universität wien

MASTERARBEIT / MASTER'S THESIS

Titel der Masterarbeit / Title of the Master's Thesis

„Equity and Deprivation Costs in Humanitarian Logistics by Gradient Search"

verfasst von / submitted by

Denisa Vlcekova, BSc

angestrebter akademischer Grad / in partial fulfillment of the requirements for the degree of

Master of Science (MSc)

Wien, 2019 / Vienna 2019

Studienkennzahl lt. Studienblatt / degree programme code as it appears on the student record sheet:	A 066 915
Studienrichtung lt. Studienblatt / degree programme as it appears on the student record sheet:	Masterstudium Betriebswirtschaft
Betreut von / Supervisor:	ao. Univ.-Prof. i.R. Dr. Walter Gutjahr

Contents

List of Figures ..... **iii**

List of Tables ..... **v**

List of Algorithms ..... **vi**

List of Abbreviations ..... **vii**

1. Introduction **1**

2. The Model **3**

2.1 The Model Constraints *5*

2.2 The Objective Function of the Model *6*

2.3 The Optimization Problem *7*

2.4 Power Functions as Deprivation Functions *7*

3. Evaluation of the Gradient of the Objective Function **9**

4. Overviews of used Optimization Algorithms **12**

4.1 The Gradient Descent *12*

4.2 The Frank-Wolfe Algorithm (FWA) *14*

4.2.1. Solution deprivation minimizing problem using FWA 16

4.3 The Bisection Method *17*

4.3.1 Performing the Bisection Method 19

5. The Application Case: The Nepal Earthquake 2015 ..... **20**

5.1 Data *21*

6. Numerical Experiments **24**

6.1 The Solution for the Utilitarian Case *24*

6.1.1 Qualitative Characteristics of the Utilitarian Solution 25

6.2 The Inequity-Averse Solution *30*

6.2.1 The Symmetric Initial Solution 31

6.2.1.1. Detecting Local Minimum 34
6.2.2. Sensitivity Analysis of η 36
6.2.3. Randomized Multistart 36

7. Conclusion 41

Bibliography 42

Appendix 43

A The Solution of the problem (15) using Karush-Kuhn-Tucker Conditions 43

B MATLAB Code 45

Abstrakt 49

Abstract 49

List of Figures

Figure 1: An example for the development of deprivation intensity over time for two different delivery intervals with p = 1, 2. 4

Figure 2: Application of the gradient descent algorithm to the example with two variables. 14

Figure 3: One step of the Bisection Method. 18

Figure 4: Map of Nepal Epicenters of the main earthquake (three circles) and of the major aftershock (two circles) [3]. 21

Figure 5: Map of the 14 most severely affected districts according to the definition by the government of Nepal [3]. 22

Figure 6: Gini Index and Relative Deprivation Cost in dependence of the exponent p of the DIF [3]. 26

Figure 7: (a) Left upper picture: Gini Index in dependence of the exponent p of the DIF solved by PSO [3]. (b) Right upper picture: GS-B with $tol = 10^{-3}$ and $maxit$ = 10,000. (c) Left lower picture: GS-B with $tol = 10^{-5}$ and $maxit$ = 10,000. (d) Right lower picture: GS-B with $tol = 10^{-8}$ and $maxit$ = 10,000. 27

Figure 8: (a) Left upper picture: Relative Deprivation Cost in dependence of the exponent p of the DIF solved by PSO [3]. (b) Right upper picture: GS-B with $tol = 10^{-3}$ and $maxit$ = 10,000. (c) Left lower picture: GS-B with $tol = 10^{-5}$ and $maxit$ = 10,000. (d) Right lower picture: GS-B with $tol = 10^{-8}$ and $maxit$ = 10,000. 28

Figure 9: (a) Left upper picture: Gini Index in dependence of the exponent p of the DIF solved by PSO [3]. (b) Right upper picture: GS-HSD with $tol = 10^{-3}$ and $maxit$ = 10,000. (c) Left lower picture: GS-HSD with $tol = 10^{-5}$ and $maxit$ = 10,000. (d) Right lower picture: GS-HSD with $tol = 10^{-8}$ and $maxit$ = 100,000. 29

Figure 10: (a) Left upper picture: Relative Deprivation Cost in dependence of the exponent p of the DIF solved by PSO [3]. (b) Right upper picture: GS-HSD with $tol = 10^{-3}$ and $maxit$ = 10,000. (c) Left lower picture: GS-HSD with $tol = 10^{-5}$ and $maxit$ = 10,000. (d) Right lower picture: GS-HSD with $tol = 10^{-8}$ and $maxit$ = 100,000. 30

Figure 11: (a) Left picture: The Objective Function in dependence of the number of iterations with *tol* = 10^{-3} and *maxit* = 1,000. (b) Right picture: The Objective Function in dependence of the number of iterations with *tol* = 10^{-8} and *maxit* = 10,000, where λ = 0.5. 35
Figure 12: (a) Left picture: Changes of two consecutive iterations with *tol* = 10^{-4} and *maxit* = 10,000, where λ = 0.5. (b) Right picture: The same changes of two consecutive iterations in detail. 35
Figure 13: The influence of the constant η in Harmonic Step Decrease with λ = 0.5 without stopping criterion..... 36
Figure 14: Tradeoff between μ and G(δ) for all three methods. 39
Figure 15: Tradeoff between λ and G(δ) for all three methods..... 40

List of Tables

Table 1: The number of beneficiaries ω_k, the parameter $\bar{c}_k$ and σ_k, and the costs d_k for all 14 districts. 23

Table 2: Comparison of G-HSD and G-B for the utilitarian solution. 25

Table 3: Numerical comparison of GS-HSD and GS-B with PSO for the inequity-averse case, where *tol* = 10^{-8} and *maxit* = 10,000..... 32

Table 4: Numerical comparison of GS-HSD and GS-B with PSO for the inequity-averse case, where *tol* = 10^{-8} and *maxit* = 200,000..... 33

Table 5: Numerical comparison of GS-HSD and GS-B with PSO for the inequity-averse case without stopping criterion. 34

Table 6: Numerical comparison of GS-HSD and the GS-B with PSO for the inequity-averse case for randomized multistart, where *k_max* = 10..... 37

Table 7: Numerical comparison of GS-HSD and GS-B with PSO for the inequity-averse case for randomized multistart, where *k_max* = 100. Note that for λ = 0.5, GS-HSD stopped at 2,159th iteration and GS-B at 10,000th. 39

List of Algorithms

1 The Gradient Descent 12
2 The Frank-Wolfe Algorithm (FWA) 14
3 The Bisection Method 17

List of Abbreviations

K	Set of Demand Points
k	Index denoting the Demand Point
ω_k	Number of Beneficiaries in each Demand Point k
N	Sum of all Beneficiaries
T	Time Horizon for Disaster Response Mission
τ_k	Delivery Interval in Location k
$g(t)$	Deprivation Intensity Function
t	Deprivation Time
δ_k	Average Deprivation Cost of an Inhabitant of Demand Node k per Time Unit
x_k	Number of Deliveries per Time Unit in Location k
y_k	Normalized Number of Deliveries per Time Unit in Location k
d_k	Cost for Covering Demand in Location k
c_k	Fixed Cost in Location k
γ_k	Variable Cost in Location k
B	Available Budget per Time Unit
μ	Average Deprivation Cost per Time Unit
λ	Inequity-Averse Penalization Parameter
Δ	Gini´s Mean Absolute Difference
G	Gini Index
p	Parameter defining Curvature of the Deprivation Intensity Function
$GS-B$	Gradient Search with the Bisection Method
$GS-HSD$	Gradient Search with the Harmonic Step Decrease
PSO	Particle Swarm Optimization

1. Introduction

The basic task of humanitarian logistics includes acquiring and delivering requested supplies and services, at the places and times they are needed, while minimizing the cost. During natural disasters, these supplies cover items that are essential for survival, such as food, water, temporary shelter and medicine, as defined in [1].

Many professional aid organizations only consider the total degree of demand satisfaction, but the concern that relief goods should be distributed as equally as possible among the affected area and people does often not find sufficient attention, as has been pointed out by some critical voices.

In works on humanitarian logistics with the aim of minimizing cost under budget constraints, the question of equity or fairness is usually neglected, although equity plays a significant role for decision making. Following the ethical idea, that every person has equal rights, nobody should be limited by imbalances in the supply of urgent goods.

This thesis on humanitarian logistics, builds on the recent article by Gutjahr and Fischer [3]. The authors used the deprivation cost concept by Holguín-Veras et al. [2] and modified this concept by a term proportional to the Gini inequity index in order to take into account equity. They performed the Particle Swarm Optimization metaheuristic for obtaining a solution.

