MASTERARBEIT

Titel der Masterarbeit

OPTIMIZATION OF WAREHOUSE LOCATIONS BASED ON WARDROP EQUILIBRIA

Verfasserin

Nađa Džubur

angestrebter akademischer Grad

Master of Science (MSc)

Wien, 2013

Studienkennzahl lt. Studienblatt: A 066 920

Studienrichtung lt. Studienblatt: Masterstudium
Quantitative Economics, Management, and Finance

Betreuer : Univ.-Prof. Dr. Walter Gutjahr

Eidesstattliche Erklärung

Ich erkläre hiermit an Eides Statt, daß ich die vorliegende Arbeit selbtständig und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe. Die aus fremden Quellen direkt oder indirekt übernommenen Gedanken sind als solche kenntlich gemacht.

Die Arbeit wurde bisher in gleicher oder ähnlicher Form keiner anderen Prüfungsbehörde vorgelegt und auch noch nicht veröffentlicht.

Wien, am 2. Dezember 2013

Nađa Džubur

ABSTRACT

Ein Warehouse Location Problem wird mit dem Konzept von Wardrop Gleichgewichten optimiert, die durch den Algorithmus von Frank-Wolfe für konvexe Optimierung bestimmt werden. Die Anwendung ist ein Katastrophenhilfe-Modell, wobei es sich bei den Standorten, die optimiert werden sollen, um Verteilungszentren handelt, die Hilfsgüter für eine bestimmte Gegend anbieten. Die Verteilung der Menschen zwischen den DCs (Verteilungszentren) erreicht seine optimale Lösung im Wardrop Gleichgewicht, wobei die Zielfunktion eine gewichtete Summe der Reisekosten und des nicht abdeckbaren Bedarfs (aufgrund der beschränkten Kapazitäten der DCs) bildet. Um die Pareto-optimalen Lösungen für die beiden Zielfunktionen, die für die Hilfsorganisationen relevant sind, DC-Kosten und Nachfragedeckung, zu finden, wird vollständige Enumeration verwendet. Dies ruft alle möglichen Kombinationen der Standorte auf. Das Modell wird mit einem Datensatz ländlicher Gemeinden in Senegal getestet.

Schlagwörter

Katastrophenhilfe, Facility Location Problem, Wardrop Gleichgewicht, Minimum Cost Multi-Commodity Flow Problem, Algorithmus von Frank-Wolfe

ABSTRACT

A Warehouse Location Problem is optimized using the concept of Wardrop Equilibria which are determined by the Frank-Wolfe Algorithm for convex optimization. The application is a disaster relief model where the facilities to be optimized, are the distribution centers, offering relief supplies for a certain area. The distribution of people among the DCs (Distribution centers) finds its optimum in the Wardrop Equilibrium where the objective function is the weighted sum of the traveling cost and non-coverable demand due to the restricted capacity of the DCs. To find the Pareto optimal solutions for the two objectives relevant for the relief organizations, DC cost and demand coverage, complete enumeration is used, calling up all variants of facility locations. The model is tested using a dataset of rural communities in Senegal.

Keywords

disaster relief, facility location problem, Wardrop Equilibrium, minimum cost multi-commodity flow problem, Frank-Wolfe Algorithm

Acknowledgments

I would like to express my sincere gratitude to my supervisor Prof. Dr. Walter Gutjahr for his patience and kind help over the whole period of work.
I would also like to thank my parents for supporting me and giving me encouragement when I needed it most.
Furthermore, I wish to thank my friends for being there for me in stressful times.

Contents

1 Introduction

When it comes to emergency relief during or after disasters such as natural catastrophes, for humanitarian organizations the question arises how to set up emergency facilities with relief supplies in the population areas, where the people are located, in the most efficient way. Rather than delivering all goods directly to the people, which is not feasible in most cases, goods are shipped to facilities which should lie as close as possible to their housings so that the maximal number of people is able to reach those facilities and is provided there with the needed supplies [19]. Furthermore, since humanitarian organizations are mostly provided with a limited budget, the overall cost containing establishment and maintenance of such DCs (=Distribution centers), both depending on the size and respectively, the capacity, should be held as small as possible. The organizations' decision on the location set-ups depends on the modelling of how the people will distribute over the DCs. The decision of the people to leave their villages in order to visit a DC depends on the distance they would need to pass as well as on their chances to be provided with relief supplies. As most of the people in those villages can't travel by anything else than by feet, their willingness to traverse many kilometres will fall monotonously with rising distance. Besides, if too many people go the same DC, the capacity will be exceeded and the demand of the people will not be covered in full but they will only get a specific percentage of their demand dependent on the amount of people traveling to this facility. In this case, the capacity is split up equally between all incoming people. This can also be interpreted in a probabilistic way by assuming that every person gets relief goods only with a certain percentage. Rising the capacities higher than a given threshold is no solution at all because such centers are mostly set up provisionally and therefore, endangered to be plundered at night. Cost rise when there is a need for people taking care of the centers overnight. Before the establishment of those DCs, it's not possible to know how people will spread over the area and therefore also, how to distribute the supplies from the supplier's side. It is assumed that the relief goods are homogeneous and everyone has the same need of the same amount of goods. The whole system is considered as a flow network where the villages represent the population nodes and certain ones out of those are set also to DC nodes.

The arcs between population and DC nodes make for the road sections and provide the distances. The demand for relief goods is the number of people within a population node. Feasible routes are those between the villages and the DCs with no limits on their capacities. The problem consists of two stages. First, within the set of all DC set-up possibilities, one combination is chosen in each iteration. After the facility location is set, one can calculate the distribution of the people leaving their villages. The Wardrop Equilibrium is the state of the people's distribution where the selfish optimum is achieved for every participant by selecting a route which minimizes a weighted sum of traveling cost and uncovered demand. This is a steady state which levels out after a short transition phase where people adapt their behaviour. This flow equilibrium is calculated using the Frank-Wolfe Algorithm which is an iterative algorithm for constrained convex optimization using first order derivatives for linear approximation. After modelling the people's distribution, it's on the humanitarian organizations to decide which location possibility to choose. This problem is a bi-objective one. In each iteration step for the current DC establishment combination, the cost for the DCs form the first objective and total uncovered demand the second. The decision whether to choose higher cost or to cover more demand lies in the hand of the humanitarian organizations. Out of all possible DC combinations, the set of Pareto optimal solutions is picked out such that the organizations can choose according to their capabilities.

2 Related Literature

This work is referring to an already existing disaster relief model by Tricoire et al. (2011) [19]. The basic model used in this paper assumes that the people in the village walk to the closest DC which is opened and are provided there with the available capacity of relief supplies which are split up equally between all the demand flow entering the DC. The model is denoted as the Nearest center model (NC). It is assumed that all people within a village will walk to the closest DC if the distance is less or equal to 6km and only 50% of the people will leave if the distance is more than 6km but not more than 15km. Above the distance-threshold of 15km, nobody will leave his home. The problem was treated as a linear 2-stage problem with recourse, considering fixed set-ups under stochastic demand. Monte

Carlo simulation was used to find the uncertain provided fraction of uncovered demand under fixed cost.
An overview of stochastic and deterministic approaches on the facility location problem are presented by Owen and Daskin (1998) [15]. A focus on one of the most popular models among facility location, the covering problem, is given in Farahani et al. (2010) [8]. A model that maximizes the amount of demand covered within the acceptable distance by locating a given fixed number of new facilities rather than covering all demand by minimizing the number of locations is the maximum coverage problem representing a better model when it comes to disaster relief and limits on expenditure. New variations of the maximum coverage facility location problem are given by Bhattacharya and Nandy (2011) [4]. As the behaviour of people facing emergency situations is more complex and rather selfish, the model should adapt to this. Introduced by Wardrop in 1952 [20], Beckmann et al. were the first to mathematically formalize Wardrop's 1st principle in 1956 [3]. The Wardrop Equilibria network model used in this thesis is based on the recent work done by Correa and Stier-Moses (2010) [6]. A game theoretical approach to the self-interested behaviour of Wardrop Equilibria is given by Sangwan (2007) [17]. The relation to the Nash Equilibrium is described by Altman et al. (2005) [1]. Statements to improve the convergence of Wardrop Equilibria with the help of sampling methods are done by Fischer et al. (2006) [9].

3 Wardrop Equilibria

From studies of behavioural assumptions of transportation and telecommunication networks, one knows that travellers or packets choose instinctively routes which they perceive as being the shortest under given traffic conditions. This results in a situation where no traveller can reduce his journey time by choosing an other route since he's already chosen the most effective. Such a situation within a traffic pattern is therefore a static one, known as a User- or Wardrop Equilibrium. The Wardrop Equilibrium is a steady state coming after an unsteady phase where travellers adjust their route choice until no improvement is possible and a situation with stable route cost and flows is reached [6].
There are two principles stated by Wardrop formalizing the notion of an equilib-

rium and introducing the alternative behaviour of the minimization of the total travel cost (cf.[20]):

1. *The journey times in all routes actually used are equal and less than those which would be experienced by a single vehicle on any unused route.*

2. *At equilibrium the average journey time is minimum.*

Ad 1: Each user tries to non-cooperatively minimize his cost and traffic flows. Flows referring to such pattern are known as "User Equilibria" flows, since each user chooses that route that is best for him. An user-optimized equilibrium is reached when no user may lower his traveling time (resp. cost) through unilateral action [6].
Ad 2: Usually, the socially-optimal pattern of route choice differs from the pattern that occurs when each individual traveller acts according to his or her own self-interest. Therefore, Wardrop articulated another principle. The 2nd principle implies cooperation of the users. Everyone aims to minimize the total travel time (resp. cost) for the whole system. Traffic flows which satisfy Wardrops 2nd principle are called "system optimal", such an equilibrium is known as System Equilibrium (SE) [6].
By talking about Wardrop Equilibria (WE), one refers to Wardrop's 1st principle. One can say, a state is a Wardrop Equilibrium, if all routes being used have the same cost for each member whereas the cost that would occur on all paths which are not used by the members are higher or equal.

3.1 The Basic Model

The model was taken from Correa and Stier-Moses (cf.[6]). A traffic assignment problem is considered with origin- destination (OD) pairs and their demand on a given network with a road geometry. G=(N,A) represents a directed network where N is the set of all population nodes given and A the set of arcs connecting them. The set of commodities $C \subseteq N \times N$ is represented by OD pairs. For each OD pair k, an amount d_k of flow demand (which represents the amount of people) has to be routed from the origin to the destination. The individual routing decision of one person has no impact on the routing decision of the others within

the same population node, hence, the demand is arbitrarily divisible. Let R_k be the set of feasible routes connecting each OD pair k in G, and let R be the union of all connections, R=$\bigcup_k R_k$. For every arc a, the non-negative vector f=$(f_a)_{a\in A}$ describes the amount of traffic passing through it. It is known as the link flow. Besides, let $t_a : R_{\geq 0} \cup \{\infty\} \mapsto R_{\geq 0} \cup \{\infty\}$ be the non-negative, non-decreasing and continuous travel cost function which maps each link flow f_a on arc a to the cost (e.g. cost in forms of travel time or effort) needed to traverse this arc. One denotes a non-negative vector h=$(h_r)_{r\in R}$ as route flow, if this vector meets the demand for each OD pair, $\sum_{r\in R_k} h_r = d_k$. This is the range of possible routes for each OD pair within the graph. Every route flow determines the quantity of traffic for each arc a of the set of arcs, which is the link flow, $f_a = \sum_{r|a\in r} h_r$, $\forall a \in A$. For a link flow f, travel cost along a route r is given by $c_r(f) := \sum_{a\in r} t_a(f_a)$. The set of feasible flows (f,h) is denoted as X and its projection on the set of link flows is denoted as X_f. Due to the flow conservation constraint holding for each OD pair, saying that, unless it's not the origin or destination node, the inflow within every node has to equal the outflow of this node, it's easy to see that X_f is a polytope. Since Wardrop's first principle states that all flows travel along shortest paths, a route flow h is called Wardrop Equilibrium if and if only for all OD pairs $k \in C$,

$$c_r(h) = \min_{q\in R_k} c_q(h),$$

holds $\forall r \in R_k$, such that $h_r > 0$. The route flow h is that one giving the minimal cost within the set of route flows of an OD pair. Note the convention that $c_r(h) := c_r(f(h))$ where f(h) is the link flow determined through route r. One considers the minimal cost multi-commodity flow problem with a separable objective function

$$\min_f \{\sum_{a\in A} \int_0^{f_a} t_a(z)dz \mid f \in X_f\}.$$

According to the proof of Beckmann et al. [3] such minimum always exists. Since the objective function of the problem is the integral of a non-decreasing function, the objective is convex, and since X_f is a convex set, the problem is a convex one. Furthermore, the domain of the problem is a bounded polyhedron and therefore a compact set (according to Heine-Borel, since X_f is a closed and bounded subset of

R^n). Because the objective is a continuous and real function on a compact set and thus, the range of the function is again compact, there always exist an minimum and maximum according to the Weierstrass Theorem [16]. Since the problem is a convex one, as mentioned above, the existence of a global minimum for the given objective is assured. Applying the Karush-Kuhn-Tucker Conditions yields $c_r(h) <= c_q(h)$ for all commodities $k \in C$ and all routes $r, q \in R_k$ such that $h_r > 0$ (cf.[1]). This is equivalent to the definition of a Wardrop Equilibrium. If h and h' are both Wardrop Equilibria, the cost for them are equal, $c_r(h) = c_r(h')$ [3]. This is due to the fact that there exists only a global minimum and no local ones. The route flow as well as the link flow are not necessarily unique. If the cost function t_a is strictly increasing, the link flow f is unique, but still, there can be different decompositions of the route flow h which lead to the same traffic quantities. If the cost function is just non-decreasing, f is not unique but the cost vector $(t_a(f_a))_{a \in A}$ of potentially different equilibria is unique. A computational solution of Wardrop Equilibria is possible using the multi-commodity minimal-cost formulation with separable objective function.

