F_2(x), ..., F_k(x)]^T \\ &subject\ to\ g_j(x) \leq 0,\ j = 1, 2, ..., m \\ &h_l(x) = 0,\ l = 1, 2, ..., e \end{aligned} \tag{3.1.0.1}$$

k in this respect denotes the number of objective functions, e and m the numbers of equality and inequality constraints, respectively, $\boldsymbol{x} \in E^n$ denotes a vector of decision variables with n independent variables x_i. $\boldsymbol{F(x)} \in E^k$ is a vector of objective functions $F_i(x)$.

$\boldsymbol{X}$ is the feasible decision space, and respects the equality and inequality constraints, and $\boldsymbol{Z}$ is the objective space and is defined as $\{\boldsymbol{F}(x)|x \in \boldsymbol{X}\}$. The difference between the feasible decision space and objective space is that the first consist of the space that is basically attainable, but only the second refers for sure to a possible realization, a point $x \in \boldsymbol{X}$ that can be realized as well.

What options are there to compare the solutions of a multi-objective problem? Basically two types: either, one follows a scalarization method or a vector optimization method (Marler and Arora, 2004, p.371). If the scalarization method is employed, some sort of a priori preference articulation has to happen, since the preferences are used to transform the objective functions values into on single scalar to make the solutions comparable. Common approaches contain for example the weighted global criterion method, in which parameters are used to model preferences and then the weighted exponential sum of the objective functions is used. (Marler and Arora, 2004, pp.373ff.).

If the vector optimization method is used, the concept of Pareto dominance comes into play. Two objective vectors z_1 and z_2 can be compared according to the different objectives values. If $z_1 \prec\prec z_2$, then z_1 strictly dominates z_2, which means all objectives of z_1 are better than z_2. If $z_1 \prec z_2$, then z_1 dominates z_2, which means that it is not worse in all objectives but better in at least one, and if $z_1 \preceq z_2$, then z_1 weakly dominates z_2, which, in turn, means that z_1 is not worse than z_2 in all objectives. In case $z_1||z_2$, then

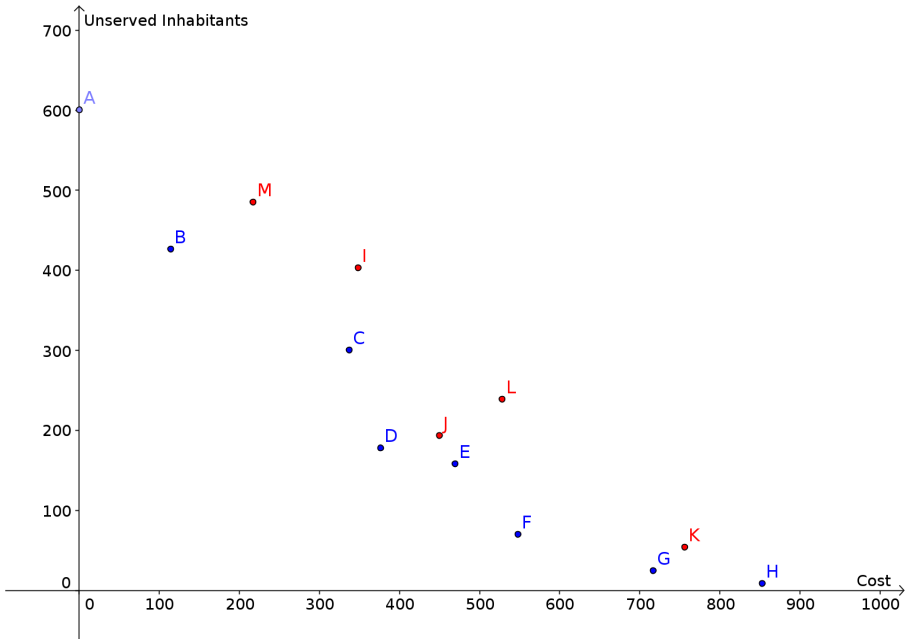


Figure 3.1: Exemplary Paretofront
(Source: own graph)

z_1 is incomparable to z_2 since it is neither $z_1 \preceq z_2$ nor $z_1 \succeq z_2$, and in the special case of $z_1 \sim z_2$, all objectives of z_1 are the same as of z_2. The minimal values according to these relations are Pareto optimal, and all Pareto optimal objective vectors together form the Pareto-optimal front. (Knowles et al., 2006, p.4).

Figure 3.1 is a good example for a pareto-optimal front in a setting with two objectives, cost and unserved demand in a minimization problem. Here, the objective vectors A to H constitute the pareto-optimal front, since none of these objective vectors is dominated by any of the other vectors, while the vectors I to M are all at least dominated by one other vector. Therefore, the vectors A to H form the pareto-optimal front.

If a heuristic provides the decision maker with a large set of objective vectors, and another heuristic provides the decision maker with a different set of objective vectors on the same problem instance, then a measure to compare the power of the heuristics is required. One such quality indicator is the Hypervolume, as presented first in Zitzler and Thiele (1999). This measure requires a bounding reference point that is at least weakly dominated by each other objective vector. In Figure 3.2, this reference point is I. The field spanning from I to A, B, C,D, E, F, G and back to I is the portion of the objective space that is weakly dominated by the set A to G. The volume of this field can be compared to

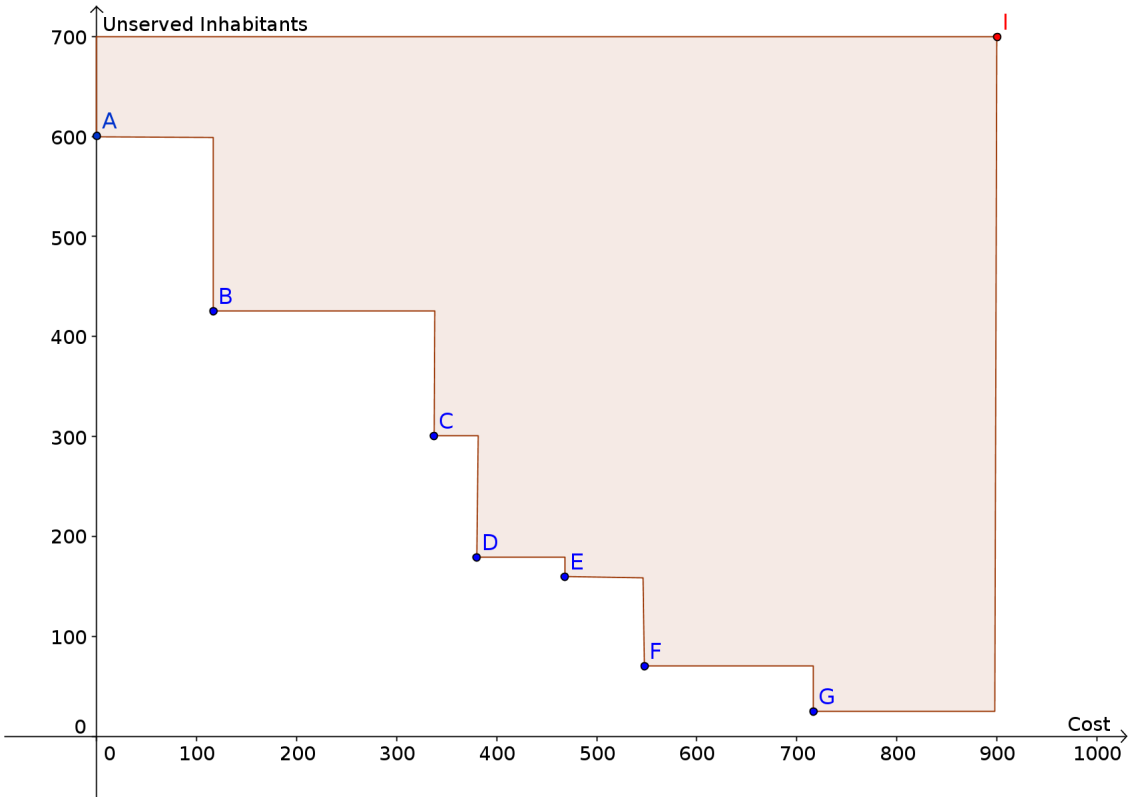


Figure 3.2: Explanatory example for the Hypervolume Indicator in the two-objective setting (Source: own graph, based on Knowles et al. (2006, p.7))

the volume of the field constructed with the same method on the pareto-optimal front from the other heuristic and serves as a unary quality indicator, which has to be maximized (Knowles et al., 2006, p.11). Another application would be to compare the hypervolume of a heuristic to the hypervolume of the exact solution, as can be seen in Figure 3.3.

3.2 Optimization with NSGA II

In this section, the Nondominated sorting genetic algorithm 2 (NSGA II) is explained. This is a heuristic multiobjective evolutionary algorithm (MOEA) developed by Deb et al. (2002) that helps finding potentially optimal non-dominated fronts in optimization problems with multiple objectives. Since the algorithm is a standardized mechanism, the whole section is based on the aforementioned paper from Deb et al. (2002) apart from the addition of a special archive at the end which will be described later. Genetic Algorithms mimic the behavior of evolution by creating 'offspring' generations from the basic population selected randomly by genetic operations like mutation, where one small change

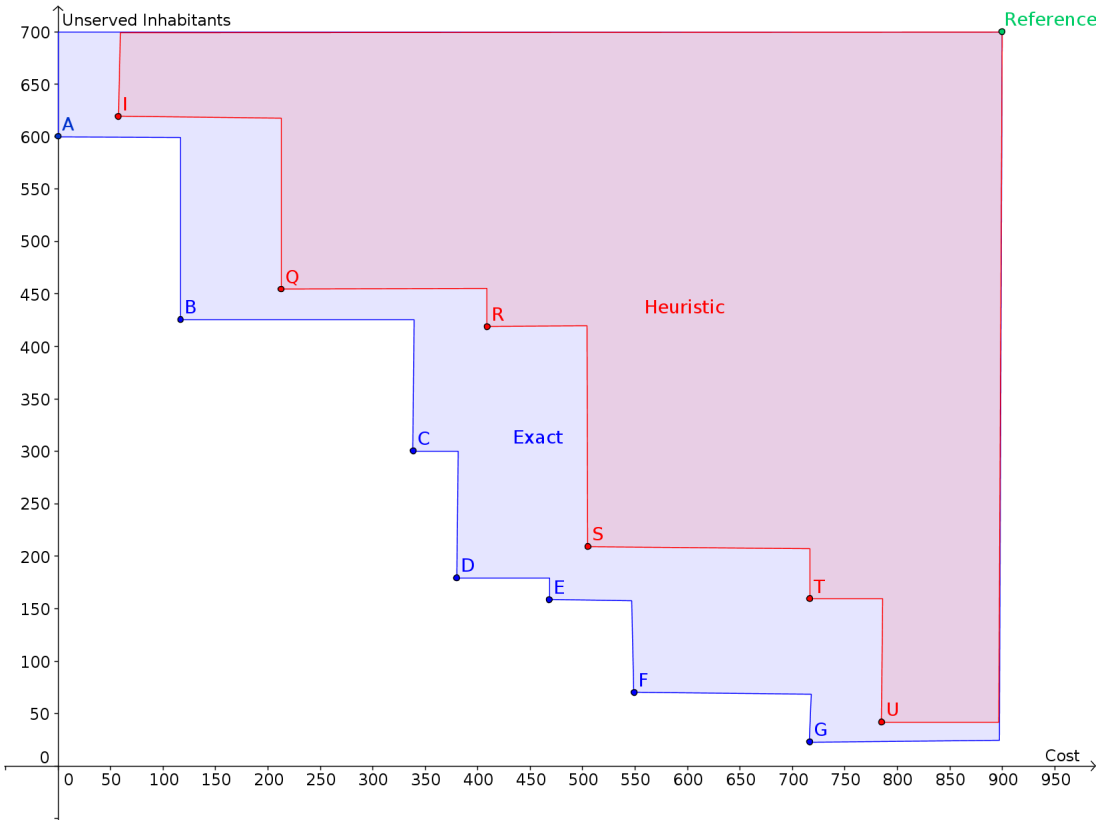


Figure 3.3: Explanatory example of Hypervolume Comparison
(Source: own graph)

is applied to the decision variables and crossover, where segments of different solutions sequence of decision variables at a crossover point are exchanged crosswise just like the process employed in nature to produce new DNA. The offspring generation is then combined with the basis of it, the 'parent' generation, and then the best of both are selected in a type of 'survival of the fittest' to form the next parent generation.

The algorithm has some valuable advantages: It is fast since it reduces the computational complexity O (i.e. the tasks (e.g. comparisons) that have to be carried out) to $O(MN^2)$ with M objectives and N population size, it does not need any parameters to achieve more evenly distributed results since it has a diversity preserving mechanism included, while it uses elitism to find better solutions in a quicker way, which means that better solutions are preferred in the genetic algorithm over worse performing solutions.

One crucial feature defining the efficiency of the NSGA II is the unique way of non-dominated sorting as described in Algorithm 1

Algorithm 1 Nondominated Sorting (Deb et al., 2002, p.184)

1: **for all** $p \in P$ **do**
2: $S_p = \emptyset$
3: $n_p = 0$
4: **for all** $q \in P$ **do**
5: **if** $p \prec q$ **then**
6: $S_p = S_p \cup \{q\}$
7: **else if** $q \prec p$ **then**
8: $n_p = n_p + 1$
9: **end if**
10: **if** $n_p = 0$ **then**
11: $p_{rank} = 1$
12: $F_1 = F_1 \cup \{p\}$
13: **end if**
14: **end for**
15: **end for**
16: $i = 1$
17: **while** $F_i \neq \emptyset$ **do**
18: $Q = \emptyset$
19: **for all** $p \in F_i$ **do**
20: **for all** $q \in S_p$ **do**
21: $n_q = n_q - 1$
22: **if** $n_q = 0$ **then**
23: $q_{rank} = i + 1$
24: $Q = Q \cup \{q\}$
25: **end if**
26: **end for**
27: **end for**
28: $i = i + 1$
29: $F_i = Q$
30: **end while**

In the presented nondominated sorting, for each solution p in the population P a set of other solutions S_p is defined which are all dominated by the solution p and a domination count n_p is also calculated which counts how many solutions in P are dominating the solution p. To achieve this set S_p and count n_p, every solution p is compared with every other solution q from the same set P, and if p dominates q, then the solution q is added to the set of solutions dominated by p, S_p. If, on the other hand, q dominates p, then the domination counter of p, n_p is increased by one. Recall that it is also possible that neither of the two solutions dominates the other (see Section 3.1).

After all comparisons have been conducted, the sets S_p and the counts n_p for each member p of P describes which solutions are dominated and how many dominate the solution p. Now, every solution p that has a domination count of $n_p = 0$ is in the first, nondominated front of solutions, forming the set of best solutions from P. The remaining question is, which of the remaining solutions with $n_p > 0$ belong to which fronts so one can easily compare a solution by the rank of the front it belongs to. For this purpose, the front counter i is used. After the comparisons, every solution with $n_p = 0$ has a p_{rank} of 1, since this solution belongs to the first front F_1.

After the first front has been filled, the next fronts are filled, as can be seen in lines 17-30 of Algorithm 1. This starts with initialization of the front counter i to 1. Every new front, a storage space for the members of the next front, Q is set up and is initially empty. Now, for every member in the front number i, F_i, all solutions $q \in S_p$ dominated by these solutions $p \in F_i$ get their domination count n_p reduced by 1. If the respective domination count is reduced to 0, then this solution q belongs to the next front and the rank of it is q_{rank} is $i + 1$ and the solution is included in the set of the next front Q. So, after iterating through all members of the current front F_i, a new front Q is filled with all members of P that were only dominated by the members in the current front, and therefore compose the new nondominated front of the remaining solutions. The front counter i is increased by one and the new front F_i is filled with the members of Q. This sorting is conducted until there is no more solution in the next front F_i.

This algorithm is a fast way to determine which solution belongs to which front and thus enables the user to sort solutions which are characterized by their multiple objectives according to the rank of the front.

It was already described that the NSGA II is a very good algorithm since it has a mechanism built in that preserves the diversity of the solutions. This is done by the crowding distance assignment operation described in Algorithm 2. The basic intuition of this approach is that a set of solutions is better, if the single solutions' objective values are more evenly distributed. This is especially advantageous if a genetic algorithm creates new offspring generations, the algorithm will create also more diverse offspring if the parent generation is more diverse, and thus increases the probability of not searching in the same neighborhood all the time.

Algorithm 2 Crowding Distance Assignment (Deb et al., 2002, p.185)

1: $l = |I|$
2: **for all** i **do**
3: $\quad I[i]_{distance} = 0$
4: **end for**
5: **for all** m **do**
6: $\quad sort(I, m)$
7: $\quad I[1]_{distance} = I[l]_{distance} = \infty$
8: $\quad$ **for** $i = 2$ **to** $i = (l - 1)$ **do**
9: $\quad\quad I[i]_{distance} = I[i]_{distance} + (I[i + 1].m - I[i - 1].m)/(f_m^{max} - f_m^{min})$
10: $\quad$ **end for**
11: **end for**

The crowding distance assignment algorithm calculates for every solution i in a front I the total sum of normalized distances to their upper and lower neighbors. For each objective m, I is sorted. The solutions at the extreme ends are assigned indefinite crowding distance since they are the extreme points from a diversity perspective and should be included in new parent generations with the highest probability. For the intermediary solutions, the normalized distances are summed up over all objectives m. Figure 3.4 shows what is meant by this: On both objectives of this example, for a solution i, the objective values of the prior solution $i - 1$ are subtracted from the objective values of the next solution $i + 1$ and then divided by the range of the values (see line 9 in Algorithm 2).

These crowding distance values for the single fronts can be used to compare the solutions in a front. Therefore, a solution a is better than a solution b ($a \prec_n b$) if

- the rank of a is lower than that of b: $a_{rank} < b_{rank}$, which means that a is in a better front, or

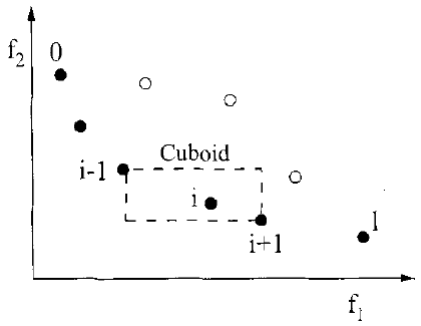


Figure 3.4: Crowding Distance Calculation
(Source: Deb et al. (2002, p.185))

- when $a_{rank} = b_{rank}$, then the crowding distance of a has to be bigger: $a_{distance} > b_{distance}$

Now, since the core parts of the NSGA II have been described, the main loop of this genetic algorithm can be explained. A graphical description can be found in Figure 3.5, while the Algorithm itself is depicted in Algorithm 3.

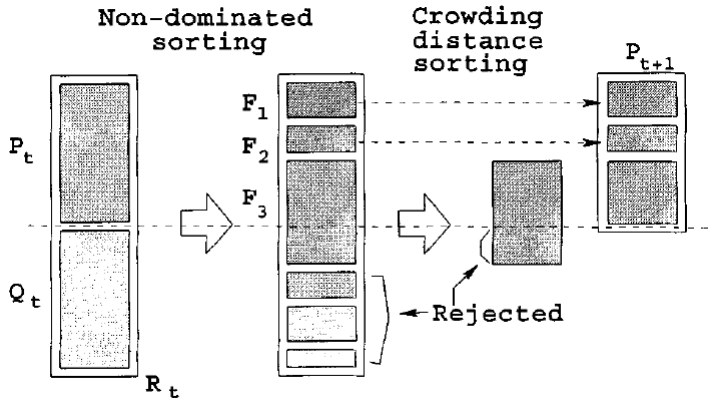


Figure 3.5: NSGA II Procedure
(Source: Deb et al. (2002, p.186))

At first, the parent and offspring generations of period t are combined (P_t and Q_t). Then, the nondominated sorting is executed with this large set. Here, this thesis inserts an archive operation to recover all solutions that are not dominated at the end of the algorithm's runtime by adding the first nondominated front to the archive, sorting it subsequently and then using only the not dominated solutions in next periods archive.

Now the regular NSGA II continues by filling in a new parent generation with full fronts until the last front does not fit entirely without exceeding the size limit N of the parent

generation. This last front F_i, which did not fit entirely, is now used for calculating the crowding distance of it so that the parent generation P_{t+1} can be filled up to the specified number N with those members of F_i, which have the highest, i.e. best, crowding distance number. Then, the rest of F_i is rejected just like the rest of R_t, and a new offspring generation Q_t is calculated using selection, crossover and mutation as described. Since a full iteration of the algorithm is now finished, the iteration or period counter t is increased by 1.

Algorithm 3 Main Loop, adapted from (Deb et al., 2002, p.186)

1: $R_t = P_t \cup Q_t$
2: $F = Nondominated\ Sorting(R_t)$
3: **for all** $i \in R_t$ **do**
4: **if** $i_{rank} = 1$ **then**
5: $Archive_t = Archive_t \cup i$
6: **end if**
7: **end for**
8: $X = Nondominated\ Sorting(Archive_t)$
9: $Archive_{t+1} = X_1$
10: $P_{t+1} = \emptyset$
11: $i = 1$
12: **repeat**
13: $Crowding\ Distance\ Assignment(F_i)$
14: $P_{t+1} = P_{t+1} \cup F_i$
15: $i = i + 1$
16: **until** $|P_{t+1}| + |F_i| \leq N$
17: Sort$(F_i, \prec_n)$
18: $P_{t+1} = P_{t+1} \cup F_i[1 : (N - |P_{t+1}|)]$
19: $Q_{t+1} = Evolutionary\ Operations\ Creating\ Offspring(P_{t+1})$
20: $t = t + 1$

The main loop can be seen also in Figure 3.5, but the archive operation are not included in this figure, they would fit in after the Non-dominated sorting. Using this approach results in choosing the best solutions and solutions that show high diversity for the next round of evolutionary operations to improve the performance of finding good solutions.

3.3 Exact Solutions with Complete Enumeration

For small instances of problems with binary decision variables (opening of DC), two objectives and a discrete set of potential DCs, an exact computational solution can be calculated

by iterating through all possible combinations of opening patterns. This approach is called Complete Enumeration. For Competitive Location Models, Drezner and Drezner (1998) describe a "Brute Force" heuristic for solving the problem to find the best location for a new facility under competition, which works similarly. However, in this case a continuous location model is used, i.e. the demand occurs at discrete locations and the new location can be established at any place using Cartesian coordinates (Daskin, 2008, p.284). So for Drezner and Drezner (1998), a grid of possible locations is created, while in this case here, already a discrete number of solutions is defined. Then, the objective functions values for every point on the grid are calculated, which translates into our situation as checking the objective functions values of all possible combinations.

This extensive computation effort is the reason why this exact solution is not advisable for bigger instances. For example, the number of possible combinations for 6 DCs accounts to 2^6 combinations, totalling to 64 combinations. By simply doubling the number of potential DC locations, one gets $2^{12} = 4096$ combinations, so by doubling the size, one results in an amount of combinations that grew with the power of two, by tripling the number, one results in growth to the power of three in combinations.

The Complete Enumeration algorithm for a two-objective optimization problem uses the concept of Pareto dominance also used in the NSGA II. The algorithm runs like the following Algorithm 4 and requires a set A, the archive with its members a which is initially empty, and the set P of all possible solutions p.

In the algorithm, for every possible solution p, the objective function values cost and unserved demand are calculated. Then a domination count is initialized with 0 and now, the comparison with all elements of the archive set A begins. As long as the domination count stays 0, the comparison with the members of the archive continues. If the new element p dominates the current archive solution a it is compared with, then a is deleted from A and the comparison process continues. If p, however, is dominated by a, then the domination count is raised to 1. Now the comparison is stopped. If after the comparison loop the domination count remained to be 0, this solution p has not been dominated by any solution in the archive and therefore is added to the archive A. Now, the next potential solution p is compared in the same way until all potential p have been examined.

In the end, the archive consists of the real Pareto front since all possible combinations

Algorithm 4 Complete Enumeration

```
1: for all p ∈ P (2^j) do
2:    Calculate Unserved
3:    Calculate Cost
4:    domCounter = 0
5:    for all a ∈ A do
6:       while domCounter = 0 do
7:          if p ≺ a then
8:             delete a from A
9:          else if a ≺ p then
10:            domCounter = 1
11:         end if
12:      end while
13:      if domCounter = 1 then
14:         Break the For Loop, continue line 17
15:      end if
16:   end for
17:   if domCounter = 0 then
18:      A = A ∪ {p}
19:   end if
20: end for
```

have been checked and all belong to the non-dominated front.

The presented concepts will be used in the following chapter presenting the problem considered and the solution procedure that has been developed for this specific problem type.