We adopt the model and the assumptions as they were discussed in [3]. Our aim is to solve the logistics model by a gradient-based method and then compare the results with the Particle Swarm Optimization (PSO) metaheuristic from [3].

The main difficulty posed by the objective function is non-differentiability in a finite set of points due to the presence of absolute value. Because of this difficulty, we modified the component of the gradient in a suitable way in the present work. After that, the resulting nonlinear problem of the constrained minimization was solved by the Frank-Wolfe algorithm with two line search routines, the Bisection Method and the Harmonic Step Decrease.

The main contribution of this thesis is to investigate the use of the gradient that can increase precision of numerical calculations.

The thesis is organized as follows: The second chapter offers overview of the presented model and its features. Firstly, the inequity-neutral part of the model is introduced. Secondly, the extended problem formulation considering equity is presented. In the third chapter, we derive a formula for the evaluation of the gradient of the objective function, and reformulate the problem into its canonical form. The fourth chapter presents all used methods to obtain the results – the Gradient Descent, the Frank-Wolfe Algorithm, the Bisection Method and the Harmonic Step Decrease. The fifth chapter describes an illustration case with data from the Nepal earthquake 2015 used for the method evaluation. In the sixth chapter, we compute the solutions for both the utilitarian and the inequity-averse case using the Frank-Wolfe Algorithm. In the inequity-averse case, we perform the numerical analysis from the initial symmetric solution. To improve a performance of the method, a randomized multistart was considered.

2. The Model

The model is kept very simple and well-structured. We assume a single relief commodity. This also includes the case of a mix of different relief commodities for which the ratios between the requirements of each commodity are constant. It means that everyone gets the same package with food, water, medicine, etc. It also contains the “output” used to satisfy the nonmaterial needs, for example, medical services.

A periodic delivery scheme was assumed, because it would not be realistic to suppose that the total demand can be covered in a single delivery. Some of the reasons for that are as follows:

a) Vehicle capacities are limited.
b) Perishable goods cannot be stored by the consumers long-term.
c) Safe and sufficiently large local storage capacity may be lacking [3].

In this model, we consider relief commodities which need to be used within short time after supply. Such commodities are perishable food or medical help. Furthermore, if the relief commodity is not available, demand disappears and does not grow due to increased future supply. For example, if medication is not provided in one time unit, beneficiaries cannot get twice the amount of medication in the following time unit.

In the model is K the set of demand points k, that need to be served from a central place (“depot”). The number of receivers in each demand point k is $w_k \left(k \in K\right)$. The sum of the numbers of all receivers is denoted by $N = \sum_{k \in K} w_k$. It is supposed that the entire disaster response mission will take T time units. The delivery interval τ_k is the time spent between two consecutive deliveries. It is assumed that every visit of demand point k always fully satisfies the current demand in point k at given time.

The concept of deprivation costs with a deprivation intensity function $g(t)$ was used from Holguín-Veras et al. [2]. If it is considered that the commodity is food, the function $g(t)$ characterizes the intensity of hunger increasing with

deprivation time t. It follows that the deprivation intensity function (DIF) is always a nondecreasing function. This can easily be seen in the following example:

Let us consider the function $g(t) = 2t^p \left(t \in [0, \tau_k]\right)$ which will be defined in the interval $[0, \tau_k]$ because of the assumption that the delivery is periodic and every visit completely covers the current demand. As a result, the function $g(t)$ drops down to zero at $t = \tau_k$. We look at two different cases of the intervals with $p = 1, 2$. The dashed curves in Figure 1 plot the case with the delivery interval $\tau_k = 3$ which means that a delivery occurs every three days. The solid ones show the delivery interval $\tau_k = 1$. The different color of the curves demonstrates the function in dependence of the exponent p. The green curves display the case for $p = 1$, the blue ones for $p = 2$. As shown in Figure 1, if $\tau_k < 1$ then the deprivation intensity (at instant) is decreasing function of p. On the contrary, if $\tau_k > 1$ then the deprivation intensity (at instant) is increasing function of p.

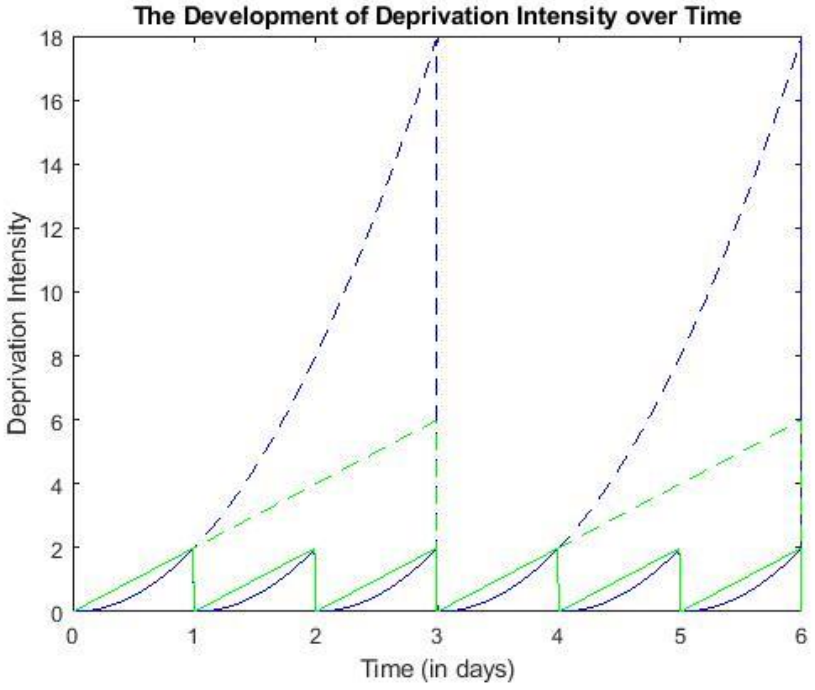

Figure 1: An example for the development of deprivation intensity over time for two different delivery intervals with *p* = 1, 2.

From the example, it is clear that the deprivation cost until time τ_k is the cumulated value of the deprivation intensity $g(t)$ in the interval $[0, \tau_k]$. That can

be mathematically formulated by the integral $\int_0^{\tau_k} g(t)$, which is nothing but as an increase between two deliveries. So the average deprivation cost of an inhabitant of demand node k per time unit can be defined as follows:

$$\delta_k = \frac{1}{\tau_k} \int_0^{\tau_k} g(t)\, dt\,. \tag{1}$$

The decision variable $x_k \in \mathbb{R}^+$ indicates the average number of deliveries per time unit. In other words, it is the distribution frequency of aid for location k. After the setting $x_k = \frac{1}{\tau_k}$, the Equation (1) can be reformulated as:

$$\delta_k = x_k \int_0^{1/x_k} g(t)\, dt\,. \tag{2}$$

In addition to that, it is assumed that the cost to cover the demand in location k is:

$$d_k = c_k + \gamma_k(\omega_k) \qquad k \in K\,, \tag{3}$$

where c_k are the fixed costs, and $\gamma_k(\omega_k)$ are the variable costs. Moreover, it is considered that the variable costs γ_k are a nondecreasing function of the number of receivers ω_k in location k.

2.1 The Model Constraints

At this point, it is easy to describe the delivery cost to demand location k during the whole mission as $d_k T x_k$, as well as the total logistics cost as $T\sum_{k\in K} d_k x_k$. What is more, there is available budget B per time unit. It means that the budget for logistic costs is BT. This produces the constraint:

$$T\sum_{k\in K} d_k x_k \le BT \tag{4}$$

And dividing by T we get a simplified expression:

$$\sum_{k\in K} d_k x_k \le B \tag{5}$$

2.2 The Objective Function of the Model

The objective function of the problem is the following

$$f(\mathrm{x}) = \mu + \lambda\Delta, \tag{6}$$

where $\mu = \frac{1}{N}\sum_{k\in K}\omega_k\delta_k = \frac{1}{N}\sum_{k\in K}\omega_k x_k \int_0^{1/x_k} g(t)\,dt$, $\Delta = 2\mu G$, G denotes the Gini Index of deprivation cost and λ is the parameter penalizing for inequity. This formulation is analogous to [3]. In our case, μ is measured in the units of the deprivation scope. The parameter λ regulates the influence of fairness in the objective function and can be tuned between 0 and $1/2$. The boundary case $\lambda = 0$ means that there will be no consideration of equity. Therefore, the penalty term will not be taken into account and we get the utilitarian optimization problem. The higher λ is, the greater the consideration of the equity is.