4 The Frank-Wolfe Algorithm

The Frank-Wolfe Algorithm [10] is one of the most widely used algorithms when it comes to routing problems in areas of traffic and telecommunication, thus, models with flow assignment problems on OD-pairs. It's a traditional algorithm for the computation of equilibria and very popular since it's rather simple and the memory requirement is modest [2]. This is due to the fact that it deals at any iteration only with a single path between each OD pair [13]. The algorithm was invented initially for quadratic optimization, but it's also applicable for all kinds of convex problems. For the usage of the algorithm, one considers the convex optimization problem

$$\{\min \omega(f) \mid f \in X \subset R^n_{\geq 0}\}$$

where ω defines a convex and continuously differentiable function and X$\subset$ $R^n_{\geq 0}$ the convex set of feasible points. By the continuity of ω and since X is bounded from below, there exists a solution for this problem, the solution is unique, if ω

is strictly convex. For the computation of Wardrop Equilibria, one can apply the Frank-Wolfe Algorithm to the multi-commodity minimal-cost problem formulation mentioned in the chapter above since the necessary conditions are given (cf. [6],[13]). The algorithm is an iterative descent method, it starts with an initial value $f^0 \in$ X and generates a sequence of feasible points $f^t \in$ X, whereby f^{t+1} is generated out the previous point f^t in order to find a feasible descent direction [2], [12]. The idea behind the algorithm is the replacement of the convex function $\omega(f)$ in each step t through the linear approximation going through the current search point f^t

$$\omega(f) \cong \omega(f^t) + \bigtriangledown\omega(f^t)(f - f^t)$$

where the righthand-side linearization gives -due to convexity- an affine minorant to the lefthand-side objective [2].
Minimizing the objective function with respect to f corresponds to minimizing the linearization with respect to f:

$$\min_{f\in X}\{\omega(f^t) + \bigtriangledown\omega(f^t)(f - f^t)\} = \min_{f\in X}\omega(f^t) + \min_{f\in X}\bigtriangledown\omega(f^t)f - \min_{f\in X}\bigtriangledown\omega(f^t)f^t.$$

Since $\omega(f^t)$ and $\bigtriangledown\omega(f^t)f^t$ don't depend on f, they can be ignored for the minimization:

$$\omega(f^t) + \min_{f\in X}\bigtriangledown\omega(f^t)f - \bigtriangledown\omega(f^t)f^t.$$

One determines the solution of the problem

$$s^t := arg\min_{s\in X}\bigtriangledown\omega(f^t)s$$

where the search point s^t is defined as $s^t := f \in$ X and the search direction in step t is given through $s^t - f^t$.
Due to the usage of the first order derivative, the algorithm is also known as the Conditional Gradient [12]. Far from the optimal solution the algorithm performs very well but it tends to slow and poor convergence around the optimum. In the near surrounding of the optimal solution, it tends to zig-zag behaviour [18]. The reason for this is that the algorithm is more attracted to constraint corners than to

the actual descent direction of the objective function once it's close to the solution in order to avoid infeasibility [13]. Modification of the step-size is used to improve the performance. In the line search approach the objective is minimized along the line segment between the current solution f^t and the search point s^t to find the optimal convex combination of them. Analytically, this means one searches for the weight $\xi \in [0,1]$ such that

$$\xi^t := arg \min_{\xi \in [0,1]} \omega((1-\xi)f^t + \xi s^t)$$

which produces the next search point

$$f^{t+1} := (1-\xi)f^t + \xi s^t$$

(cf. [2]). The objective value is decreasing significantly in each step when using line search. Clarkson [5] showed that also the following variant of the Frank-Wolfe Algorithm ensures certain convergence bounds: Instead of performing line search to find the best ξ, in each iteration t, a fixed step-size is used, $\xi^t = \frac{2}{t+2}$. This stepsize is converging to zero as t goes to infinity, this means that after a certain number of iterations, far more weight will be put on the previous flow rather than on the new search point. This saves the expenses of calculating the function value in each step because only the derivatives need to be calculated. In conclusion, this leads to the following algorithm (Frank-Wolfe 1956 [10]): Let $f^0 \in$X and $\xi \in$ [0,1] be arbitrarily chosen initial values.

- Set t=1.
- Compute $s^t := arg \min_{s \in X} \bigtriangledown \omega(f^t)s$.
- Set $\xi^t := \frac{2}{t+2}$.
- Set $f^{t+1} := (1-\xi)f^t + \xi s^t$.
- If the termination condition is not met yet, set t:=t+1.

(cf. [12]).
The iterates of the previous algorithm satisfy $\omega(f^t) \leq \omega(f^*) + O(\frac{1}{t})$ where f^*

defines an optimal solution of the algorithm [5][12]. The same holds also for the sub-problem solved with the linear search method [12]. This is relatively slow compared to other algorithms. The Frank-Wolfe Algorithm is especially useful when the number of variables is very large, like in the disaster model discussed in this work, as such problems require rather small effort per iteration than a fast convergence rate in terms of iterations [5].

The algorithm stops after a certain precision condition is achieved. Since a convex problem is treated, upper and lower bounds of the problem's objective value can be determined and used as a termination criterion. Convexity of the function ω implies that $\omega(f) \geq \omega(f^t) + \bigtriangledown\omega(f^t)(f - f^t)$ for all $f, f^t \in X$. Assuming that s^t is the optimal search point at iteration step t and $\omega(f^*)$ the optimal solution, one gets

$$\omega(f^t) + \bigtriangledown\omega(f^t)(s^t - f^t) \leq \omega(f^t) + \bigtriangledown\omega(f^t)(f^* - f^t) \leq \omega(f^*) \leq \omega(f^{t+1}),$$

where the first inequality results from the fact that $s^t - f^t$ is the optimal search direction at step t but $f^t - f^*$ is not necessarily optimal as a search direction in step t. The second inequality follows from the convexity of ω and the third from the optimality of $\omega(f^*)$ [12].

Thus, in each iteration t, $LBD_t := \omega(f^t) + \bigtriangledown\omega(f^t)(s^t - f^t)$ denotes a lower bound and $UBD_t := \omega(f^{t+1})$ denotes an upper bound to the optimal solution. While in each iteration step, the optimal value is descending and therefore, UBD_t is decreasing towards $\omega(f^*)$, the sequence LBD_t approaches $\omega(f^*)$ from below. The sequence of lower bounds is not necessarily monotonically growing and also the sequence of upper bounds is not necessarily monotonically falling. Nevertheless, there exists always an interval $[LBD_t, UBD_t]$ which contains the optimal value. Therefore, making $\frac{UBD_t - LBD_t}{LBD_t} > 0$ smaller than a certain value $\epsilon > 0$ can be used as a termination criterion for the algorithm [12].

A smarter version of a termination criterion can be obtained by considering the best found upper (resp. lower) bound so far instead of the upper (resp. lower) bound in each iteration step. This guarantees monotonicity, the gap between the best lower bound so far and the best upper bound so far is iteratively shrinking until they are sufficiently close to each other and, as a result, sufficiently close to

the optimal solution contained in the interval. This means, the value

$$\frac{UBD - LBD}{LBD}$$

with $UBD \leq UBD_t, UBD_{t-1}, ..., UBD_1$ and $LBD \geq LBD_t, LBD_{t-1}, ...LBD_1$, defined as the relative gap at iteration t, is decreasing. As above, the algorithm can be stopped when the relative gap is smaller than a certain threshold $\epsilon > 0$ [14],[2], [12].

In contrast to the termination criterion which uses the current upper and lower bounds, the sequence of relative gaps considering the so far best upper and lower bounds is converging in a monotonically decreasing way. Therefore, one can also approximate after how many steps the algorithm starts converging rather slowly and doesn't make any significant steps any more.

5 The Disaster Relief Model

5.1 Problem Formulation

A complete graph $(V \cup \{t\}, A)$ is considered where V=$\{1, ..., n\}$ denotes the set of villages with a certain given population, t is the depot which is definitely given in advance, and A are the arcs representing the potential roads between the villages as well as between villages and depot. The set of commodities can be defined as $C = \{(k, t) \mid k \in V\}$, thus, every village k represents an origin and the depot is always representing the destination. The demand d_k denotes the number of inhabitants of each village k in V. After the decision in which village nodes to set up DCs has been made, they are brought together in the set D=$\{1, ..., m\}$, which is a subset of the set of villages V. The routes are going from a village k to a DC j, and from there, one can determine the ask for demand. The node t only serves as an artificial dummy node with the purpose of extending the graph by arcs to which the total incoming flows in each DC can be assigned. Therefore, the set of routes for each OD pair k is given by $R_k = \{(k, j, t) \mid j \in D\}$. The set of all routes is given by $R = \{R_1, ..., R_n\}$, assuming n to be the number of villages, and therefore also the number of OD-pairs in G. A route $r \in R$ is uniquely defined

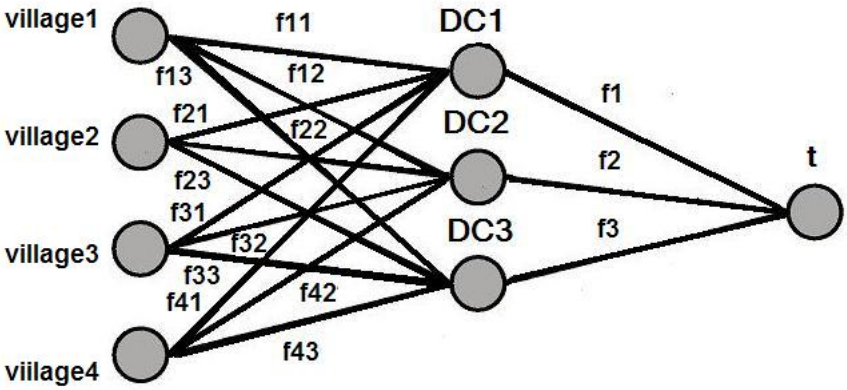

Figure 1: Expample of 4 OD-pairs with 3 DCs

through the pair (k,j). There are two kinds of flow within the network representing f_a: $f_{(k,j)}$ or shortly, f_{kj}, represents the flow between a village k and a DC j. $f_{(j,t)}$, shortly f_j, represents the flow from a DC j pseudo-ingoing into the depot. The amount f_{kj} is therefore the number of people who decided to travel from village k to DC j and f_j the call for supplies within DC j, thus, the number of incoming people. The route flow can be defined as $h = (h_{kj})_{k \in V, j \in D}$ since every route r is uniquely defined through the pair (k,j), which is the arc going through village k to DC j. As every route is well-defined, the demand constraint can be written as $\sum_{j \in D} h_{kj} = d_k \ \forall k \in V$. Every link flow f_{kj} is uniquely determined through a route flow h_{kj}, it holds $f_{kj} = h_{kj}$ because the entire route flow h_{kj} is only passing through one link. For f_j, the flow conservation constraint has to hold, this means, the inflow into a DC j has to be equal to the link outflow, which is only flowing through one link (j,t). Therefore, the sum of all routes visiting DC j has to be equal to the link flow f_j, which yields $f_j = \sum_{k \in C} h_{kj} = \sum_{k \in C} f_{kj}$. Hence, there is a bijective relation between route flows and link flows. Therefore, everything can be expressed in terms of link flows. From the view point of the people, there are two quantities which need to be optimized.

The first one is distance, where the cost function could be assumed as journey cost which could be traveling time or effort needed. It is assumed to be independent from the amount of flow f_{kj}: $t_{kj}(f_{kj}) = \phi(a_{kj})$, where a_{kj} represents the distance

between village k and DC j . Since ϕ is independent of the flow, it is constant regarding to flow and thus, a continuous and non-decreasing function. Besides, since travel cost should be reasonable, it's a non-negative function. It makes sense to define ϕ as a constant value $\alpha > 0$ in order to have a linear function regarding to distance. Instead of maximizing the covered demand as the second factor, the cost of uncovered demand for the people is considered in order to express the overall objective as a sum of traveling cost and cost for uncovered demand which has to be minimized. Furthermore, since uncovered demand can either be zero or positive and rising with higher flow of people, everything can be expressed through non-negative and non-decreasing cost with respect to the flow. There are three factors influencing the uncovered demand, f_j denotes the number of incoming persons into a DC, b is the requirement of relief goods for each person, which is given in advance, and δ_j stands for the capacity of a DC, also given in advance. Let $\psi(f_j, b, \delta_j)$ be the function of uncovered demand. Then, one gets

$$\psi(f_j, b, \delta_j) = \begin{cases} 0 & f_j \geq \frac{\delta_j}{b} \\ b - \frac{\delta_j}{f_j} & else. \end{cases}$$

If the number of incoming people is not exceeding the needed capacity per person, the uncovered demand is zero, whereas if more people arrive at a DC than capacity is available, the uncovered demand for each person at this DC is the difference between requirement and capacity per person. This function is not a convex one, but it's at least non-decreasing. One can therefore assume a cost function $t_j(f_j) = p\psi(f_j, b, \delta_j)$ with $p > 0$ denoting a constant price. It's easy to see, that this cost function is fulfilling the needed requirements of being non-negative, continuous and non-decreasing. The range of the function is constantly zero until the point $\frac{\delta_j}{b}$. From this point on, the uncovered demand is growing continuously with rising incoming demand. In conclusion, the total cost function for the flow on a route k is given by

$$c_{kj}(f) = \phi(a_{kj}) + p\psi(f_j, b, \delta_j).$$

Both terms are non-negative, non-decreasing and continuous and therefore, also the sum of them is non-negative, non-decreasing and continuous and consequently, fulfilling the assumptions needed in order to get a Wardrop Equilibrium. Since X

is the set of feasible flows (f,h), its projection X_f on the set of feasible link flows $f = ((f_{kj}), (f_j))_{k \in V, j \in D}$ is given by the demand constraints for all commodities $k \in C$. Due to the flow conservation constraint, f_j results from f_{kj}, and a feasible link flow is already fully given through (f_{kj}) satisfying its demand constraint. The demand constraints are linear and each forms a simplex

$$S^{(d_k)} = \{(f_{k1}, ..., f_{km}) \mid \sum_{j \in D} f_{kj} = d_k, f_{kj} \geq 0, j \in D\}$$

where each DC denotes one corner point. Therefore, X_f is a product of n enlarged simplices of the length d_k for each k:

$$X_f = \prod_{k \in V} S^{d_k}.$$

Interpreting Wardrop's First Principle, due to the fact that for each commodity k, the routes in R_k are uniquely defined through (j,k), f is a Wardrop Equilibrium if and if only

$$c_{kj}(f) = \min_{j'} c_{kj'}(f)$$

holds for all pairs (k,j) with $f_{kj} > 0$. This means that if there exists a flow which is bigger than zero on a route $k \to j \to t$ as well as on another route $k \to j' \to t$, the reduced cost for the route with the shorter path compensate for the higher uncovered demand on this route. The minimum-cost multi-commodity flow problem is given by

$$\min_f \{\sum_{k,j} \int_0^{f_{kj}} \phi(a_{kj}) dz + \sum_j \int_0^{f_j} p\psi(z, b, \delta_j) dz \mid f \in X_f\}.$$

Since the first integrand of the two combined integrals is independent from the flow f_{kj}, one gets

$$\int_0^{f_{kj}} \phi(a_{kj}) dz = \phi(a_{kj}) f_{kj}$$

for the first term. This is linear and thus, also convex. If $z < \frac{\delta_j}{b}$, then the integrand $\psi(z, b, \delta_j)$ is zero. Hence, one gets

$$\int_0^{f_j} p\psi(z, b, \delta_j)dz = p\int_{\frac{\delta_j}{b}}^{f_j}(b - \frac{\delta_j}{z})_+ dz = p[b(f_j - \frac{\delta_j}{b})_+ - \delta_j(\int_{\frac{\delta_j}{b}}^{f_j}\frac{dz}{z})_+]$$

$$= p[b(f_j - \frac{\delta_j}{b})_+ - \delta_j(log(z)|_{\frac{\delta_j}{b}}^{f_j})_+] = p[b(f_j - \frac{\delta_j}{b})_+ - \delta_j(log(f_j) - log(\frac{\delta_j}{b}))_+]$$

for all j∈D. In the case $f_j < \frac{\delta_j}{b}$, the integral is zero. This is a convex function due to the fact that for one thing, $pb(f_j - \frac{\delta_j}{b})_+$ is a linear function with regard to f_j, and for another thing, since $log(f_j)$ is a concave function, $-log(f_j)$ is convex. Because δ_j is positive and $log(\frac{\delta_j}{b})$ is just a constant factor, $-\delta_j(log(f_j) - log(\frac{\delta_j}{b}))_+$ is convex with regard to f_j. This means, both parts of the integrated uncovered demand are convex and the sum of two convex function is again convex. Besides, it is also known that for a function f which is non-decreasing (which holds for both factors, as mentioned above, and also for the sum of them), the integral $\int f(x)dx$ is convex. The feasible set is a product of simplices and therefore a convex polyhedron. Thus, this is a convex optimization problem. Since the function is also continuously differentiable and the decision variables in f are non-negative, the needed prerequisites are fulfilled and one can apply the Frank-Wolfe Algorithm.