Chapter 4

The Problem

In this chapter, first the concepts that build the basis for the problem will be explained in Section 4.1, the Covering Tour Problem and the Competitive Location Model. Then, in Section 4.2, the problem of designing opening patterns of locations and the routing of delivery trucks will be presented. The problem will be solved showing the minimal decision requirements of decision makers: First, after calculation of the capacities of the DCs, a decision on parameters has to be made by which the attractiveness of the potential DCs is calculated, then a variation of the Covering Tour Problem (CTP), a capacitated CTP with competitive demand allocation, is used to find a non-dominated front of solutions regarding cost and unserved demand and the corresponding opening pattern before, finally, the decision maker can use the resulting non-dominated front of solutions (or, for complexity reasons, a subset of the non-dominated front) to choose from according to his/her resources. An overview of this problem solution is depicted in Algorithm 5, which specifically emphasizes the necessary decisions from the decision maker coordinating the logistics operations. Finally, in Section 4.3, the problem solution will be assessed using the concepts of justice presented in Chapter 2.

Algorithm 5 General Overview of the presented solution procedure with special focus on the Decision Makers Tasks

Data Input (e.g. transport network, possible DCs etc.)
Potential Parameter Decisions (see Section 4.2.1)
Capacity Calculation (see Section 4.2.2)
Allocation Parameter Decision (see Section 4.2.3)
Multiobjective Optimization Heuristic / Exact Algorithm
Decision Maker's Choice

4.1 Background

Two basic concepts that work together in this thesis are presented with their theoretical background to facilitate the understanding of the core concept presented later.

4.1.1 Competitive Location Model

The idea to determine the location of a new facility based on the preferences of the customers is not new at all. Already Hotelling (1929, p. 45) described, concerning two sellers situated near each other on a one-dimensional market, possibly a main street in a town: "No customer has any preference for either seller except on the ground of price plus transportation cost." Since transportation cost is defined by the distance of the buyer to the seller, on a one-dimensional market always the nearer seller will be chosen. With this paper, Hotelling was the first to describe that only distance has an influence on buyers decision as long as both seller are equally attractive. After Hotelling, Huff (1966) was a major contributor to the development of the field of Competitive Location Models by introducing gravity models into competitive demand allocation. It is based on four propositions (Huff, 1966, p.294):

- The probability that a consumer shops at a certain place:

  1. The probability P of a given alternative j being chosen from among all alternatives in the subset J_0 is proportional to u_j [the utility that the alternative provides the consumer with, comment C.B.]. That is,

$$P_j = \frac{u_j}{\sum_{j=1}^{n} u_j} \tag{4.1.1.1}$$

  such that, $\sum_{j=1}^{n} P_j = 1$; and , $0 < P_j \leq 1$. (Huff, 1966, p.294)

- Ratios of consumers choosing alternative A relative to alternative B depend only on the utilities of the two alternatives and not on the existence of other alternatives. This, of course, is only true for the relations *between* different alternatives, and not in absolute terms.

- The utility of an alternative depends on its size and distance to the customers loca-

tion.

- The utility of an alternative can be calculated by dividing its size with the distance to the power of a constant λ.

With this last proposition, the distance decay function, here in the form of a power decay, is introduced. This concept of decreasing utility / attractiveness with the distance between consumer and seller is especially important in the context of competition between multiple sellers for the demand of the consumers in geographical space. This, however, assumes that every consumer has to go to a seller and cannot choose to stay at home.

Therefore, the type of Competitive Location Model employed in this thesis stems from a very recent development in the area employing at the same time a gravity based model using attractiveness measures and a distance decay function similar to the model proposed in Huff (1966), but is extended to account for the option that consumers are not willing to go anywhere ("lost demand"), since the option of not going has an attractiveness level of itself as well, resulting in a probability that 'not going' is chosen (Drezner and Drezner, 2012).

In this paper, a "Dummy location" is introduced that has a certain attractiveness measure α and a distance D assigned for the dummy facility. Therefore, the utility value β used to determine the proportion of lost demand in a PC can be calculated when assumptions concerning the attractiveness level α, the dummy distance D and the constant distance decay function parameter λ are made, since here the relation is

$$\beta = \alpha f(D) \tag{4.1.1.2}$$

with $f(D)$, the distance decay function, described as a preferably exponential decay (Drezner and Drezner, 2012, p.202):

$$f(D) = e^{-\lambda d} \tag{4.1.1.3}$$

where d describes the distance, here between customer and dummy (fixed), in the case of a real facility the real distance between seller and customer.

An adapted version of the model of Drezner and Drezner (2012, p.203) can be used in this thesis to define the ratio of potential demand allocated to a certain DC in reach (within dummy distance D) or to be considered as 'lost demand'. The adaptation is necessary, since the original model considers also opened locations of competing sellers, which results in competition not only from other opened DCs of ones own operations, but of other companies as well. This is not considered in the case of disaster relief operations of this thesis.

However, the adapted version of the model of Drezner and Drezner (2012) can be formulated as:

$$A_{jh} = \frac{a_j f(d_{hj})}{\sum\limits_{j \in N_{h_j}} (a_j f(d_{hj})) + \beta} \tag{4.1.1.4}$$

As we can see, the utility measure A_{jh} responsible for the ratio of demand of the Population Centers (PC)s total population p_h allocated to DC j depends on the attractiveness measure a_j of the DC in question and is decreased proportional to the sum over all DCs j in reach of a certain PC h (N_{h_j}, which is further described in Section 4.2.1) of attractiveness measures multiplied with the respective distance decay function's value. To increase the denominator further, the utility measure for the dummy, β is added as well. Therefore, if the value of β is greater than 0, for sure some proportion of demand will not be covered in any circumstances.

Figure 4.1: Explanatory example for the Competitive Location Model's calculation of demand allocation

(Source: Own figure)

Figure 4.1 provides a good example for the calculation of demand allocation ratios between the points DC_1 and DC_2. Assuming that the attractiveness level of the dummy, α is equal to zero, which eliminates β in the formula and the attractiveness for the DCs

is given by their respective capacity, $a_1 = 2000$ and $a_2 = 5000$, and $\lambda = 0.001$. Then the ratio of inhabitants of the PC going to DC_1, assuming that only these two DCs are within D of the PC, is given by

$$A_{1PC} = \frac{2000 * e^{-0.001*2000m}}{2000 * e^{-0.001*2000m} + 5000 * e^{-0.001*3500m} + 0} = 0.642 \tag{4.1.1.5}$$

Due to the unfavorable distance decay function parameters for larger distances, approximately two thirds of the demand from the PC will be allocated to DC_1 and only a little bit more than one third in DC_2.

Concluding, this model heavily depends on the choice of the factors α, D and λ, which determine the ratio of dummy attractiveness to attractiveness of other alternatives and the behavior of the distance decay function. The greater λ, the less attractive a DC seems to be for a PC even if the distance between them, d_{hj}, stays the same.The greater α, the higher the ratio of lost demand, with the reverse relation being true for the dummy distance D. Therefore, the choice of these parameters is a very sensitive decision, as becomes clear in Section 4.2.3.

4.1.2 Covering Tour Problem

The Covering Tour Problem (CTP) is a routing problem which uses the concept of not directly visiting some nodes of an undirected graph but nevertheless considering them as "covered" when a visited vertex is within a specified covering distance. Gendreau et al. (1997) describe the CTP as a problem, in which a set V of vertices can be visited, a subset of the vertices V,T has to be visited and the subset W of the vertices V may be visited, but must be covered. They emphasize that the CTP is especially useful in designing bilevel transportation networks, since in cases where there is a certain degree of mobility available from other parties involved in the transportation process, e.g. people bringing their letters to post boxes, travelers may drive some distance to the next big airport etc., it is not necessary that all nodes have to visited. Prior to Gendreau et al. (1997), Current and Schilling (1989) presented the same problem under the name of Covering Salesman Problem, although here, all vertices that are not visited must be within the coverage distance S. Figure 4.2 shows a solution of a problem instance, presented by Current and Schilling (1989).

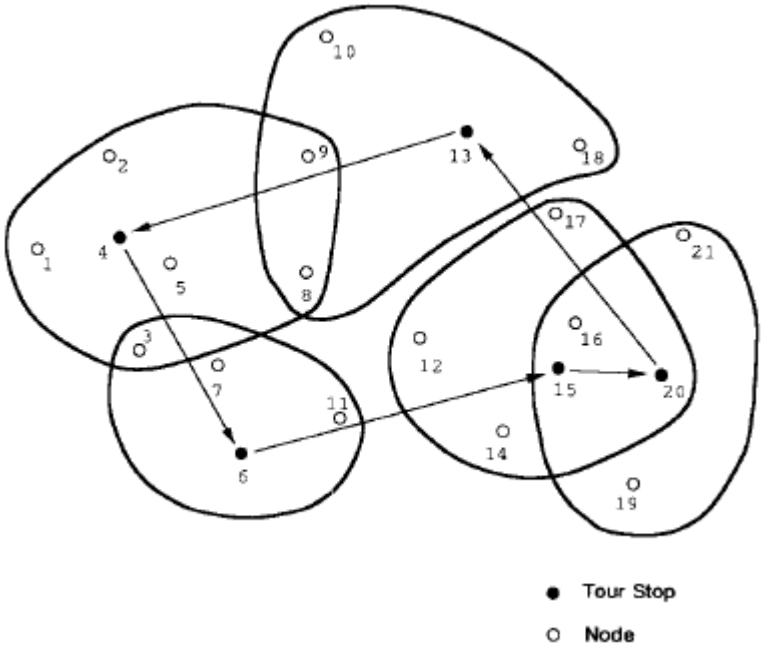


Figure 4.2: Covering Salesman Problem
(Source: Current and Schilling (1989, p.211))

Today, CTPs are used frequently in disaster relief operations, since it can be assumed that beneficiaries can traverse some distance to collect relief goods from location centers. Examples for this application can be found in Tricoire et al. (2012), where disaster relief operations in Senegal are considered and Nolz et al. (2010), where the water distribution after a disaster in Aceh, Indonesia, has been modeled.

Considering the CTP used in this thesis, some differences to the presented 'standard' CTPs have to be pointed out. The visiting or covering of a PC in the context of the Competitive Location Model does not mean that all possible demand is allocated to the respective DC. Since this model provides the inhabitants with the opportunity to choose between potentially different DCs and not going at all, the allocation of demand behaves quite different than in usual CTPs. Furthermore, the existence of more than one DCs within the sphere of influence distance of a PC results in splitting the demand according to the attractiveness measures of the DCs for the PC. Considering these differences, the calculation of objective functions values is different, but the task of the CTP remains the same: Finding which nodes to visit (i.e. which DCs to open) so that costs (amongst others for the routing of vehicles) and the amount of unserved demand becomes minimized.Furthermore, since the sphere of influence denotes a maximum distance over which

an opened DC can affect the decision making of inhabitants in a PC, this sphere of influence determines the covering distance in usual CTPs.

4.2 The Problem Description

This section is split up in five subsections. First, the parameters and variables used are explained and the notation described, then the required calculations prior to the actual problem are explained in detail. Subsequently, solving the problem on which opening pattern of DCs to choose is explained in three sections, following the decisions necessary from the viewpoint of a decision maker. Here, the methods from the previous chapter are employed.

4.2.1 Parameters and Variables

The following problem has many similarities from the viewpoint of the structure of the problem with the problem proposed in Tricoire et al. (2012), although the model employed there, a bi-objective two-stage stochastic optimization problem with recourse, differs from the approach in this thesis. The similarities result from the same problem structure: The determination of DCs in a disaster relief situation with multiple vehicles, tour stops that correspond to the DCs with attached opening costs, inhabitants that can travel to the DCs if they are in proximity of it and decreasing numbers of inhabitants visiting DCs with increasing distance to the DC. This is the reason why the notation of the problem in this thesis is based on the notation in Tricoire et al. (2012).

The problem requires two complete undirected graphs $G = (V_0, E_1)$ with edges E_1 and nodes $V_0 = V_1 \cup 0$, where 0 is the depot and V_1 the set of DCs. However, all potential DCs are at the same time Population Centers (PCs), while this is not necessarily true in the other direction. This means that G is the graph on which vehicles can be routed between the depot and all of the DCs, but this does not necessarily include all PCs. Therefore, a second graph $U = (V_2, E_2)$ is needed, where V_2 determines the PCs and the edges E_2 determine the pedestrian network on which inhabitants can walk to the DCs. Every PC has a certain maximum demand, namely the total population p_h assigned to it.

The index h describes a PC (i.e. $h \in U$), no matter if it is a DC at the same time or

not, while the indexes i and j only describe DCs (with nodes $j \in U$ and nodes $i, j \in G$, since DCs are PCs as well). Consequently, d_{hj} and d_{ij} describe the walking and driving distance to the DCs respectively.

In the problem, a single commodity (e.g. a CARE package), will be delivered by a fleet K consisting of homogeneous vehicles, each vehicle $k \in K$ having the same capacity Q. Delivery will take place in the same tours every fixed time period, in our example one week was chosen.

The demand W_j at the DCs is calculated in the second step of the problem solution after the second decision, the DC opening decision and uses the attractiveness resulting from the first step of the problem solution after the allocation parameter decision, which resulted in the parameters λ and β from the model of Drezner and Drezner (2012). The values for γ_j, the capacity in the DC j can be calculated before this step. This capacity is the maximum amount of relief goods that can be distributed from the DC in the time between two deliveries (i.e. one week).

The costs that have to be considered occur for driving a certain distance ($\tau * d_{hj}$), and for opening a DC the costs c_j and $b_j * \gamma_j$ occur, the first denoting the weekly fixed cost for opening a DC, the second the variable costs adjusting for the size of the capacity of the DC. Furthermore, a per unit cost for delivered relief goods r is multiplied with the amount of relief goods delivered t.

Since the problem has characteristics of a Covering Tour Problem, it is not unnatural that some form of threshold value for the coverage distance is used. In this problem, the parameter D denotes the 'sphere of influence', described in the competitive location model of Drezner and Drezner (2012, p. 203). DCs that are further away from a PC than D ($D < d_{hj}$) are not considered by the people in this PC. An explanation for this behavior can be found in Figure 4.3. This figure shows two potential DCs, the points A and B. The points C,D,E and F are PCs, where C and D are covered by A and B, E is only covered by B and F is outside of the sphere of influence D of both DCs. Note that coverage does not mean that all inhabitants of a PC are going to the DC. This demand estimation and allocation procedure is one of the core computational results of the second step of the problem solution.

For the first step, following the attractiveness related decision, all parameters from the

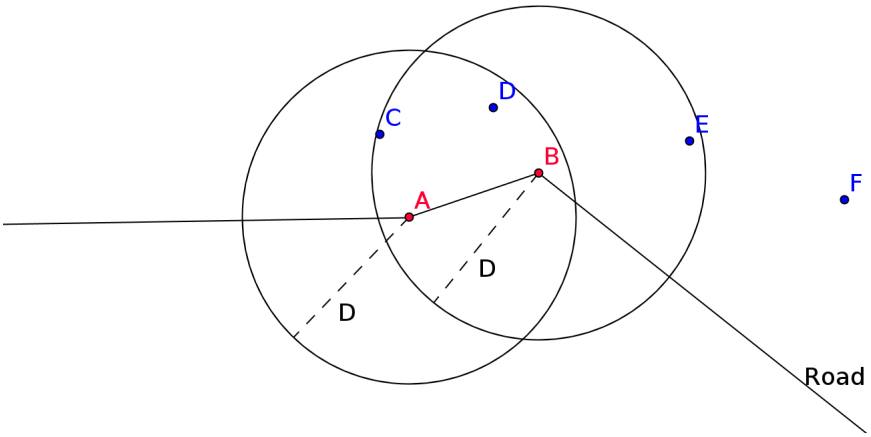


Figure 4.3: Explanatory example of the sphere of influence
(Source: Own figure)

model of Drezner and Drezner (2012, pp. 202ff.) are needed as well as various others. The number n_h describes the sum of DCs within the sphere of influence D of the PC h, while the number m_j describes the number of PCs within the sphere of influence of the DC j. In the case of Figure 4.3, the n of PC D would be 2, while the m of DC B would be 3. Similarly, the sets N_{h_j} and M_{j_h} comprise the points $j \in V_1$ and $h \in V_2$ that are within distance D of the points $h \in V_2$ and $j \in V_1$ respectively. A simple example can help the understanding of these sets of points: Using Figure 4.3, the set N of the PC E comprises only the DC B. This means $N_{E_j} = \{B\}$. Similarly, the set M of the point B, is $M_{B_h} = \{C, D, E\}$, which means the set of PCs within distance D of the DC B consists of C,D and E.

Additionally, the distance decay function ($f(d_{hj}) = e^{-\lambda * d_{hj}}$) parameter λ is calculated in the first step as well as the dummy locations attractiveness level α. As in Drezner and Drezners model, the dummy will be chosen with a probability proportionally to the value $\beta = \alpha * f(D)$, where the dummy attractiveness is used in the distance decay function for the sphere of influence D.

Since all variables and parameters have been described, the three steps can now be described in more detail, but first, the capacity calculation has to be presented. A complete list of parameters and variables can be found in Table 4.1.

4.2.2 Capacity Calculation

The whole problem solution has to begin with the calculation of the capacities of the potential DCs. In the following, an intuitive approximation method for determining the capacity of the single DCs is presented. Afterwards, the attractiveness decision will be presented.

The capacity γ_j of a specific DC j could be calculated using a straightforward and rational approach, employing the numbers for total population $P = \sum_{h \in V_2} p_h$. Since the decision maker has neither an interest in providing too little possible total capacity ($\Gamma = \sum_{j \in V_1} \gamma_j$) nor too much and waste valuable resources this way, another approach has been used.

A first approach was to assign all possible demands in reach (in the sphere of influence D) to the DCs:

$$\gamma_j = \sum_{h \in M_{j_h}} p_h \tag{4.2.2.1}$$

This showed to result in very large capacities γ_j and a very high ratio of unused relief packages, while leading to very high costs. Therefore, the total amount of possible demand was multiplied with an arbitrary capacity ratio number cr between 0 and 1 to limit the size to a certain percentage of possible demand for each DC.

This, of course, resulted in the situation that PCs that could not chose between different DCs would be chronically in lack of supplies. Thus, another approach has been suggested to calculate the optimal capacity ratio cr_j for each individual DC. For every DC, the PCs in reach of the DC in question were checked upon how many DCs where in reach of them. Then the ratio would be calculated for this specific DC by dividing 1 by the average of DCs in reach of the PCs in reach of this one specific DC. The following formula shows this in simpler form:

$$cr_j = \left(\frac{\sum_{h \in M_{j_h}} n_h}{m_j} \right)^{-1} \tag{4.2.2.2}$$

This resulted in very low ratios, barely providing enough supply for the whole potential demand if all DCs were opened at the same time.

Therefore a final improvement of this approach is proposed that takes up the question which ratio of DCs should be opened by relying on a concept borrowed from the calculation of optimal order quantities (Kummer et al., 2013, p.170), where the optimal order quantity (OOQ) is essentially a trade off between inventory cost and order cost. Applied to the problem in this thesis, one could say that the relation of fixed cost to set up a DC (per delivery) c_j divided by the cost to serve DC j with a Full Truckload (FTL)

$$c_{FTL_j} = 2 * \tau * c_{0j} \tag{4.2.2.3}$$

shows pretty clearly, how many DCs should be opened. The cost of a FTL does not necessarily reflect the cost to serve the DC, since a truck could have other deliveries on board as well, but it serves as a justifiable approximation.

If the ratio

$$OOQ = \frac{c_j}{c_{FTL_j}} \tag{4.2.2.4}$$

is high, fixed costs are very high in comparison to the cost of getting the goods there, so opening up less would be advisable, which should also correspond to a higher capacity γ_j at the small amount of opened DCs j, so as to be able to serve the population nevertheless. On the other hand, if the ratio OOQ is small, the routing cost are critical compared to the fixed DC cost, so opening up more DCs with less capacity and therefore combining the load into consolidated shipments should be considered.

Therefore, the mode to include this ratio OOQ is to multiply the old capacity ratio cr_j with it to adjust the capacities at the DCs,

$$cr_j^* = cr_j * OOQ \tag{4.2.2.5}$$

which is also beneficial due to the no longer necessary choice of a seemingly random

parameter, which has to be reevaluated for every new problem setup. The only problem, arising with this procedure, is that the new ratio can be lower than one, so decreasing the maximum available capacity Γ to below the potential demand P. This can be the case if the fixed cost c_j is negligible and/or the routing cost τ very high. On the other hand, if the ratio is very high (due to the opposite reasons), the capacity in one DC can exceed the maximal potential demand.

These very undesirable situations can easily be avoided if maximum and minimum values for the cr_j^* are set:

$$1 \leq cr_j^* \leq \left(\frac{\sum\limits_{h \in M_{j_h}} n_h}{m_j} \right) \tag{4.2.2.6}$$

with a minimum of one for the lower bound and a ratio of the average number of DCs that can be reached by the PCs in reach of the DC in consideration, as calculated already earlier. So in the end, the capacity ratio itself will be a value between near to 0 and 1. Concluding, the capacity calculation looks like the following:

$$\gamma_j = \left(\sum_{h \in M_{j_h}} p_h \right) * cr_j^* \tag{4.2.2.7}$$

subject to:

$$cr_j^* = \left(\frac{\sum\limits_{h \in M_{j_h}} n_h}{m_j} \right)^{-1} * \frac{c_j}{2 * \tau * c_{0j}} \tag{4.2.2.8}$$

and

$$1 \leq cr_j^* \leq \left(\frac{\sum\limits_{h \in M_{j_h}} n_h}{m_j} \right) \tag{4.2.2.9}$$

4.2.3 The Allocation parameters

Now, since the capacities of the potential DCs are known, the problem solution can proceed to the decision in the first step of the problem. In the allocation parameter decision, the

decision maker specifies two pairs of values r_1, d_1 and r_2, d_2 required for the attractiveness calculation. These pairs are the only required decision input, since the rest will be either calculated (e.g. λ) or has to be provided as data (e.g. the population data p_h).

The pairs have both a 'ratio part', r_1 and r_2, and a distance part, d_1 and d_2. These pairs define how big the numbers for λ and β should be. Those two are the only two open parameters missing in the model of Drezner and Drezner (2012), assuming that the attractiveness levels a_j of the DCs j are given by their capacity γ_j and the dummy attractiveness is known as soon as β is known. If we assume that the dummy distance is equal to the sphere of influence, one unknown variable is eliminated. Choosing the value of the outermost point influencing the rational decision of the inhabitants as the point associated with the dummy responsible for allocating lost demand seems to be the intuitive right way to approach the topic of lost demand in a situation of disaster relief, where the option of not going to a DC might be considered just in a situation where it is really nearly infeasible to traverse the distance to get access to relief goods.

For determining the two last remaining parameters, λ and α, an approximation method using the aforementioned two pairs of ratios and distances is employed. The decision maker has to provide the problem solution method with 2 different pairs of values, which he determines to be fitting for the following case:

Assuming a PC has only one DC in its sphere of influence, and this DC has average capacity $\frac{\Gamma}{\sum j}$ as attractiveness level a_j. Furthermore, the option to stay at home is viable. The pairs d_1, r_1 and d_2, r_2 have to describe values for the ratios r_1, r_2 of people of this PC that would go to the single available DC with average capacity if it was situated in a distance of d_1, d_2 respectively.

Modifying Function 4.1.1 for this special setting, the simplified equation gives:

$$r_{1,2} = \frac{a_j * e^{-\lambda * d_{1,2}}}{a_j * e^{-\lambda * d_{1,2}} + \alpha * e^{-\lambda * D}} \tag{4.2.3.1}$$

Since the two pairs of values entered by the decision maker, such as "60 % of people will go to such a DC if it is in 2000m distance", open up the possibility to create a system of equations consisting of two equations with two unknown variables. For this purpose, the

Function 4.2.3 is further transformed to suit the purpose in the systems of equations:

$$
\begin{aligned}
r_{1,2} &= \frac{a_j}{a_j + \alpha * e^{-\lambda*(D-d_{1,2})}} \\
a_j &= r_{1,2} * a_j + r_{1,2} * \alpha * e^{-\lambda*(D-d_{1,2})} \\
a_j - r_{1,2} * a_j &= r_{1,2} * \alpha * e^{-\lambda*(D-d_{1,2})} \\
\frac{a_j}{r_{1,2}} - a_j &= \alpha * e^{-\lambda*(D-d_{1,2})} \\
\frac{\frac{a_j}{r_{1,2}-a_j}}{\alpha} &= e^{-\lambda*(D-d_{1,2})} \\
-\lambda * (D - d_{1,2}) &= \ln\left(\frac{\frac{a_j}{r_{1,2}-a_j}}{\alpha}\right) \\
-\lambda &= \frac{\ln\left(\frac{\frac{a_j}{r_{1,2}-a_j}}{\alpha}\right)}{D - d_{1,2}}
\end{aligned}
\tag{4.2.3.2}
$$

Now, as we have reached the expression for λ, we can set the λs for the two pairs of decision variables equal and solve for α.

$$
\begin{aligned}
\frac{\ln\left(\frac{\frac{a_j}{r_1-a_j}}{\alpha}\right)}{D - d_1} &= \frac{\ln\left(\frac{\frac{a_j}{r_2-a_j}}{\alpha}\right)}{D - d_2} \\
(D - d_2) * \ln\left(\frac{\frac{a_j}{r_1-a_j}}{\alpha}\right) &= (D - d_1) * \ln\left(\frac{\frac{a_j}{r_2-a_j}}{\alpha}\right) \\
(D - d_2) * \ln(\frac{a_j}{r_1} - a_j) - (D - d_2) * \ln\alpha &= (D - d_1) * \ln(\frac{a_j}{r_2} - a_j) - (D - d_1) * \ln\alpha \\
(D - d_2) * \ln(\frac{a_j}{r_1} - a_j) - (D - d_1) * \ln(\frac{a_j}{r_2} - a_j) &= (D - d_2) * \ln\alpha - (D - d_1) * \ln\alpha \\
(D - d_2) * \ln(\frac{a_j}{r_1} - a_j) - (D - d_1) * \ln(\frac{a_j}{r_2} - a_j) &= \ln\alpha * (d_1 - d_2) \\
\frac{(D - d_2) * \ln(\frac{a_j}{r_1} - a_j) - (D - d_1) * \ln(\frac{a_j}{r_2} - a_j)}{d_1 - d_2} &= \ln\alpha \\
e^{\frac{(D-d_2)*\ln(\frac{a_j}{r_1}-a_j)-(D-d_1)*\ln(\frac{a_j}{r_2}-a_j)}{d_1-d_2}} &= \alpha \\
a_j * (\frac{1}{r_1} - 1)^{\frac{D-d_2}{d_1-d_2}} * (\frac{1}{r_2} - 1)^{\frac{d_1-D}{d_1-d_2}} &= \alpha
\end{aligned}
\tag{4.2.3.3}
$$

Now we can use the value of α and fill it into the last step of Transformation 4.2.3 to get a solution for α and λ, which can be used in the rest of the problem solution.