The Gini Index G is a statistical measure of distribution. The coefficient ranges from 0 (or 0%) to 1 (or 100%), with 0 representing perfect equality (everyone has the same share of goods) and 1 representing perfect inequality (only one person accounts for all the consumption, and all the others have none).

For losses a_n assigned to single persons $n = 1,...,\mathrm{N}$, the Gini Index is defined as

$$G = \frac{1}{2N\sum_{n=1}^{N} a_n}\sum_{n=1}^{N}\sum_{n'=1}^{N}\left|a_n - a_{n'}\right|. \tag{7}$$

In our case, where the losses are deprivation costs δ_k, aggregation over the inhabitants of a demand point transforms the Gini Index (9) into

$$G = \frac{1}{2\mu N^2}\sum_{k\in K}\sum_{k'\in K}\omega_k\omega_{k'}\left|\delta_k - \delta_{\mathrm{k}'}\right| \tag{8}$$

The objective function (6) with Gini Index (10) takes the form

$$f(\mathrm{x}) = \frac{1}{N}\sum_{k\in K}\omega_k x_k \int_0^{1/x_k} g(t)\,dt + \lambda 2\mu\frac{1}{2\mu N^2}\sum_{k\in K}\sum_{k'\in K}\omega_k\omega_{k'}\left|x_k\int_0^{1/x_k} g(t)\,dt - x_{k'}\int_0^{1/x_{k'}} g(t)\,dt\right| \tag{9}$$

2.3 The Optimization Problem

Combining the objective function (6) with constraint (5) and with non-negativity of x, results in a representation of the optimization problem:

$$\begin{aligned} \min_{x} \quad & \mu + \lambda 2\mu G \\ s.t. \quad & \sum_{k \in K} d_k x_k \le B \\ & x_k \ge 0 \qquad \forall k \in K \end{aligned} \tag{P_λ}$$

If $\lambda = 0$, then we obtain the minimization of the average deprivation costs per time unit without the consideration of the penalty term. It is also called the utilitarian optimization problem. It is given by the following nonlinear mathematical programme:

$$\begin{aligned} \min_{x} \quad & \mu \\ s.t. \quad & \sum_{k \in K} d_k x_k \le B \\ & x_k \ge 0 \qquad \forall k \in K \end{aligned} \tag{P_0}$$

2.4 Power Functions as Deprivation Functions

We assume a special case of deprivation intensity functions

$$g(t) = g_1 t^p, \tag{10}$$

where $p > 0$. Since $g''(t) = g_1 p(p-1)t^{p-2}$ and $t \ge 0$, a deprivation intensity function given by (12) is convex if $p > 1$ and concave if $0 < p < 1$. The case $p = 1$ produces the linear special case. Since for $g(t)$ as above,

$$\int_0^{1/x} g(t)\,dt = g_1 \int_0^{1/x} t^p dt = \frac{g_1}{(p+1)x^{p+1}}, \tag{11}$$

and it is

$$\delta_k = \frac{g_1}{(p+1)x_k^p}, \tag{12}$$

$$\mu = \frac{g_1}{N\left(p+1\right)} \sum_{k \in K} \frac{\omega_k}{x_k^p} \tag{13}$$

and

$$G = \frac{g_1}{2N^2(\mathrm{p}+1)\mu} \sum_{k,k' \in K} \omega_k \omega_{k'} \left| \frac{1}{x_k^p} - \frac{1}{x_{k'}^p} \right|. \tag{14}$$

After omission of constant factors in the objective function, the utilitarian optimization problem becomes

$$\begin{aligned} \min_x \quad & \sum_{k \in K} \omega_k x_k^{-p} \\ s.t. \quad & \sum_{k \in K} d_k x_k \leq B \\ & x_k \geq 0 \qquad \forall k \in K . \end{aligned} \tag{15}$$

This optimization problem can easily be solved by the Karush-Kuhn-Tucker conditions (see Appendix A).

3. Evaluation of the Gradient of the Objective Function

In a closer analysis of the objective function, it has to be noted that the part with the Gini Index is not differentiable because of the absolute value, while the first part, the minimization of the average deprivation costs, is easy to differentiate. This means that it is not easy to find an optimal solution as in the case of the utilitarian problem.

In this section, we will explain in detail how the absolute value is removed by using several reformulations and by introducing new auxiliary variable σ and $\bar{\sigma}$.

Firstly, the objective function is divided into two parts:

a) the average deprivation costs per time unit → differentiable

$$\mu = \frac{g_1}{N(p+1)} \sum_{k \in K} \frac{\omega_k}{x_k^p}$$

b) the average deprivation costs per time unit multiplied by the Gini Index → not differentiable

$$\mu G = \frac{g_1}{2N^2(p+1)} \sum_{k,k' \in K} \omega_k \omega_{k'} \left| \frac{1}{x_k^p} - \frac{1}{x_{k'}^p} \right|$$

We can replace the constraint (5) "smaller or equal" by "equal", $\sum_{k \in K} d_k x_k = B$. Thus, the optimal solution fully consumes the available budget. Inequity can be decreased by using the extra resources to satisfy the needs of the currently disadvantaged. Then, we apply the substitution using the new variable: $y_k := \frac{d_k}{B} x_k$, which transforms (5) into $\sum_{k \in K} y_k = 1$. Using this variable, optimization will be over the standard simplex. As a result, the total deprivation costs function looks like

$$\mu G = \frac{g_1}{2N^2(p+1)B^p} \sum_{k,k' \in K} \omega_k \omega_{k'} \left| \frac{d_k^p}{y_k^p} - \frac{d_{k'}^p}{y_{k'}^p} \right|, \tag{16}$$

where

$$\overline{C}_p := \frac{g}{2N^2(p+1)B^p}. \tag{17}$$

Further, we continue with the b) to remove the absolute value. We can see that there are two elements k and k', which belong to the set K. The set K^2 can be divided in three disjoint sets:

- $K_= := \{(k,k') \in K^2 \,|\, k = k'\}$
- $K_> := \{(k,k') \in K^2 \,|\, k > k'\}$
- $K_< := \{(k,k') \in K^2 \,|\, k < k'\}$

When we consider double summation in (16), terms from the set K_0 sum up to zero and terms from $K_<$ can be counted twice due to the symmetry between $K_>$ and $K_<$, based on the property of the absolute values – evenness (the reflection symmetry of the graph). The remaining terms in the double summation can be divided into three subsets:

- $K^+ := \left\{(k,k') \in K_< \,\middle|\, \frac{d_k^p}{y_k^p} - \frac{d_{k'}^p}{y_{k'}^p} > 0\right\}$,
- $K^- := \left\{(k,k') \in K_< \,\middle|\, \frac{d_k^p}{y_k^p} - \frac{d_{k'}^p}{y_{k'}^p} < 0\right\}$,
- $K^0 := \left\{(k,k') \in K_< \,\middle|\, \frac{d_k^p}{y_k^p} - \frac{d_{k'}^p}{y_{k'}^p} = 0\right\}$.

When evaluating (16) in general, $K^0 = \varnothing$. If that is not the case, we need to change the current search point slightly. This is necessary for a simple reason: the case $K^0 \neq \varnothing$ is not possible to differentiate**.**

Moreover, we use the auxiliary function σ to include the both cases of summation (K^+, K^-) in b):

$$\mu G = 2\overline{C}_p \sum_{(k,k') \in K_<} \sigma_{kk'} \omega_k \omega_{k'} \left| \frac{d_k^p}{y_k^p} - \frac{d_{k'}^p}{y_{k'}^p} \right|,$$

where

$$\sigma_{kk'} := \begin{cases} 1, & (k,k') \in K^+ \\ -1, & (k,k') \in K^- \end{cases}.$$

After that, we compute the partial derivation of b) with respect to y_l. We distinguish two different cases:

- $l = k$
- $l = k'$

$$\frac{\partial(\mu G)}{\partial y_l} = -2\overline{C}_p \sum_{k'} \sigma_{lk'} \omega_l \omega_{k'} d_l^p p y_l^{-p-1} + 2\overline{C}_p \sum_{k} \sigma_{kl} \omega_k \omega_l d_l^p p y_l^{-p-1} = 2\overline{C}_p \sum_{k \in K} \bar{\sigma}_{lk} \omega_l \omega_k d_l^p p y_l^{-p-1}$$

Finally, we define a new auxiliary function $\bar{\sigma}$ to simplify the notation:

$$\bar{\sigma}_{lk} := \begin{cases} -\sigma_{lk}, & l < k \\ \sigma_{kl}, & l > k \\ 0, & l = k \end{cases}.$$

4. Overviews of used Optimization Algorithms

In this section, we give an overview of used optimization algorithms:

- The Gradient Descent
- The Frank-Wolfe Algorithm (FWA)
- The Bisection Method

4.1 The Gradient Descent

One of the points of our thesis is the use of the gradient descent in the above mentioned problem. We try to find out the effect of the method on the results and its quality.