Another consideration which has to be taken into account is that the model by now omits the possibility that some persons don't visit any DC since at equilibrium, the cost for not visiting a DC is always higher than the utility. In order to enable this option, the network needs to be extended with a further DC node, hence, the set of DC nodes is extended to D∪$\{z\}$ where z is a dummy node. It represents the possibility of staying at home. Since every DC yields for every person a benefit of b at full covered demand, which means for the system a benefit of pb for each peron and each DC, the cost of uncovered demand by a visit of the dummy DC are $t_{zt} := pb$ per person. On the other hand, no travel cost arise, i.e., $t_{kz} = 0 \quad \forall k \in V$. In total, this yields $c_{kz} := pb$. People, whose travel costs exceed the gain from the supply by the most suitable DC, will choose the dummy DC.

5.2 Application Of The Frank-Wolfe Algorithm

Considering the assumptions which had been made in advance, assuming the number of opened DC is m and $z := m + 1$ denotes the dummy DC, one gets the problem

$$\min \omega(f) := \min_f \{\alpha \sum_{k=1}^{n} \sum_{j=1}^{m+1} a_{kj} f_{kj} + p \sum_{j=1}^{m+1} [b(f_j - \frac{\delta_j}{b})_+ - \delta_j (log(f_j) - log(\frac{\delta_j}{b}))_+]\}$$

such that

$$\sum_{k=1}^{n} f_{kj} = f_j \qquad \forall j = 1, ..., m+1$$

$$\sum_{j=1}^{m+1} f_{kj} = d_k \qquad \forall k = 1, ..., n.$$

As f_j results from $\sum_{k=1}^{n} f_{kj}$, everything can be expressed in terms of f_{kj}. Thus, the only constraint needed for the Frank-Wolfe Algorithm is the demand constraint for all k=1,..,n. $(f_{kj})_{k=1,..,n;j=1,...,m+1}$ is the n×(m+1)-matrix describing the number of people going from village k to DC j. The column vector k of this matrix $f_k = (f_{k1}, ..., f_{km+1})'$ is an element of

$$S_{m+1}^{(d_k)} = \{(x_1, ..., x_{m+1}) \in R^{m+1} \mid x_1 + ... + x_{m+1} = d_k, x_j \geq 0,\ j = 1, .., m+1\}$$

where $S_{m+1}^{d_k}$ represents a standard simplex in R^{m+1} enlarged by the factor d_k (cf. model in [11]). Therefore, the feasible set X_f of the problem is a Cartesian product $S_{m+1}^{d_1} \times ... \times S_{m+1}^{d_n}$ of enlarged simplices, which is a polyhedron. In the case where the feasible set X_f would be only a simplex, it is easy to see that $s^t := arg\min_{s \in X_f} \bigtriangledown \omega^\top (f^t) s$, is attaining its optimum at a corner point of the simplex (which is the extremal point of a simplex) in each iteration. The problem considered in this work is slightly more complicated since X_f is a product of simplices. Still, finding the optimal s^t in each iteration is not difficult at all: Under the conditions given above, the problem is of the form

$$s^t := arg\min \{\sum_{k=1}^{n} \sum_{j=1}^{m+1} \frac{\partial \omega(f^t)}{\partial f_{kj}} s_{kj} \mid s = (s_{kj}) \in S_{m+1}^{d_1} \times ... \times S_{m+1}^{d_n}\}$$

in each iteration step t. The search point s is also a matrix of dimension $n \times (m+1)$. Since the gradient is only dependent on f_j and independent of f_{kj}, one can split this problem for each k [6], [11]. It decomposes into n partial problems of the form

$$s_k^t = arg\min\{\sum_{j=1}^{m+1} \frac{\partial \omega(f^t)}{\partial f_{kj}} s_{kj} \mid s_k = (s_{k1}, ..., s_{km+1})' \in S_{m+1}^{d_k}\} \quad k = 1, ..., n$$

where the column vector s_k denotes the transposed k-th row vector of the matrix s. The solution of each of these problems k=1,...,n is attained at a corner point of the enlarged simplex $S_{m+1}^{d_k}$, denoting one of the DCs. After calculating the partial derivatives, one gets n problems of the form

$$s_k^t = arg\min\{\alpha \sum_{j=1}^{m+1} a_{kj} s_{kj} + p \sum_{j=1}^{m+1} (b - \frac{\delta_j}{\sum_{k=1}^{n} f_{kj}})_+ s_{kj} \mid s_k = (s_{k1}, ..., s_{km+1})' \in S_{m+1}^{d_k}\}.$$

A capacity of zero is assumed for the dummy DC. The solution of the k-th problem is attained at a corner point of the simplex $S_{m+1}^{d_k}$, which means, the solution for the matrix s^t is of the form $(s_1^t, ..., s_n^t)' = (d_1 e_{j^*(1)}, ..., d_n e_{j^*(n)})'$ where $j^*(k)$ is the index of the optimal corner point of the k-th problem and e_j denotes the j-th unit vector in R^{m+1}. Thus, in every iteration step, for each search direction s_k, one has to test all of the m+1 corner points, in order to find the steepest descent direction (cf.[11]). As a starting value for the flows within each simplex, the center $f_{k\cdot} = \frac{1}{m+1} \sum_{j=1}^{m+1} d_k e_j = \frac{1}{m+1} d_k (1, ..., 1)'$ is chosen for each k.
For the initial increment $\xi = 1$, this yields the adapted Frank-Wolfe Algorithm version:

- For t=1:t_{max}\{
  - For k=1:n\{
    - Find the optimal corner point $s_k = d_k e_{j^*(k)}$ within the simplex $S_{m+1}^{d_k}$ such that

$$j^*(k) = arg\min_j \{\alpha a_{kj} d_k + p(b - \frac{\delta_j}{f_j})_+ d_k\};$$

  - \}

- Set $s = (s_1, ..., s_n)$
- Set $f = (1 - \xi)f + \xi s \quad$ for $\xi = \frac{2}{t+2}$
- Set $\sum_{k=1}^{n} f_{kj} = f_j \quad$ for j=1,...,m+1

- }

After a certain number of iterations t_{max}, depending on how much exactness is aimed, the flow matrix f won't change significantly any more. The resulting f is (close to) the Wardrop Equilibrium of the distribution of people over the DCs.

5.3 Optimization Of The Warehouse Locations

By now, it was only explained how the determine the people's distribution for a fixed combination of DC establishments. Optimality in this context refers to the two competing objectives taken into account by the humanitarian organizations, cost and uncovered demand. The cost implies the overall cost for set-up and maintenance of the DCs. Since a DC having higher capacity requires higher effort of set-up as well as possibly more people working, the cost rises monotonously. It is assumed that the cost is growing linearly with a positive constant $\eta > 0$. Thus,

$$\Phi_1(f, D) = \eta \sum_{j \in D} \delta_j$$

is the first objective. Obviously, this objective is not dependent of the flow distribution. The second objective consists of all that demand of the people which stay at home, which is all the inflow in the dummy DC times the constant per-capita need for relief goods b>0, plus all the uncovered demand within all DCs where demand for goods exceeds capacity. Hence,

$$\Phi_2(f, D) = \sum_{j \in D \cup \{z\}} (bf_j - \delta_j)_+$$

is the second objective with $\delta_z = 0$.
Complete enumeration is used to cover all $2^n - 1$ possibilities of facility location combinations assuming that in every village there is either a DC established or not. Since the option of not opening any DC makes no sense at all, it can be

ignored. For each possibility, the Wardrop Equilibrium is determined. Out of the resulting f, the values for the objectives $\Phi_1(f, D)$ and $\Phi_2(f, D)$ are calculated. As the problem is bi-criterial, there exists not only one optimal solution but all non-dominated solutions are "optimal". A solution is dominated, if there exists another solution for which (i) cost is lower or equal, (ii) uncovered demand is lower or equal, and (iii) either cost is strictly lower or uncovered demand is strictly lower. The set of solutions which are not dominated is called Pareto set. Within this Pareto set, one can either choose lower cost at the expense of higher uncovered demand or conversely, cover more demand by having higher cost. The decision to choose one Warehouse Location solution out of the Pareto set is left open to the organizations so they can adapt it to their financial capabilities and to the degree of urgency.

6 Computational Experiments

6.1 Test Instances

The data provided is from the region of Thiès which is located in western Senegal. This region is split into 32 "communautés rurales" and each of them consists of a certain number of villages [19]. As the transportation of goods to DCs is not considered in this work and the depot is only used as a dummy node, its location is not specified. The instance Mbayene, consisting of 11 villages, is used to provide some insight information in terms of parameters and convergence, which is then applied to the other instances.

The Nearest center (NC) model used in Tricoire et al. [19] assumes that the people from one village walk to the closest DC which is opened, where supply is split up equally between all the people entering. It is assumed that 100% of the population within a village will walk to the closest DC if the distance is less than 6km. Only 50% of the population within a village will walk if the distance is 6km or more but less than 15km and if the distance is more than 15km, nobody will walk. The number of people who are not leaving their homes flows into the uncovered demand. In order to get some information about the quality of the model in this work, it is compared with the NC model which is also solved using complete enumeration

instead of Branch&Cut, which was used in Tricoire et. al [19]. Since the Wardrop Equilibria (WE) model assumes more flexibility, one can expect better results in terms of uncovered demand.
In Fig.2 (Fig10,Fig.11), red crosses (+) represent the Pareto front of the WE model and green crosses (×) the front of the NC model. Blue crosses (×) denote the evaluation of the NC model under the assumption that the WE model holds.

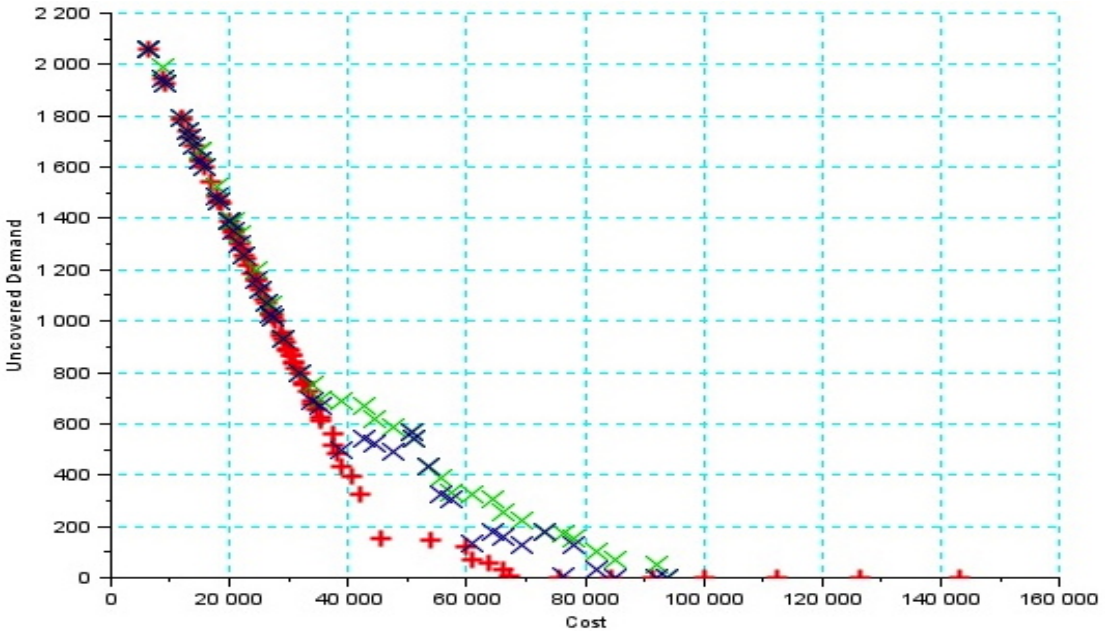

Figure 2: Comparison of the Pareto fronts of the WE model and the NC model (red/green) and their facility location solutions in uncovered demand (red/blue) on the region of Mbayene with 11 nodes

6.2 Parameter Settings

Parameters used for the model are adapted to the assumptions about the people's willingness to leave their villages made in [19]. Opening cost for a DC is assumed to be independent of the location but dependent of the capacity of the DC where one unit of capacity has a price of 20 cost units. This price weight does not have any impact on the model as long as price is assumed to depend linearly on capacity. For simplicity reasons, the need for relief supply b is assumed to be 1. Just as in the NC model, the capacity of a DC is assumed to be linearly dependent of

the number of inhabitants of the village in which the DC is located and is set to 3 times demand, thus, $\delta_j = 3d_j$ holds for all DCs j which are opened. The parameters α and p are the weights representing the people's willingness whether to invest their energy to visit a DC or not. Since the dummy DC, which stands for staying at home, has a travel cost of $\alpha a_{kz} = 0$, the cost for uncovered demand, pb=p, should be adapted in such a way, that following conditions (resulting out of the assumptions on people's willingness to leave made in the NC model) hold:

$$600\alpha + p(1-x) \leq p \qquad even\, if \quad 0 < x < \frac{1}{2}$$

$$1500\alpha + p(1-y) \leq p \qquad if \qquad \frac{1}{2} < y \leq 1,$$

where the distance is given in units of 10m and x (resp. y) is the fraction of covered demand for a distance of 6km (resp. 15km). The first inequality yields that if there is a DC within a distance of 6km, it will be visited even if demand is covered for less than 50%. The second inequality yields that a DC within a distance of 15km will be visited, if more than 50% of the demand is covered. Out of those two inequalities, one gets

$$\frac{p}{\alpha} \geq \max\{\frac{600}{x}, \frac{1500}{y}\}.$$

$\frac{p}{\alpha} \geq 2400$ results by assuming that the people are indifferent between x=$\frac{1}{4}$ and 6km.

Setting α=0.58 and p=1400, one gets that a person would traverse a distance of 6km if this person can expect that at least 25% of his demand is covered. In order to undertake a distance of 15km, at least 58% of the demand needs to be covered. The willingness to walk 20km is only present if the person can expect that his demand is covered to at least 83%. From distances of 21.14km and upwards, nobody will leave. If a person can only expect 10% of his demand to be covered, he or she wouldn't accept a distance higher than 2.41km.