A legitimate question remains: Why is this transformation necessary? It is perfectly true that the decision maker could also just estimate the values of λ and α, without going this long way. But estimating parameters that have no correspondent in the real world is a difficult task and will likely result in opportunistic decision making to achieve one's goals. However, estimating a ratio of inhabitants that are willing to traverse a certain distance to get relief goods is a more practical and well justifiable decision. Even better, the beneficiaries can answer the questions for themselves, contributing to the design of their own relief network.

4.2.4 Multiobjective Optimization Applied: Heuristic and Exact

In the second stage, the multiobjective optimization algorithm is executed. The decision variable here is z_j, which is a binary variable, where $z_j = 0$ if the DC is not opened and $z_j = 1$ otherwise. The coverage distance used is the sphere of influence, already presented and used in the Competitive Location Model. Basically, in this step the optimization algorithm calculates the objective function values for the solutions it elects or, as in the case of the exact algorithm, the Complete Enumeration, all solutions that are possible.

First, the unserved demand and the demand at each DC have to be calculated as described in Algorithm 6.

The pseudocode presented for the unserved demand and demand at DC calculation starts with summing up all utility measures of certain DCs for a PC A_{j_h} for every PC, since, in line 7, this number is used to calculate how many people in each PC decide to stay at home. This number is added to the count of unserved demand. As can be seen, the Competitive Location Model's gravity model is used here to determine the numbers of unserved demand. Further on in line 11, the same model is employed to determine, how high the demand at each DC will turn out to be according to the Competitive Location Model. As the demand at the DCs might turn out higher than the capacity, the differential in this case cannot be served, thus has to be added to the total number of unserved demand. If the demand is too high for the capacity level, then the capacity will be used as demand figure for the delivery of goods in the cost calculation routing, if the demand at the DC is

Algorithm 6 Calculation of unserved demand and demand at DCs

1: **for all** $j \in V_0$ **do**
2: **for all** $h \in V_2$ **do**
3: $CumAttractiveness_h = CumAttractiveness_h + (A_{j_h} * z_j)$
4: **end for**
5: **end for**
6: **for all** $h \in V_2$ **do**
7: $Unserved = Unserved + p_h * (\alpha/(CumAttractiveness_h + \alpha))$
8: **end for**
9: **for all** $j \in V_0$ **do**
10: **for all** $h \in V_2$ **do**
11: $CumDemand_j = CumDemand_j + (p_h * z_j * (A_{j_h}/(CumAttractiveness_h + \alpha)))$
12: **end for**
13: **end for**
14: **for all** $j \in V_0$ **do**
15: **if** $CumDemand_j > \gamma_j$ **then**
16: $Unserved = Unserved + (CumDemand_j - \gamma_j)$
17: $Demand_j = \gamma_j$
18: **else**
19: $Demand_j = CumDemand_j$
20: **end if**
21: **end for**

lower than the capacity, then the demand figure will be used in the routing.

The cost calculation, as depicted in Algorithm 7, is structured into four parts: The cost resulting from the supply of relief goods (line 5), which are accounted for in unit cost, the fixed costs related to opening a DC (line 2) and the variable costs that increase with the size of the DC (line 3) and finally, the routing cost (lines 6-21). The routing cost is multiplied with a fixed factor indicating the cost per distance unit traveled, which in the end appears easy but requires the most detailed calculation behind it since the routing decisions have to be included here as well. Here, it is crucial to note that the mode, how the routing is calculated depends on the choice of routing algorithm, since different algorithms provide different qualities of routing and speed of solutions. However, since this routing problem is a capacitated CTP with multiple vehicles, we need an algorithm that is capable of respecting the vehicle capacity constraints of Q and is able to create not only one tour, but different tours for k vehicles if necessary.

Since the algorithm basically does not allow for split deliveries, but the demand at single DCs can possibly get larger than one FTL of size Q, we need to adjust such big demand points by sending FTLs to these DCs before the actual routing algorithm starts.

Algorithm 7 Calculation of Cost

1: **for all** $j \in V_0$ **do**
2: $Cost = Cost + z_j * c_j$
3: $Cost = Cost + z_j * \gamma_j * b_j$
4: **end for**
5: $Cost = Cost + (P - Unserved) * r$
6: $count = 1$
7: **while** $count > 0$ **do**
8: $count = 0$
9: **for all** $j \in V_0$ **do**
10: **if** $Demand_j > Q$ **then**
11: $Cost = Cost + 2 * d_{0j} * \tau$
12: $Demand_j = Demand_j - Q$
13: $count = 1$
14: **end if**
15: **end for**
16: **end while**
17: Routing Algorithm produces tours on k vehicles that respect the quantity limit of Q of the trucks.
18: **for all** $k \in K$ **do**
19: Calculate $Tourlength_k$
20: $Cost = Cost + Tourlength_k * \tau$
21: **end for**

Subsequently, the demand at those DCs is decreased by one FTL amounting to Q. This procedure is reflected in lines 6-16 of Algorithm 7. After the routing algorithm finished, the distances traveled in the tours created are multiplied with the distance cost and added to the total cost of this specific DC opening pattern. Therefore, both objective values have been calculated in the course of step 2. These are now used in the heuristic algorithm explained in the Section 3.2, in which the NSGA II is explained. In the case of the computation of the exact solution, the Complete Enumeration of Section 3.3 is employed here. It is important to mention that the multiobjective optimization carried out in this chapter didn't require any interference from the side of the decision maker.

4.2.5 The final decision: Decision maker's choice

The final choice on how the disaster relief logistics operation is carried out rests with the decision maker. Depending on how the multiobjective optimization was carried out, different options exist for this final decision. The two options here are, again, either the heuristic NSGA II or the exact solution algorithm Complete Enumeration.

In both, a non-dominated front is found that is used as input for the final decision. Since the front itself might be very large, a subset could be selected for this decision. In the case of the exact solution, the front found is the real Pareto optimal front, while the front found by the heuristic is hopefully close to it.

However, the decision maker has to choose one of the solutions of the presented front since it is not possible to follow two solution patterns at the same time. Resource restrictions can serve here as limiting factors on which opening pattern to choose.

4.3 Critical Assessment

This section takes a critical look at the proposed algorithms and methods to determine demand allocation in disaster relief logistics from two viewpoints: First, the proposed algorithms are assessed concerning their implications on justice in the disaster relief operations, and second, more general remarks concerning weaknesses and strengths of the approach proposed are discussed.

What rational thinking lies behind this approach? A computational solution to the problem of distributing relief goods using attractiveness values of different DCs is found through minimizing the objective values of cost and unserved demand, changes in the opening pattern are recommended if the objective values profit from it: Everything seems to follow the principle of maximizing pleasure and minimizing pain, as the Utilitarians would say. But does the presented approach really try to optimize the balance of pleasure and pains? The theory of utility suggests that the highest amount of pleasure can also come at the expense of certain parts of society, and this is exactly what happens in this approach as well. Small villages in remote locations will incur much higher costs if served while reaching not substantially more people. It would be easier to focus on densely populated regions since the routing cost are reduced and more people can be served with the same amount of resources. In multiobjective optimization algorithms, as proposed in the next chapter, such inefficient solutions that would provide a more equitable distribution across regions are officially sorted out, since they are 'dominated' by more unequal solutions that provide better objective function values in both, cost and unserved demand.

This critique, however, is very unfair towards the approach presented. Simply because

it is difficult to define and include quantitative measures of justice does not mean that it should not be attempted. And, one has to admit, the way that the preferences of the beneficiaries are incorporated into the decision making process to a much further extent as usual in such decision support systems can be counted as a deviation from usual mechanistic decision making. It even opens up the possibility to include 'soft' values into the attractiveness calculation mechanism, since who determines that only distance and capacity have a say in what attracts people to one or another DC? This feature could also be employed in more complex disasters, for example civil wars, where different DCs belong to regions with different political background and therefore some DCs are very unattractive for some PCs.

However, other schools of thought concerning justice would maybe rather continue criticizing the utilitarian view of the presented approach. Socialists would regret supporting a distribution planning system that does not take the difference in needs of the affected population into account. Competitive Location Problems would rather not target vulnerable members of society, but vice versa: The fitter a person is, the easier it will be to traverse large distances to retrieve food aid, while the elderly, the weak and the young would be less able to organize the trip to a further away DC. When Socialists proclaim to distribute to each according to the needs, then they forget that also targeting has its downsides. As Jaspars and Shoham (1999) show, targeting food aid to vulnerable households is difficult because of a myriad of reasons. From faster declining resources for logistics due to higher distribution cost to the challenge of identifying the real vulnerable target group, internal redistribution after targeting through exploitation to the increased fear of creating dependencies on donors aid. These inefficiencies would not qualify as just for libertarians, who would especially employ the last argument to talk against disaster relief operations in general.

Libertarians would neither find the greatest happiness principle appropriate, nor favor unequal distribution to target the vulnerable. They would find any disaster relief aid that was paid for with tax money, equally inappropriate, since it takes away the freedom of possession of the individual by taxing the fruits of labor and giving it to somebody else. Libertarians wouldn't think that a situation, in which people die because of a disaster is good or fair, but as long as the freedom of possession remains intact, at least it is just. Therefore, arguments like those presented by Raschky and Schwindt (2009a) are

to the liking of libertarian thinkers: The anticipation of aid after a disaster reduces the efforts to insure against disasters or reduces other protection activities. This, as Raschky and Schwindt (2009a) show, eventually results in even higher death tolls in disasters where foreign aid was anticipated. This game theoretical dilemma is called Samaritan's Dilemma. Another point that that this paper makes is that people underestimate the potential threat of disasters due to a lack of information and interest.

The last conception of justice discussed in this thesis is John Rawls' conception of "justice as fairness". Rawls would argue that in a situation of the formation of a social contract to organize disaster relief operations, society would opt for a very egalitarian distribution of relief goods, since the potential threat is, even without a 'veil of ignorance', exists for pretty much everyone, even though richer nations usually resist disasters in a better way than poorer ones, where preparation and mitigation of disaster risks is too costly considering the economic shortcomings of everyday business. However, Rawls would most likely be in favor of the influenceable attractiveness level if it would really reflect the preferences of everybody.

A few general remarks are obvious when critically reflecting on the proposed approach. In the following, a short list of major reasonable conflicts is presented, although a certain degree of abstraction is necessary to facilitate the computational implementation.

- Walking and Driving distance

  The approach considers that inhabitants of PCs are only able to walk to the DCs. If there would be other modes of transport, bicycles, cars or even hitch-hiking, then the whole Competitive Location Model would have to be changed accordingly. In this respect, it is also interesting that PCs, i.e. villages and cities, are represented as zero-dimensional vertices, so that the exact position in the city has no influence.

- Combination of procurement and sharing

  The option that single inhabitants take goods not only for themselves but also for other members of their PC is not allowed in this form now, since it would change the coverage mechanism towards much faster coverage of larger parts of the total demand. At the same time, sharing of relief goods is not envisaged.

- Leftover capacity and unserved demand

Since the calculated demand that is expected to realize at the DCs and the capacity of the DC are not necessarily the same, there must be high safety stocks at DCs since the demand is replenished weekly. However, the approach suggests that even if there is high safety stock in one DC, it might be that there is excess demand in another DC due to its superior attractiveness. This means that people would rush towards the bigger DCs even if smaller and nearer DCs have excess capacity while the big DCs might run out of stock. Another difficult topic is that no matter how much capacity and how many DCs exist, due to the properties of the gravity model of the Competitive Location Model, some demand will always go unserved.

- Rational decision making

  The whole approach assumes that people are perfectly informed about the characteristics relevant to their decision making on where to go to: They know which DC is opened, how much capacity it has and how far it is away from his/her PC. Funnily, inhabitants don't anticipate the decision making of other beneficiaries, so that situation of excess demand occurs.

- Data requirements

  As will be seen in Chapter 5, the requirements from the data viewpoint are substantial. Two routable networks, one for vehicles and one for pedestrians is used as well as data on population counts.

Although there are many points where deficiencies can be detected, all those can be used as hints for further research to amend the processes to repair these deficiencies. For example, projected future research should encompass changing the capacity in order to react to too high or low demand compared to capacity. Similarly, the quality of demand prediction will very likely be better, if better and more factors are included in the attractiveness calculation model apart from distance and capacity. Maybe it is possible to achieve a method to target vulnerable groups in the presented algorithm.

And some points about the approach already convince: If data is available on the affected region, implementation can happen relatively fast and the two objectives used can help finding efficient solutions that are even able to take the preferences of the local beneficiaries into account to some extent. And the model of Drezner could be employed

in its original form if more than one provider of relief items is on the scene, since this competition between companies was envisioned in the original model.

However, in the next chapter the application of this problem solution approach will be presented using the Gaza province in southern Mozambique.

Parameters Variables	Meaning
a_j	Attractiveness value of a DC j
A_{jh}	Utility measure of a DC j for a certain PC h
α	Dummy attractiveness
β	Probability factor of dummy selection
b_j	Variable operation cost of DC j
c_{FTL_j}	Cost to serve DC j with one FTL
c_j	Opening cost of DC in j
$cr_j(cr_j^*)$	(Improved) Capacity Ration Multiplier
$Cost$	Cost figure resulting from second step
d_1, d_2	Distance decision variable of first step
d_{hj}	Distance PC-DC
d_{ij}	Distance DC-DC
D	Sphere of Influence
$Demand_j$	Total demand in DC j
γ_j	Capacity in DC j
Γ	Total capacity of all DCs
G	Graph for vehicle routing
h	Index of PCs in U
i	Index of DCs in G
j	Index of DCs in G and U
k	Index of vehicles in K
K	Set of vehicles k used
λ	Distance decay function parameter
m_j	Number of PCs within distance D of DC j
M_{j_h}	Set of PCs h within distance D of DC j
n_h	Number of DCs within distance D of PC h
N_{h_j}	Set of DCs j within distance D of PC h
OOQ	Optimal order quantity factor
P	Total population
p_h	Population at PC h
Q	Capacity of trucks
r	Unit cost for relief goods
r_1, r_2	Ratio decision variable of first step
t	Total delivered goods
τ	Distance cost for trucks
U	Graph for pedestrian routing
$Unserved$	Total number of unserved demand
V_0	Set of nodes from G
V_2	Set of nodes from U
W_j	Demand in j
z_j	Opening decision variable of step 2

Table 4.1: Problem parameters and variables
(Source: own table)

Chapter 5

Case Study Mozambique

In this chapter, the case study, where the methods presented in Chapter 3 were applied to solve a problem as described in Chapter 4, is described. First, the situation in the case study area, the Gaza province of Mozambique, is briefly described, before, second, the tasks necessary for the implementation of the problem solution algorithm are explained. In this section, the composition of the instances used in the computational solution are explained and the parameter requirements are investigated.

In the next section, the implementation of the solution method is described briefly, before the results for both instances are presented. Concluding, the practical relevance of the presented method is assessed.

5.1 Case Study Description

The method proposed is very well suited for drought disasters: Since the transportation network needs to remain intact without taking any stochastic or dynamic routing into consideration, the type of disaster cannot include disruptions in the transport network, e.g. resulting from flooding or fighting areas in situations of warfare. Therefore, a hypothetical drought disaster has been chosen as case study object, with the task to deliver food items, i.e. standardized packages containing different food items, to distribution centers. Mozambique has been chosen as the location out of many reasons: First, droughts with severe effects on the population occur regularly, so the assumptions for this case study rest

on realistic circumstances. Furthermore, the contact to a worker from a relief organization of high renown operating in the area was established, helping in determining some key assumptions used in this thesis. Last but not least, the area chosen, Gaza province in southern Mozambique, has been investigated for another relief related diploma thesis by Späthling (2007). In his thesis, data on the region such as location of population centers and the road network have been gathered, and this information forms the cornerstone of the data used in this thesis.

Figure 5.1: Geographical situation of Mozambique
(Source: Own Screenshot from ArcMap 10.1, using the National Geographic Basemap)

Figure 5.1 shows the geographical location of Mozambique, while the red dots depict the population centers in Gaza province considered in this thesis. Even though there have been substantial depots of natural gas and coal detected in the recent past, the gross national income per inhabitant is only 980 $ with 54.5% of population living beneath the poverty line. (Berié, 2012, p. 530 and pp. 321)

The region of Gaza with its 75000 km^2 is nearly as big as the whole country of Austria, but compared to Austria, Gaza has a very small population of about 1.219 million inhabitants. The topography of the region is characterized by very flat lands crossed by various meandering river basins, most importantly the river Limpopo, which is a potential source of flooding. The capital of the province is Xai-Xai ("Governo da Provincia de Gaza", 2014). A more detailed map of the region considered in this thesis, the southern part of Gaza province, can be seen in Figure 5.2, again highlighting the population centers.

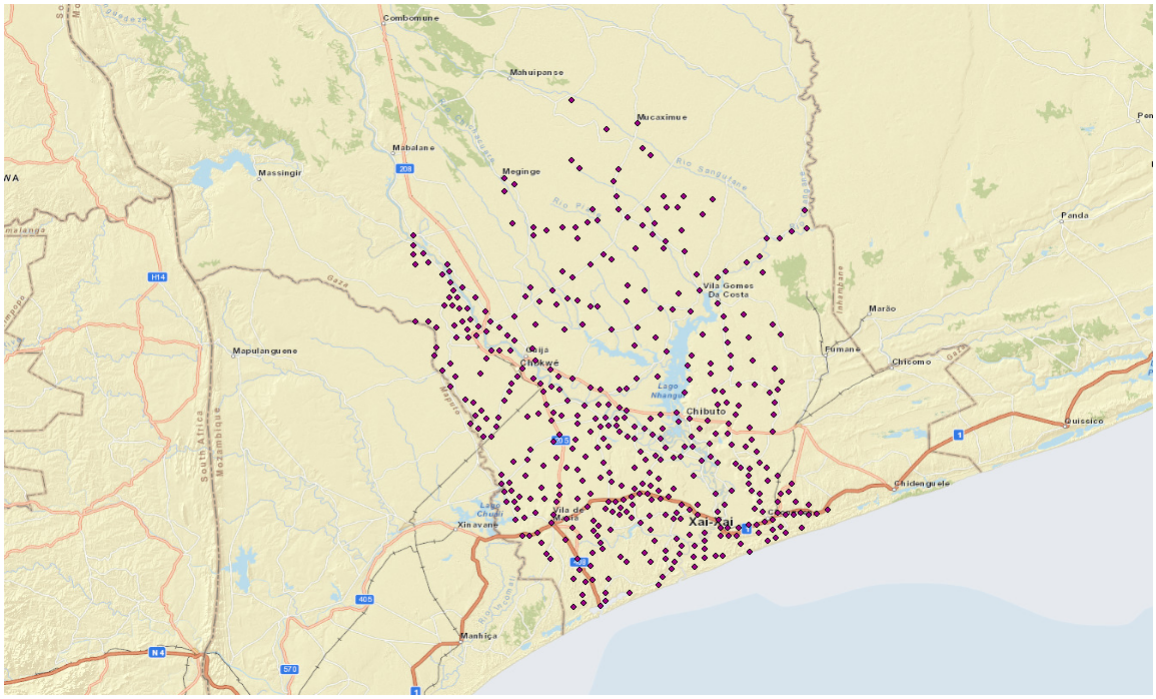


Figure 5.2: Geographical situation of case study area
(Source: Own Screenshot from ArcMap 10.1, using the World Streetmap Basemap)

As Aragon et al. (1998) show, the southern part of Mozambique, which contains the case study area of Gaza, is the most affected by very frequent droughts, resulting from various factors but crucially including El Niño. In the year 2005, the fourth consecutive year of drought affected the southern region of Mozambique, leading to a complex situation of famine, followed by illnesses such as cholera due to the intake of contaminated water and simple diarrhea due to the intake of inedible foods (IRIN, 2005).

According to the international disaster database EM-DAT, run by the Centre for Research on the Epidemiology of Disasters (CRED), the most severe disaster affecting Mozambique between 1900 and 2014 was indeed a drought in 1981, killing 100000 people, while this drought, together with the other 11 droughts registered between 1900 and 2014

affected a total of more than 17 million people. The last entry for a severe drought in the region was in August 2007, affecting 520000 people (EM-DAT: The OFDA/CRED International Disaster Database – www.emdat.be, 2014).

Obviously and unfortunately, the presented problem solution method encounters fitting disaster environments quite regularly, thus proving that assuming such a special problem is a realistic constraint. In the next section, the preparation tasks for the implementation in southern Gaza province are explained.

5.2 Preparation Tasks

In order to use the approach presented so far, a few requirements from a data point of view have to be fulfilled to enable the decision maker to employ the solution method. This means that data on the position and size of the population centers has to be readily available, just as on the street network to provide the routing of trucks with a network to reach the DCs. Furthermore, a decision has to be made on where to position possible DCs. Since it can be assumed that pedestrians not necessarily stick to the road network to walk to the DC they favor, a solution to this problem has to be found as well:

- Population Centers

  In the diploma thesis of Späthling (2007), population data on the southern part of Gaza province has been used and is employed again in this thesis. The point data with respective population counts has been calculated using an interpolation technique using census data, imagery and various other types of data to assign an average population count to specific points in the cells (raster data) under consideration. Therefore, if the cells are smaller than the whole surface area covered by a city, the city might be represented by several points, each indicating the population in the part of the city that is within the cell that the population center point belongs to. The points, which can be seen in Figure 5.2 have been taken from the thesis of Späthling without any changes, apart from the changes necessary in the next step, the definition of distribution centers.

- Distribution Centers

The choice on where to position possible DCs that can be chosen by the solution algorithm is a difficult and influential one. The approach followed in this thesis is that a DC should be established in a PC, since villages tend to have the required facilities to store the food items (e.g. buildings like schools or administrative buildings) and therefore not tents or other facilities have to be created. Furthermore, it is assumed that villages also provide staff for the DCs who can distribute food and guard the DCs to avoid theft.

In order to provide the Geographical Information System (GIS) ArcGIS with data that can be used for routing the vehicles between DCs, the DCs have to be at a road, which is also a reasonable requirement, since one would not establish a DC in a location where no goods can be delivered to easily. Therefore, all PCs that are within one kilometer of a road from the road network qualify as potential DCs. In Figure 5.3, the highlighted points are DCs that are not within one kilometer of a road, and therefore are considered to be just PCs, while the rest are DCs and PCs at the same time. The potential DCs have therefore been moved onto the closest road by a so-called 'snap' operation in the software ArcGIS 10.1. Since villages are not zero dimensional like the point data suggests, for all PCs less than a kilometer away from the street network it has been assumed that those points actually lie on the street network.

- Vehicle Network

  Not only PCs and DCs are needed for the execution of the solution algorithm, but also distance matrices. These are calculated using the software ArcGIS and especially its extension 'Network Analyst', which has a functionality for Origin-Destination (OD) matrix calculation, which creates a matrix consisting of all distance values on the network between all points indicated. It is assumed that time is not as crucial as distance in the recurrent delivery plans, therefore the OD matrix consists of the distances.