The gradient method, also called the steepest descent method, is a method used in the numeric analysis to solve general unconstrained optimization problems. Moreover, the gradient descent is a first-order iterative optimization algorithm for finding the minimum of a function. The gradient descent algorithm can be applied for solving the following problem:

$$\min \quad f(x) \qquad x \in \mathbb{R}^n \tag{18}$$

with a continuous differentiable function $f(x)$.

The gradient of the function $f(x)$ will be denoted as $g(x) = \nabla f(x)$, $g_k = g(x_k)$ with the direction $s_k = -\nabla f(x_k) = -g_k$, and the step length α_k.

Inputs: - The objective function $f(x)$

- The initial point x_0
- The tolerance constant $tol > 0$
- Setting the iterator counter $k = 0$

Algorithm: 1) Test the accuracy. If $\|\nabla f(x_k)\| < tol$, stop.

2) Calculate the direction s_k.

3) Compute the step length α_k.

4) Determine the new approximation $x_{k+1} = x_k + \alpha_k s_k$.

5) Repeat the cycle $k := k+1$ and go to 1).

Now, we will consider the specific example with two variables and the optimal step length and use the previous steps from the algorithm.

Inputs: - The function to minimize: $f(x,y) = 2x - y + 2x^2 - 2xy + y^2$

- Initial points: $x_0 = -2$, $y_0 = -5$

- The tolerance constant: $tol = 0.1$

Application of the algorithm:

1) Enumerate the norm of gradient $\|\nabla f(x_0)\| = 3.59$ which is more than our $tol = 0.1$.
2) Calculate the direction $s_0 = (3.12, 1.78)$.
3) Compute the step length $\alpha_0 = 0.24$.
4) Determine the next approximation
   $x_1 = (-2,-5) + 0.24(3.12,1.78) = (-2.95,-3.34)$.
5) Repeat the cycle $k := 1$ and go to 1).

Iterations and contours of the objective function from the example above are shown in Figure 2. Note that gradients and level sets are perpendicular to each other.

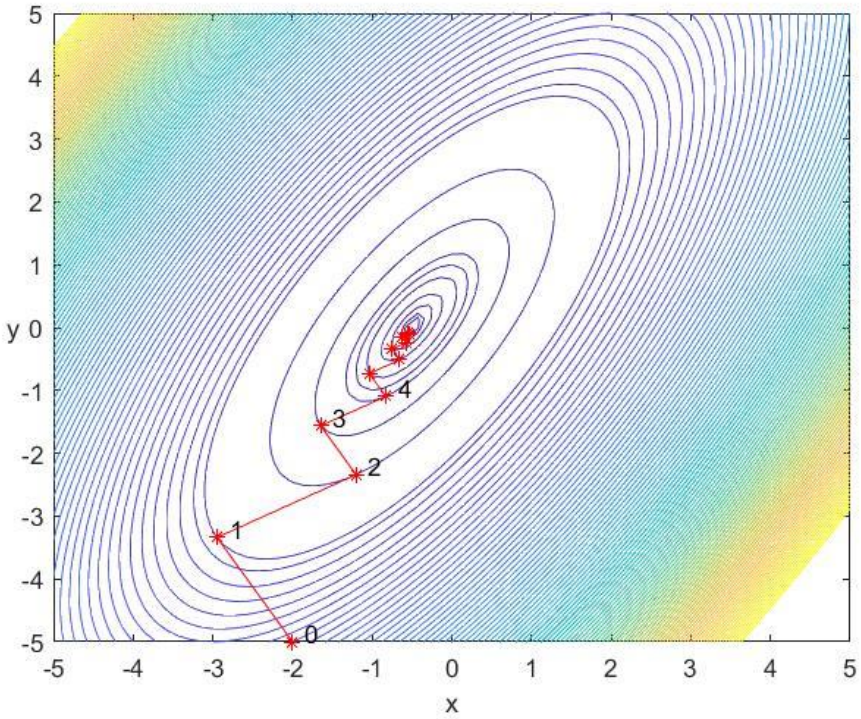

Figure 2: Application of the gradient descent algorithm to the example with two variables.

We have a nonlinear constrained optimization problem. Because of this fact, we will use the Frank-Wolfe algorithm, which includes the idea of the gradient descent.

4.2 The Frank-Wolfe Algorithm (FWA)

The Frank-Wolfe method was originally published by Marguerite Frank and Philip Wolfe [4] in the 1950´s. They applied the algorithm to solve the problem of minimization of a convex quadratic function under linear constraints.

In general, it can be used for nonlinear constrained optimization. That is why the algorithm is appropriate - our objective function is nonlinear with linear constraints. The Frank-Wolfe is an iterative first-order optimization algorithm. First, there is an initial point x_0. Then, it constructs a sequence of estimates $x_1, x_2, ..., x_n$ that converges to the optimal solution. The algorithm can solve problems of the form

$$\min \quad f(y)$$

$$s.t. \qquad y \in S,$$

where $f:\mathbb{R}^n \mapsto \mathbb{R}$ is a continuously differentiable function, over a convex[1] and compact[2] set $S \subset \mathbb{R}^n$. More information about why the function is defined in this way can be found in [5]. FWA approximates the objective function by first two terms of Taylor expansion around y_k:

$$f(\xi) \approx f(y_k) + \nabla f(y_k)^T (\xi - y_k).$$

Then, FWA searches for a solution of the approximated problem

$$\min_{\xi} \quad f(y_k) + \nabla f(y_k)^T \xi - \nabla f(y_k)^T y_k$$

$$s.t. \quad y \in S$$

$$\Leftrightarrow$$

$$\min_{\xi} \quad \nabla f(y_k)^T \xi$$

$$s.t. \quad y \in S$$

When the set S is a polyhedron[3], then each step of FWA is reduced to solve the linear programme.

Inputs: - The objective function $f(y)$

- The set of feasible solutions $y \in S$
- The initial point $y_0 \in S$
- The tolerance constant for the stopping criterion $tol > 0$

Algorithm: 1) Choose an initial solution $y_0 \in S$. Let $k := 0$.

2) Calculate a direction s_k.

3) Perform the line search to find a step length α_k, such that

$$f(y_k + \alpha_k s_k) < f(y_k).$$

4) Determine a new iteration point: $y_{k+1} = y_k + \alpha_k s_k$.

5) Check the stopping criterion. If it is fulfilled → Stop! Otherwise, let $k := k+1$, and go to 2).

---

[1] A set is called convex if one for which any segment between two points lies within the set. While the FW algorithm does not require the objective function f to be convex, it does require the set of feasible solutions to be a convex.

[2] For any subset A of Euclidean space $\mathbb{R}^n$, A is compact if and only if it is closed and bounded.

[3] Definition of a polyhedron: $P = \{x \in \mathbb{R}^n : Ax \le b\}$, where A is m x n matrix and b is the vector of length m.

4.2.1. Solution deprivation minimizing problem using FWA

At this point, we apply the algorithm to our problem using [10]:

$$\begin{aligned} &\text{minimize} && f(y) = \mu + \lambda 2\mu G \\ &\textit{subject to} && y \in S \ , \end{aligned}$$

where

$$S = \left\{ y \middle| \sum y_i = 1, y_i \geq 0, \quad \forall i \in K \right\}.$$

We can prove that the set of feasible solutions S is polyhedron:

$$S = \left\{ \mathrm{y} \middle| v^T y = 1, \mathrm{y}_k \geq 0, \quad \forall k \in K \right\} \ \textit{where} \ \ v = (1,1,...,1)^T$$

$$\Leftrightarrow S = \left\{ \mathrm{y} \middle| v^T y \leq 1 \wedge v^T y \geq 1, \mathrm{y}_k \geq 0, \quad \forall k \in K \right\}$$

$$\Leftrightarrow S = \left\{ \mathrm{y} \middle| v^T y \leq 1 \wedge -v^T y \leq -1, -\mathrm{y}_k \leq 0, \quad \forall k \in K \right\},$$

$$S = \left\{ y \middle| \begin{pmatrix} v^T \\ -v^T \\ -I_{|K|} \end{pmatrix} y \leq \begin{pmatrix} 1 \\ -1 \\ 0_{|K|} \end{pmatrix} \right\}.$$

1) Choose an initial solution $y_0 \in S$. Let $k := 0$.

2) Calculate a search direction s_k.