As shown in Fig.2, the model using Wardrop Equilibria provides more efficient, but not apparently better solutions than the NC model for uncovered demand if the cost is low and thus, the number of opened DCs is rather small. This is due to the

fact that capacities are limited, which means, in the WE model almost all people in surroundings of a 6km-radius will leave their homes, even if the ratio of covered demand is very small. Everbody will leave his home in the NC model within this radius. Within surroundings between 6 and 15km, a rather small number will leave their homes in the WE model. This compensates for the uncovered demand in the NC model, where 50% will go within this radius and the supply has to be split over more persons in the case where there is too few supply. Solutions with unused capacity are not optimal for the lower cost section of the NC model (except for the solution with a cost of 8760, where 9.7% of the total capacity remains unused, which is slightly above the WE solution). When some supply is still left, even if there are few DCs but only 50% leave their homes in the NC model, the WE model covers the remaining supply and gives therefore far more optimal solutions. Comparing the optimal facility locations for the solutions with lower cost under the assumption that the WE model holds yields the same amount of uncovered demand (further explanations in section 6.3). At a cost of 39 060, 103 units of capacity remain unused for the NC model and the Pareto curve is not linear any more. Between 40 000 and 80 000, the gap in uncovered demand between the models is high. At a cost of around 45 000, the WE model has an optimal solution for which the uncovered demand is around 500 units (which corresponds to 500 persons) below the optimal solution from the NC model (evaluated with the WE model), which has even even higher cost. Opening all or almost all DCs leads to solutions which are only optimal for the WE model. Since everyone is provided with supply at a DC in his own town or at one in the near surroundings, there remains a lot of unused capacity in the NC model.

6.2.1 Comparison of traveling Cost

In Fig.3, 3 different weights on distance, corresponding to traveling cost, are compared. Thick blue crosses represent the chosen parameter α=0.58. Green crosses stand for α=0.1. This leads to optimal solutions of equal quality for a budget lower than 45 000. Between a cost of 50 000 and 70 000 α=0.1 gives better solutions in terms of uncovered demand since there is a lower weight on the factor distance and thus, people are willing to traverse higher distances even if the fraction of supply

they will get is very small. In this case, there is a willingness to pass a distance of 20km even if one can only get 14% of the needed supply. Though the optimal solutions are significantly better, putting such a low weight on the factor distance is not realistic at all. Red crosses represent α=2. This leads to very high traveling cost and as a result, people are willing to leave their villages only if there is a DC relatively close to them. A distance of 6km in this case would only be traversed if demand is covered to 86%. From 7km upwards, nobody will go, even if demand could be covered in full and thus, a lot of supply remains unused. Again, for people in need, this is not realistic.

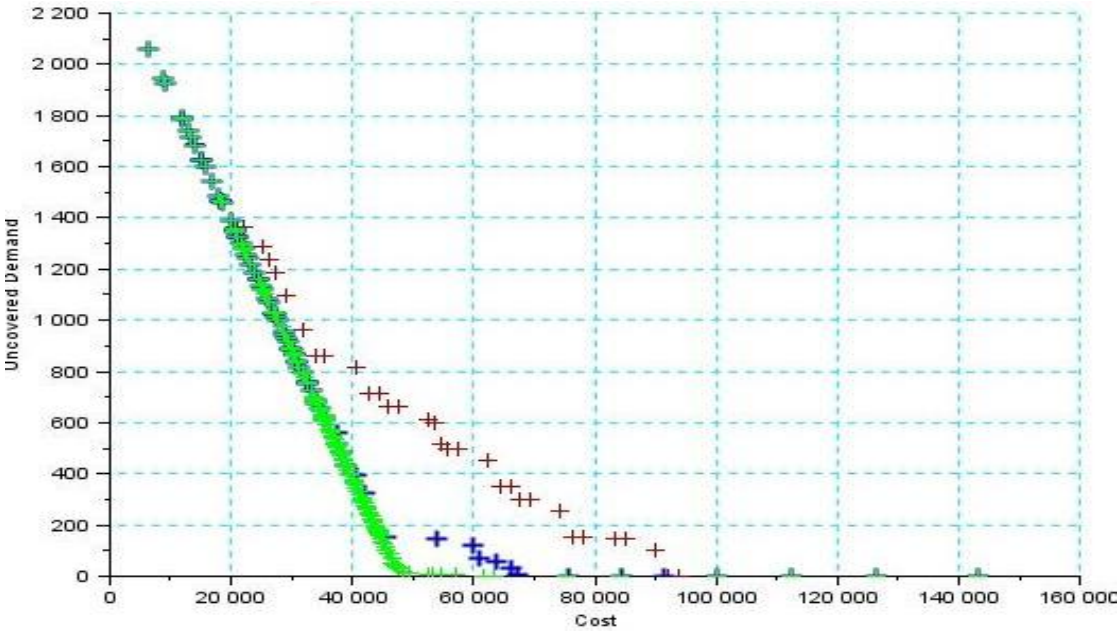

Figure 3: Comparison of the Price Factors for the traveling Cost, where blue denotes α=0.58, red α=2 and green α=0.1

6.2.2 Comparison of Price Weights

In Fig.4, 3 different weights on the perceived price for uncovered demand are compared. Thick red crosses correspond to the chosen parameter p=1400, light blue crosses stand for p=3000. The high weight p=3000 has very similar relation to the chosen parameter p=1400 as in Fig.3, the chosen parameter α=0.58 to the lower traveling cost α=0.1 by keeping the price weight unchanged. This is due

to the fact that putting higher weights on uncovered demand leads to the same effect as putting lower weights to traveling cost, people are willing to traverse higher distances even if the supply they can expect is not high. In comparison, at weights of p=1400 and α=0.1, people would pass 6km if their received supply will be more than 4% of their actual need. With the parameter combination p=3000 and α=0.58, they would pass for a supply higher than 11.4% of their need. For the chosen price weights it would need 25% for a willingness to traverse 6km. This approach is leading to good solutions but it's also not very realistic. Pink crosses represent p=100. This means that the cost for the dummy DC are very low (c_{kz}=100) and thus, the willingness to leave one's village is almost non-existent. The highest distance people would traverse is 1.72km. They would pass this distance only if their demand is covered to more than 99.6%. Therefore, people would go only to a DC which is opened in their own village, which is leading to a linear Pareto front, the more capacity is provided, the higher the cost is and the higher the uncovered demand is. If people have the privilege to have a DC in their village, their getting their needed supply and if not, they're not leaving. Since the capacity of the DCs is set to 3 times the number of inhabitants at a village, a lot of supply capacity remains unused. Again, this approach is not making much sense for people in need.
traveling cost and Price weights could also be analyzed jointly by considering the ratio $\frac{p}{\alpha}$, as both parameters depend on each other.

6.2.3 Comparison of DC Capacities

In Fig.5, one can find a comparison of 4 possibilities of DC capacities, where all of them are dependent of the number of inhabitants of a DC. Dark blue crosses represent the chosen model with capacities of 3 times the demand of a DC, red crosses stand for capacities of 4 times the demand and yellow ones for 6 times the demand. The relation is obvious: the more the capacity is raised, the lower the uncovered demand will be, leading to a higher willingness of people to leave their own villages in order to get supply. In this way, if the capacity is set too high, a lot of it remains unused which is not optimal for the relief organizations with limited budgets. For a factor of 4 or 6, one gets optimal solutions with totally covered

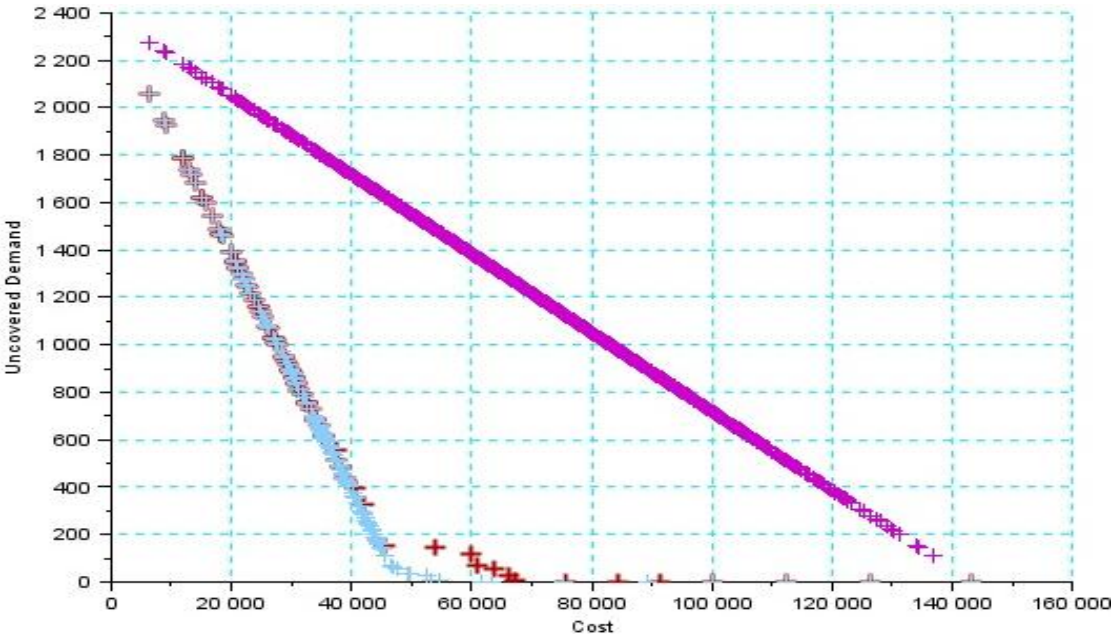

Figure 4: Comparison of the price weights for uncovered demand with the price factor p=100 in pink, p=1400 in red, and p=3000 in light blue

demand and very high cost compared to optimal solutions for full coverage with a factor 3, where an expenditure of almost 110 000 can be saved in comparison to the factor 6 and an expenditure of 60 000 for the factor 4. Besides, it makes more sense to set up more DCs with lower capacities than few larger ones since more people can be reached and splitting up relief supplies is more useful than providing some people full supply and some nothing. An additional issue, though it is not captured by the model, is that DCs shouldn't be too large since they're endangered to be plundered over night. As for reasonable budgets, all 3 possibilities lead to almost the same Pareto front, setting up DCs with capacities of 3 times the demand is optimal within this set of choices. Now, the question arises if the capacity can be set to less than 3 times the demand. Light blue crosses denote capacities of 2 times the demand within a village. Fig.5 shows that this leads to worse solutions for a cost between 40 000-60 000, yielding uncovered demand which is higher with a difference of almost 200 units in the worst case (which corresponds to the full demand of 200 people). Hence, capacities of 3 times the demand are optimal within the given data set.

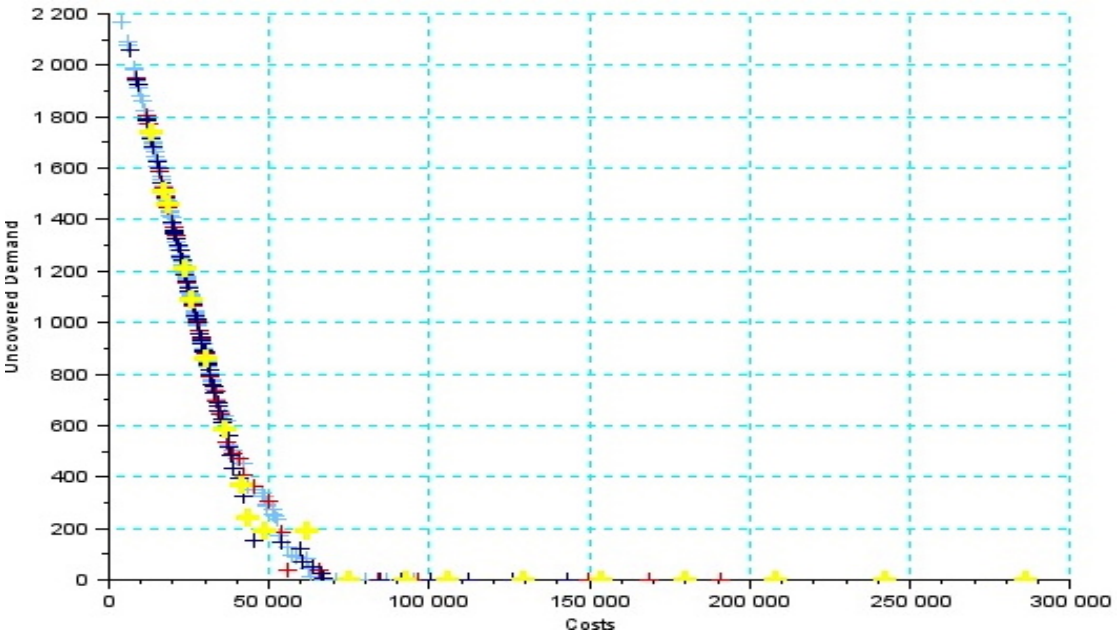

Figure 5: Comparison of DC Capacities, all linearly dependent of the number of village inhabitants with the factors: 2=light blue, 3= dark blue, 4=red and 6=yellow

6.2.4 Comparison of Relief Supply Needs

In Fig.6, 3 different amounts of relief need are tested. Besides the chosen amount of b=1 per person, which is denoted by blue crosses, the double amount (b=2, marked by green crosses) and half of the amount (b=$\frac{1}{2}$, marked by red crosses) were trialled. If people would need twice as much as in the chosen model (b=2), the willingness to leave would rise. Considering the same investment cost as for b=1, a higher amount of demand would remain uncovered, leading to a Pareto curve which is moved in parallel to the actual curve with a difference in covered demand of almost 2400 units at the same cost (corresponding to the uncovered need of 1200 persons if b=2). Lowering the need to b=$\frac{1}{2}$ leads to a decrease in uncovered demand for 1200 units for low cost, which corresponds to a linear relation, but for an average budget (which means an average number of DC set-ups), the relation is not linear any more since the difference in uncovered demand between b=1 and b=$\frac{1}{2}$ gets smaller. This is due to the fact that there is supply capacity which remains unused by setting up an average number of DCs. From a budget of 40

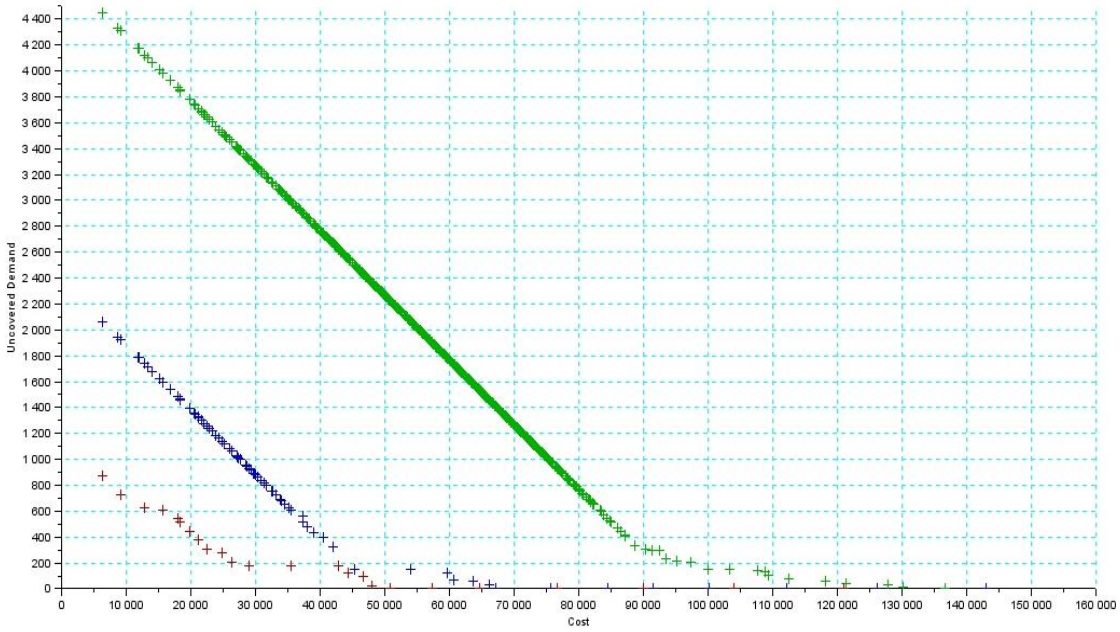

Figure 6: Comparison of Different Needs of Relief Supplies with b=1 in blue, b=2 in green and b=0.5 in red

000 and upwards, this difference in uncovered demand is not significant any more.