  The polylines representing the street network have been taken, again, from the work of Späthling (2007), while here, a manual data update has been done. This means that the lines have been contrasted to two sources of street data provided by ArcGIS as part of their 'Basemap' functionalities. World Street Map and Open Street Map

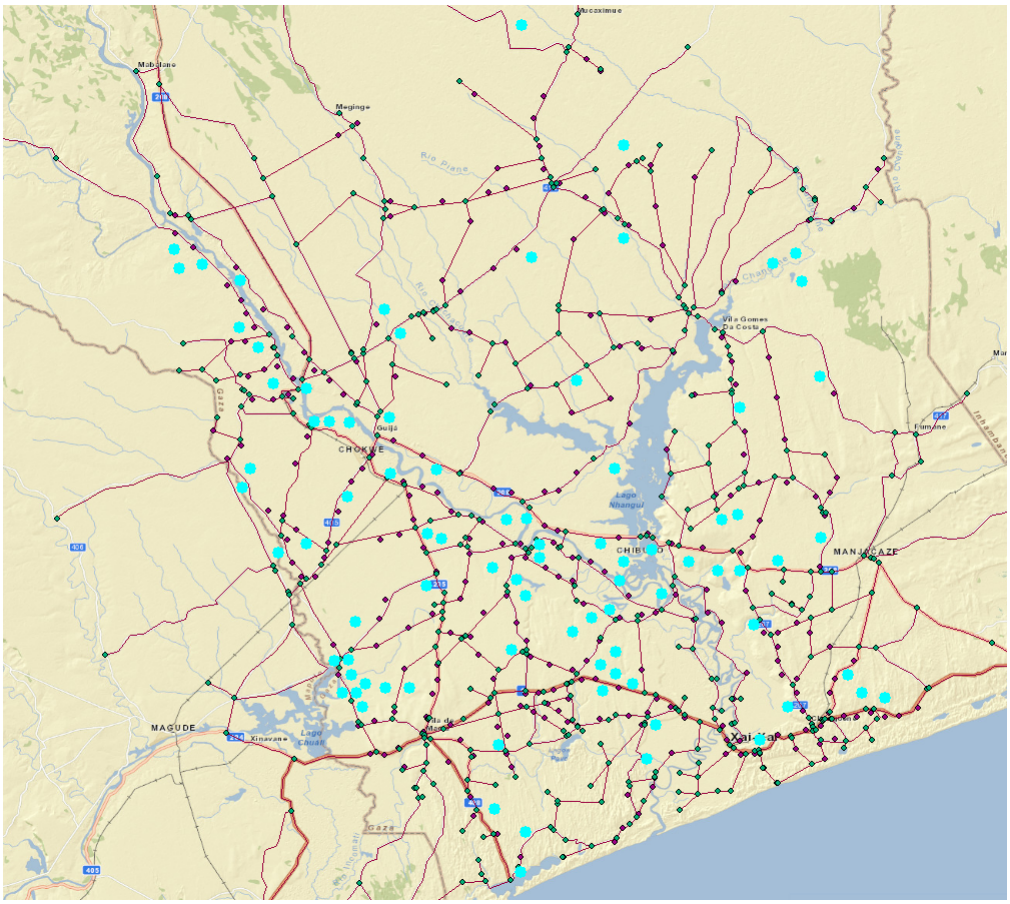


Figure 5.3: Population centers without direct road access
(Source: Own Screenshot from ArcMap 10.1 with World Streetmap Basemap)

have been used to compare the provided street data to the existing information provided by these two sources, and errors have been repaired. This functionality to contrast the data used with basemaps was not available when the data was gathered by Adrian Späthling in 2007. In the course of these amending operations, some data errors (e.g. where polylines would not connect at their endpoints, which inhibits the routing over these corners) have been corrected. Especially many connection problems, as depicted in Figure 5.4, needed close attention, since if routing over only one connection is not possible, it is not easily discovered, since only some distance values are longer as a result of routing on other, longer roads between origins and destinations affected by this connection problem. Therefore, as in Figure 5.4, the endpoints have to be snapped to the other features to allow for routing. This manual checking requires much time to process each possible connection.

After the data has been checked for connection problems, the OD matrix can be

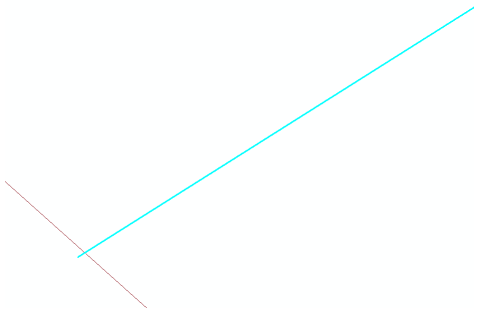


Figure 5.4: Required 'snap to' action
(Source: Own Screenshot from ArcMap 10.1)

calculated using ArcGIS. The output is the proof of a functioning, routable network and the network can be seen in Figure 5.5. This, however, answers only the question on which network to route the trucks delivering the relief goods, but not how the beneficiaries reach the DCs.

- Pedestrian Network

  Since it can be assumed that people will not always follow the course of the road network to reach DCs, as a solution a pedestrian network has been created, taking into account that the topography of the region suggests that shortcuts can be taken. However, adding links to the existing road network is just considered between close PCs and only, if no barriers are in between, like mountains, lakes or the seashore. In Figure 5.6, the network created for pedestrians is depicted, showing a nearly five-fold increased number of possible travel links. The creation of these additional links consumes a considerable amount of time, since every link has to be checked for feasibility. However, after the creation of the pedestrian network, again ArcGIS was used to calculate an OD matrix for the pedestrian network.

Both matrices were exported as .txt files to be used as input for the implementation of the solution approach in C++. The implementation code can be seen in Appendix A. However, first the approach has been tested on a smaller instance, before a bigger instance has been used to show the capabilities of the approach in problems with considerable complexity. The two instances are briefly described in the following.

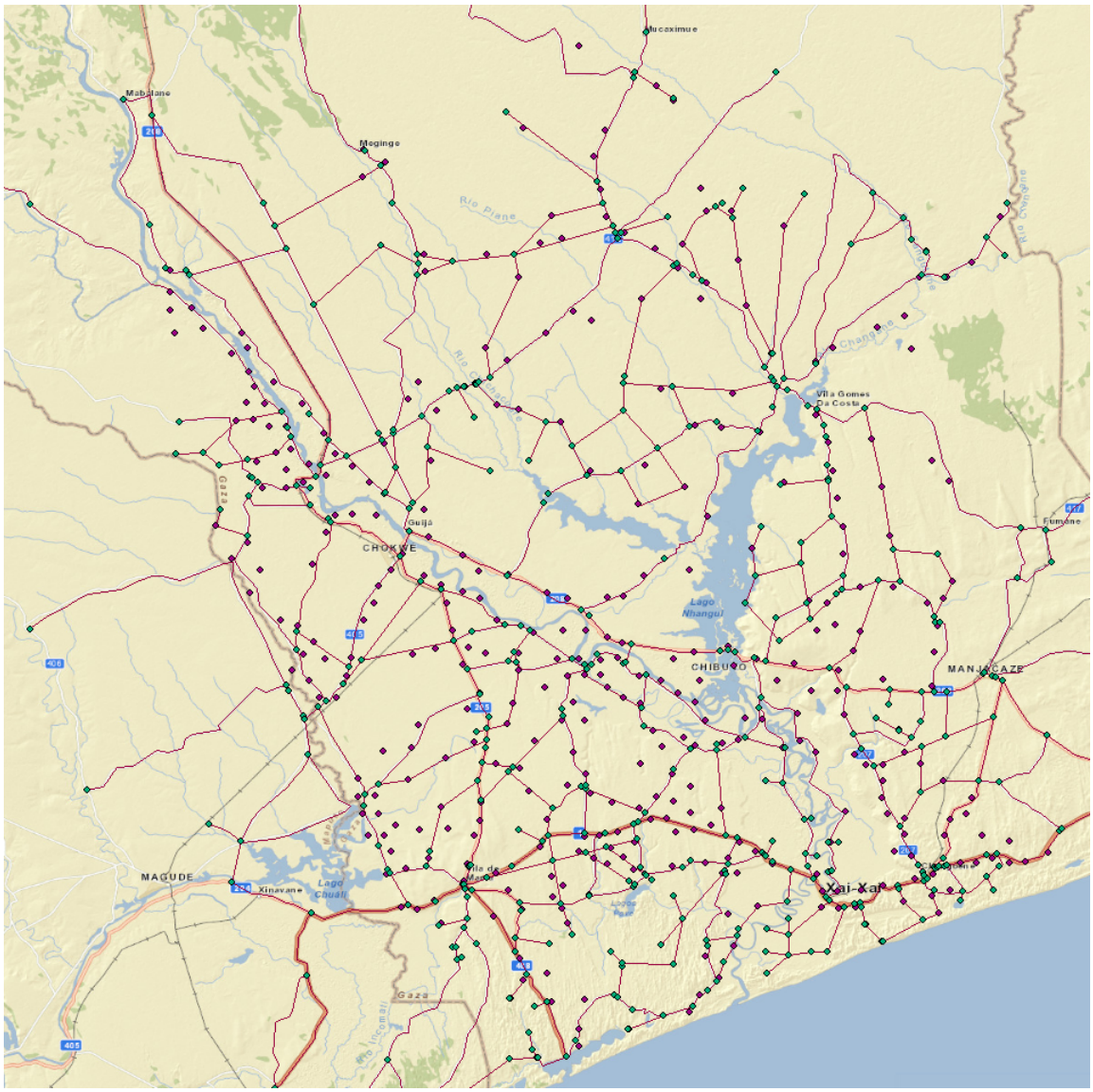


Figure 5.5: Vehicle Network
(Source: Own Screenshot from ArcMap 10.1 with World Streetmap Basemap)

5.2.1 Big Instance Generation

The easier task is to define, which instance will be used as the big instance later on. Here, all PCs that proved to be inside the routable network were chosen, amounting to 422 PCs, whereof 340 were snapped to the road network since they were elected as potential DCs due to their proximity to the vehicle routing network. The full networks can, again, be seen in Figures 5.5 and 5.6. For the depot location, which is at the same time also a PC, a very central position near the second largest city, Chokwe, has been chosen, since a central location seemed to be promising in terms of reducing total distance traveled. However,

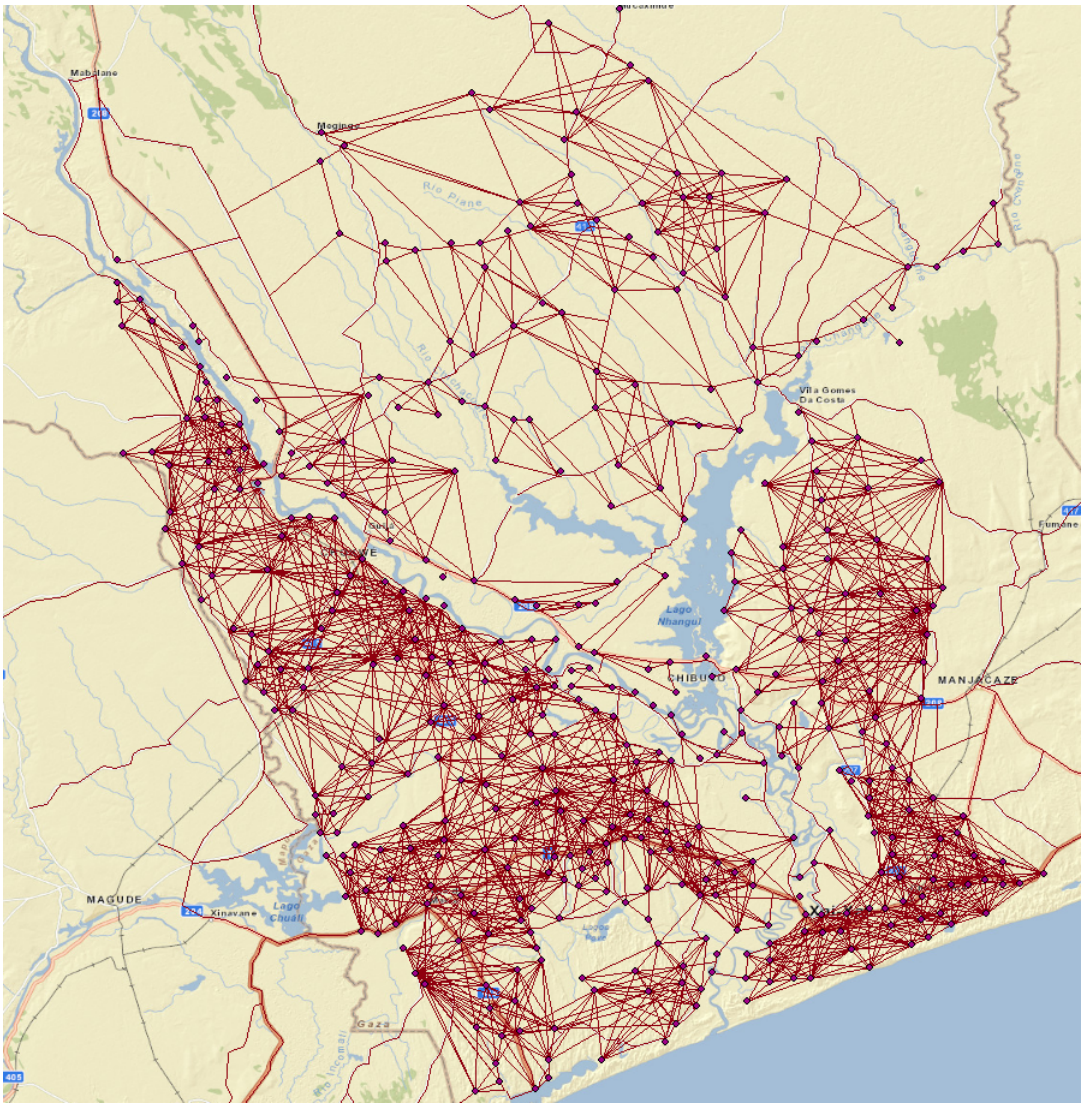


Figure 5.6: Pedestrian Network
(Source: Own Screenshot from ArcMap 10.1 with World Streetmap Basemap)

the choice of a perfect depot location would deserve more attention, but would have gone beyond the scope of this thesis.

5.2.2 Small Instance Generation

The small instance, as can be seen in Figure 5.7, is a selection of 25 PCs, whereof 21 qualified as DCs at the same time. The sample has been chosen for different reasons: First, it is close to the sea shore, which might be of high importance for logistical reasons, since relief goods might arrive by ship. Then, the sample instance is very homogeneous in terms of distribution of PCs and their size, while being next to the largest city in the

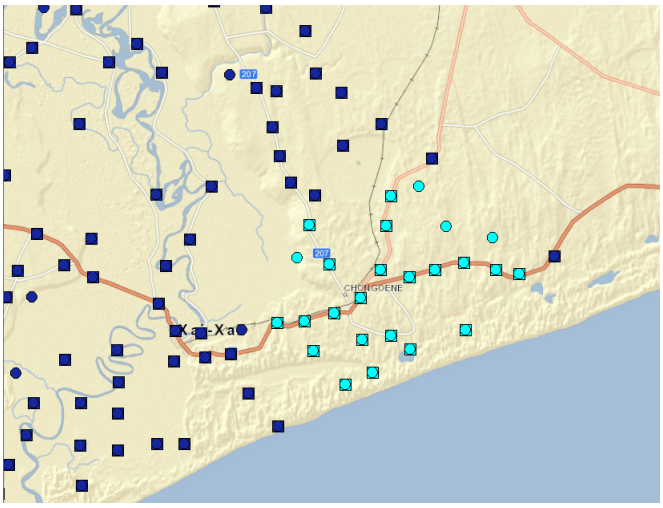


Figure 5.7: Instance used for exact solution and heuristic
(Source: Own Screenshot from ArcMap 10.1 with World Streetmap Basemap)

whole region, Xai-Xai.

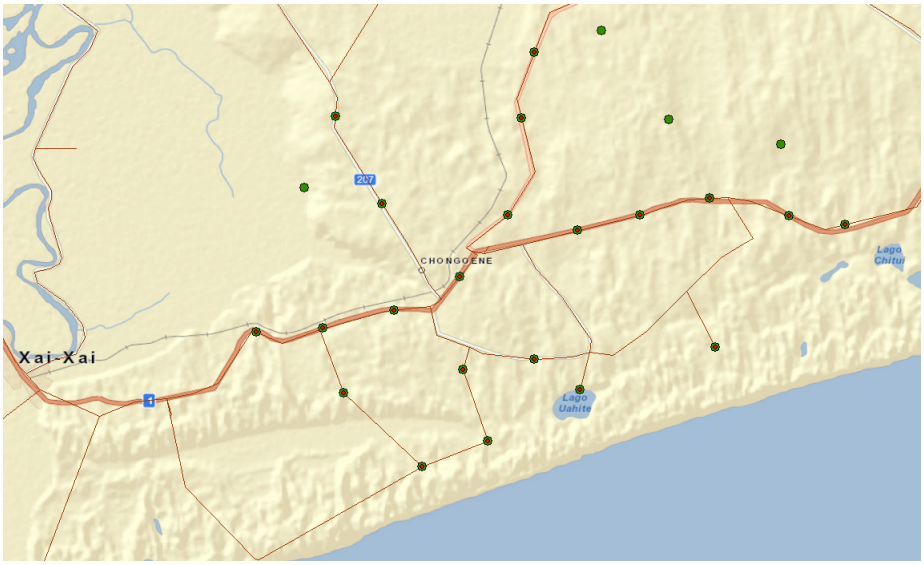


Figure 5.8: Small instance and street network
(Source: Own Screenshot from ArcMap 10.1 with World Streetmap Basemap)

In Figure 5.8, the street network can be seen. Obviously, the instances' PCs are in close proximity to two main roads in the region. This was the reason for choosing the depot in this small instance as the DC near to the crossing of these main roads. In Figure 5.9, the pedestrian network for the small instance is shown, compared to the street network one can see the vast amount of opportunities for pedestrians to use shortcuts to reach their preferred DC. This is especially true since the instance has been chosen in a way that no rivers would cross the region and the topography is also very flat.

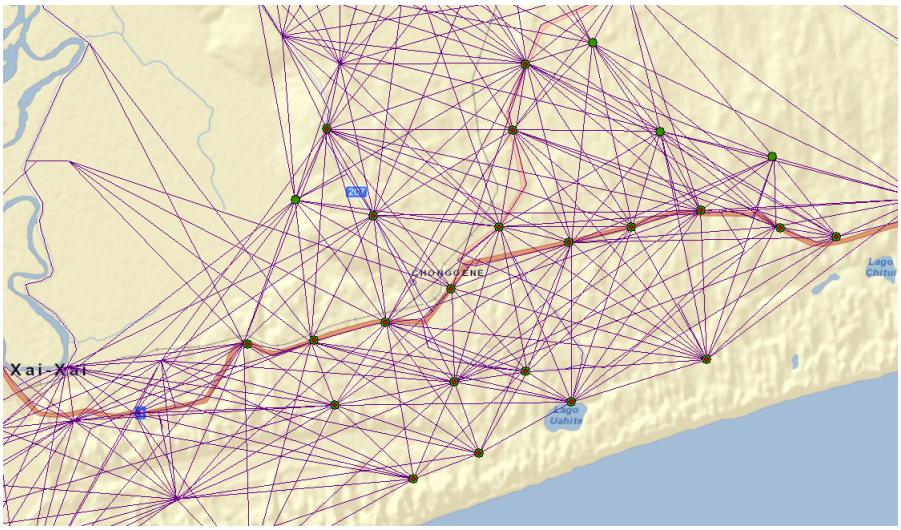


Figure 5.9: Small instance and pedestrian network
(Source: Own Screenshot from ArcMap 10.1 with World Streetmap Basemap)

In the following section, the parameters needed are described, since the parameters are the last important information before the algorithm can take up its work.

5.2.3 Parameter Estimation

For establishing reasonable calculations on the objective values and the functioning of the optimization algorithm in general, a wide range of parameters have to be estimated or filled in according to available knowledge. Cost parameters for example will most likely be available for decision makers, while other parameters need to be estimated. Table 5.1 provides an overview of the necessary parameters.

The parameters necessary stem from three areas: The Competitive Location Model, the Cost calculation and the NSGA II itself.

The parameter pairs for the Competitive Location Model have to be estimated by the decision maker as described earlier. In the course of the calculation, two pairs have been taken: First, it was estimated that if a DC with average capacity is 1000 meters away from a PC, then 99% of the population will go there. Second, the ratio of people going to an average sized DC which is situated in 10 kilometer distance has been estimated to be 66%. These two value pairs form the basis of lambda and alpha calculation, while one parameter was defined as given, the sphere of influence. Here, it was assumed that DC further away than 15km would not be considered, since going back and forth would total

Parameters	**Abbrev.**	**Value**
Competitive Location Model		
Sphere of influence	D	15000m
Ratios A,B	$r_{1,2}$	user specified
Distance A,B	$d_{1,2}$	user specified
Cost calculation parameters		
Unit cost	r	1
Distance cost	τ	0.001
Fix DC Cost	c_j	$\forall j \in G = 200$
Variable DC Cost	b_j	$\forall j \in G = 0.1/unit\ capacity$
Truck Capacity	Q	10000 units
NSGA II		
Parent/Offspring Generation Size	N	50
Iterations	I	user specified
Crossover Probability		0.8

Table 5.1: Parameters
(Source: own table)

30km, a distance that even the fittest would not march easily, especially carrying food items on the way back.

The cost parameters could also be decided by the decision maker, although here, historical values will leave little room for interpretation and these values can be estimated rather easily. For example, the distance cost, accounting for 1 Euro per kilometer, is a rather convenient estimation. Trucks, most likely imported from South Africa, 6x4 traction and 24 tons of cargo carrying capacity cost around 1.06 million South African Rand, which translates into roughly 70000 Euros (Mercedes-Benz South Africa, 2013). If one deducts salvage value at the end of usage and the duration of use following the recommendation of staying below the usual ad hoc replacement policy of 5 years or 150000kms as described by Pedraza Martinez et al. (2011), then the fixed cost for the vehicle per kilometer will be less than 50 cents. This, however, is a pessimistic estimation, since the values from Pedraza Martinez et al. (2011) are valid for smaller 4x4s, while trucks usually endure a higher mileage, but maybe the rougher driving conditions are taken into account by the higher fixed cost value. Since modern trucks consume around 30 liters of diesel per 100 kilometers, the total cost of 1 Euro per kilometer seems reasonable, but might deserve some fine tuning.

The fixed and variable costs of opening of a DC are to be considered per week, assuming that the 200 Euro fixed cost are consumed by security personnel to guard the DC day and

night, while the variable cost is used for the increasing personnel requirements for food distribution with increasing DC size.

The unit cost of a package of relief goods crucially depends on what goes into a package for one person for one week. Facts from the World Food Programme (Devex, 2014) suggest that since a family of five receives around 65kg of goods for a month, a single person would receive 3kg per week. The cost for three kilograms of basic food supplies including oil, sugar and salt will not total up to more than 1 Euro. Therefore, a truck as outlined above, will for sure not be able to take more than 10000 packages, which is already an optimistic guess.

For the NSGA II, some parameters have to be defined as well, but those will most likely not be defined by the decision maker, but by the developers of the solution method, since it is tested for various parameter settings. The number of iterations depends crucially on the size of the problem, thus a clear number cannot be given, while parent and offspring generation size of 50 solutions seems to be a regular number, while the crossover probability with 0.8 is relatively low, but resulted in good solutions.

5.3 Decision Steps and Algorithm Execution

In this section, a quick overview of how the algorithm runs in real-life application is given. First, it is described what decisions have to be made, then the execution and the final decision are scrutinized.

5.3.1 Allocation Parameter Decision

A decision maker employing the presented solution method for the hypothetical drought disaster in Gaza province of Mozambique would first need to take the data that he has at hand and provide the algorithm with it. He would specify the Competitive Location Models' value pairs and define the other parameters as described in the section above. Then, program execution would start.

5.3.2 Algorithm Execution

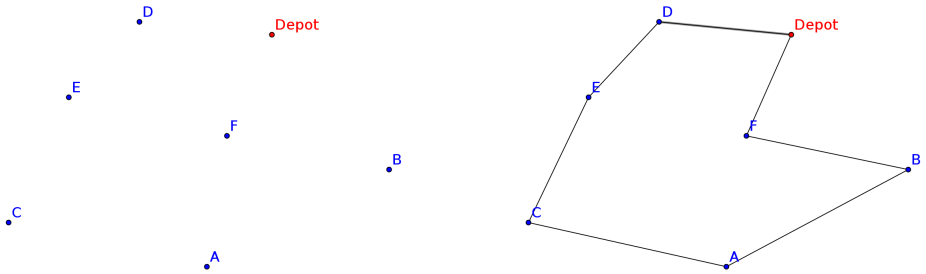
Now the NSGA II would start with a set of randomly created solutions as a set of parent solutions. The Competitive Location Model would enable the NSGA II automatically to compute the numbers of unserved inhabitants for the solutions, while the cost calculation still needs an algorithm for the routing itself, since the tour distances need to be calculated for the second objective value. Here, the Clarke Wright Savings Algorithm is employed (Clarke and Wright, 1964).