   The Frank-Wolfe algorithm approximates the objective function by the first two terms of Taylor expansion and search for a solution, s_k, of approximated
   problem (19).

$$\begin{aligned} &\min_{\xi_k} && \nabla f(y_k)^T \xi_k \\ &\textit{subject to} && \xi_k \in S \end{aligned} \tag{19}$$

   This is an LP problem. The solution of such a problem can be found using the Simplex Method [7]. The search direction is $s_k = \xi_k - y_k$, that is, the vector from the feasible point y_k to the solution of partial problem (19). Observe that this is a feasible direction, because both y_k, ξ_k belong to S and S is convex. Since S is unit simplex/polyhedron, the optimal solution

of (19) will be in one of the vertices of the form $e_i = (0,...,0,1,0,...,0)^T$, where all components are equal to 0, except the i^{th}, which is 1.

Let $\left[\nabla f(y_k)\right]_j$ be the smallest element of $\nabla f(y_k)$ the solution of approximated problem (19) will be e_j.

3) Solve the line search problem to find a step length α_k, such that

$$f(y_k + \alpha_k s_k) < f(y_k).$$

Here, we must limit the step length to be at most 1, because for $\alpha_k > 1$ the solution becomes infeasible; the line search problem therefore has the form

$$\underset{\alpha_k \in [0,1]}{\text{minimize}} \qquad f(y_k + \alpha_k s_k) \tag{20}$$

a) We will solve (20) using the Bisection Method.

b) As an alternative approach we can set $\alpha_k = \frac{\eta}{k+2}$, where η is a constant. We will call this method the Harmonic Step Decrease.

4) Determine a new iteration point: $y_{k+1} = y_k + \alpha_k s_k$.

5) If a stopping criterion is fulfilled → Stop! y_{k+1} is the approximation of the optimal solution. Otherwise, let $k := k+1$, and go to 1).

In our case, we select the following stopping criterion: $\|\alpha_k s_k\|_2 < tol$. This means if we move less than the given tolerance, then stop the algorithm.

4.3 The Bisection Method

In this part, we describe the Bisection Method and its features. The Bisection Method also called the interval halving method, the binary search method, or the dichotomy method, is a simple root finding method. It reduces an interval that includes a roof of the function $f(x)$. The method is based on the theorem called Bolzano's theorem [9], which states that if the values of $f(a)$ and $f(b)$ have opposite signs, the interval must contain at least one root.

Theorem (Bolzano): *If a function* $f(x)$ *is continuous on an interval* $[a,b]$ *and* $f(a) * f(b) < 0$*, then a value* $c \in (a,b)$ *exist for which* $f(c) = 0$*.*

The proof can be found in [9].

If $f(a)$ and $f(b)$ have different signs, then $f(a)*f(b)<0$. This means that one of them is above the $x-axis$ and the other one below the $x-axis$. In this case, if we sketch the $f(x)$ function, at some point, it will cut the *x*-axis. The x value, where the $x-axis$ is cutting, is the root of the equation $f(x)=0$. The Figure 3 clarifies the one-step scheme of the Bisection.

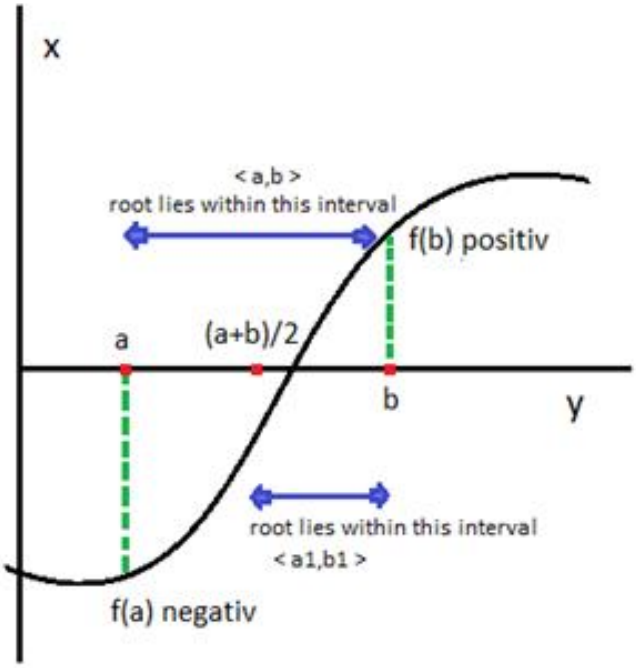

Figure 3: One step of the Bisection Method.

Based on the above information, we can summarize the following:

Inputs: - $a\in\mathbb{R}$, $b\in\mathbb{R}$ such that $f(a)*f(b)<0$

Algorithm: 1) Compute the midpoint c of the given interval using formula: $c=(a+b)/2$

2) Determine the function value at the midpoint $f(c)$.

3) Stop the algorithm if $f(c)=0$ and c is the root.

4) Set $a:=c$ if $f(b)*f(c)<0$, and set $b:=c$ if $f(b)*f(c)>0$.

5) Go to 1).

4.3.1 Performing the Bisection Method

This method is easy to implement. It begins by calculating the midpoint, $x\_help = \frac{a+b}{2}$, of the interval. The function is then evaluated at that point $f(s)$. Recalling Bolzano's theorem, if $f(a)$ and $f(x\_help)$ have different signs, the method replaces b with the calculated midpoint, or if $f(b)$ and $f(x\_help)$ have different signs, a is replaced by the midpoint. This step guarantees that there is still a root within the interval. The procedure then continues to the next iteration. The solution is found when the function is equal to 0 at $f(x\_help)$ or it is small enough to be sufficient.

Before implementing this method into the programme MATLAB, it is necessary to determine the number of iterations in the Bisection Method. This can easily be deduced from the following inequality $\left(\frac{1}{2}\right)^n (b-a) < tolerance$, where n stands for the number of iteration.

How close we will get to the real root depends on the value of the tolerance we set for the algorithm. The disadvantage of this method is that it is relatively slow.

5. The Application Case: The Nepal Earthquake 2015

As an application case we chose the same illustrative example as in [3]. The example is based on data from the Nepal earthquake of 2015. The selection is not random. Although the rescue teams responded immediately, it was criticized that the most supplied areas were the best accessible ones. This resulted in significant discrepancies of supplied amounts among affected regions. Johnsson [6] declares that “the material relief aid was distributed unevenly between the crisis-hit districts. The disproportionate distribution among districts was likely related to the geographic accessibility of the districts, as many of the districts that received comparably fewer relief items were mainly located in the mountainous regions of northern Nepal”. This indicates that the distribution of help was strongly influenced by geographical reachability. Therefore, Fischer and Gutjahr [3] decided to take into account the fairness using the Gini Index. They showed that in this way the results can be considerably improved and allows a fair distribution of aid. On the basis of their observation, we are trying to find out whether the use of the gradient descent method for the same problem will also produce favorable results or even better ones.

On April 25, 2015, an earthquake of magnitude 7.8 (the main shock) hit large parts of Nepal. More than 8,800 people died and about 22,000 were hurt. Many houses were damaged and hundreds of thousands of people became homeless. Several days later, on May 12, 2015, the main shock was followed by a major aftershock of magnitude 7.3. The Figure 4 illustrates the map of Nepalese districts, and the epicenters. The rescue mission was established primarily by the Nepalese government and was operated by the Nepalese army. Furthermore, not only other countries, but also a lot of international non-governmental organizations provided support. Most rescue actions were organized from the capital of Kathmandu.

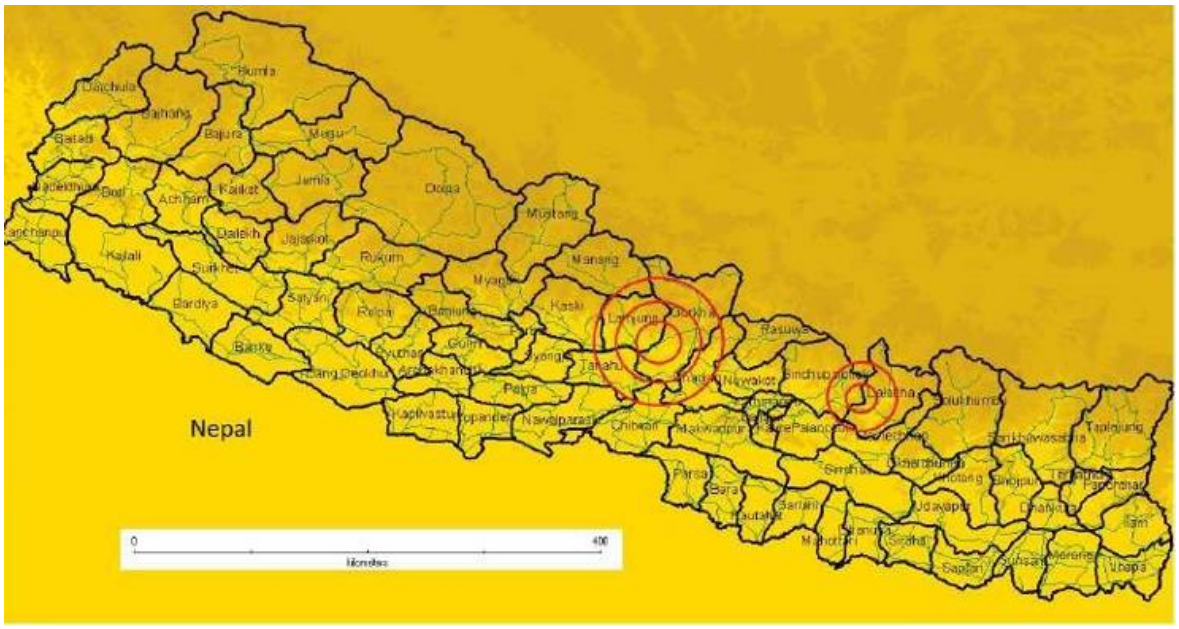

Figure 4: Map of Nepal Epicenters of the main earthquake (three circles) and of the major aftershock (two circles) [3].