6.3 Comparison of Facility Locations

As the optimization needs to be done in terms of facility locations, one has to compare the optimal locations of both models. In order to get a fair comparison in uncovered demand, both location solutions need to be compared with one model. Since the WE model allows more flexibility in terms of distribution, the comparison is assumed under the fact that the WE model holds. Thus, the optimal facility location solutions from the NC model are evaluated based on the WE model. In order to get an approximation about the improvement of facility locations with the WE model, the solutions for 20 fixed budget constraints are compared. The lowest uncovered demand is evaluated for both models for a solution the cost of which is smaller or equal to the given budget constraint. Next, the optimal location from the NC model is evaluated based on the WE model. Afterwards, the best solution for uncovered demand is compared with the uncovered demand in the optimal location solution which was chosen by the WE model considering the same budget

constraint. In many cases the models provide the same solutions but in some cases, the WE model yields solutions with significantly lower uncovered demand, as one can see in Fig.7. The green lines stand for the NC model and the red ones for the WE model's solutions. The first 10 budget constraints were chosen equidistant with budget steps of 10 000 in a budget interval of 10 000-100 000. 10 more budget constraints were added between 20 000 and 80 000 as the optimal solutions differ apparently in this interval (see Fig.2). The budget constraints get even more crowded within the subinterval 30 000-60 000, as the optimal solutions differ most within this interval. For a budget constraint of 52 000, the uncovered demand is almost 390 units less than for the solution chosen with the NC model. This is an improvement of more than 71.68% relative to the NC model. Between 70 000- 80 000, the improvement is almost 100% relative to the NC solutions. The approximation yields an average improvement of 37.78% for each location with regard to the NC model. This are on average more than 122 units of uncovered demand for each location which corresponds to 122 persons which are absolutely unprovided.

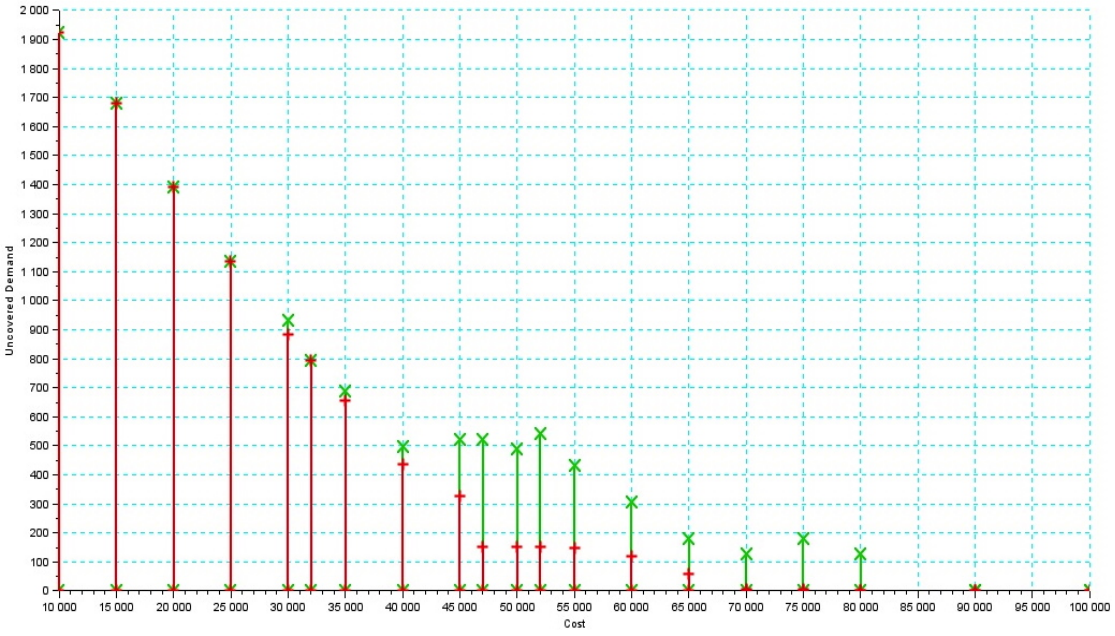

Figure 7: Comparison of the chosen facility locations of the WE and NC model for different budget constraints

6.4 Convergence Analysis

Since the Frank-Wolfe Algorithm is converging relatively slowly in the near surroundings of the optimum [18], a termination criterion has to be determined which provides sufficiently good solutions for a moderate time level. In order to get an impression about the performance of the maximum-iteration constraints of 50,75,100,150 and 200 iterations, the relative gap between the best lower bound so far, which is the so far highest optimal linear approximation, and the lowest objective value which was found, which is the best upper bound so far, is determined. Furthermore, the relative gap between the current lower bound and the current upper bound is evaluated, and also the relative maximal change of the flow components between the last two steps, since at equilibrium, the flow is static. Conversely, the maximal change within all flow components was tested as a stopping criterion by setting it smaller than ϵ=1 and ϵ=0.1, and evaluating the number of iterations needed. For both of them, a comparison to the 100-iterations termination constraint model in terms of uncovered demand for the chosen locations was done. The CPU time was evaluated for all models and the tested instance was again Mbayene with 11 villages.

Table 1: Component with maximal change within last 2 steps of fixed # of iterations, considering the average over all solutions, the maximum, the time and the change rate relative to total demand

iterations	avg MAX	max MAX	CPU time	rate avg	rate max
50	2.610	10.605	11,80min	0.44%	0.11%
75	1.847	8.850	17.73min	0.37%	0.08%
100	1.340	6.750	28.80min	0.28%	0.06%
150	0.947	4.098	37.60min	0.17%	0.04%
200	0.731	2.881	56.06min	0.12%	0.03%

After 50 iterations, the maximal difference between two components of the flow matrix in the last two iteration steps is on average 2.6 units of flow for all possible solutions and 10.6 for the worst solution. The calculation of the Equilibrium for one solution takes only some seconds, the calculation of the Pareto front for all 2047 possibilities takes 11.8min. For 75 iterations, the overall program runs in 17.7min and the maximal difference in flow between two iterations amounts to

Table 2: Improvement of the average maximum component failure of last 2 steps of all solutions

iterations steps	Improvement
50-100	48.65%
100-150	29.30%
150-200	22.89%

Table 3: Average relative gap between UBD and LBD relative to LBD

iterations	current step LBD/UBD	best LBD/UBD so far
50	0.6717439%	0.5007494 %
75	0.3398941%	0.2305570%
100	0.2078946%	0.1325446%
150	0.1111295%	0.0613630%
200	0.0677436%	0.0356486%

1.85 on average and 8.85 in the maximum. After 100 iterations, the maximal difference between two components for the last two steps averages to 1.3 and is 8.85 in maximum, while the CPU time rises to 28.8min. The improvement of the maximal difference from 50 to 100 iterations amounts to 48.65%. The difference for the last two steps after 150 iterations is 0.95 on average and maximal 4.1, which is an improvement of 29.3% to the maximal difference after 100 iterations at a CPU time of 37.6min. Hence, the difference between the flow components didn't change very much within the last 50 iterations. For 200 iterations, the program takes already 56.1min. The maximal maximum difference is 2.9 and the average maximum difference is 0.7 for all possibilities, which results in an improvement of 22.9% between 150 and 200 iterations. It's likely that the solution is already not far away from the equilibrium but considerably more iterations (and far more time) would be needed to get static flows. But there is still the possibility that for some iterations, the flows are barely changing and get more dynamic again later. This is also due to the fact that the stepsize is lowering the component change more and more. Therefore, more analytical convergence tests were done: The relative gap between the best found upper and lower bound so far as well as the relative gap between the current lower bound and the current upper bound were evaluated for t=50,75,100,150 and 200. Both of them should converge but

Table 4: Minimal # of iterations needed for a fixed maximal component change between 2 iterations

max. diff.	avg# iterations	max# iterations	min# iterations	CPU time
ϵ=1	104	346	9	25.7min
ϵ=0.1	992	3429	19	8.15h

the second one not necessarily monotonically. As one can see in Table 3, the relative gap for the current bounds is 0.67% after 50 iterations, falling by 0.33% after the next 25 iterations and by another 0.13% from 75 till 100 iterations, which is a decrease of 0.46% from 50 to 100. In the next 50 iterations from 100 to 150, the decrease is getting much weaker, falling by only 0.10%, and there is almost no observable change (0.04%) for the next 50 iterations from 150 to 200. Setting this gap lower than 0.2% as a termination criterion would be another possibility to get similar results as the ones in this work, but this could lead to very high CPU times for other instances, especially for larger ones, which is not realizable in practice. The same holds for the relative gap between the best upper bound so far and the best lower bound so far by setting it lower than the threshold 0.1%, but this gap provides monotonically convergence towards zero and is therefore more useful when the question of the number of iterations needed for significant changes arises. In Table 3, one can see that within the first 25 steps from 50 to 75, the gap is more than halved, it falls from 0.50% to 0.23% and after the next 25 iterations, again almost halved, to 0.13%. This means, from 50 to 100 steps, the gap has decreased by 0.37%. Within the next 50 steps from 100 to 150, there is a decrease by only 0.07%, and for 150 to 200 steps, no significant improvement can be achieved (0.03%). As all results show that the highest fluctuations happen within the first 100 steps (and the CPU time is acceptable), this number is used as a termination criterion for all results of the instance Mbayene, the other instances were also tested with t=200 (section 6.5).

The question of the impact of this chosen criterion on the Pareto optimal solutions for facility locations arises. Therefore, the 100 steps constraint was tested against the termination criterion of running the program until the maximum change between two components is smaller than $\epsilon = 1$ and $\epsilon = 0.1$, respectively. For $\epsilon = 1$, an average number of 104 iterations is needed but there is a large difference be-

Table 5: Comparison of Pareto optimal locations in uncovered demand for criteria t=100 and ϵ=1

solution index	improvement f. t=100	abs. difference	iterations for ϵ=1
280	0.12%	0.45871	208
265	-0.01%	0.03952	229
776	-0.03%	0.03952	229
281	-0.03%	0.04184	229
1728	0.02%	0.0202	339
1544	-5.03%	3.492548	10
1234	-12.06%	6.123063	256
1732	-17.65%	4.492567	288
1800	18.20%	1.5615563	10
1305	98.72%	5.9480507	10
1535	98.49%	2.7387757	11
2047	98.49%	2.510544	11
266	-0.01%	0.03952	229
193	-34.91%	1.8028273	344
1085	97.96%	3.1800855	13
1760	-65.63%	5.5192721	10
1307	98.72%	5.0983292	10
1437	98.86%	4.4674715	10
1439	98.72%	3.9653671	10
1503	98.49%	3.0126533	11
AVG.	0.33%	0.026649937	

tween the different solutions. The maximum number of iterations needed is 346 but there are also solutions which need only 9 iterations to meet this condition. The CPU time for this criterion is 25.7min, hence, almost the same as for the fixed iteration number criterion t=100. $\epsilon = 0.1$ requires 991.6 iterations on average with a maximum of 3429 iterations but there are also solutions with only 19 iterations needed. The CPU time gets very long, the program needs more than 8 hours for this rather small instance.

The comparison of both models against t=100 is given in Table 5 and Table 6. The coloured lines denote the solutions which appear as Pareto optimal solutions in only one of the models. The darker ones were only chosen by the epsilon-constraints while the brighter ones were only chosen by the 100-iterations constraint. The

Table 6: Comparison of Pareto optimal locations in uncovered demand for criteria t=100 and ϵ=0.1

solution index	improvement f. t=100	abs. difference	iterations f. ϵ=0.1
280	0.15%	0.59793	2050
265	-0.01%	0.04892	2291
776	-0.03%	0.04892	2291
281	-0.04%	0.05181	2291
1728	0.02%	0.02212	2093
1544	-1.11%	0.800683	22
1234	-13.03%	6.561148	1426
1732	-19.87%	4.963439	2957
1800	4.85%	0.357993	22
1305	94.30%	1.2783542	23
1535	94.18%	0.6809202	23
2047	94.18%	0.6241769	23
266	-0.01%	0.04892	2291
1085	91.56%	0.7174066	28
1307	94.18%	1.0700175	23
1437	94.82%	0.9426988	23
1439	94.18%	0.8322358	23
1503	94.18%	0.7490122	23
AVG.	0.35%	0.00996420	

fixed iteration criterion gives on average solutions for which the uncovered demand is 0.33% less than by using $\epsilon = 1$. The performance of t=100 is far better for solutions which stop after 10 or 11 iterations with the other constraint. Here, the already mentioned problem arises that after the algorithm already got almost stationary (the stepsize is also an affecting factor), larger changes happen later which are not captured by the constraint. Those solutions which get even worse in terms of uncovered demand by rising the number of iterations, emerge when people decide to go to the dummy DC rather than to another one in the equilibrium since it's the optimal solution for them but not for the system.