In each iteration of the NSGA II, after determining the solutions' opening pattern for the DCs, the costs of the opening situation have to be evaluated. One factor in the cost structure are routing cost. Here, two types of problems will likely be of use: The Traveling Salesman Problem (TSP) and the Vehicle Routing Problem (VRP).

The Traveling Salesman Problem, which was first described as "Botenproblem" by Karl Menger (Menger, 1931) consists of a set of points a hypothetical salesperson has to pass through, with known distances between the points. Now the shortest path linking those points has to be found, which is usually not the tour that results from always choosing the closest neighboring point as the next visited point (Menger, 1931, 23). In general, this TSP starts from a depot and returns at the end of the tour to the same depot, thus the TSP is solved by a Hamiltonian circle. Using Figure 5.10, one can see one instance of a TSP with 6 customers and one depot. In part b of the figure, one solution is presented. Upon close examination, one can determine the solution that would have resulted from the nearest neighbor heuristic: Depot-F-A-C-E-D-B-Depot. Since problems can get so big that more than one tour is required, either because the load would exceed the trucks' limit or the time would violate some time restrictions, a more sophisticated problem type is discussed next:

The VRP is a generalized TSP that has been introduced by Dantzig and Ramser in 1959 under the name of Truck Dispatching Problem (Dantzig and Ramser, 1959).

The formulation of the general VRP relies on the network made up by a graph $G(V, E)$, where
$V = \{v_0, v_1, \ldots, v_n\}$ is the set of vertices with v_0 representing the depot and
$E = \{(v_i, v_j) | v_i, v_j \in V; i < j\}$ is the set of edges. If the measure of impedance (e.g. cost)



(a) Problem (b) Solution

Figure 5.10: Example of the TSP and one possible solution (Source: own graphs)

to travel from v_i to v_j, c_{ij} is the same as from v_j to v_i, c_{ji} for all possible pairs of i and j, then the network is symmetric and no separate arc matrix has to be introduced that captures the second direction that the arc can be used.
C is the matrix that defines the (non-negative) cost for passing the edges and
$V' = V \setminus \{v_0\}$ is the set of n cities that have to be visited by vehicles.
d is the vector representing the demand at the customers and
m identical vehicles are assigned with
R_k, the route of vehicle k

Now every demand point V' has to be visited exactly once (if demand at the demand points is lower than the maximum vehicle capacity), and if cost is interpreted as the travel time, then a maximum total duration D for each vehicle can be set (Networking and Emerging Optimization Research Group, 2013). Dantzig and Ramser (1959) originally used a capacity constraint C per vehicle related to the demand of each demand point, which must not be confused with the cost matrix C described earlier (Burkart, 2013, pp.33f.).

For the solution of such problems very often heuristics are used, since the complexity grows very fast. In a situation with two customers, there are 2 possible tours: visiting first customer 1 and then the other one, or visiting first customer 2 and then the remaining one. These combinatorial possibilities increase very fast: with three customers, it's already 6 possible solutions, with 4 it's 24 solutions. As can be seen, the complexity (number x

possible solutions) grows with the factorial of the problem size (i customers):

$$x = 1 * 2 * 3 * ... * (i-1) * i = i! \tag{5.3.2.1}$$

The Clarke Wright Savings Algorithm is such a heuristic, and is widely used. Even modern Transport Management Systems, software designed to support the management of large fleets and also used for dispatching them, very often uses a savings heuristic (Burkart, 2013).

The Clarke Wright Savings Algorithm, as described in Larson and Odoni (1981), starts out with computing a list of savings that can be realized, if two DCs are combined in a tour compared to visiting every DC with a single tour. The list then is ordered with descending order of magnitude and then, the algorithm starts at the highest possible saving.

The savings pair is included in a route if one of the following situations occurs (and no capacity or time restrictions are violated by the combination):

- Both DCs are not yet included.
- Exactly one of the two DCs is included and is at the border of the tour, which means that it is either the first to be visited in this tour after the truck starts at the depot or the last before the truck returns.
- Both DCs are included in separate tours, but both DCs are at the edge of their tours as described in the previous bullet point, then the tours are merged.

This is done for every point on the savings list providing a positive savings value and when finished, the remaining DCs that are not included in tours get their own depot-DC-depot tours assigned.

Now the costs have been calculated for all solutions in the NSGA II, which can start over again until the specified number of iterations are finished. The resulting non-dominated front of solutions which gathered in the archive is then exported so the decision maker can have a closer look at the front or a subset of solutions from this front.

5.3.3 Final Decision

Now, the decision maker has to decide on which of the remaining solutions from the non-dominated front he wants to realize in the course of the disaster relief operations. One approach to decide on which solution to take will for sure be to recapitulate how much resources are available for the operations and then, the decision maker might decide on taking a solution that results in least unserved inhabitants at a given amount of money that can be spent. It is important to state that from data input and parameter specification to this point of choosing the appropriate solution no further tasks have to be carried out by the decision maker, which is a big advantage of the solution approach presented due to the reduced complexity. However, the quality of the computational results is highly important in such cases where decision makers can only trust the algorithms' results. Therefore, the results for the small instance are contrasted to the exact solution in the next section and the results for the big instance are also assessed.

5.4 Computational Results

5.4.1 Small Instance

The small instance has been solved with complete enumeration to receive the exact Pareto front, and with the heuristic with 10, 100, 500, 1000 and 10000 iterations. Each number of iterations has been used in six runs to achieve plausible average values for runtime, archive size at the end and the hypervolume value. In Tables 5.2 and 5.3, the computational core results are described. All computations have been conducted on an Intel Core i5-2540M with four CPUs with a clock speed of 2.6 GHz and 8 GB of RAM with Ubuntu Linux 12.10 running on it.

As can be seen, the runtime requirements for the heuristic are substantially lower than for the exact solution for every amount of iterations. The exact solution was computed in a little bit more than 12 hours, while 10000 iterations took not exactly 10 minutes. However, the runtime does not increase in a linear way, since for nearly every iteration, the archive grows and thus more comparison operations have to be conducted in the next iteration. Nevertheless, at some point the amount of solutions in the archive does not grow too fast

	Exact	Heuristic 10000 iterations	Heuristic 1000 iterations
Runs		6	6
Avg. Runtime	43682.7s	597.7283s	40.104s
% of exact runtime	100	1.3683	0.0918
avg. Archive Size	609	484.33	388
% of exact arch. size	100	79.529	63.711
Avg. Hypervolume Value	3007849772	3007650199.75	3006373778.4
% Hypervolume of Hypervolume of exact	100	99.993365	99.950929
Hypervolume Reference Point	Cost 87400, Unserved Inhabitants 53000		

Table 5.2: Results from the Hypervolume calculation for the exact solution and the Heuristic's results for 10000 and 1000 iterations

(Source: own table)

	Heuristic 500 iterations	Heuristic 100 iterations	Heuristic 10 iterations
Runs	6	6	6
Avg. Runtime	16.975s	2.047s	0.057s
% of exact runtime	0.03886	0.00469	0.0001
avg. Archive Size	329	226.5	99
% of exact arch. size	54.023	37.192	16.256
Avg. Hypervolume Value	2983098429.7	2975478882	2971774795
% Hypervolume of Hypervolume of exact	99.177108	98.923786	98.800639
Hypervolume Reference Point	Cost 87400, Unserved Inhabitants 53000		

Table 5.3: Results from the Hypervolume calculation for the Heuristic's results for 500, 100 and 10 iterations

(Source: own table)

anymore, as can be seen in Figure 5.19 for the big instance and 10000 iterations.Therefore, 1000 iterations do not even require more than 40 seconds, and 100 iterations are finished in 2 seconds.

The archive of the exact solution, i.e. the Paretofront, comprises 609 solutions, while 10000 iterations were able to find 484 solutions, 1000 iterations 388, 500 iterations 329, 100 iterations 226 and 10 iterations were able to find an astonishing 99 solutions, which is especially astonishing when compared to the real Paretofront, since they are very close. In Figure 5.11, the non-dominated fronts of the heuristic for every amount of iterations as well as the exact front can be seen. It is very interesting that the heuristics are generally very close to the optimal front, and only the 10 and 100 iterations' fronts show obvious deviations, especially around the bend. This bend, however, is due to the situation that

at some point, opening even more DCs does not improve the situation too much anymore, since the Competitive Location Model makes sure that a certain ratio of demand must always remain unserved as soon as the dummy facility has a positive attractiveness value.

To have a closer view on the bend, Figure 5.12 shows a closeup view of this region. The hypervolume value, calculated for the exact solution as well as as an average value for the different number of iterations in the heuristic, has by itself no meaning. Therefore, the most important information that could be retrieved from it is the ratio of the heuristics' hypervolume to the exact solutions' hypervolume. As can be seen in the tables, 10000 iterations heuristic achieves 99.993% of the hypervolume value of the exact solution, and even 1000 and 500 reach exceptional values like 99.95% and 99.17% respectively. Only the ratios for 100 and 10 iterations are below 99% with 98.92% and 98.8% respectively. However, these values show that the heuristic converges very soon towards a good approximation of the exact Paretofront.

In Figure 5.13, the non-dominated front for 1000 iterations is shown and one solution is emphasized. This solution is used for showing how the routing in the small instance works in Figure 5.14. The solution results in two tours, one starting at the depot, going to the DC in the north-west, then to the north-east and subsequently back to the depot, while the other tour goes to the far south near Praia do Chogoene and then back up to the far east on the EN1 and then back to the depot. As can be seen in this instance, the roads from the network do not fit exactly to the roads in the basemap.

5.4.2 Big Instance

Compared to the small instance with 21 possible DCs and 25 PCs, the complexity increased tremendously when compared to the big instance of 340 possible DCs and 422 PCs. While the amount of possible opening patterns for the 21 DCs was a already big

$$2^{21} = 2097152$$

in the case of 340 possible DCs this number increases to

$$2^{340} = 2.24 * 10^{102}$$

possibilities, exceeding the total number of atoms in the universe by far ($\sim 10^{77}$ according to Zöllner (2001))

	Heuristic 10000 iterations	Heuristic 1000 iterations	Heuristic 500 iterations
Avg. Runtime	100964s	3084.39s	1044.06s
% of 10000 iter. runtime	100	3.05494	1.03409
avg. Archive Size	2106	1198.5	998
% of archive size of 10000 iterations	100	56.908	47.388
Avg. Hypervolume Value	77948858004.4	72198484349.4	66955703339
% Hypervolume of Hypervolume of 10000 iter.	100	92.62289	85.89696
Hypervolume Reference Point	Cost 500000, Unserved Inhabitants 420000		

Table 5.4: Heuristic's results for 500, 1000 and 10000 iterations on the big instance (Source: own table)

	Heuristic 100 iterations	Heuristic 10 iterations
Avg. Runtime	201.66s	24.33s
% of 10000 iter. runtime	0.19973	0.0241
avg. Archive Size	460	133
% of archive size of 10000 iterations	21.842	6.315
Avg. Hypervolume Value	56051398103	49351999780
% Hypervolume of Hypervolume of 10000 iter.	71.90791	63.31330
Hypervolume Reference Point	Cost 500000, Unserved Inhabitants 420000	

Table 5.5: Heuristic's results for 10 and 100 iterations on the big instance (Source: own table)

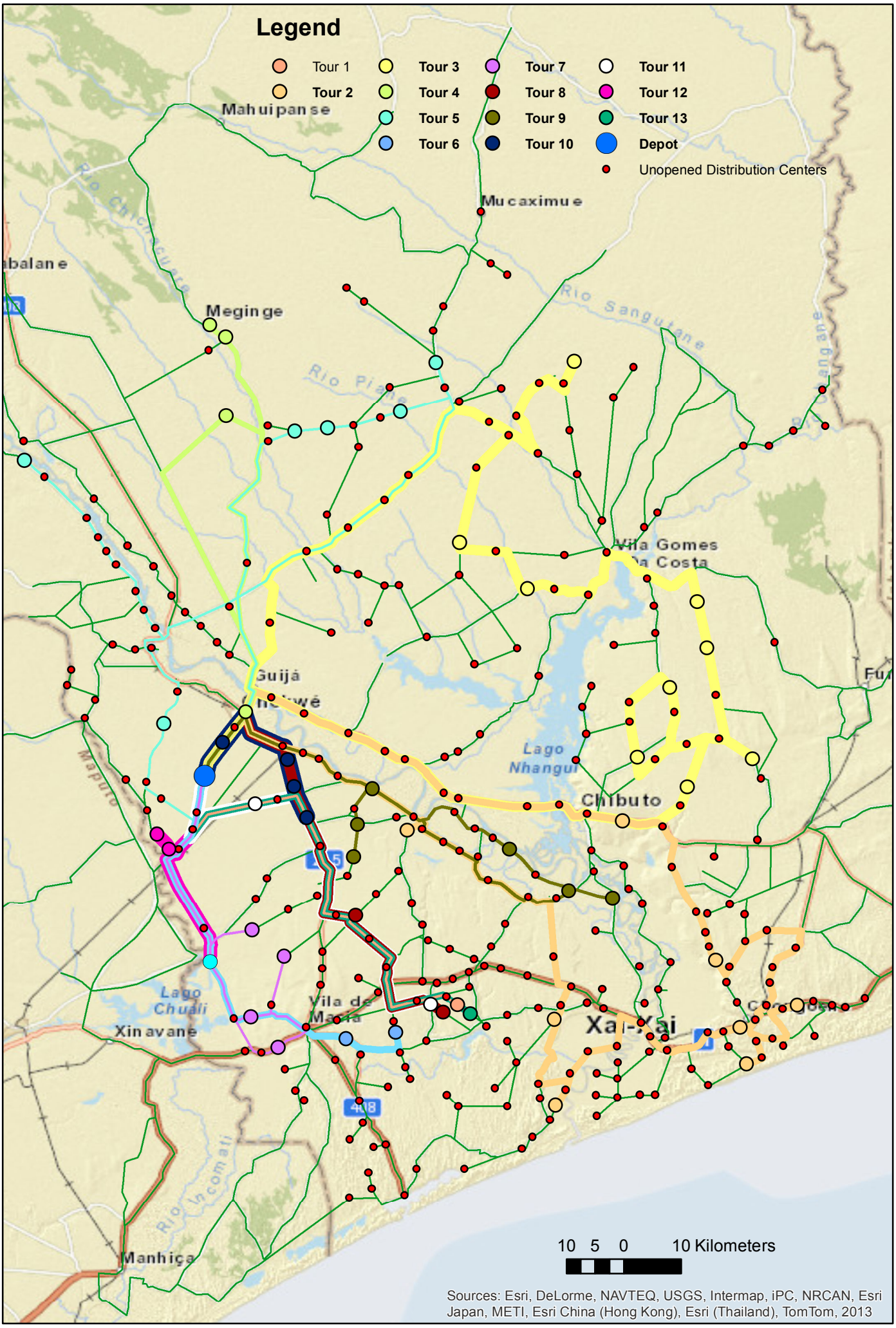
The results for the big instance in Tables 5.4 and 5.5 show similar characteristics as the results of the small instance. However, since the exact solution cannot be calculated for obvious reasons, the calculation of the heuristic for 10000 iterations serves as the benchmark for the results of the other amounts of iterations. This 10000 iterations took more than 28 hours of computation time, which can be explained by the fast growing amount of solutions in the archive, which is depicted in Figure 5.19. There, one can see that the size of the archive increases fast towards 2000 solutions, but then stays relatively constant over more than 5000 iterations. The small downward oriented spikes denote sudden findings of superior solutions removing older members of the archive.

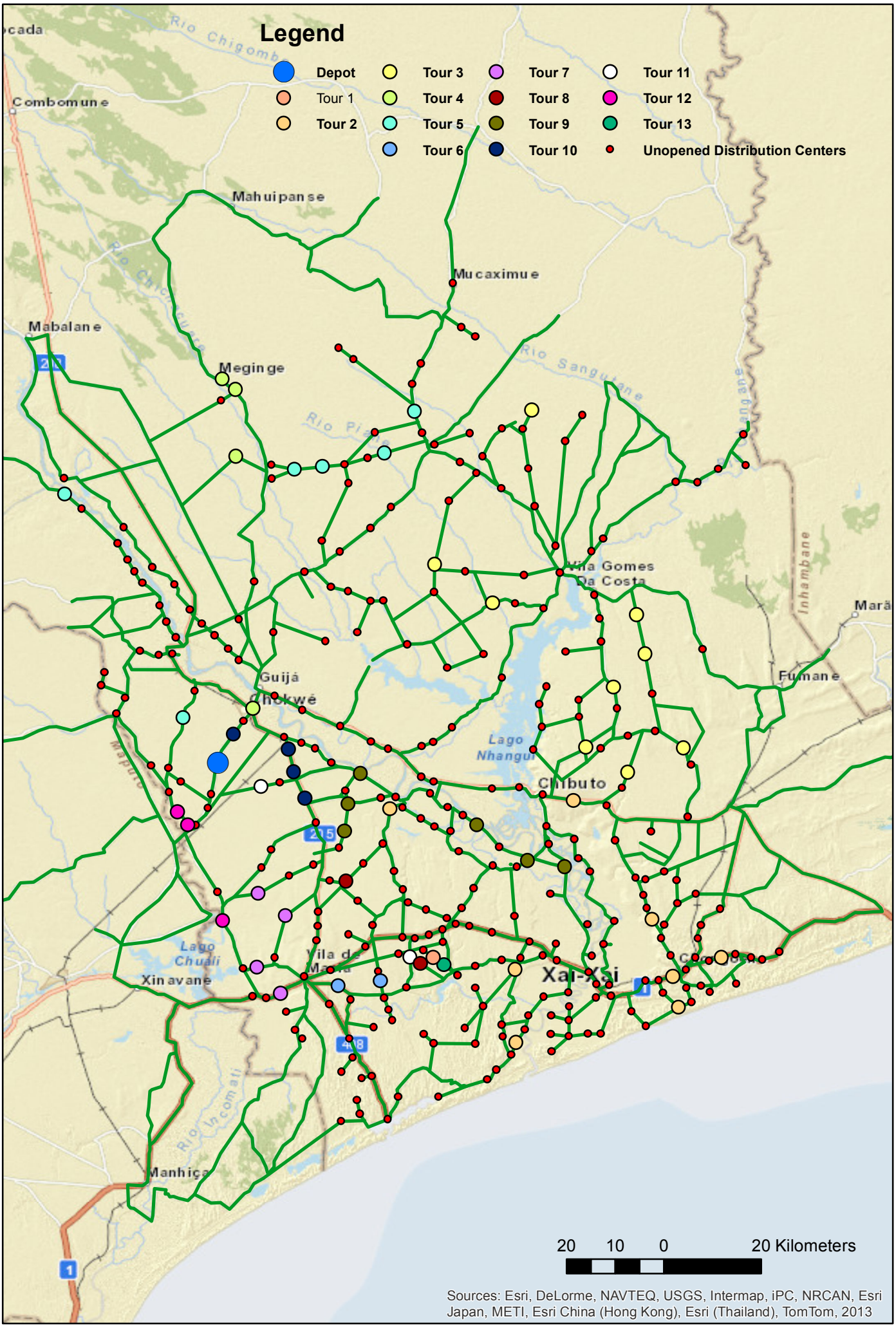
However, smaller amounts of iterations result in considerably less runtime, already 1000 iterations has a runtime of on average 51 minutes, 500, 100 and 10 iterations are finished in 17 minutes, 3.3 minutes and 24 seconds respectively. The archives are smaller, with around half the size for 1000 iterations at 1198 solutions, 998 solutions in the archive of the heuristic with 500 iterations and 460 and 133 solutions in the non-dominated front of the 100 and 10 iterations heuristic.

However, the relatively small amount of solutions in the archive compared to the biggest amount of iterations results in worse hypervolume ratios than in the small instance (1000 iterations: 92.6%, 500 iterations: 85.89%, 100 iterations: 71.9% and 10 iterations: 63.31%), although there is another factor for the bad performance here, which can be seen in Figure 5.15. The non-dominated front for the big instance seems to slowly stretch towards more expensive solutions with less unserved demand. This is due to the processes of the NSGA II, where evolutionary methods are used to find new solutions. Since with crossover and mutation only sequences of existing solutions are exchanged or single values of the solution are switched, the solutions develop slowly into more expensive solutions, which is also true in the other direction. The reason that it is faster evolving into more expensive solutions is that in the beginning, on average 30 % of DCs have been opened, so there is more opportunity to open up more than to open less.

As can be seen in Figure 5.16, the non-dominated front does not only stretch towards more extreme values, but also improve in quality by finding solutions that serve more for the same cost.

Again, the non-dominated front of 1000 iterations has been used to select one specific solution to check how the opening pattern and the related routing looks like. In Figure 5.18, one can see the opened DCs and, differentiated by colors, the other DCs that are served in the same tour, while in Figure 5.17, the single tours are depicted to allow for checking if the routing of the 13 created tours is reasonable.


Legend
Tour 1
Tour 2
Tour 3
Tour 4
Tour 5
Tour 6
Tour 7
Tour 8
Tour 9
Tour 10
Tour 11
Tour 12
Tour 13
Depot
Unopened Distribution Centers
Mahuipanse
Mucaximue
Meginge
Rio Sangutane
Rio Pisse
Vila Gomes
Costa
Guijá
Lago
Nhangui
Chibuto
Lago
Chuali
Xinavane
Vila de
Manica
Xai-Xai
Rio Incomati
Manhiça
10 5 0 10 Kilometers
Sources: Esri, DeLorme, NAVTEQ, USGS, Intermap, iPC, NRCAN, Esri Japan, METI, Esri China (Hong Kong), Esri (Thailand), TomTom, 2013


Legend
Depot
Tour 1
Tour 2
Tour 3
Tour 4
Tour 5
Tour 6
Tour 7
Tour 8
Tour 9
Tour 10
Tour 11
Tour 12
Tour 13
Unopened Distribution Centers
Combomune
Mahuipanse
Mucaximue
Mabalane
Meginge
Rio Sangutane
Rio Inhambane
Vila Gomes Da Costa
Marã
Guijá
Chokwé
Lago Nhanguí
Chibuto
Fumane
Lago Chuali
Xinavane
Vila de Maxia
Xai-Xai
Rio Incomati
Manhiça
20 10 0 20 Kilometers
Sources: Esri, DeLorme, NAVTEQ, USGS, Intermap, iPC, NRCAN, Esri Japan, METI, Esri China (Hong Kong), Esri (Thailand), TomTom, 2013

5.5 Assessment of Practical Relevance

The solution method explained in this thesis can be of practical relevance, since it enables the decision maker to use a different method to assign demand to the DCs as is usually employed. The possibility to take the preferences of the beneficiaries into consideration pays off for the more complex method of calculation.

One potential problem associated with solving the problem with such a type of model is the high requirement in terms of data. Since decisions have to be made fast in a situation of disaster, this requires the data on the transportation network, the PCs and DCs and, most of all, the potential pedestrian network to exist in a ready-to-use way. As soon as this information is not available, the process of developing such data is very time consuming and might inhibit the usage of the solution approach.

However, if the requirements can be met, then this solution approach provides the decision maker with a tool to come up with solutions by estimating parameters that have a real life meaning and can be estimated in a reasonable way, while maximizing the efficiency of the available funds and, at the same time, meeting the preferences of the beneficiaries.

Unfortunately, the presented approach only helps in a situation of deterministic information, which means that demand, travel distances and other factors are known with certainty, while in reality, such information will most likely not be available and, if at all, a probability distribution for the values can only be estimated.

A few limitations of the approach appear reasonable: People can only take relief goods for themselves and not for others in their PC. This might help prevent reselling of goods, while it requires everybody to come to a DC to get goods. And the assumption that all people can only walk to a DC can easily be relaxed: If some people are able to go to a DC with a car or even a bicycle, then the value pairs for the attractiveness parameter calculation have to be adapted, or the sphere of influence expanded due to the possibility to reach DCs that are further away.