5.1 Data

In order to get a well-structured illustration example, we assume that mobile teams are created in the capital and are ready to go to support other affected districts according to a specific feature of the Nepal earthquake 2015. During the earthquake many public health facilities were destroyed or partially damaged. The teams visit the districts regularly, staying one day with district-specific frequencies x_k, where k is the index of district. During their visit in a district, they help not only in the central location and provide the health care, but also at home if necessary. The attention is mainly focused on the 14 districts that the Nepalese government has stated to be the regions in the biggest need. On the one hand, some of the regions are easily accessible, but on the other hand there are several districts that are located in the mountain area and the access to them is very complicated. Figure 5 shows the variety of landscape as well as the 14 most severely affected districts.

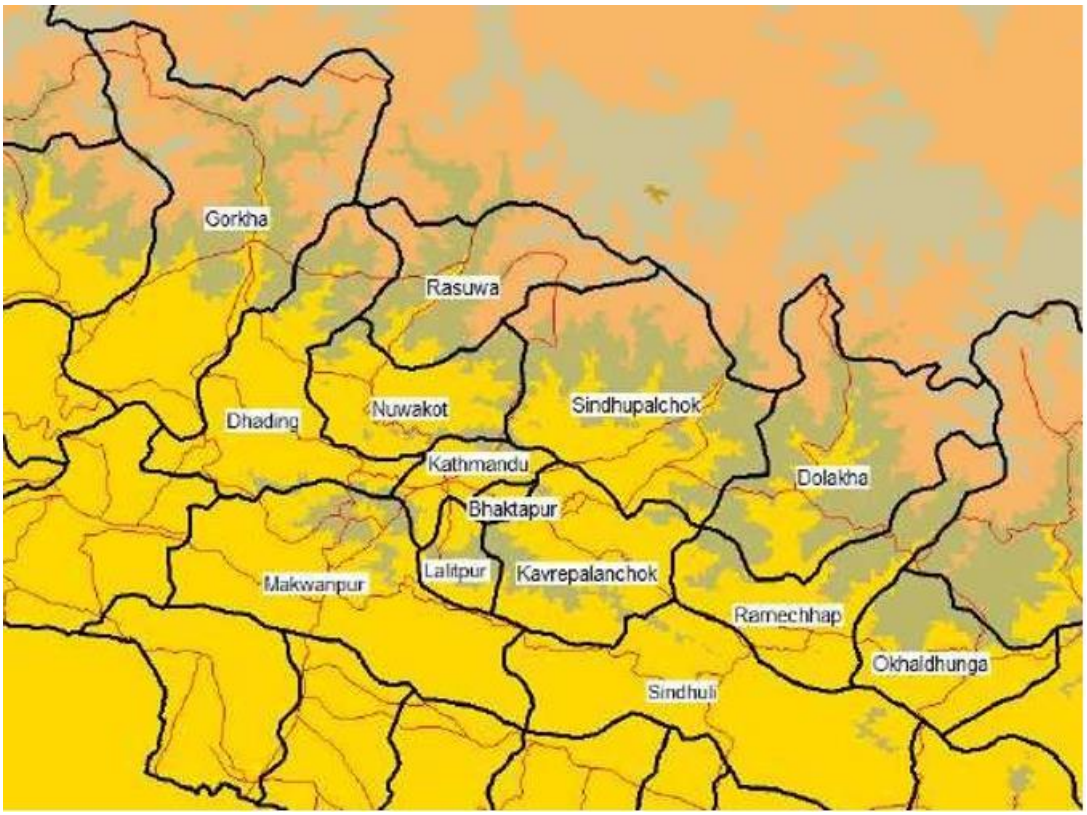

Figure 5: Map of the 14 most severely affected districts according to the definition by the government of Nepal [3].

We will use the data from Johnsson [6]. We consider the following parameters:

- K is the set of the 14 selected districts.
- ω_k (the number of beneficiaries) is equal to the number of the Health Centre fully or partially destroyed in the district k.
- d_k are the costs to cover the demand in the district k by a single visit:

$$d_k = c_k + \mathrm{X}\left(1+\beta\left(\tan\sigma_k\right)^{\rho}\right)\omega_k,$$

  where $c_k = \bar{c}_k + c_{bas}$ and it represents the distance.
- $\bar{c}_k$ is equal to the distance between Kathmandu and the main city of the district k in kilometers.
- c_{bas} is the base value which is added to the distance from/to the capital.
- σ_k is the average geographical slope (in degrees) in the district k.
- X defines the variable costs per a destroyed Health Centre (in kilometers).
- β indicates a weighting factor for the increased costs of a home visit compared to the health care at the central location.

- ρ describes the power factor of increasing unavailability for the tangent of the mean geographic slope σ_k.

The parameters c_{bas}, X, β and ρ are constant and in our model we suppose:

- $c_{bas} = 10$,
- $\mathrm{X} = 2$,
- $\beta = 200$,
- $\rho = 3$.

In terms of travel costs, this means that the base value of 10 kilometers is added to the distance from/to the capital. The variable costs to satisfy one Health Centre are equal to the round trip journey costs for 2 kilometers of distance. The inaccessibility of an region increases with the third power of the tangent of the slope. The cost effort for home visits is higher by 13.4%, if an average slope is 5° compared to an average slope of 0°.

The values for ω_k, $\bar{c}_k$ and σ_k according to Johnsson [6]:

District	ω_k	$\bar{c}_k$	σ_k	d_k
Bhaktapur	25	16	3.98	79.40
Dhading	106	90	10.79	605.50
Dolakha	83	183	13.86	857.70
Gorkha	79	144	15.70	1,013.80
Kathmandu	63	0	4.79	150.80
Kavrepalanchok	131	38	8.62	492.50
Lalitpur	39	7	8.12	140.30
Makwanpur	59	88	5.89	241.90
Nuwakot	99	61	10.39	513.10
Okhaldhunga	30	217	10.67	367.30
Ramechhap	66	154	11.14	497.60
Rasuwa	27	87	16.88	452.80
Sindhuli	59	134	5.57	283.90
Sindhupalchok	97	67	13.20	771.60

Table 1: The number of beneficiaries ω_k, the parameter $\bar{c}_k$ and σ_k, and the costs d_k for all 14 districts.

6. Numerical Experiments

We use the programme MATLAB R2017b to get results for the utilitarian solution as well as results for the inequity-averse solution. The programme was started on the computer Lenovo T400 with processor Intel Core 2 Duo CPU P8400 and 4 GB RAM. All MATLAB´s codes are in the appendices.

The Gradient Search (GS) has several parameters that can have a major impact on results, for example, the choice of the step calculation – the Bisection Method (GS-B) or the Harmonic Step Decrease (GS-HSD). The tolerance constant (*tol*) stands for a certain number of decimal numbers in results. Furthermore, the parameter *maxit* represents the maximum iteration count needed to stop the method if it does not find the solution earlier. And the constant τ describes the starting step size for the GS-HSD. We will check them in a closer analysis in the case of the utilitarian solution as well as in the inequity-averse case.

- Methods: GS-B, GS-HSD.
- *tol*: tolerate the result for a certain number of decimal numbers.
- *maxit*: maximum iteration count.
- η: starting step size for the GS-HSD.

We set the parameter in the deprivation intensity function (12) as follows: $g = 2$.