Although $\epsilon = 0.1$ is far more precise than $\epsilon = 1$, it also leads to solutions providing on average 0.35% more uncovered demand relative to t=100. 2 out of the 3 Pareto optimal locations which are only captured by the ϵ-criteria disappear by tightening the criterion from 1 to 0.1. The absolute difference for both comparisons to t=100

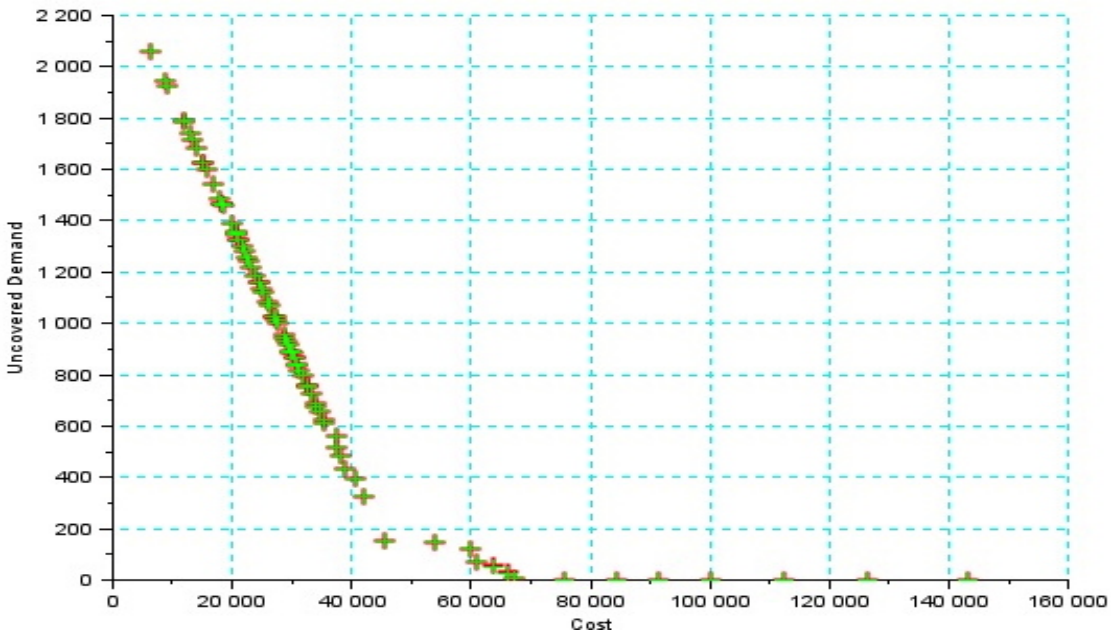

Figure 8: Comparison of the optimal solutions for the termination criteria t=100 in red and t=200 in green

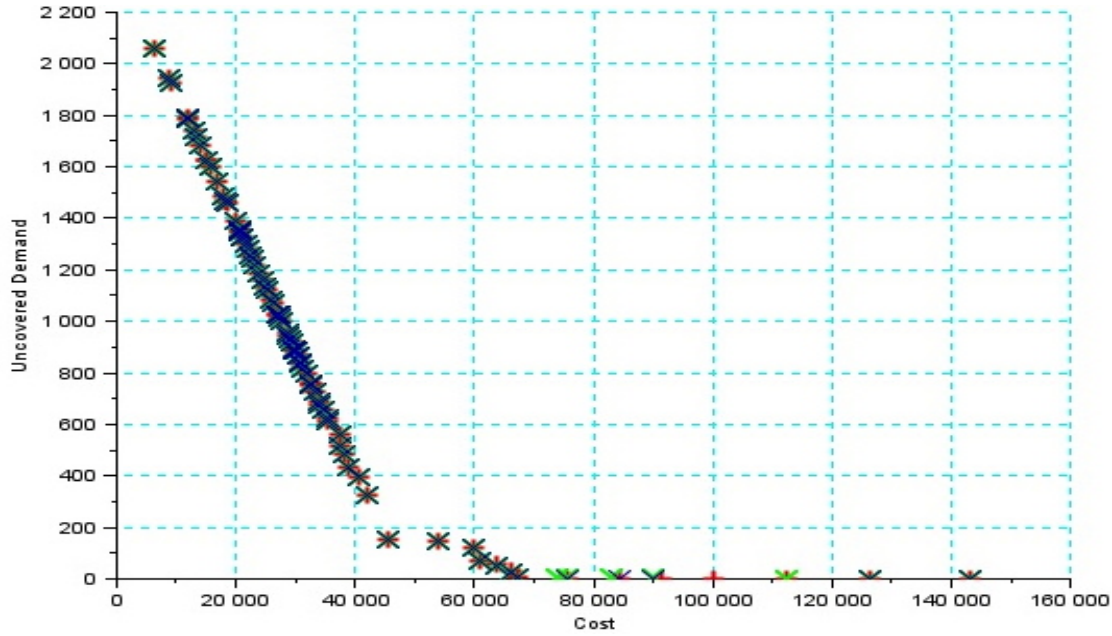

Figure 9: Comparison of the optimal solutions for termination criteria t=100 in red, $\epsilon = 0.1$ in blue and $\epsilon = 1$ in green

is on average smaller than 0.03 and for $\epsilon = 0.1$, in maximum less than 7. Since this termination constraint is, due to the high CPU time, not realizable and since it's not providing a large difference in the Pareto optimal solutions, using a fixed iteration size makes most sense.

6.5 Further Results

The program has an overall computation time of order $O(2^n n^2)$, which leads to a growth in time of about $2 + \frac{4}{n} + \frac{2}{n^2}$ when the number of villages rises from n to n+1. From the empirical results one gets the same solution by comparing the growth in CPU times.
The program is solved with Scilab 5.4.1. and run on a 1.66 GHz GENUINE Intel(R) CPU. The model was tested for instances of 8-16 nodes. Bigger instances would exceed a limit of 2 days on this processor, a faster one may manage also an instance with 17 nodes within this limit. All instances with 8-15 nodes were tested for a number of t=100 and t=200 iterations to see if there is a significant difference within the Pareto optimal locations. Some instances may require more steps to get so close to the equilibrium that the flow distribution doesn't make any significant steps that have an impact on the optimality of the facility locations. As the required CPU time is doubled by changing from 100 iteration steps to 200, testing for more than t=200 is inefficient in time.
Table 7 provides the solutions for both termination conditions, including also the relative gap for the solutions. It is obvious that the relative gap is not shrinking significantly between t=100 and t=200 for any of those instances. The relative gap provides only information about the convergence towards the equilibrium but it's not a reliable indicator of the changes within the Pareto set. The solution with the highest difference in the relative gap is the instance Thienaba with 9 nodes, with a change of 0.37% between t=100 and t=200. Nonetheless, the location positions for this instance differ for only 15 out of 148 solutions in t=200. For the instance Sandira with 14 nodes, the relative gap changes for only 0.13%, while out of those 623 solutions for t=100 only 578 remain the same for t=200. For Koul with 15 villages, the relative gap changes by 0.18%, and out of those 1175 Pareto optimal

Table 7: Problem size, Pareto front size, CPU effort and convergence level for Senegal instances

Region	#nodes	iterations	front size	CPU time	average gap
Ndiakhene	8	100	58	2.7min	0.2843145%
		200	58	5.4min	0.0896376%
Cherif Lo	9	100	129	7.6min	0.2664942%
		200	131	9.0min	0.0808580%
Thienaba	9	100	145	9.4min	0.5729961%
		200	148	19.8min	0.1991609%
Notto Gouye Diama	10	100	169	10.1min	0.3051723%
		200	168	19.8min	0.0803099%
Ndiene Shirak	10	100	87	16.8min	0.2588612%
		200	87	25.7min	0.0701270%
Mbayene	11	100	79	28.8min	0.1325446%
		200	81	56.1min	0.0356486%
Malicounda Wolof	11	100	141	32.3min	0.1138617%
		200	141	58.6min	0.0143713%
Pekesse	11	100	328	26.7min	0.2838319%
		200	329	62.4min	0.0949108%
Thiadiaye	12	100	654	54.8min	0.3180985%
		200	653	1.74h	0.1043530%
Thilmanka	12	100	355	1.23h	0.2017322%
		200	348	4.70h	0.0680869%
Tiba Ndiaye	13	100	591	2.99h	0.1295353%
		200	586	5.27h	0.0882761%
Mont Roland	14	100	1215	5.82h	0.4360950%
		200	1192	14.52h	0.1293622%
Sandira	14	100	623	3.60h	0.1902916%
		200	633	10.96h	0.0542638%
Koul	15	100	1175	11.37h	0.2772559%
		200	1136	23.88h	0.0930075%
Meouane	16	100	1761	26.30h	0.2269415 %
		200		>2d	

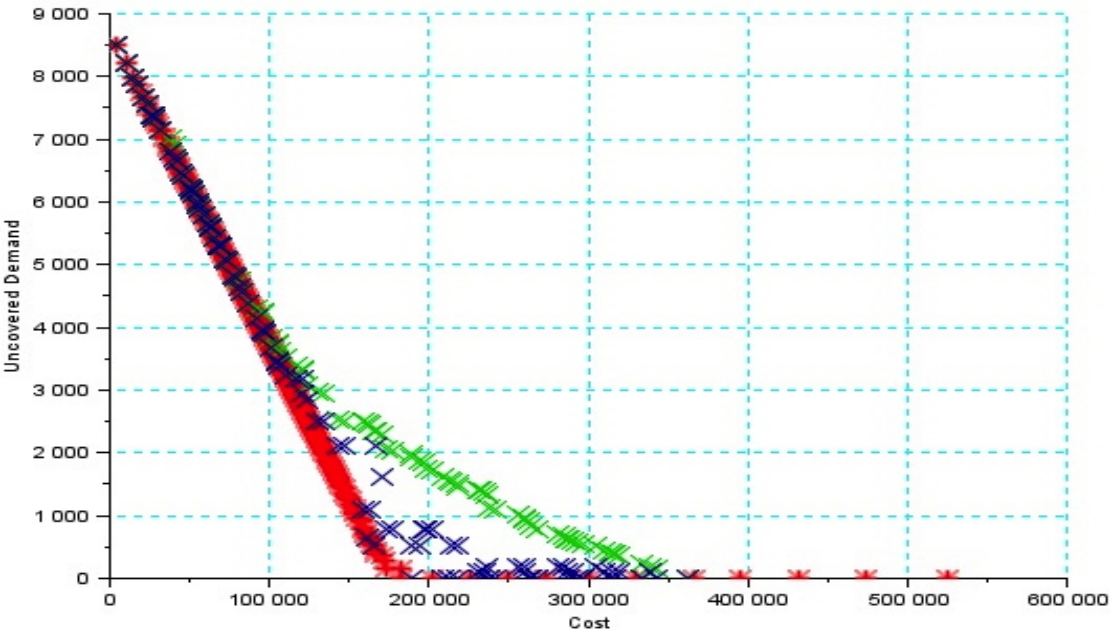

Figure 10: Comparison of the Pareto fronts for the instance Sandira with 14 villages, the WE model with t=100 in red, t=200 in light pink, the NC front in green + evaluation of the NC-optimal locations with the WE model in blue

solutions, there are only 794 which are the same for t=200. The Pareto front has the same appearance for t=100 and t=200 (see Sandira in Fig.10 and Koul in Fig.11), but several solutions are replaced by others. Thus, a larger number of villages requires more iterations in order to get better results. The instance Meouane with 16 nodes was not tested for 200 iterations because more than 2 days would be needed using the same processor, pushing the time limit up too far. Nonetheless, even results for t=100, which differ a lot from t=200, as in the instance of Koul, are still far better than the solutions that one gets with the Nearest center model [19]. This model provides only 202 Pareto optimal solutions. Out of these 202, only 128 are the same for t=100, and 123 for t=200. A scenario for the instance of Koul using 100 iterations, showing the gain in uncovered demand when using the NC model instead of the WE model, is given in the next chapter.

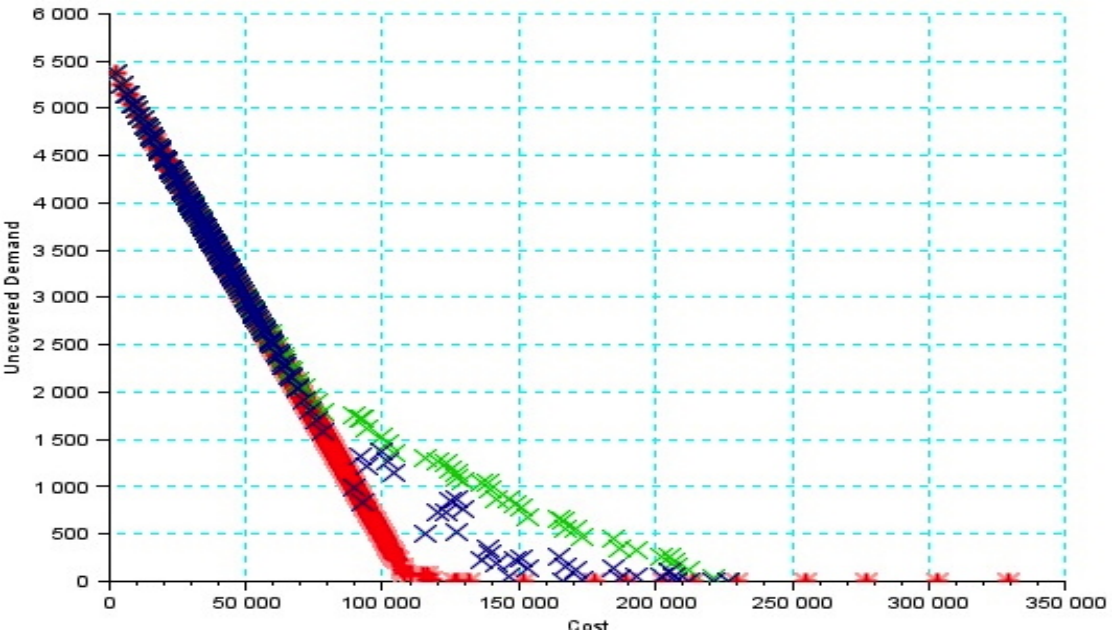

Figure 11: Comparison of the Pareto fronts for the instance Koul with 15 villages with t=100 in red, t=200 in light pink, the NC front in green + evaluation of the the NC-optimal locations with the WE model in blue

6.6 Practical Application

It's up to the humanitarian organizations to decide which selection of locations is actually chosen. The model can be used as a Decision Support System helping the Decision Maker to exclude suboptimal solutions. Calculating the Pareto front is the most challenging part within this process. The overall planning shouldn't require much more than one day. As soon as the Pareto front is evaluated, all other decisions can be done very fast. Adding further constraints, such as budget constraints or constraints depending on the region, requires only few seconds, since the evaluation of the Wardrop Equilibrium for one particular solution requires also only few seconds. Also solutions providing too much uncovered demand can be excluded. The final decision is always scenario-dependent and should be done by people and not by the Decision Support System (cf.[19]).
In order to present such a simulated scenario, a closer look on the instance of Koul (consisting of 15 villages) focusing on those solutions where the uncovered demand between the WE and NC model differs most, is instructive (see Fig.11).