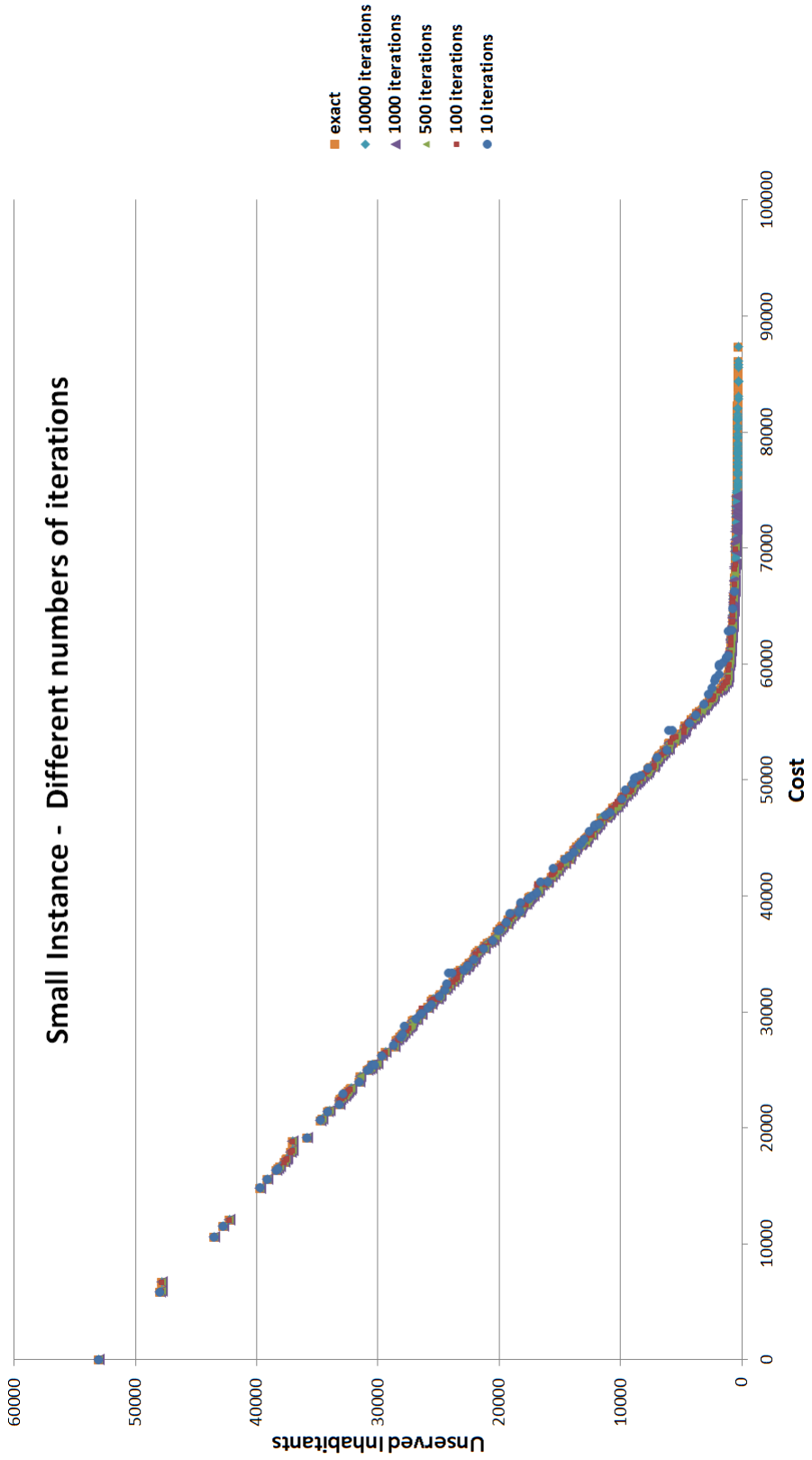


Figure 5.11: Paretofronts of exact solution and Heuristic using 10, 100, 500, 1000 and 10000 iterations

(Source: own graph)

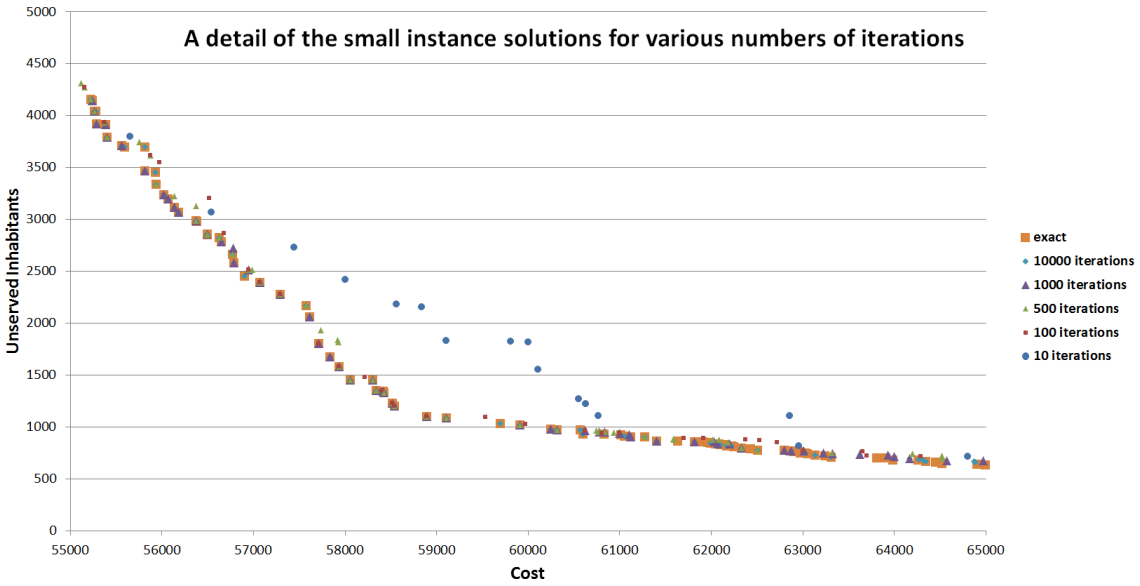


Figure 5.12: Detail of Figure 5.11
(Source: own graph)

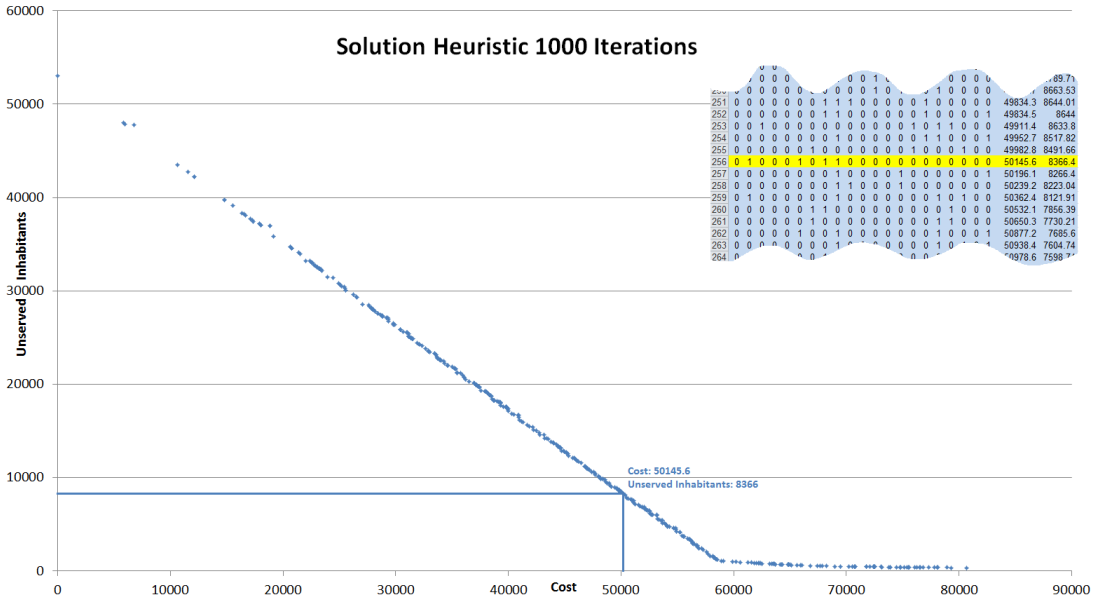


Figure 5.13: Paretofront of Heuristic with 1000 Iterations and single random solution
(Source: own graph)

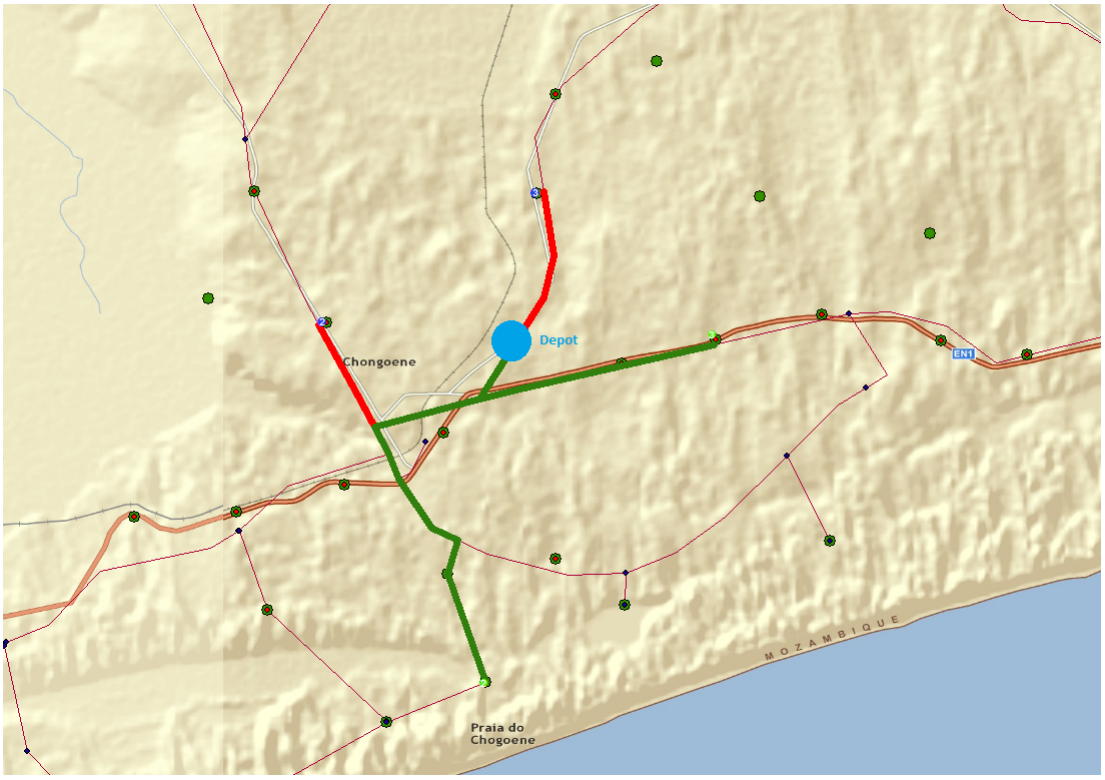


Figure 5.14: Routing solution of single random solution
(Source: own graph from ArcMap 10.1 with World Streetmap Basemap)

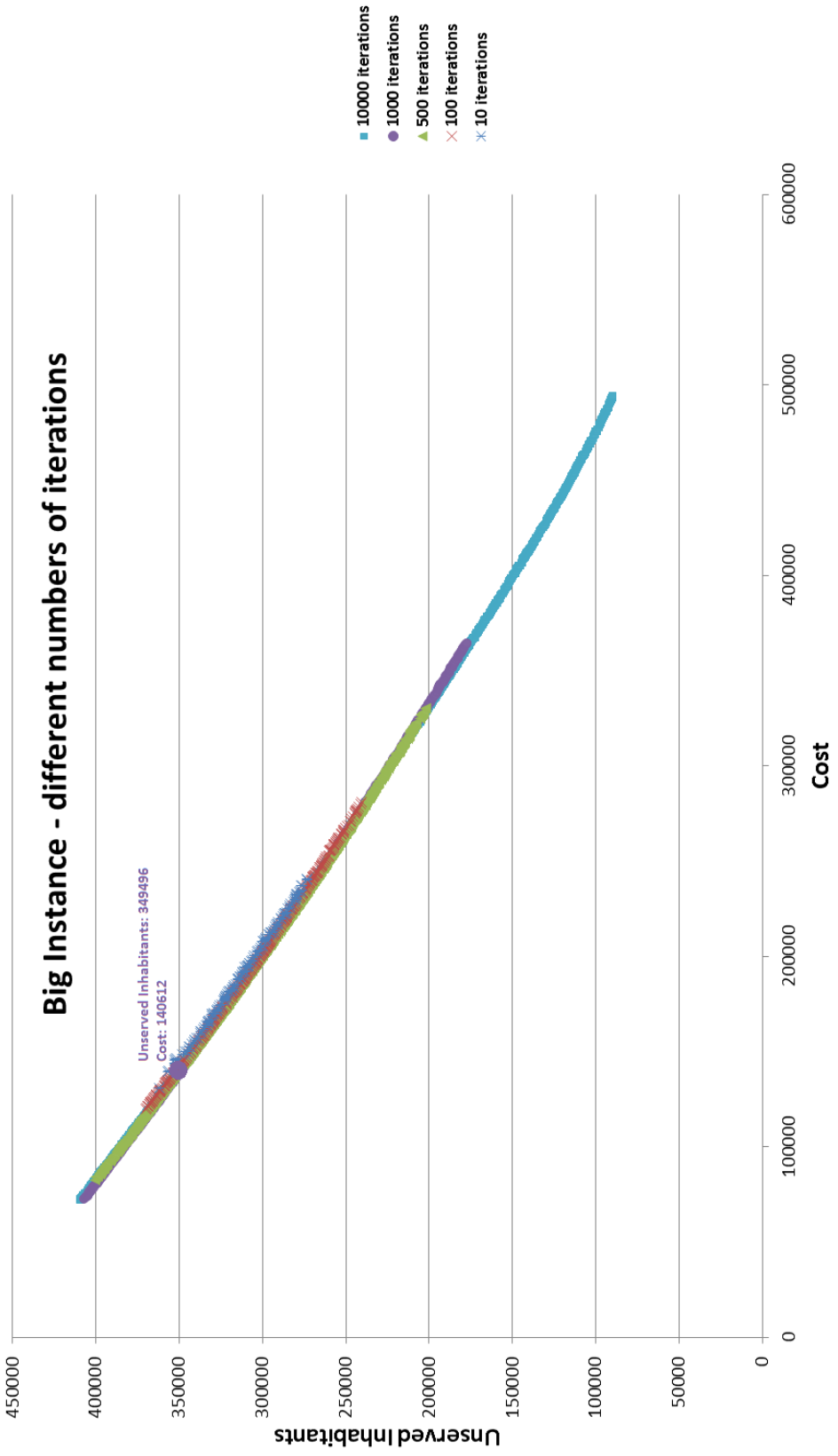


Figure 5.15: Paretofront of heuristic with 10, 100, 500, 1000 and 10000 iterations on big instance (340 DCs / 422 PCs)

(Source: own graph)

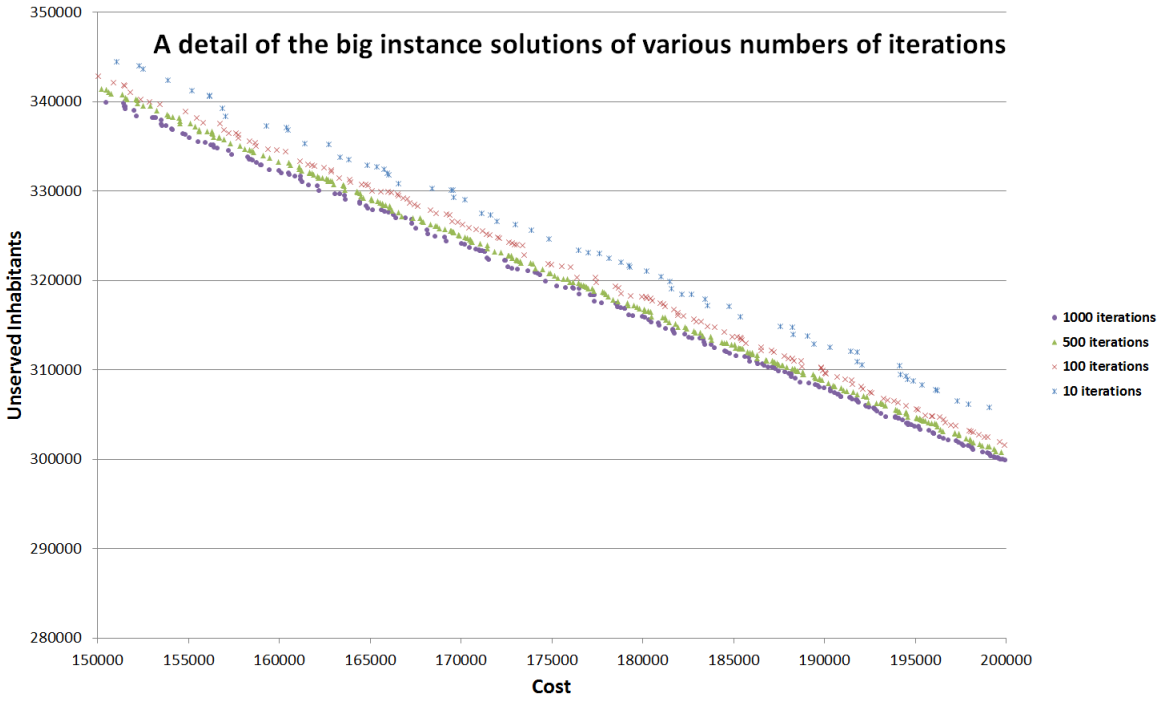


Figure 5.16: Detail of Figure 5.15
(Source: own graph)

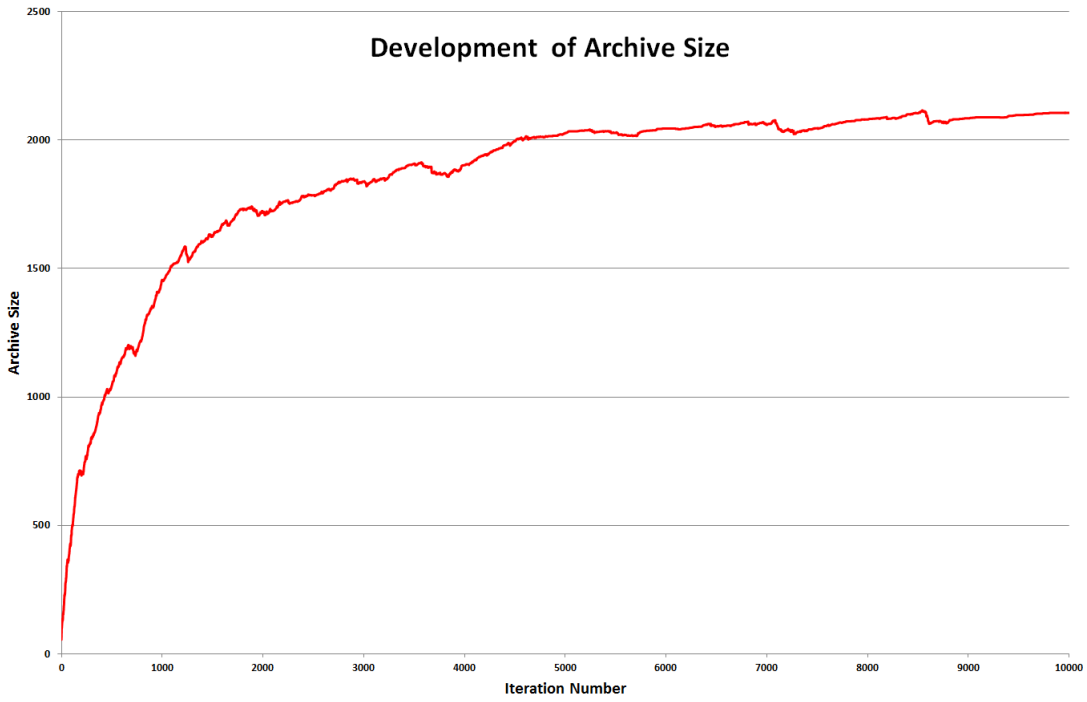


Figure 5.19: Development of Archive Size over 10000 Iterations for the big instance
(Source: own graph)

Chapter 6

Conclusion

In the course of this thesis, it has been shown how a new type of CTP, employing the allocation of demand based on the modeled preferences of the beneficiaries, can help solving the decision which possible distribution centers to open. The thesis has not only grounded the approach used in the area of Humanitarian Logistics, but has shown what implication the use of a Competitive Location Model has on different concepts of justice. From a utilitarian viewpoint, this approach is very useful, since it increases the utility of the beneficiaries by taking their utility into account by incorporating their preferences, or at least attempting to do so. However, speaking from the view of other definitions of justice, there might be much room for improvement. Since the two objective values can achieve highest values even if the geographical distribution ignored certain areas that are not easy to reach, Socialists would not see their requirement, to give everybody according to their needs, fulfilled, and Rawls would object that society would maybe favor a more equitable distribution if society could decide on it under the veil of ignorance.

However, one of the big advantages of this solution method is the easy application if the required data is available, but data availability might be required for any planning of relief goods distribution networks. The methodology has been described in great detail in the Chapters 3 and 4, and it has become clear that the use of this solution approach requires, apart from data input and the setting of a few parameters, very little interference of the decision maker apart from the final decision. On top of this, the inclusion of beneficiaries' preferences is an achievement in its own right, since usually only distances are included in such problem solution approaches.

Concerning limitations and assumptions, it is clear that the approach in this form is only valid for deterministic data, which means that everything, from demand to travel distances, is known beforehand. It is not clear how the approach behaves under uncertainty and this might be a very valuable research area for the future.

The assumption of a drought disaster is legitimate, but disasters are very often more complex in nature, and drought and the existence of hunger can often be attributed to man-made causes. So a very promising research area would be if disruptions would be included in some form, either by stochastic data or dynamic information on transport network disruptions. This could make the approach even more powerful and applicable in many different scenarios.

Another very important future research area would be to investigate how the preferences can be modeled better, since in this thesis it has been assumed that the inhabitants' preferences can be expressed as the capacity of the DC. This can and should be discussed further, but would have the great benefit that the beneficiaries and their preferences, who should form the addressee of disaster relief operations, get the attention they deserve.

All these limitations can be, without doubt, encountered with adaptations of the proposed method, which brings out the high practical potential this approach carries. A CTP that has the ability to take the real preferences of the beneficiaries into account and provides efficient solutions in a accessible, fast and easy way for decision makers in disaster relief operations would definitely be considered as a huge improvement.

Bibliography

Aragon, M., Barreto, A., Epstein, P. R., 1998. Drought and health implications in mozambique. Medicine and Global Survival 5, 42–49.
URL http://www.ippnw.org/pdf/mgs/5-1-aragon-barreto-epstein.pdf

Balcik, B., Beamon, B. M., Krejci, C. C., Muramatsu, K. M., Ramirez, M., 2010. Coordination in humanitarian relief chains: Practices, challenges and opportunities. International Journal of Production Economics 126, 22–34.

Beamon, B. M., 2004. Humanitarian relief chains: issues and challenges. In: Proceedings of the 34th International Conference on Computers and Industrial Engineering. University of Washington, pp. 1–6.

Bentham, J., 2007 [1789]. An introduction to the principles of morals and legislation. In: Sandel, M. J. (Ed.), Justice. A Reader. Oxford University Press, pp. 9–14.

Berger, M., 2004. Karl Marx: "Das Kapital". W. Fink.

Berié, E., 2012. Der neue Fischer Weltalmanach 2013: Zahlen, Daten, Fakten. Fischer Taschenbuch.

Burkart, C., 2013. Disruption management in transport management systems. Master's thesis, Vienna University of Economics and Business.

Chopra, S., Meindl, P., 2010. Supply Chain Management. Strategy, Planning and Operation., 4th Edition. Pearson.

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

Current, J., Schilling, D., 1989. The covering salesman problem. Transportation Science 23, 208–213.

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

Daskin, M. S., 2008. What you should know about location modeling. Naval Research Logistics 55, 283–294.

Deb, K., Pratap, A., Agarwal, S., Meyarivan, T., 2002. A fast and elitist multiobjective genetic algorithm: "NSGA-II". Evolutionary Computation, IEEE Transactions on 6 (2), 182–197.

Devex, 2014. Crisis in syria: What's in wfp food aid package?
URL `http://de.slideshare.net/Devex/wfp-food-aid-package-for-syria-crisis`

Drezner, T., Drezner, Z., 1998. Facility location in anticipation of future competition. Location Science 6, 155–173.

Drezner, T., Drezner, Z., 2012. Modelling lost demand in competitive facility location. Journal of the Operational Research Society 63 (2), 201–206.

Drezner, T., Eiselt, H., 2002. Consumers in competitive location models. Berlin: Springer, Ch. 5, pp. 151–178.

EM-DAT: The OFDA/CRED International Disaster Database – www.emdat.be, Université Catholique de Louvain, B. B., January 2014. Em-dat country profile.
URL `http://www.emdat.be/result-country-profile`

Fink, G., Redaelli, S., 2009. Determinants of international emergency aid. humanitarian need only? World Bank Policy Research Working Paper 4839, 1–36.

Friedman, M., Friedman, R., 2007 [1980]. Created equal, in: Free to choose, a personal statement. In: Sandel, M. (Ed.), Justice. A Reader. Oxford University Press, pp. 49–60.

Gendreau, M., Laporte, G., Semet, F., 1997. The covering tour problem. Operations Research 45, 568–576.

"Governo da Provincia de Gaza", 2014. Gaza. Portal do Governo da Provincia de Gaza. URL http://www.gaza.gov.mz/gaza

Hotelling, H., 1929. Stability in competition. The Economic Journal 39 (153), 41–57.

Huff, D. L., 1966. A programmed solution for approximating an optimum retail location. Land Economics 42 (3), 293–303.

IRIN, October 2005. Mozambique: Drought after drought. URL http://www.irinnews.org/report/56812/mozambique-drought-after-drought

Jaspars, S., Shoham, J., 1999. Targeting the vulnerable: A review of the necessity and feasibility of targeting vulnerable households. Disasters 23, 359–372.