6.1 The Solution for the Utilitarian Case

In this section, we will consider only the case $\lambda = 0$. It means that we minimize the average deprivation costs per time unit without the consideration of the penalty term. We can check the correctness of our results by comparing them with the results from [3]. Although two different methods are used (Gradient Descent, Particle Swarm Optimization), in this case both approaches should yield the same results. Our aim is to replicate the results from [3] concerning the exponent $p = 2$.

We consider an initial solution $y_0 = \frac{\mathbf{1}}{|K|}$, where $\mathbf{1}$ is a vector of 1, which means that the aid is distributed equally to each location. The value of $\eta = 2$ was used for calculations done by GS-HSD.

Table 2 shows the overview of algorithm runs for both, GS-B and GS-HSD, with the tolerance set to 10^{-3} and 10^{-5}. The k stands for the number of iterations in

which the methods stops, the t is an execution time of the algorithm, the |μF-μ| is the absolute difference between average deprivation costs μ_F from [3] and obtained μ and the $|\mu_F-\mu| / \mu_F$ is the relative difference against μ_F.

When we compare both the methods, we can observe differences in performance. Using weak tolerance criteria applied, the results of the GS-B are much worse than those obtained with the GS-HSD. But if we toughen the tolerance, we can see that the GS-B is faster and matches the GS-HSD precision. It is interesting to note that the average iteration of the GS-B takes 125 times longer than iteration of the GS-HSD.

GS-HSD	**tol = 10^{-3}**	**tol = 10^{-5}**	**tol = 10^{-8}**
k	1,822	182,144	1,000,000
t [s]	0.4877	21.4637	114.2054
$\|\mu_F-\mu\|$	7.9782E+03	554.2328	553.9534
$\|\mu_F-\mu\| / \mu_F$ [%]	0.0311	0.0022	0.0022
GS-B	**tol = 10^{-3}**	**tol = 10^{-5}**	**tol = 10^{-8}**
k	130	1,818	4,322
t [s]	1.1147	18.3737	61.4564
$\|\mu_F-\mu\|$	4.5741E+05	622.6194	553.9474
$\|\mu_F-\mu\| / \mu_F$ [%]	1.7847	0.0024	0.0022

Table 2: Comparison of G-HSD and G-B for the utilitarian solution.

6.1.1 Qualitative Characteristics of the Utilitarian Solution

The following Figure 6 displays two pictures. The left one describes the Gini Index of the deprivation costs (red) and the Gini Index of the visiting frequencies (blue) depending on the exponent p of the deprivation intensity function (12). The right one shows the worst-off districts (upper), the best-off (lower) districts and the average deprivation costs (middle) scaled by average deprivation costs over the all 14 districts depending on p.

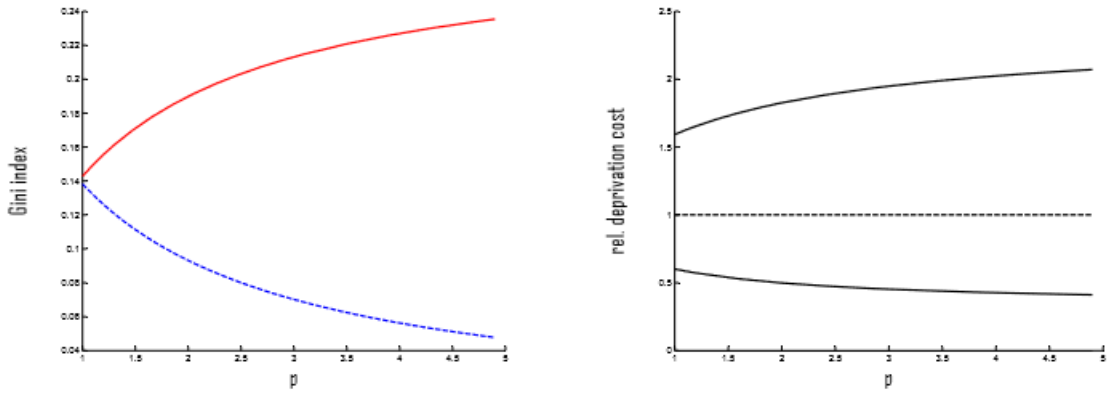

Figure 6: Gini Index and Relative Deprivation Cost in dependence of the exponent *p* of the DIF [3].

The red curve in the left picture demonstrates that the higher the exponent p, the higher the inequality of deprivation costs is. This phenomenon occurs if the goods to cover the basic human needs are lacking and the relief system and the local sources fail to supply them. In contrast to the effect of the red curve, the dashed blue curve drops with the increase of p. It means that the large curvature of the DIF causes less inequity among aid supply frequencies. In the right picture, we can see that this effect does not cover the increased sensibility of beneficiaries. A Gini Index in the range of 0.14 – 0.23 does not seem worrying, but in the case of humanitarian aid distribution it can be considered justifiable.

The curves in the right picture show that the deprivation costs in the worst-off locations grow with increasing p values, and the deprivation costs in the best-off locations decrease. As a result, residents of the worst-off locations suffer at least twice as much as those of the best-off locations. This state is, of course, unacceptable.

As mentioned above, we use the gradient descent method to get the results for the utilitarian solution. Because of the iterative optimization process of the algorithm, the speed of obtaining the results depends on the number of iterations and the tolerance constant in the stopping criterion.

We will present Figures 7-10, which plots numerically obtained Gini Index and Relative Deprivation Cost in dependence of the exponent p of the DIF. On these Figures each point was calculated with prescribed *tol* and *maxit* value.

Figures 7, 8 show the results of the GS-B for three different values of the tolerance constant. We set the maximum of iterations in each case equal to 10,000. This means that the programme takes 10,000 iterations if it does not find the solution matching the used tolerance constant. We change the tolerance constant in order to improve the results. Firstly, we set the tolerance to 10^{-3}. That means the algorithm stops if the step size is less than 0.001. Then, we modify it to 10^{-5}, and finally, we change it to 10^{-8}. We suppose that the results should improve with increasing tolerance.

When we take a look at the results, we can really see a significant difference between the three cases. While the curves in Figure 7(b) are very rugged, the ones in Figures 7(c)-(d) are smoother. This indicates that our hypothesis is right and the tolerance constant has a considerable effect on the results.

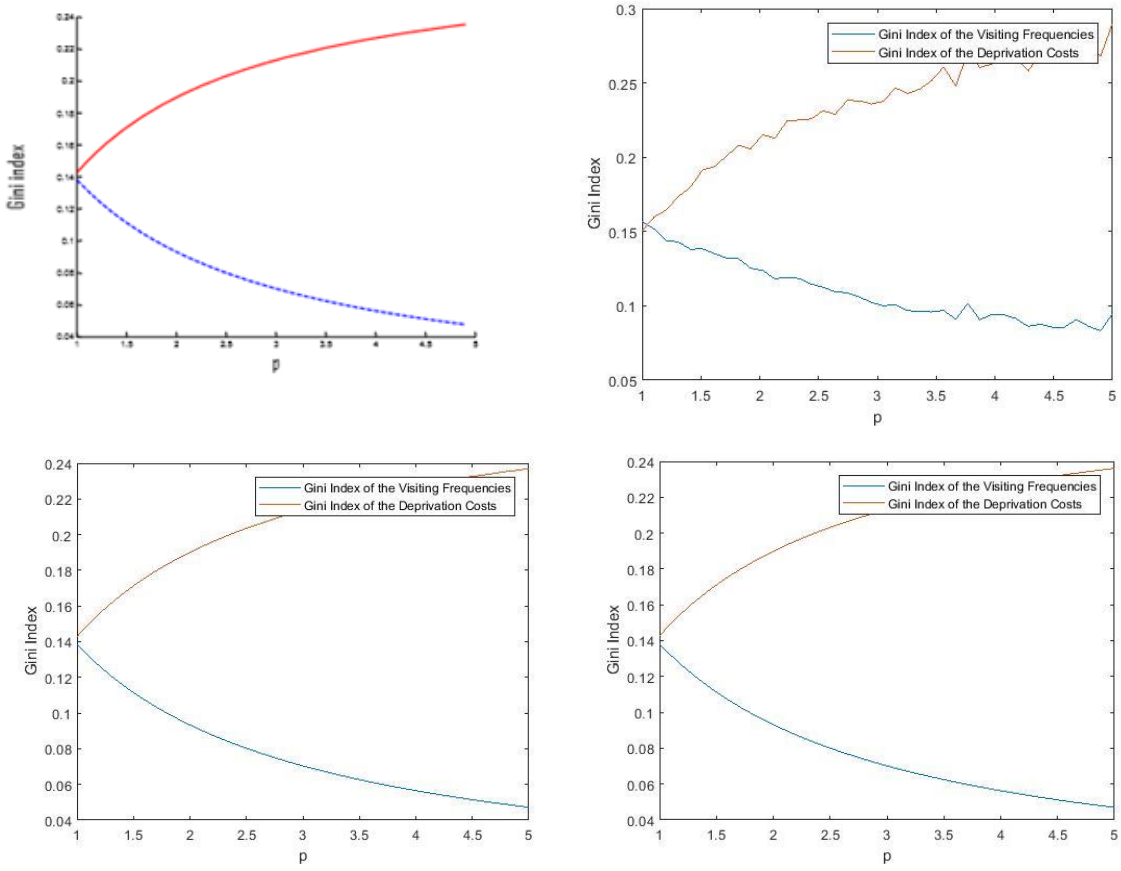

Figure 7: (a) Left upper picture: Gini Index in dependence of the exponent *p* of the DIF solved by PSO [3]. (b) Right upper picture: GS-B with *tol* = 10^{-3} and *maxit* = 10,000. (c) Left lower picture: GS-B with *tol* = 10^{-5} and *maxit* = 10,000. (d) Right lower picture: GS-B with *tol* = 10^{-8} and *maxit* = 10,000.