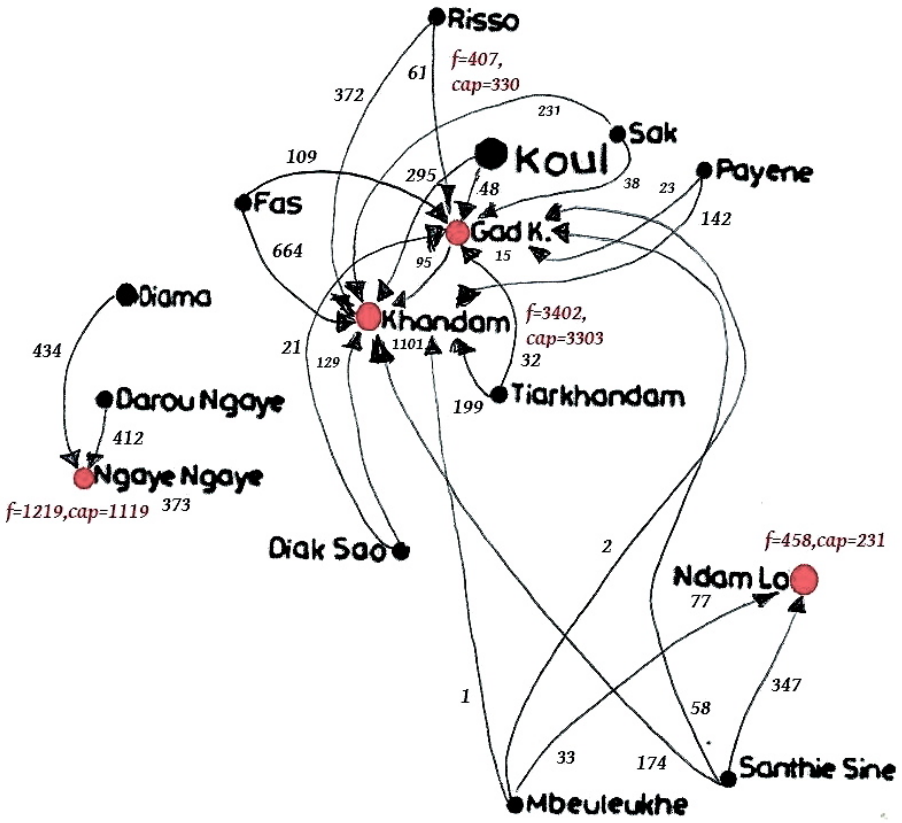

Figure 12: Flow distribution of an optimal solution of the WE model with a cost of 99660 and an uncovered demand of 503 considering a budget limit of 100 000

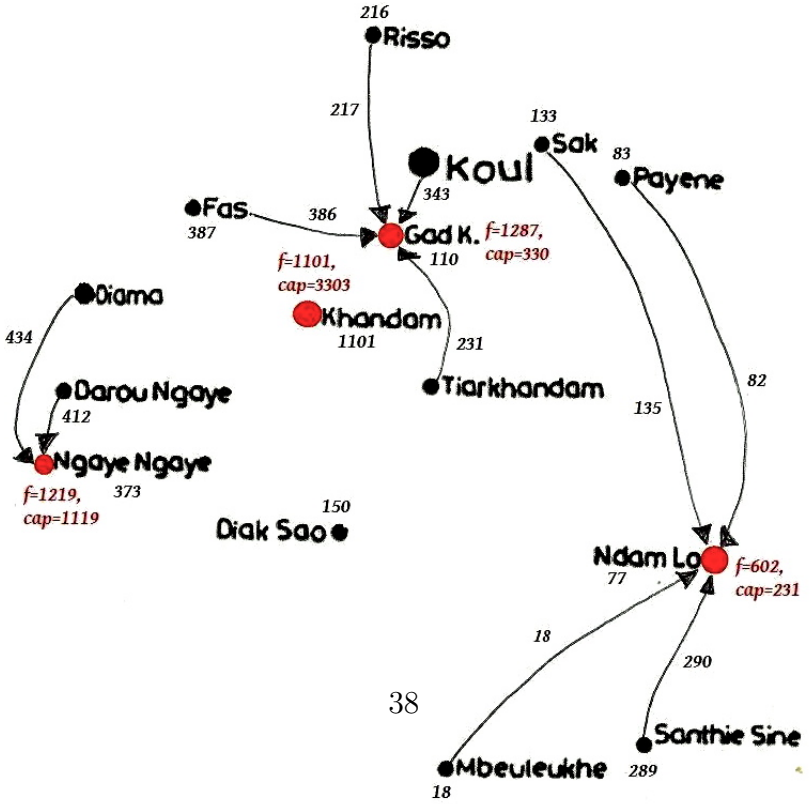

Figure 13: Flow distribution for the solution above with an uncovered demand of 2705, evaluated with the NC model

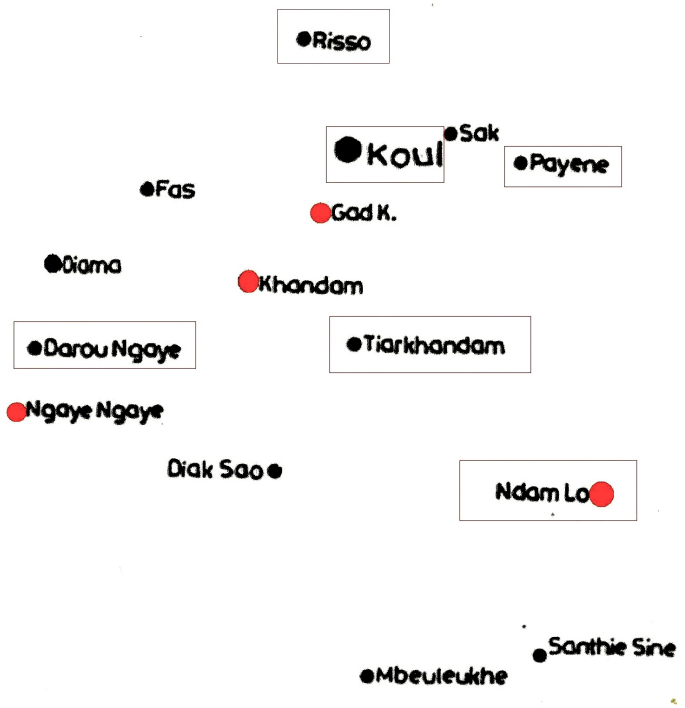

Figure 14: Optimal solution of the NC model for the same budget limit

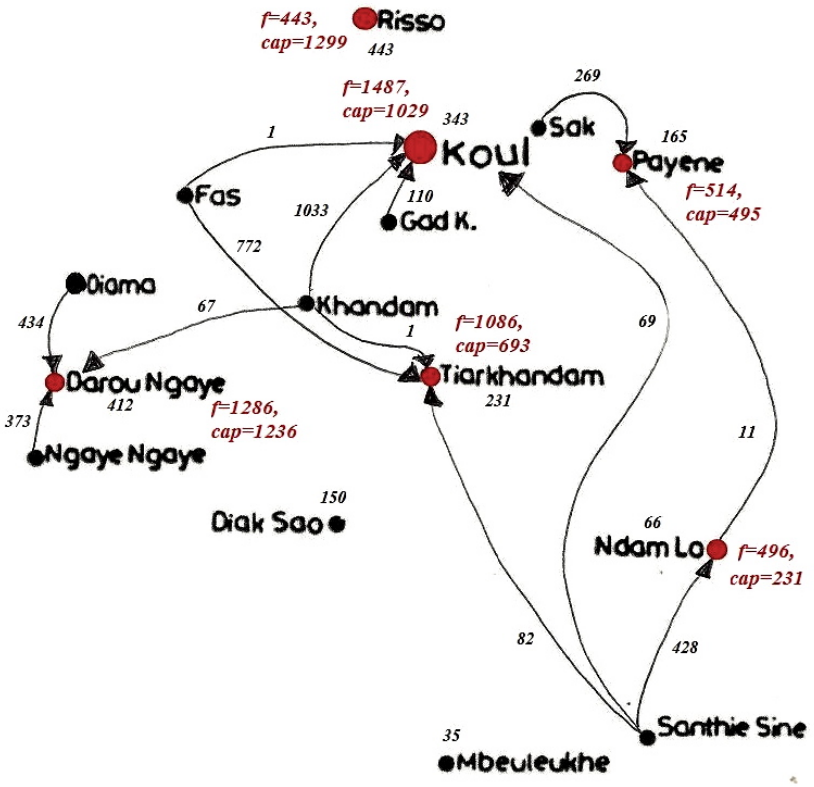

Figure 15:
Optimal solution of the NC model for the same budget limit, with the same budget and an uncovered demand of 1369, evaluated with WE model

It is assumed that the humanitarian organization has a budget limit of 100 000. The Pareto front of the WE model provides a solution with a cost of c_1 =99660 and uncovered demand of the amount c_2=503 with 4 opened DCs. The precise distribution can be seen in Fig.12. Evaluating this DC set-up with the NC model (by assuming that the NC model holds) leads to a rise of uncovered demand to 2705, which is 5.4 times the uncovered demand in the WE model (see Fig.13). In the largest DC in Khandam, a capacity of 2202 remains unused. This solution is not efficient any more if one assumes that the NC model holds.
Assuming now that the Decision Support System uses the other model to find the optimal solution under the same budget constraint, one gets a solution with 6 opened DCs (see Fig.14). This solution has 1522.5 units of uncovered demand. In order to compare this solution with the previously chosen one with 4 DCs, which was evaluated under the assumption that the WE model holds, one has to evaluate also this one with the WE model. As it can be seen in Fig.15, nobody leaves Mbeuleukhe in this solution, and furthermore, $\frac{2}{3}$ of the capacity in Risso remains unused. The uncovered demand amounts to 1369 in total. Compared to the previously chosen solution with c_2=503, this yields a rise of more than 63% in uncovered demand.

7 Conclusions and Future Work

A deterministic bi-objective model for disaster relief facility location planning has been introduced as well as a solution technique consisting of 2 levels, the construction of a solution and the calculation of its quality under the assumption that people will adapt their behaviour such that a Wardrop Equilibrium results. For the computation of the Wardrop Equilibrium, the Frank-Wolfe algorithm is used. Complete enumeration is used for determining Pareto optimal solutions on the upper level. This is only possible for small communities with at most 16-17 villages. For larger instances, an approximation to the Pareto front could be found by a heuristic, such as the Non-Sorting Genetic Algorithm 2 (NSGA2) for multi-objective optimization (cf.[7]). In such a metaheuristic framework, one might increase the number of iterations for the Frank-Wolfe Algorithm to get closer to the equilibrium, as the change in the Pareto set was quite high for all larger

instances used in this work. Increasing the iteration number to twice or more as much as used in this work is not overshooting since the NSGA2 is running in polynomial time with a computational complexity of $O(2n^2)$. Future research should be directed towards solving instances with a larger number of villages using this approach.

References

[1] Altman, E.; El-Azouzi, R.; Jimenez, T. and Wynter, L. (Jan. 2005): A survey on networking games in telecommunications

[2] Arrache, S. and Ouafi, R.: Accelerating Convergence of the Frank-Wolfe Algorithm for Solving the Traffic Assignment Problem (IJCSNS International Journal of Computer Science and Network Security, VOL.8 No.5, May 2008)

[3] Beckmann, M.; McGuire, C. and Winsten,C. (1956): Studies in Economics of Transportation (Yale University Press, New Haven, 1956)

[4] Bhattacharya, B. and Nandy, S. (Dec. 2011): New variations of the maximum coverage facility location problem (European Journal of Operational Research 224 (2013) 477-485)

[5] Clarkson, K.: Coresets (July 2008), Sparse Greedy Approximation, and the Frank-Wolfe Algorithm

[6] Correa, J. and Stier-Moses, N. (May 2010): Wardrop Equilibria ("ENCYCLOPEDIA OF OPERATIONS RESEARCH AND MANAGEMENT SCIENCE. EDITED BY J. J. COCHRAN. WILEY" July 2010)

[7] Deb, K.; Pratap, A.; Agarwal, S. and Meyarivan, T. (2002): A fast and elitist multiobjective genetic algorithm: NSGA-II (Evolutionary Computation, IEEE Transactions on 6(2):181–197)

[8] Farahani, R.; Asgari, N.; Heidari, N.; Hosseininia, M.; Goh, M. (Aug. 2010): Covering problems in facility location: A review (Computers and Industrial Engineering 2011)

[9] Fischer, S.; Räcke, H.; Vöcking, B. (May 2006, Seattle, Washington, USA): Fast Convergence to Wardrop Equilibria by Adaptive Sampling Methods

[10] Frank, M.; Wolfe, P. (1956): An algorithm for quadratic programming (Naval Research Logistics Quarterly 3: 95)

[11] Gutjahr, W. and Froeschl, K.: Project Portfolio Selection under Uncertainty with Outsourcing Opportunities

[12] Jaggi,M. (2013): Revisiting Frank-Wolfe: Projection-Free Sparse Convex Optimization (Journal of Machine Learning Research: Workshop and Conference Proceedings 28 (1): 427-435JMLR W+CP 28 (1): 427-435, 2013)

[13] Jayakrishnan, R.; Tsai, W.; Prashker, J. and Rajadhyaksha, S. (Jan. 1994): A faster path-based algorithm for traffic assignment (UCI-ITS-TS-WP-94-1)

[14] Mitradjieva, M. and Lindberg, P. O. (2004): The Stiff is Moving - Conjugate Direction Frank-Wolfe Methods with Applications to Traffic Assignment (Journal Transportation Science Volume 47 2, 05. 2013 280-293)

[15] Owen S. and Daskin, M. (April 1998): Strategic facility location: A review (European Journal of Operational Research 111 (1998) 423-447)

[16] Roughgarden, T. (May 2007): Selfish Routing and the Price of Anarchy (Optima Mathematical Programming Society Newsletter 2007)

[17] Sangwan, A.(May 2007): Wardrop Equilibrium and Potential Games in Wireless Ad-hoc Networks (B.Tech Project Thesis)

[18] Sheffi, Y. (1985): Urban Transportation Networks (Prentice-Hall, Englewood, NJ, 1985)

[19] Tricoire, F.; Graf, A. and Gutjahr,W. (Sep. 2011): The bi-objective stochastic covering tour problem (Computers and Operations Research 39 (2012) 1582-1592)

[20] Wardrop, J. (1952): Some theoretical aspects of road traffic research (Proceedings of the Institution of Civil Engineers, Part II 1, 325-378)