Knowles, J., Thiele, L., Zitzler, E., 2006. A tutorial on the performance assessment of stochastic multiobjective optimizers. TIK-Report 214, Computer Engineering and Networks Laboratory ETH Zurich. URL http://www.tik.ee.ethz.ch/sop/publicationListFiles/ktz2006a.pdf

Kovacs, G., Spens, K., 2009. Identifying challenges in humanitarian logistics. International Journal of Physical Distribution & Logistics Management 39, 506–528.

Kummer, S., Groschopf, W., Grün, O., Brunner, J.-C., Jammernegg, W., Poiger, M., 2013. Grundzüge der Beschaffung, Produktion und Logistik. Pearson.

Larson, R. C., Odoni, A. R., 1981. Urban operations research. No. Monograph. Prentice-Hall. URL http://web.mit.edu/urban_or_book/www/book/

Marler, R., Arora, J., 2004. Survey of multi-objective optimization methods for engineering. Structural and Multidisciplinary Optimization 26, 369–395.

Marx, K., 1875. Critique of the gotha programme. URL http://www.marxists.org/archive/marx/works/1875/gotha/ch01.htm

Marx, K., Engels, F., 2012 [1848]. The communist manifesto. Socialist Register 34, 240–268.

Menger, K., 1931. Bericht über ein mathematisches kolloquium 1929/30. Monatshefte für Mathematik 38 (1), 17–38.

Mercedes-Benz South Africa, 2013. Truck price card 2013.
URL `www.mercedes-benz.co.za/content/media_library/south_africa/mpc/trucks/trucks_price_card.object-Single-MEDIA.download.tmp/Truck+Price+Card+2013.pdf`

Mill, J. S., 2007 [1861]. Utilitarianism. In: Sandel, M. (Ed.), Justice. A Reader. Oxford University Press, pp. 14–47.

Networking and Emerging Optimization Research Group, 2013. Vehicle routing problem.
URL `http://neo.lcc.uma.es/vrp/vehicle-routing-problem/`

Nolz, P. C., Doerner, K. F., Hartl, R. F., 2010. Water distribution in disaster relief. International Journal of Physical Distribution & Logistics Management 40, 693–708.

Nozick, R., 2007. Anarchy, state and utopia. In: Sandel, M. (Ed.), Justice. A Reader. Oxford University Press, pp. 60–82.

Pedraza Martinez, A. J., Stapleton, O., van Wassenhove, L. N., 2011. Field vehicle fleet management in humanitarian operations: A case-based approach. Journal of Operations Management 29, 404–421.

Raschky, P. A., Schwindt, M., 2009a. Aid, natural disasters and the samaritan's dilemma. The World Bank Working Paper 4952, 1–41.

Raschky, P. A., Schwindt, M., 2009b. On the channel and type of international disaster aid. The World Bank Working Paper 4953, 1–29.

Rawls, J., 2007 [1971]. A Theory of Justice. Oxford University Press, Ch. 7, pp. 203–221.

Späthling, A., 2007. Modelling and analysis of network data after natural disasters. Master's thesis, University of Vienna.

Thomas, A. S., Kopczak, L. R., 2005. From Logistics to Supply Chain Management: The Path Forward in the Humanitarian Sector. Fritz Institute.

Tomasini, R., van Wassenhove, L., 2009. Humanitarian Logistics. Palgrave Macmillan.

Tricoire, F., Graf, A., Gutjahr, W. J., 2012. The bi-objective stochastic covering tour problem. Computers & Operations Research 39, 1582–1592.

van Wassenhove, L., 2006. Blackett memorial lecture. humanitarian aid logistics: supply chain management in high gear. Journal of the Operational Research Society 57, 475–489.

Zitzler, E., Thiele, L., 1999. Multiobjective evolutionary algorithms: A comparative case study and the strength pareto approach. IEEE Transactions on Evolutionary Computation 4, 257–271.

Zöllner, D., 2001. Verfahren der kryptographie.
URL `http://fma2.math.uni-magdeburg.de/~bessen/krypto/krypto8.htm`

Appendices

Appendix A

Program Codes

```
//*************************************************************************
//Project: Competitive location models for transportation in disaster relief logistics
//Author: Christian Burkart
//*************************************************************************


// library includes
#include <iostream>
#include <cmath>
#include <cstdlib>
#include<cstring>
#include <fstream>
#include <vector>
#include <map>
#include <cfloat>
#include <algorithm>
#include <ctime>
using namespace std;

// ***** global variables, structures and constants *****
#define V_ROWS 21       //number of origins
#define V_COLUMNS 21    //number of destinations
#define P_ROWS 25
#define P_COLUMNS 21 //destinations of pedestrians = length of binary vector
#define D_DIST 15000 //dummydist = sphere of influence in meters, used also for capacitycalc
#define INITIAL_RATIO 0.3 //probability for having a dc built in the initial solution generation
#define SIZE_2N 100 //size of the full set of solutions, must be (SIZE_2N % 2 = 0)
#define N 50        //size of half set, e.g. parent or offspring set
#define ITERATIONS 10000
//################Lambda calc global variables
#define RATIO_A 0.99  //these values describe, which ratio x of population should go to a dc of
average capacity in the distance of DIST_X in meters. has to be adjusted if Dummy dist changes
#define RATIO_B 0.66  //there are three ratios to build the average of the corresponding lambda values.
#define DIST_A 1000    // ratios are required
#define DIST_B 10000
//################Cost calc global variables
#define UNITCOST 1          // cost for one delivered unit
#define DIST_COST 0.001     // cost for one meter driven by truck
#define FIXED_DC 200         // fixed cost for starting a dc, but divided onto the usage horizon, f.e.
cost per week if weekly deliveries
#define VARIABLE_DC 100      // variable cost for every 1000 units capacity, divided over the usage
horizon, f.e. cost                                    per week if weekly deliveries, CAVE! linear
cost evolution assumed!!!
#define TRUCK_CAP 10000
#define CROSSOVERPROB 0.8      //the mutation probability ist 1-CROSSOVERPROB

struct solution  //used as content of the set of solutions.
{
   bool openings[P_COLUMNS];
   float demand[P_COLUMNS];   //demand at dcs, calculated in unserved calc function
   float cost;
   float unserved;
   vector<int> setDominated;         //used in NSGAII sorting
   int dominationCounter;            //used in NSGAII sorting
   int frontCounter;                 //used in NSGAII sorting
};

struct savingsPair          //used in clarke wright savings for the multimap as mapped value, while
savings is key value                                    there
{
  int O;
  int D;
};

struct crowdingStruct
{
  float unserved;
  float cost;
  int ID;
};

// ***** functions and function prototypes *****
int dataInput (float vdistmatrix[V_COLUMNS][V_ROWS], float pdistmatrix[P_COLUMNS][P_ROWS], int
population[P_ROWS])
```

```
{
    int origin = -1;
    int destination = -1;
    float distance = -1;
    char waste;
    char buffer[1000];
    int totalpopulation = 0;
    ifstream vdata("vdata.txt");
    vdata.getline(buffer,1000);
    while (!vdata.eof())
        {
           vdata >> origin;
           origin--;                    //adjust for matrix layout
           vdata.get(waste);
           vdata >> destination;
           destination--;               //adjust for matrix layout
           vdata.get(waste);
           vdata >> distance;
           vdistmatrix[destination][origin] = distance;
        }
    //########################pedestrian matrix input
    ifstream pdata("pdata.txt");
    pdata.getline(buffer,1000);
    while (!pdata.eof())
        {
           pdata >> origin;
           origin--;                    //adjust for matrix layout
           pdata.get(waste);
           pdata >> destination;
           destination--;               //adjust for matrix layout
           pdata.get(waste);
           pdata >> distance;
           pdistmatrix[destination][origin] = distance;
        }
    //######################population number input part
    ifstream popdata ("population.txt");
    popdata.getline(buffer,1000);
    for (int i = 0; i < P_ROWS;i++)
    {
        popdata >> population[i];
        totalpopulation = totalpopulation + population[i];
    }
    return totalpopulation;
}


void calcCAP_RATIO(float pdistmatrix[P_COLUMNS][P_ROWS], float CAP_RATIO[P_COLUMNS], int
totalpopulation, float vdistmatrix[V_COLUMNS][V_ROWS])       // just needed if capratio endogeneous
variable, also transmission of CAP_RATIO to                     calccapacities and the variable
itself!!!
{                                                                                   // = 1 / (how
many dcs have the              pcs in reach of this dc in reach (d_dist) on average)  for each dc->
capratio is an array
                                                                                    // correction
for ratio of fixed                      dc cost to delivery cost calc
    int tempDCsums[P_ROWS];             //dcs in reach of every pc
    float tempDCavg[P_COLUMNS];         // avg for dcs available to pcs in reach of dc...
    float avgDisttoDepot = 0;
    float count = 0;
    float count2 = 0;
    float count3 = 0;
    float ratioMultiplier = 0;
    // average distance from depots to dcs is calculated
    for (int fo = 0; fo < V_COLUMNS; fo++)
    avgDisttoDepot += vdistmatrix[fo][0];               // depot is situated at 0
    avgDisttoDepot = avgDisttoDepot / V_COLUMNS;
    //first, dcs in reach for every pc are summed up
    for (int fa = 0; fa < P_ROWS; fa++)
    {
        for (int fe = 0; fe < P_COLUMNS; fe++)
        {
            if (pdistmatrix[fe][fa] <= D_DIST)
            count++;
        }
```

```
        tempDCsums[fa] = count;
        count = 0;
    }
    // then, for every dc, sums of possible dcs for every pc in reach are summed and divided by number
of pcs in reach
    for (int fu = 0; fu < P_COLUMNS; fu++)
    {
        for (int fi = 0; fi < P_ROWS; fi++)
        {
            if (pdistmatrix[fu][fi] <= D_DIST)
            {
                count3++;
                count2 += tempDCsums[fi];
            }
        }
        tempDCavg[fu] = count2 / count3;
        CAP_RATIO[fu] = 1 / (count2 / count3);
        count3 = 0;
        count2 = 0;
    }
    // now, ratio is adjusted for ratio of fixed dc cost to approximated delivery cost
    ratioMultiplier = (FIXED_DC)/(2 * DIST_COST * avgDisttoDepot);
    for (int fb = 0; fb < P_COLUMNS; fb++)
    {
        if (ratioMultiplier < 1)
        continue;
        if (ratioMultiplier > tempDCavg[fb])
        CAP_RATIO[fb] *= tempDCavg[fb];
        if (ratioMultiplier >= 1 && ratioMultiplier <= tempDCavg[fb])
        CAP_RATIO[fb] *= ratioMultiplier;
    }
    return;
}


float calcCapacities(float capacities[P_COLUMNS], int population[P_ROWS], float pdistmatrix[P_COLUMNS]
[P_ROWS], float CAP_RATIO[P_COLUMNS])
{   // sum all pop in the next D_DIST meters from possible DC on pedestrian netwrok and multiply with
CAP_RATIO for this dc, so
    // stays the same for all calculations if CAP_RATIO[i] remains the same
    float tempSum = 0;
    float averagesum = 0;
    float averagecapacities = 0;
    for(int i = 0; i < P_COLUMNS; i++)
    {
        tempSum = 0;
        for (int j = 0; j < P_ROWS; j++)
        {
            if (pdistmatrix[i][j] <= D_DIST)
            {
                tempSum = tempSum + population[j];
            }
        }
        tempSum = tempSum * CAP_RATIO[i];
        capacities[i] = tempSum;
    }
    for(int z = 0; z < P_COLUMNS; z++)
       {
           averagesum = averagesum + capacities[z];
       }
    averagecapacities = averagesum / P_COLUMNS;
    return averagecapacities;
}

void createInitialSolutions(solution solutionS[SIZE_2N])
{
    int random = -1;
    for(int i = 0; i < SIZE_2N; i++)
    {
        srand(i);              //srand(time(NULL)); results in always the same sequence of binaries.
so different way has been selected
        for (int j = 0; j < P_COLUMNS; j++)
        {
            random = rand();
```

```
            if (random < RAND_MAX * INITIAL_RATIO)
            solutionS[i].openings[j] = 1;
            else
            solutionS[i].openings[j] = 0;
        }
        solutionS[i].dominationCounter = 0;
        solutionS[i].frontCounter = 0;
    }
    return;
}

float calcAttractivity(float capacities[P_COLUMNS], float pdistmatrix[P_COLUMNS][P_ROWS], float
attractivityMatrix[P_COLUMNS][P_ROWS], float dummyA, float lambda)
        // the calculation is based on drezner drezner 2012,
{
    float dummyAttractivity = dummyA * exp(lambda * D_DIST);
    for(int x = 0; x < P_ROWS; x++) //every pc
    {
        for(int y = 0; y < P_COLUMNS; y++) //every dc
        {
            if(pdistmatrix[y][x] > D_DIST)
               attractivityMatrix[y][x] = 0;
            else
            {
                attractivityMatrix[y][x] = capacities[y] * exp(lambda * pdistmatrix[y][x]);
            }
        }
    }
    return dummyAttractivity;
}

float calcLambda (float & dummyA, float averagecapacities)
{
    float lambda = 0;
    dummyA = averagecapacities * (pow ((1/RATIO_A)-1 , (D_DIST- DIST_B) / (DIST_A - DIST_B))) * pow
((1 / RATIO_B) - 1 , (DIST_A - D_DIST) / (DIST_A - DIST_B)); //first, the Dummy attractivity value is
calculated
    lambda = log(((averagecapacities/RATIO_A)-averagecapacities)/dummyA)/(D_DIST-
DIST_A);               // second, fill in dummy attractivity to calculate lambda
    return lambda;
}

void calcUnserved(float capacities[P_COLUMNS], int population[P_ROWS], solution solutionS[SIZE_2N],
float attractivityMatrix[P_COLUMNS][P_ROWS], float dummyAttractivity)
{
    //for each pc calc how much demand will go to which dc, save value (in cumDemandDC) and add it upp
and add up dummydemand fo each PC
    // after every PC, sum of demands at DC is compared to capacity. if less cap, then add difference
to dummydemand and write value into
    // solutionS[i].unserved. this procedure allows no insight into who gets a package at the dc and
who not, it also assumes that people
    // do not organize to spread demand towards the underutilized dcs, and stuff might be left over!
    // also every person is entitled to one unit, does not differentiate if needs are bigger or
smaller, very egalitarian! same for max walking dist
    float tempSumDummy = 0;
    float cumDemandDC[P_COLUMNS];
    for(int y = 0; y < P_COLUMNS; y++)  //initialisierung
    cumDemandDC[y] = 0;
    float cumAttractivity[P_ROWS];
    for(int u = 0; u < P_ROWS; u++)
    cumAttractivity[u] = 0;
    for(int k = 0; k < SIZE_2N; k++) // unserved calc
    {
        tempSumDummy = 0;
        for(int b = 0; b < P_COLUMNS; b++)
        cumDemandDC[b] = 0;
        for(int c = 0; c < P_ROWS; c++)
        cumAttractivity[c] = 0;             // cleaning up for every new solution assessed
        // now, for every pc, the total attractivity is calculated, then the ratio of the dummy times
population is added to tempSumDummy

        for (int i = 0; i < P_COLUMNS; i++)
        {
           for (int u = 0; u < P_ROWS; u++)
```

```
            {
                cumAttractivity[u] += (attractivityMatrix[i][u] * solutionS[k].openings[i]);
            }
        }

        for (int w = 0; w < P_ROWS; w++)
        {
            tempSumDummy += (population[w]* (dummyAttractivity / (cumAttractivity[w] +
dummyAttractivity)));          //now all people that do not go to a DC are summed up
        }

        // now, the demand at the DCs has to be summed up to show if demand exceeds capacity at some
DCs, excess is added to tempSumDummy as well
        for (int n = 0; n < P_COLUMNS; n++)
        {
           for (int m = 0; m < P_ROWS; m++)
           {
                cumDemandDC[n] += (population[m] * solutionS[k].openings[n] * (attractivityMatrix[n]
[m] / (cumAttractivity[m] + dummyAttractivity)));
           }
        }
        for (int p = 0; p < P_COLUMNS; p++)
        {
            if (cumDemandDC[p] > capacities[p])
            {
                tempSumDummy += (cumDemandDC[p] - capacities[p]);
                solutionS[k].demand[p] = capacities[p];             //this makes sure the expected
demand is calculated and stored right, excessive demand over capacity will be ignored!!!!!
            }
            else
            solutionS[k].demand[p] = cumDemandDC[p];
        }

        // now, the value for the unserved has to be written into the solutionS file
        solutionS[k].unserved = tempSumDummy;
    }
return;
}

void calcCost(float capacities[P_COLUMNS], int population[P_ROWS], float vdistmatrix[V_COLUMNS]
[V_ROWS], solution solutionS[SIZE_2N], int totalpopulation)
{
// cost per thousand units cap as a global var, fixed opening cost small or not existing, plus tour
cost in faktor * km, cost per packet?
// tour cost has to be calculated from expected demand, not capacities in the dcs, since this exceeds
maximum possible demand largely, depending on CAP_RATIO
// starting point for the  tours is the first dc, assuming there is a depot in very close proximity
float cost = 0;
float savings = 0;
savingsPair paar;
bool oIncludable = false;
bool dIncludable = false;           // these two bool are used to identify if a savings origin or
destination can be included.
int tourNumberorigin = -1;
int tourNumberdestination = -1;     // these two are used to identify the position of the tour to be
merged in the iteration, the last 4 variables have to be set to 0/false every iter.
int tourNumbermissing = -1;         //used for checking if a dc is forgotten in tours
multimap<float, savingsPair, greater<float> > savingsList;          // attention, map orders
descending with argument greater<float>!
vector<float> tourDemand;           //used to add up the demand on the tours of the CWS for each tour
vector< vector<int> > tours;        //used to collect all tours
vector<int> neu;                    // used for putting new tours into tours vector
float cumTourmeters = 0; // used for calculating the total CWS cost
for (int z = 0; z < SIZE_2N; z++)
{
    tours.clear();
    tourDemand.clear();
    cost = 0;
    cost += UNITCOST * (totalpopulation - solutionS[z].unserved); //here unit cost are included for
all served people
    for (int o = 0; o < P_COLUMNS; o++)
    {
        cost += FIXED_DC * (solutionS[z].openings[o]);      // fixed opening costs included
        cost += VARIABLE_DC * (solutionS[z].openings[o] * capacities[o] / 1000); // variable costs for
```

```
dc capacity included
    }
// Clarke-Wright Savings
// first, a new multimap savingslist with savings as key and  savingsPair as mapped value is
calculated for each combination, in which both origin and
// destination are either open dcs or the depot, but not the same
    for (int a = 0; a < V_ROWS; a++)
    {
        if (solutionS[z].openings[a] == 0)
        continue;                                       // use just connections that actually include
opened dcs
        for (int b = 0; b < V_COLUMNS; b++)             //this double loop creates savingsList,
depending on solutionS[z]-openings[]
        {
            if (solutionS[z].openings[b] == 0 || a == b)          //same as with a: just include
opened dcs in savingsList
            continue;
            savings = vdistmatrix[0][a] + vdistmatrix[0][b] - vdistmatrix[a][b];        //
savingscalculation
            paar.O = a;
            paar.D = b;
            savingsList.insert(pair<float, savingsPair> (savings, paar));
        }
    }
    for (multimap<float,savingsPair>::iterator it=savingsList.begin(); it!=savingsList.end(); +
+it)          //iteration through all savings
    {
        oIncludable = true;
        dIncludable = true;
        tourNumberorigin = -1;
        tourNumberdestination = -1;         //reset for every iteration
        for (unsigned int d = 0; d < tours.size(); d++)              //this section checks if dc is
already included and lies on the edge of a tour
        {
            for (unsigned int e = 0; e < tours[d].size(); e++)
            {
                if (tours[d][e] == (*it).second.O)
                {
                    if(e == 0 || e == (tours[d].size()-1))
                    {
                       tourNumberorigin = d;
                    }
                    else
                    oIncludable = false;
                }
                if (tours[d][e] == (*it).second.D)
                {
                    if(e == 0 || e == (tours[d].size()-1))
                    {
                        tourNumberdestination = d;
                    }
                    else
                    dIncludable = false;
                }
            }
        }
        if (oIncludable == true && dIncludable == true && (*it).first > 0) //both points are not in
the inner part of an existing tour and savings is positive
        {
            if (tourNumberdestination == -1 && tourNumberorigin == -1) //if both dcs are not yet
included, inside these ifs the capacity violation has to be
            {
                if ((solutionS[z].demand[(*it).second.O]+ solutionS[z].demand[(*it).second.D]) <=
TRUCK_CAP)
                {
                    tourDemand.push_back(solutionS[z].demand[(*it).second.O]+ solutionS[z].demand
[(*it).second.D]);
                    neu.push_back((*it).second.O);
                    neu.push_back((*it).second.D);
                    tours.push_back(neu);
                    neu.pop_back();
                    neu.pop_back();                 //neu has to be cleared for next use
                }
                else
```

```
                continue;
            }
            if (tourNumberdestination == -1 && tourNumberorigin != -1 ) // one dc is included at an
edge, the other not
            {
                if ((solutionS[z].demand[(*it).second.D] + tourDemand[tourNumberorigin]) <= TRUCK_CAP)
                {
                    tourDemand[tourNumberorigin] += solutionS[z].demand[(*it).second.D];
                    if (tours[tourNumberorigin][0] == (*it).second.O)           //add at begin
                    {
                        std::vector<int>::iterator it3;
                        it3 = tours[tourNumberorigin].begin();
                        tours[tourNumberorigin].insert (it3,1,(*it).second.D);
                    }
                    if (tours[tourNumberorigin][tours[tourNumberorigin].size() -1] ==
(*it).second.O)         //add at end  !!!! Symmetric dist matrix!!!!!
                    {
                        tours[tourNumberorigin].push_back((*it).second.D);
                    }
                }
                else
                continue;
            }
            if (tourNumberdestination != -1 && tourNumberorigin == -1) // one dc is included at an
edge, the other not. the other way around
            {
                if ((solutionS[z].demand[(*it).second.O] + tourDemand[tourNumberdestination]) <=
TRUCK_CAP)
                {
                    tourDemand[tourNumberdestination] += solutionS[z].demand[(*it).second.O];
                    if (tours[tourNumberdestination][0] == (*it).second.D)           //add at begin
                    {
                        std::vector<int>::iterator it4;
                        it4 = tours[tourNumberdestination].begin();
                        tours[tourNumberdestination].insert (it4,1,(*it).second.O);
                    }
                    if (tours[tourNumberdestination][tours[tourNumberdestination].size() -1] ==
(*it).second.D)         //add at end  !!!! Symmetric dist matrix!!!!!
                    {
                        tours[tourNumberdestination].push_back((*it).second.O);
                    }
                }
                else
                continue;
            }
            if (tourNumberdestination != -1 && tourNumberorigin != -1 && tourNumberdestination !=
tourNumberorigin)  // if both are in (different) tours that can be connected
            {
                if ((tourDemand[tourNumberdestination] + tourDemand[tourNumberorigin]) <= TRUCK_CAP)
                {
                    //4 possibilities how tours should be merged: simple stick together O-D tours, D-O
tours, reverse order of O-D, O-reverse order of D, due to symmetric distcost
                    if ((tours[tourNumberdestination][0] == (*it).second.D) && (tours[tourNumberorigin]
[0] == (*it).second.O))      // both at the begin of the tours
                    {
                        for (unsigned int aa = 0; aa < tours[tourNumberorigin].size(); aa++)
                        {
                            std::vector<int>::iterator it5;
                            it5 = tours[tourNumberdestination].begin();
                            tours[tourNumberdestination].insert(it5,1,tours[tourNumberorigin][aa]);
                        }
                        tourDemand[tourNumberdestination] += tourDemand[tourNumberorigin];          //
sum up tour demand in new tours
                        tourDemand.erase (tourDemand.begin() +
tourNumberorigin);                      // delete old tour and associated demand
                        tours.erase (tours.begin() + tourNumberorigin);
                    }
                     if ((tours[tourNumberdestination][0] == (*it).second.D) && (tours
[tourNumberorigin][tours[tourNumberorigin].size() -1] == (*it).second.O)) //origin at end, destination
at begin
                    {
                        for (int ab = (int)(tours[tourNumberorigin].size() - 1); ab > -1 ; ab--)
                        {
                            std::vector<int>::iterator it6;
```

```
                        it6 = tours[tourNumberdestination].begin();
                        tours[tourNumberdestination].insert(it6,1,tours[tourNumberorigin][ab]);
                    }
                    tourDemand[tourNumberdestination] += tourDemand[tourNumberorigin];          //
sum up tour demand in new tours
                    tourDemand.erase (tourDemand.begin() +
tourNumberorigin);                  // delete old tour and associated demand
                    tours.erase (tours.begin() + tourNumberorigin);
                }
                 if ((tours[tourNumberdestination][tours[tourNumberdestination].size() -1] ==
(*it).second.D) && (tours[tourNumberorigin][0] == (*it).second.O))    //origin at begin, destination
at end
                {
                    for (unsigned int ac = 0; ac < tours[tourNumberorigin].size(); ac++)
                    {
                        tours[tourNumberdestination].push_back(tours[tourNumberorigin][ac]);
                    }
                    tourDemand[tourNumberdestination] += tourDemand[tourNumberorigin];          //
sum up tour demand in new tours
                    tourDemand.erase (tourDemand.begin() + tourNumberorigin);                    //
delete old tour and associated demand
                    tours.erase (tours.begin() + tourNumberorigin);
                }
                 if ((tours[tourNumberdestination][tours[tourNumberdestination].size() -1] ==
(*it).second.D) && (tours[tourNumberorigin][tours[tourNumberorigin].size() -1] ==
(*it).second.O))    //origin and destination at end
                {
                    for (int ad = (int)(tours[tourNumberorigin].size() - 1); ad > -1 ; ad--)
                    {
                        tours[tourNumberdestination].push_back(tours[tourNumberorigin][ad]);
                    }
                    tourDemand[tourNumberdestination] += tourDemand[tourNumberorigin];          //
sum up tour demand in new tours
                    tourDemand.erase (tourDemand.begin() +
tourNumberorigin);                  // delete old tour and associated demand
                    tours.erase (tours.begin() + tourNumberorigin);
                }
                //tourdemand has to be reduced
            }
            else
            continue;
        }
    }

    // the dcs that require their own tour have to be included too! check for dcs that are not
included in the existing tours at the end and create these tours then
}
     // check for dcs that are not included in tours and add cost for visit
     for (int ba = 0; ba < P_COLUMNS; ba++)
     {
         tourNumbermissing = -1;
         if (solutionS[z].openings[ba] == 1)            //just if the dc is opened
         {
            for (unsigned int bb = 0; bb < tours.size(); bb++)
            {
                for (unsigned int bc = 0; bc < tours[bb].size(); bc++)
                {
                    if (tours[bb][bc] == ba)
                    {
                        tourNumbermissing = 1;
                    }
                }
            }
            if (tourNumbermissing == -1)
            {
                cost += (2 * DIST_COST * vdistmatrix[0][ba]);       //assuming symmetric distmatrix
            }
        }
     }

      // cost calculation for the tours
    for (unsigned int r = 0; r < tours.size(); r++)
    {
        cumTourmeters = 0;
```

```
            cumTourmeters += vdistmatrix[0][tours[r][0]] + vdistmatrix[tours[r][tours[r].size() -1]]
[0];          //includes meters from and to the depot, not possible for multiple depots!
            if (tours[r].size() >= 2)
            {
                 for (unsigned int m = 0; m < tours[r].size() -1; m+
+)                                  //includes meters on the rest of the tour (if there is more than
one dc in tour)
                 {
                    cumTourmeters += vdistmatrix[tours[r][m]][tours[r][m+1]];
                 }
            }
            cost += DIST_COST * cumTourmeters;
        }

    savingsList.clear();    // empties the savingslist after the end of the iteration
    solutionS[z].cost += cost;                          // += very important since it adds to FTL costs
}
return;
}

void dotheSorting (solution solutionS[SIZE_2N], vector< vector<int> >& paretofronts)
{
    vector<int> neu2;               //used to store elements of a front
    for (unsigned int jh = 0; jh < paretofronts.size(); jh++)
    {
        paretofronts[jh].clear();           // paretofronts have to be empty since they are filled
from solutionS
    }
    paretofronts.clear();
    for (int z = 0; z < SIZE_2N; z++)
    {
        solutionS[z].setDominated.clear();
        solutionS[z].dominationCounter = 0;
        for (int y = 0; y < SIZE_2N; y++)
        {
            if ((solutionS[z].cost < solutionS[y].cost) && (solutionS[z].unserved < solutionS
[y].unserved))
            {
                solutionS[z].setDominated.push_back(y);
            }
            if ((solutionS[z].cost > solutionS[y].cost) && (solutionS[z].unserved > solutionS
[y].unserved))
            {
                solutionS[z].dominationCounter++;
            }
        }
        if (solutionS[z].dominationCounter == 0)
        {
            solutionS[z].frontCounter = 0;
            neu2.push_back(z);
        }
    }
    paretofronts.push_back(neu2);
    int i = 0;                          //initilize front counter
    while (paretofronts[i].size() > 0)
    {
        neu2.clear();                   //used to store next paretofront
        for (unsigned int j = 0; j < paretofronts[i].size(); j++)
        {
            for (unsigned int k = 0; k < solutionS[paretofronts[i][j]].setDominated.size(); k++)
            {
                solutionS[solutionS[paretofronts[i][j]].setDominated[k]].dominationCounter--;
                if ( solutionS[solutionS[paretofronts[i][j]].setDominated[k]].dominationCounter == 0)
                {
                    solutionS[solutionS[paretofronts[i][j]].setDominated[k]].frontCounter = i+1;
                    neu2.push_back(solutionS[paretofronts[i][j]].setDominated[k]);
                }
            }
        }
        i++;
        paretofronts.push_back(neu2);
    }
    paretofronts.pop_back();                                // while loop creates an empty last
front... pop it... but done by line if neu2.size() != 0
```

```
    return;
}

void archiveSort (vector<solution> &archive)
{
    int stopcriterium = 1;
    while (stopcriterium > 0)
    {
        stopcriterium = 0;
        for (unsigned int et = 0; et < archive.size() ; et++)                       //eliminate
archive solutions with same objective values
        {
             if (stopcriterium > 0)                                                 //needed to exit 2
loops to start again since there were double solutions-
                break;
            for (unsigned int rt = 0; rt < archive.size() ; rt++)
            {
                if ((archive[et].cost == archive[rt].cost) && (archive[et].unserved == archive
[rt].unserved) && et != rt)
                {
                    archive.erase(archive.begin() + rt);
                    stopcriterium++;
                    break;
                }
            }
        }
    }
    for (unsigned int zt = 0; zt < archive.size() ; zt++)
    {
        archive[zt].dominationCounter = 0;
        for (unsigned int yt = 0; yt < archive.size() ; yt++)
        {
            if ((archive[zt].cost > archive[yt].cost) && (archive[zt].unserved > archive[yt].unserved))
            {
                archive[zt].dominationCounter++;
            }
        }
    }
    int kt = 1;
    while (kt > 0)
    {
        kt = 0;
        for (unsigned int xt = 0; xt < archive.size(); xt++)
        {
            if (archive[xt].dominationCounter != 0)
            {
                archive.erase(archive.begin() + xt);
                kt++;
                break;
            }
        }
    }
}

bool cost_sorter(const crowdingStruct &a, const crowdingStruct &b)          //comparison function for
crowding dist assignment
{
    return a.cost < b.cost;
}

void crowdDistAssignment (int rest, int counta, solution solutionS[SIZE_2N], vector<solution>
newParents, vector< vector<int> > paretofronts, vector<int>& additionalSolutions)
{       //in here, the crowding dist assignment is done and the remaining space of the parent
generation is filled
    multimap<float, int, greater<float> > crowdingranklist;         //from this, the new parents are
selected
    multimap<float, int>::iterator it;
    vector<crowdingStruct> crowdingList;
    crowdingStruct temp;
    float temp2;
    float costmin, costmax, unservedmin, unservedmax;
    for (unsigned int ds = 0; ds < paretofronts[counta].size() ; ds++)
    {
        temp.cost = solutionS[paretofronts[counta][ds]].cost;
```

```
        temp.unserved = solutionS[paretofronts[counta][ds]].unserved;
        temp.ID = paretofronts[counta][ds];
        crowdingList.push_back(temp);
    }
    std::sort(crowdingList.begin(), crowdingList.end(), cost_sorter);           //cost_sorte
    crowdingranklist.insert (std::pair<float, int>(FLT_MAX, crowdingList[0].ID));       //add the
border values with max float values as crowding dist
    crowdingranklist.insert (std::pair<float, int>(FLT_MAX, crowdingList[paretofronts[counta].size
()-1].ID));
    costmin = crowdingList[0].cost;
    costmax = crowdingList[paretofronts[counta].size()-1].cost;
    unservedmax = crowdingList[0].unserved;
    unservedmin = crowdingList[paretofronts[counta].size()-1].unserved;
    for (unsigned int i = 1; i < crowdingList.size()-2; i++)
    {
        temp2 = 0;
        temp2 = (crowdingList[i+1].cost - crowdingList[i-1].cost) / (costmax - costmin);
        temp2 += (crowdingList[i-1].unserved - crowdingList[i+1].unserved) / (unservedmax -
unservedmin); // since second objective is sorted descending if first is ascending, reverse order!!
        crowdingranklist.insert (std::pair<float, int>(temp2, crowdingList[i].ID));
    }
    it = crowdingranklist.begin();
    for (int z = 0; z < rest; z++)
    {
        additionalSolutions.push_back((*it).second);
        it++;
    }
    return;
}

void createParent (vector <solution> &newParents, solution solutionS[SIZE_2N], vector< vector<int> >
paretofronts)
{
    int rest = 0;
    unsigned int counta = 0;            // counts front
    unsigned int countb = 0;                        // counts sum of solutions in parent
    while ((countb + paretofronts[counta].size()) <= N)
    {
        for (unsigned int countc = 0; countc < paretofronts[counta].size(); countc++)
        {
            newParents.push_back(solutionS[paretofronts[counta][countc]]);
            newParents[newParents.size() -1].cost = 0;
            newParents[newParents.size() -1].unserved = 0;
        }
        countb += paretofronts[counta].size();
        counta++;
    }
    if (N - countb > 0)         //there is the possibility, that only full fronts are sufficient to
fill parent up
    {
        rest = N - countb;
        vector<int> additionalSolutions;
        crowdDistAssignment(rest, counta, solutionS, newParents, paretofronts, additionalSolutions);
        for (int gh = 0; gh < rest; gh++)
        {
            newParents.push_back(solutionS[additionalSolutions[gh]]);
            newParents[newParents.size() -1].cost = 0;
            newParents[newParents.size() -1].unserved = 0;
        }
    }
}

void createOffspring (vector <solution> newParents, vector <solution> &newOffspring)
{
    vector <int> rouletteChances;
    int sumChances = 0;
    int randomChoice = 0;
    int random = -1;
    int count = 0;
    int crossovercount = 0;
    int where = -1;
    solution offspring;
    srand(time(NULL));
    for (int k = 0; k < N; k++)
```

```
    {
        rouletteChances.push_back(newParents[N-1].frontCounter - newParents[k].frontCounter + 1); //
each front gets chances accoring to spread of sets, so if 3 sets, then best sets solutions get value 3
and worst get 1.
        sumChances += rouletteChances[k];                                                   //benefit:
it takes into account the number of fronts in the set. best have n times the chnce of getting chosen
in a situation with n fronts
    }
    for (int z = 0; z < N; z++)         //for each place in the offspring generation
    {
        randomChoice = (rand() % sumChances)+1;
        count = 0;
        while (randomChoice > 0) // the count, at which random choice gets 0 is the selected roulette
wheel selection!!!
        {
            randomChoice -= rouletteChances[count];
            count++;
        }
        if (count == N)
        count = N - 1;
        offspring = newParents[count];
        random = rand();
        where = rand() % P_COLUMNS;             // decides where the crossover / mutation starts
        if (random <= RAND_MAX * CROSSOVERPROB)     //Here, a crossover is initiated
        {
            crossovercount = 0;
            randomChoice = (rand() % sumChances)+1;     //define the crossvoerpartner
            while (randomChoice > 0) // the count, at which random choice gets 0 is the selected
roulette wheel selection!!!
            {
                randomChoice -= rouletteChances[crossovercount];
                crossovercount++;
            }
            if (crossovercount == N)            //corrected as above for which parent to take...
            crossovercount = N - 1;
            for (;where < P_COLUMNS; where++)
            {
                offspring.openings[where] = newParents[crossovercount].openings[where];
            }
        }
        if (random > RAND_MAX * CROSSOVERPROB)      //here, a mutation is initiated
        {
             offspring.openings[where] = !offspring.openings[where];        //flips the where value...
        }
        newOffspring.push_back(offspring);
    }

}

void dataOutput (vector<solution> archive, double time)
{

    ofstream output("output.csv");
    for (unsigned int rt = 0; rt < archive.size() ; rt++)
    {
        for (int rz = 0; rz < V_COLUMNS; rz++)
        {
            output << archive[rt].openings[rz] <<",";
        }
        output << "," << archive[rt].cost << "," << archive[rt].unserved << endl;
    }
    output << "time," << time<<endl;
    output.close();
}

// ###################################### main function
#########################################################################################################
int main()
{
    //########################## variable definition part
#####################################################################
    clock_t start = clock();
    float vdistmatrix[V_COLUMNS][V_ROWS];
    float pdistmatrix[P_COLUMNS][P_ROWS];
```

```
    int population [P_ROWS];                    //used for dataInput
    int totalpopulation = 0;                    //used for lambdacalculation
    float capacities[P_COLUMNS];                //used for capacity calculation
    float averagecapacities = 0;                // used for lambdacalculation
    float attractivityMatrix[P_COLUMNS][P_ROWS];              //used for attractivity and unserved
calculation
    float dummyAttractivity = 0;                    //includes distance decay
    float lambda = 0;
    float dummyA = 0;                               //just dummy Attractivity
    solution solutionS[SIZE_2N];                //used for set of solutionS, array of struct type
solution
    float CAP_RATIO[P_COLUMNS];                         // used only if CAP_RATIO endogenous
    vector<vector<int> > paretofronts;           //used in nondominated sorting
    vector<solution> newParents;                    //used for new parent generation
    vector<solution> newOffspring;                  //used for new offspring generation
    int combinecount = 0;

    //########################### distance matrix & population input begin
#####################################################
    totalpopulation = dataInput(vdistmatrix, pdistmatrix, population);
    //########################### distance matrix & population input end
#####################################################



    //########################### capacity calculation part
#####################################################################
    calcCAP_RATIO (pdistmatrix, CAP_RATIO, totalpopulation,
vdistmatrix);                               // since capratio is calculated internaly (1/n, where n is
average number of possible DCs nearby for each dc) this is necessary, otherwise
    averagecapacities = calcCapacities(capacities, population, pdistmatrix, CAP_RATIO);
    //########################### capacity calculation part end
#####################################################################



    //########################### create initial solution and archive, attracitvity measure
#####################################
    // use a struct of an array size P_COLUMNS type binary, plus float cost and float unserved, fill
binary with 0 and 1s by chance,
    // use INITIAL_RATIO as factor for having a 1 in binary
    createInitialSolutions(solutionS);
    vector<solution> archive;
    lambda = calcLambda(dummyA, averagecapacities);  // used for unserved calc part
    dummyAttractivity = calcAttractivity(capacities, pdistmatrix, attractivityMatrix, dummyA, lambda);
    //############################create initial solution and archive and attractivity measure end
#################################



    //############################begin of main part
#########################################################################################################
    //########################### cost and unserved calculation part begin
#####################################################
    for (int mastercount = 0; mastercount < ITERATIONS; mastercount++) //this is the main loop
    {
        for (int j = 0; j < SIZE_2N; j++)           // cost and unserved are initialized with  value
of 0, necessary, since split deliveries cost have to be stored before actual cost calc
        {
            solutionS[j].cost = 0;
            solutionS[j].unserved = 0;
        }
        calcUnserved(capacities, population, solutionS, attractivityMatrix, dummyAttractivity);
        // here be structure for split deliveries just in case of exceeding TRUCK_CAP: COST of this
solution is increased by the tour-retour cost to the cost of the specific solution
        for (int q = 0; q < SIZE_2N; q++)
        {
            int count = 1;
            while (count > 0)           // this loop runs until no more truck capacity excesses happen
            {
                count = 0;
                for(int b = 0; b < P_COLUMNS; b++)
                    {
                        if (solutionS[q].demand[b] > TRUCK_CAP)
```

```
                        {
                            count++;
                            solutionS[q].cost += ((2 * vdistmatrix[0][b]) * DIST_COST);
                         // cost increased for split delivery (one FTL)
                            solutionS[q].demand[b] -=
TRUCK_CAP;                                                  // remaining demand at dc after FTL is
reduced by one FTL
                        }
                    }
                }
        }
        calcCost(capacities, population, vdistmatrix, solutionS, totalpopulation);
        //############################ cost and unserved calculation part end
#########################################################


        //############################ non dominated sorting part begin
#################################################################
        dotheSorting(solutionS, paretofronts);
        //############################ non dominated sorting part end
#####################################################################


        //############################ comparison with archive and replacement if necessary part begin
################################
        for (unsigned int abd = 0; abd < paretofronts[0].size() ; abd++)
        {
        archive.push_back(solutionS[paretofronts[0][abd]]);
        }
        archiveSort(archive);
        //############################ create parent generation start
#####################################################################
        createParent(newParents, solutionS, paretofronts);
        //############################ create parent generation end
#######################################################################

        //############################ create offspring generation start
#####################################################################
        createOffspring(newParents, newOffspring);
        //############################ create offspring generation end
#######################################################################

        //############################ combine offspring and parent
start#######################################################################
        combinecount = 0;
        for (int jk = 0; jk < N; jk++)
        {
            solutionS[combinecount] = newParents[jk];
            combinecount++;
            solutionS[combinecount] = newOffspring[jk];
            combinecount++;
        }
        newParents.clear();
        newOffspring.clear();
    }

    //############################ combine offspring and parent end
#####################################################################

//############################  end of main part
#############################################################################################
clock_t clockcount = clock() -start;
double time = clockcount / (double)CLOCKS_PER_SEC;
//############################ data output
#############################################################################################
    dataOutput (archive, time);
    //############################ data output part end
#########################################################################
getchar();
return EXIT_SUCCESS;
}
```