8 Appendix

1. Wardrop Equilibria code for the instance Mbayene

```
timer();         //start counting time
demand=[363
281
108
154
199
235
200
369
108
146
223];                    //demands within villages
distance=[0 3262 2194 3261 2603 2971 3995 2637 2921 2580 2993
        3262 0   2324 3895 3986 3618 733  2767 4281 1396  269
        2194 2324  0 1571 1662 1294  3057  443 1957 1642 2055
        3261 3895 1571  0  657  290  4628 1129 952  3213 3626
        2603 3986 1662 657  0   368  4718 1219 318  3304 3717
        2971 3618 1294 290 368  0   4351  851  663  2936 3349
        3995 733  3057 4628 4718 4351 0   3500 5013 2129 1002
        2637 2767 443  1129 1219 851 3500  0   1514 2085 2498
        2921 4281 1957 952  318 663  5013 1514 0    3599 4012
        2580 1396 1642 3213 3304 2936   2129 2085 3599 0 1127
        2993  269 2055 3626 3717 3349 1002 2498 4012 1127 0];
n=length(demand);                    //number of villages
alpha=0.58;                             //multiplicationfactor of costs per distance
p=1400;                                 //price of uncovered demand
b=1;                                 //need per person
eta=20;                         //fixcost for the opening of a DC per 1 unit of capacity
u=ones(2,1)*1000000;                    //vector of Pareto optimal solutions
u=[u;0];                            //added index row to identify Pareto optimal locations
cap=zeros(n,1);                                    //set-up of capacity vector
for k=1:n                                          //capacities of possible DCs in all villages
cap(k)=3*demand(k);
end
open=zeros(2^n-1,n);      //initialization of indicator vector of all of DC set-ups per solution
open(1,n)=1;
w=n;
for i=1:2^n-2         //set-up of binary vector which is 1 if there is a DC in the village
if open(i,w:n)==ones(n-w) then
open(i+1,w-1)=1;
open(i+1,w:n)=zeros(n-w);
w=w-1;
else
open(i+1,w:n)=open(i,w:n);
for v=n:-1:w+1
if(open(i,v)==0)then
open(i+1,v)=1;
open(i+1,v+1:n)=zeros(n-v-1);
break
end
end
end
end
GAP=zeros(2^n-1,1);                //initialization of relative gap for all solutions
for i=1:2^n-1
m=nnz(open(i,:));                                  //amount of opened DCs
z=zeros(n,n);                //initialization of indicator matrix of DC set-up possibilities
for k=1:n                 //indicator matrix where diagonal value=1, if DC is opened, else =0
if(open(i,k)>0) then
z(k,k)=1;
end
end
distanceDC=z*distance;       //leaves only the entries of distances to opened DCs, others=0
```

```
delta=z*cap;                          //capacities for DCs which are open are >0, all others =0
test=ones(n,1);                         //testing vector
for k=n:-1:1
if(distanceDC(k,:)*test==0)           //testing, if the column of distances is one leading to a DC
distanceDC(k,:)=[];                   //if the column contains only zero-entries, they are erased
end
end
distanceDC=[distanceDC;zeros(1:n)];    //expanding distance matrix with dummy DC with distance 0
for k=n:-1:1
if(delta(k)==0)                     //if village is not an opened DC...
delta(k)=[];                            //...the entry of the capacity vector is erased
end
end
delta=delta';
f=zeros(m+1,n);                         //set-up of a FLOW-matrix
g=zeros(m+1,n);                         //set-up of the matrix of search directions
flow=zeros(m+1,1);                    //set-up of incoming flows into each DC
xi=1;                                 //initialization of increment
for k=1:n
cornerpoint=zeros(m+1,m+1);                  //setup of a m*m simplex for each village
for j=1:m+1
cornerpoint(j,j)=demand(k); //simplex is an extended with the length of the demand values
end
for j=1:m+1
f(j,k)=(1/(m+1))*sum(cornerpoint(:,j));
 //for each k, the starting flow value is chosen to be the median point within each simplex k
end
end
for j=1:m+1                     //initialization of total flows within each DC
flow(j)=sum(f(j,:));
end
f1=zeros(m+1,n);
LBD=-1000000;                                    //initialization of lower bound
UBD=10000000;                                    //initialization of upper bound
for t=1:100         //t=100 or t=200, number of iterations
f1=f;                                 //defining the previous flow
flow1=flow;                           //defining the previous incoming flow into each open DC
xi=2*xi/(xi+2);            //determination of increment per iteration
uncovered=zeros(m+1,1);                 //set-up of the vector of uncovered demand
uncovered(m+1,1)=b;      //demand per person is completely uncovered at dummy DC
for j=1:m
if ((b*flow(j,1)-delta(1,j))>0);
//if demand is covered within a DC, the entry of uncovered remains zero
uncovered(j,1)=b-delta(1,j)/flow(j,1);
//for each DC, the amount of uncovered demand with resp. to the previous flows is considered
end
end
for k=1:n
//determination of search direction (vector to cornerpoint) with optimal value within each simplex k
minval=100000000;             //starting value for search of the optimal descent (very large number)
cornerpoint=zeros(m+1,m+1);        //value of previous cornerpoints is overwritten with zeros
for j=1:m+1
cornerpoint(j,j)=demand(k);  //for each simplex k, the cornerpoint have the value of demand k
end
for j=1:m+1
descent=alpha*distanceDC(:,k)'*cornerpoint(:,j)
+p*uncovered(:,1)'*cornerpoint(:,j);     //descent=gradient*search direction for each cornerpoint
if (descent<minval)
//testing, if cornerpoint is better than previous one (the first is always better than 100000000)
minval=descent;   //if the value for descent is better than minval, it is set to the new minimal value
g(:,k)=cornerpoint(:,j);     //the cornerpoint with the minimal value becomes the new search direction
end
end
f(:,k)=(1-xi)*f(:,k)+xi*g(:,k);
//flows are determined as a linear combination of the previous flow and the search direction
end
for j=1:m+1                      //flow values within each DC are updated
```

```
flow(j)=sum(f(j,:));
end
LBD1=0;
for k=1:n
for j=1:m+1
LBD1=LBD1+alpha*distanceDC(j,k)*f1(j,k);
end
end
for j=1:m
if ((b*flow1(j,1)-delta(1,j))>0);
LBD1=LBD1+p*b*flow1(j,1)-p*delta(1,j)
-p*delta(1,j)*log(flow1(j,1))+p*delta(1,j)*log(delta(1,j)/b);
end
end
distgrad=0;
for j=1:m
distgrad=distgrad+distanceDC(j,:)*g(j,:)';
end
distgrad2=0;
for j=1:m
distgrad2=distgrad2+distanceDC(j,:)*f1(j,:)';
end
LBD1=LBD1+p*b*sum(flow1(m+1,:))-(alpha*distgrad2+p*sum(uncovered'*f1))
+(alpha*distgrad+p*sum(uncovered'*g)); //determination of current lower bound
UBD1=0;
for k=1:n
for j=1:m+1
UBD1=UBD1+alpha*distanceDC(j,k)*f(j,k);
end
end
for j=1:m
if ((b*flow(j,1)-delta(1,j))>0);
UBD1=UBD1+p*b*flow(j,1)-p*delta(1,j)-p*delta(1,j)*log(flow(j,1))
+p*delta(1,j)*log(delta(1,j)/b);
end
end
UBD1=UBD1+p*b*sum(flow(m+1,:));  //determination of current upper bound
if(UBD1<UBD)                          //determination of the so far best upper bound
UBD=UBD1;
end
if(LBD1>LBD)                          //determination of the so far best lower bound
LBD=LBD1;
end
end                                  //end of main loop for t
GAP(i)=(UBD-LBD)/LBD;   //determination of the relative gap
c1=eta*sum(delta);                                    //overall costs for DC set-ups
c2=0;                             //initializations of uncovered demand
for j=1:m
if(b*flow(j)>delta(j))
c2=c2+(b*flow(j)-delta(j));
//determination of all people who get only a part of their demand at a DC
end
end
c2=c2+b*flow(m+1);
//adding of the uncovered demand of people who didn't leave their homes
//=overall uncovered demand
x=length(u(1,:));                     //current amount of Pareto optimal solutions
if ((c1==u(1,1)& c2<=u(2,1))|(c1<u(1,1))) then
u=[[c1;c2;i],u];
 //put current solution on first place if its first objective is better and it's not dominated
elseif(c1>u(1,x) & c2<u(2,x)) then
u=[u,[c1;c2;i]];
//put current solution on last place if its first objective is worse than all but it's not dominated
else
for l=2:x
if ((c1==u(1,l) & c2<=u(2,l))|(c1<u(1,l) & c2<u(2,l-1))) then
u=[u(:,1:l-1),[c1;c2;i],u(:,l:x)];
```

```
//if solution is not dominated, put it in the right order within the Pareto set,
// if better, break loop
break
end
end
end
x=length(u(1,:));                //output new amount of Pareto optimal solutions
for y=x:-1:1                              //delete all dominated solutions
if (c1<u(1,y)& c2<=u(2,y)) then
u(:,y)=[];
elseif (c1==u(1,y) & c2<u(2,y)) then
u(:,y)=[];
end
end
end                        //end loop for i DCs
cputime=timer()               //determine CPU time
u
plot2d(u(1,:),u(2,:),style=-1);          //plot Pareto front
xgrid(i=4);                              //add grid
xtitle('Pareto Front of DC Setups','Cost','Uncovered Demand');      //add labels
```

2. Code of the NC model for the instance Mbayene

```
timer();         //start counting time
demand=[363
281
108
154
199
235
200
369
108
146
223];                    //demands within villages
distance=[0 3262 2194 3261 2603 2971 3995 2637 2921 2580 2993
        3262 0   2324 3895 3986 3618 733  2767 4281 1396  269
        2194 2324  0 1571 1662 1294  3057  443 1957 1642 2055
        3261 3895 1571  0  657  290  4628 1129 952  3213 3626
        2603 3986 1662 657  0   368  4718 1219 318  3304 3717
        2971 3618 1294 290 368  0   4351  851  663  2936 3349
        3995 733  3057 4628 4718 4351 0   3500 5013 2129 1002
        2637 2767 443  1129 1219 851 3500  0   1514 2085 2498
        2921 4281 1957 952  318 663  5013 1514 0    3599 4012
        2580 1396 1642 3213 3304 2936   2129 2085 3599 0 1127
        2993  269 2055 3626 3717 3349 1002 2498 4012 1127 0];
n=length(demand);                    //number of villages
b=1;                                 //need per person
eta=20;                         //fixcost for the opening of a DC per 1 unit of capacity
u=ones(2,1)*100000000;                    //vector of Pareto optimal solutions
u=[u;0]                            //added index row to identify Pareto optimal locations
cap=zeros(n,1);                          //set-up of capacity vector
for k=1:n                                 //capacities of possible DCs in all villages
cap(k)=3*demand(k);
end
open=zeros(2^n-1,n); //initialization of indicator vector of all of DC set-ups per solution
open(1,n)=1;
w=n;
for i=1:2^n-2            //set-up of binary vector which is 1 if there is a DC in the village
if open(i,w:n)==ones(n-w) then
open(i+1,w-1)=1;
open(i+1,w:n)=zeros(n-w);
w=w-1;
else
open(i+1,w:n)=open(i,w:n);
for v=n:-1:w+1
if(open(i,v)==0)then
open(i+1,v)=1;
open(i+1,v+1:n)=zeros(n-v-1);
break
end
end
end
end
for i=1:2^n-1
m=nnz(open(i,:));                        //amount of opened DCs
z=zeros(n,n);          //initialization of indicator matrix of DC set-up possibilities
for k=1:n                     //indicator matrix where diagonal value=1, if DC is opened, else =0
if(open(i,k)>0) then
z(k,k)=1;
end
end
distanceDC=z*distance;            //leaves only the entries of distances to opened DCs, others=0
delta=z*cap;                           //capacities for DC which are open are >0, all others =0
test=ones(n,1);                          //testing vector
for k=n:-1:1
if(distanceDC(k,:)*test==0)     //testing, if the column of distances is one leading to a DC
distanceDC(k,:)=[];                 //if the column contains only zero-entries, they are erased
end
```

```
end
for k=n:-1:1
if(delta(k)==0)              //if village is not an opened DC...
delta(k)=[];                //...the entry of the capacity vector is erased
end
end
a=zeros(3,n);
//matrix with distances to chosen DCs in 1st entry...
//...demand of people who are willing to leave in 2nd (or all stay) and indices of DC in 3rd row
indexDC=0;      //initialization of index showing from which village to go to which DC
for k=1:n
minval=100000000;
//initialization of objective distance, which has to be minimized, with very large value
for j=1:m
if (distanceDC(j,k)<minval) then
minval=distanceDC(j,k);
 //the minval is chosen as the smallest distance from all opened DCs to village k in each iteration k
indexDC=j;     //the index of the chosen DC is saved
end
end
if(minval<=600)then
a(1,k)=minval*demand(k,1);
  //if the distance to the nearest DC is smaller than 6km, all people face this distance
a(2,k)=demand(k,1);          //all of the people will go, the demand for goods is a full one
a(3,k)=indexDC;                          //in the 3rd row, the index of the nearest DC is saved
elseif(minval>600)&(minval<=1500)then
a(1,k)=minval*demand(k,1)/2;       //if the distance to the nearest DC is between 6 and 15km...
//... it is assumed that only half of the people will leave their homes
a(2,k)=demand(k,1)/2;            //only half of the population will leave to visit a DC, thus....
//... only half of the people face the capacity
a(3,k)=indexDC;                  //the index of the chosen DC is saved
else
a(2,k)=demand(k,1);
 //if the distance is bigger than 15km, nobody will go, thus....
//... the index remains 0, the distance too, not fulfilled demand is saved
end
end
uncovered=zeros(1,m+1);              //initialization of vector of uncovered demand
demandinDC=zeros(1,m);               //all the people coming to one DC
for j=1:m
for k=1:n
if(a(3,k)==j)
demandinDC(1,j)=demandinDC(1,j)+a(2,k);
 //all people coming to this DC j (with same index j in 3rd row of a) are added to the demand in DC j
end
end
end
for j=1:m
 if (b*demandinDC(1,j)>delta(j)) then
uncovered(1,j)=b*demandinDC(1,j)-delta(j,1);
//the uncovered demand within a DC results from the gap between need of people and capacity
else
uncovered(1,j)=0;
 //if demand for goods doesn't exceed the capacity, the uncovered demand is zero
end
end
for k=1:n
//in the m+1 row, the uncovered demand of people who don't leave for a DC is saved (dummy DC)
if a(3,k)==0 then
uncovered(1,m+1)=uncovered(1,m+1)+b*a(2,k);
//if the distance is >15km (index==0), all people times their need for goods are added to the dummy DC
elseif (a(2,k)==demand(k,1)/2)then
 uncovered(1,m+1)=uncovered(1,m+1)+b*a(2,k);
 //if only half of the people don't leave their homes...
//... the other half times their need is added to the dummy DC
    end
end
```

```
c1=eta*sum(delta);          //overall costs for DC set-ups
c2=sum(uncovered(1,:));        //determination of overall uncovered demand
x=length(u(1,:));                    //current amount of Pareto optimal solutions
if ((c1==u(1,1)& c2<=u(2,1))|(c1<u(1,1))) then
u=[[c1;c2;i],u];
 //put current solution on first place if its first objective is better and it's not dominated
elseif(c1>u(1,x) & c2<u(2,x)) then
u=[u,[c1;c2;i]];
//put current solution on last place if its first objective is worse than all but it's not dominated
else
for l=2:x
if ((c1==u(1,l) & c2<=u(2,l))|(c1<u(1,l) & c2<u(2,l-1))) then
u=[u(:,1:l-1),[c1;c2;i],u(:,l:x)];
 //if solution not dominated, put it in the right order within the Pareto set, if better, break loop
break
end
end
end
x=length(u(1,:));             //output new amount of Pareto optimal solutions
for y=x:-1:1                          //delete all dominated solution
if (c1<u(1,y)& c2<=u(2,y)) then
u(:,y)=[];
elseif (c1==u(1,y) & c2<u(2,y)) then
u(:,y)=[];
end
end
end
cputime=timer();          //determine CPU time
plot2d(u(1,:),u(2,:),style=-2);        //plot Pareto front
xgrid(i=4);                        //add grid
xtitle('Pareto Front of DC Setups','Cost','Uncovered Demand');    //add labels
```

Lebenslauf

Schulische Ausbildung und Studium

09/1999 – 06/2007	AHS BG/BRG Leibnitz
21.06.2007	Matura mit ausgezeichnetem Erfolg
10/2007 – 09/2011	Bachelorstudium Mathematik an der KF Universität Graz
10/2009 – 09/2010	Erasmus-Aufenthalt an der TU Berlin
03.10.2011	Bachelorabschluss in Mathematik
10/2011 – 12/2013	Masterstudium Quantitative Economics, Management & Finance mit Schwerpunkt Operations Research an der Universität Wien

Berufliche Erfahrungen

seit 2006	Betreuerin bei Schreibwerkstätten, sowie Jurorin bei internationalen Schreibwettbewerben der Jugend-Literaturwerkstatt Graz
08/2012 – 09/2012	Praktikantin bei Ziviltechniker KG Daninger& Partner
10/2007 – 09/2011	Nachhilfelehrerin in Mathematik am IFL Graz
2007–2009	administrative Mitarbeiterin im Büro der Jugend-Literaturwerkstatt Graz

Sprachkenntnisse

Bosnisch/Kroatisch/Serbisch (Muttersprache), Deutsch (Muttersprache), Englisch (fließend in Wort und Schrift)

Französisch (fließend in Wort und Schrift), Spanisch (fortgeschritten)