Appendix B

Abstract

This thesis achieves three things. First, it develops an extension of the well-known Covering Tour Problem, a problem in which the decision to visit a node in a transport network results in the neighboring nodes to be considered as covered. This approach is used in a setting of Humanitarian Logistics, where people from neighboring villages are covered by distribution centers for relief goods. Here, however, the coverage is calculated using a Competitive Location Model, in which the attractiveness of a distribution center results in certain ratios of the neighboring villages to be considered as covered (i.e. a certain ratio of the inhabitants will move to the distribution center). At the same time, the neighboring villages' inhabitants can choose not to go to a distribution center and stay at home, which is modeled using a dummy facility.

Second, this approach is applied to a realistic case study for Gaza province in southern Mozambique. The setting is a drought disaster and the task is to define a Pareto optimal set of solutions according to the two objectives cost and unserved demand for relief goods. A genetic algorithm is used for the heuristic solution for two instances, implemented in C++. The application is analyzed from the view of a decision maker in Humanitarian Logistics and the necessary steps, information and decision tasks are explained and limitations are pointed out.

Finally, several definitions of (distributive) justice are explained and compared, and the approach presented is analyzed according to the different schools of thought, ranging from Utilitarianism to Libertarianism and Rawls Liberal Egalitarianism.

Appendix C

Deutsche Zusammenfassung

Diese Arbeit beschäftigt sich mit drei Themen. Zuerst wird eine Erweiterung zum Covering Tour Problem entwickelt. Dieses ist ein Routenfindungsproblem, in dem die Entscheidung einen Knoten des Transportnetzes zu besuchen, dazu führt, dass die umliegenden Knoten abgedeckt sind. Damit wird angenommen, dass die Bewohner von benachbarten Ortschaften von Verteilungszentren zu diesen gehen können um ihren Bedarf an Hilfsgütern zu decken. Dieses Problem wird um ein Competitive Location Model erweitert, welches nicht zu den Verteilungszentren zuteilt, sondern die Bewohner anhand eines Attraktivitätsmaßes eine Entscheidung zu welchen Verteilungszentren sie gehen wollen treffen lässt. Die Möglichkeit, gar nirgends hinzugehen wird durch einen Dummy inkludiert.

Für eine hypothetische Dürrekatastrophe in der Provinz Gaza in Mozambique wird die Aufgabe, ein Set an Pareto optimalen Lösungen für die beiden Ziele Kosten und unbefriedigte Nachfrage zu finden, angenommen. Ein genetischer Algorithmus wird als heuristische Lösungsmethode für zwei Instanzen in C++ benutzt. Diese Anwendung wird aus der Sicht eines Entscheidungsträgers in humanitärer Logistik analysiert und die notwendigen Schritte, Informationen und Entscheidungen erklärt sowie Probleme aufgezeigt.

Schlussendlich werden mehrere Definitionen von (Verteilungs-)Gerechtigkeit beschrieben und der entwickelte Lösungsansatz wird daraufhin untersucht, wie er sich auf die Gerechtigkeit in der Verteilung von Katastrophenhilfsgütern auswirkt. Dabei kommen unter anderem utilitaristische aber auch libertäre Ansätze sowie John Rawls „Schleier des Nichtwissens“ zum Einsatz.

Appendix D

Curriculum Vitae

Christian Burkart

Curriculum Vitae

Ausbildung

2007–Heute **Diplomstudium Internationale Entwicklung (IE)**, *Universität Wien*.
Geplantes Studienende: März 2014

2011–2014 **Masterstudium in Supply Chain Management (SCM)**, *WU Wien*.

2007–2011 **Bachelorstudium Wirtschafts- und Sozialwissenschaften**, *WU Wien*.
Spezialisierung in Volkswirtschaftslehre und Sozioökonomie (Volkswirtschaftszweig)

2006 **Matura**, *Höhere Internatsschule des Bundes (HIB) Graz-Liebenau*, Bundesgymnasium mit vier obligatorischen Fremdsprachen.
Matura und jede Schulstufe mit ausgezeichnetem Erfolg bestanden

Master Thesis

Titel *Disruption Management in Transport Management Systems*

Betreuer Univ. Prof. Dipl.-Ing. Dr. Werner Jammernegg & Dr. Pamela Nolz

Beschreibung Diese Arbeit untersucht wie elektronische Transport Management Systeme mit regelmäßigen und unregelmäßigen Transportunterbrechungen umgehen und legt dabei den Schwerpunkt auf Robustheit und Resilienz in der Routenfindung.

Diplomarbeit

Titel *Competitive Location Models for Transportation in Disaster Relief Logistics*

Betreuer Univ. Prof. Dr. Walter Gutjahr & Dr. Pamela Nolz

Beschreibung Ein Covering Tour Problem wird durch den NSGA II Algorithmus in C++ gelöst. Basis dafür ist ein Competitive Location Model, das die Nachfrage nach Hilfspaketen in einem Dürrekatastrophenfall in den potenziellen Verteilungszentren modeliert.

Bachelorarbeit

Titel *Konzepte der Einkommensverteilung und ihre methodische Anwendung am Beispiel Österreichs. Konzepte, Indikatoren, Beispielrechungen.*

Betreuer Dr. Sascha Sardadvar

Beschreibung Verschiedene Methoden um die Einkommensverteilung zu berechnen werden verglichen und am Beispiel Österreichs mit dem EU-SILC Datensatz in R berechnet.

St. Ulrichsplatz 5/2/6 – Wien, 1070
(0043) 664 6569873 • christian@burkart.at

Arbeitserfahrungen

01/2014– Heute **Universitätsassistent prae doc**, *WU Wien*.
Institut für Transportwirtschaft und Logistik

03/2013– 01/2014 **Wissenschaftlicher Mitarbeiter**, *WU Wien*.
Institut für Produktionsmanagement

2012–2013 **Betreuungstutor**, *WU Wien*.

08/2011 **Praktikum**, *Österreichischer Integrationsfonds*, Wien.

2011-2012 **Ferialjobs**, *Rotes Kreuz Graz-Stadt*, Rettungssanitäter.

2005–2010 **Mehrere Ferialjobs**, *KBO Ostermann GmbH*, Hart b. Graz.

Weiteres

2006–Heute **Ehrenamtlicher Rettungssanitäter**, *Rotes Kreuz Graz-Stadt*.

2006–2007 **Zivildienst**, *Rotes Kreuz Graz-Stadt*.

Auszeichnungen

2011/2012 Leistungsstipendium für das Masterstudium Supply Chain Management

2009/2010 Leistungsstipendium für das Bachelorstudium der Wirtschafts- und Sozialwiss.

Softwarekenntnisse

Office	LaTeX, Microsoft Office, LibreOffice, GeoGebra
Statistik	MiniTab, R, SPSS
Ökonometrie	EViews
GIS	ESRI ArcGIS, GeoDa, QGIS
OR Software	Rapid Modeler, @Risk, Excel Solver
Optimierung	C++

Sprachen

Deutsch	Muttersprache	
Englisch	Cambridge Certificate in Advanced English	*(entspricht Europäische Kompetenzstufe C1)*
Französisch	DELF 1er Degré	*(entspricht Europäische Kompetenzstufe B1)*
Spanisch	Basiskenntnisse	

Interessen

- Rotes Kreuz
- Kochen
- Volleyball
- Lateinamerikanische Tänze

St. Ulrichsplatz 5/2/6 – Wien, 1070
(0043) 664 6569873 • christian@burkart.at