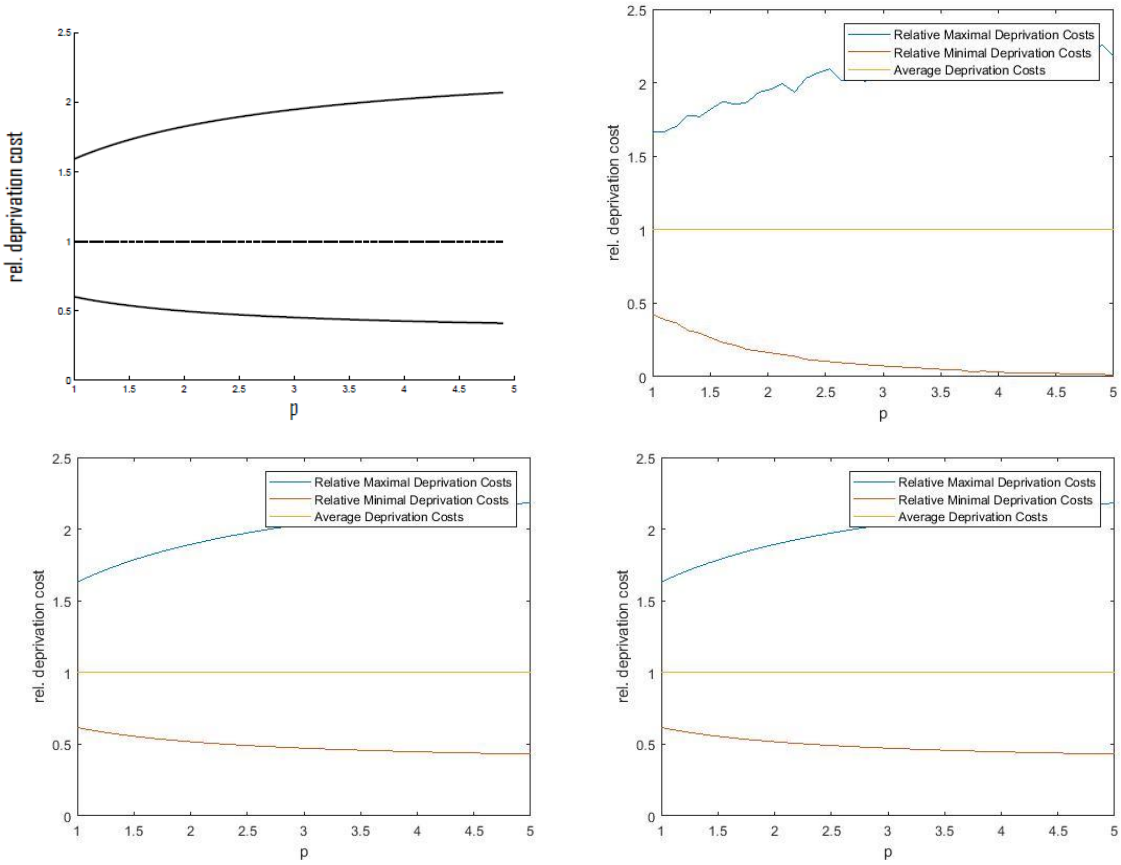

Figure 8: (a) Left upper picture: Relative Deprivation Cost in dependence of the exponent *p* of the DIF solved by PSO [3]. (b) Right upper picture: GS-B with *tol* = 10^{-3} and *maxit* = 10,000. (c) Left lower picture: GS-B with *tol* = 10^{-5} and *maxit* = 10,000. (d) Right lower picture: GS-B with *tol* = 10^{-8} and *maxit* = 10,000.

Figures 9 and 10, on the other hand, display the results of the GS-HSD also in three cases. The maximum of iterations is set in the first two cases to 10,000 and in the last one to 100,000. Again we will change the tolerance constant as mentioned above. Using the GS-HSD confirmed that the results are better with the increasing tolerance.

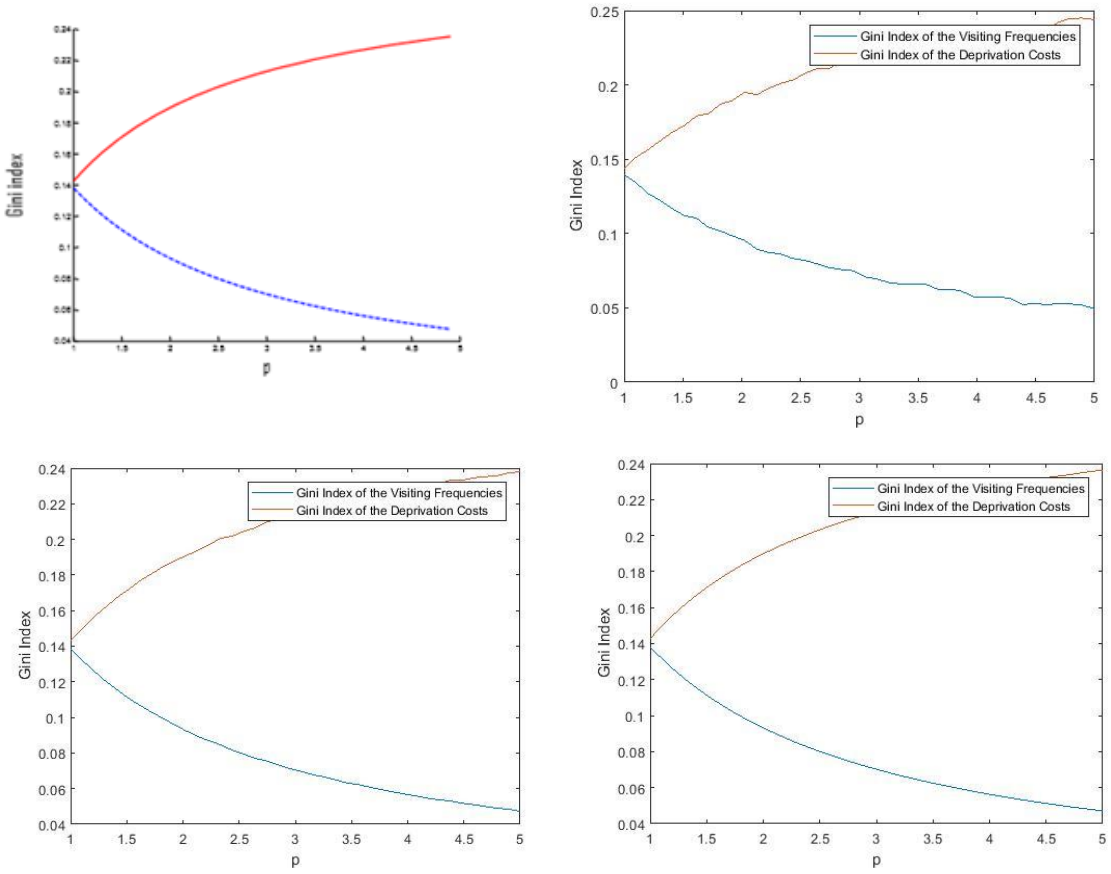

Figure 9: (a) Left upper picture: Gini Index in dependence of the exponent *p* of the DIF solved by PSO [3]. (b) Right upper picture: GS-HSD with *tol* = 10^{-3} and *maxit* = 10,000. (c) Left lower picture: GS-HSD with *tol* = 10^{-5} and *maxit* = 10,000. (d) Right lower picture: GS-HSD with *tol* = 10^{-8} and *maxit* = 100,000.

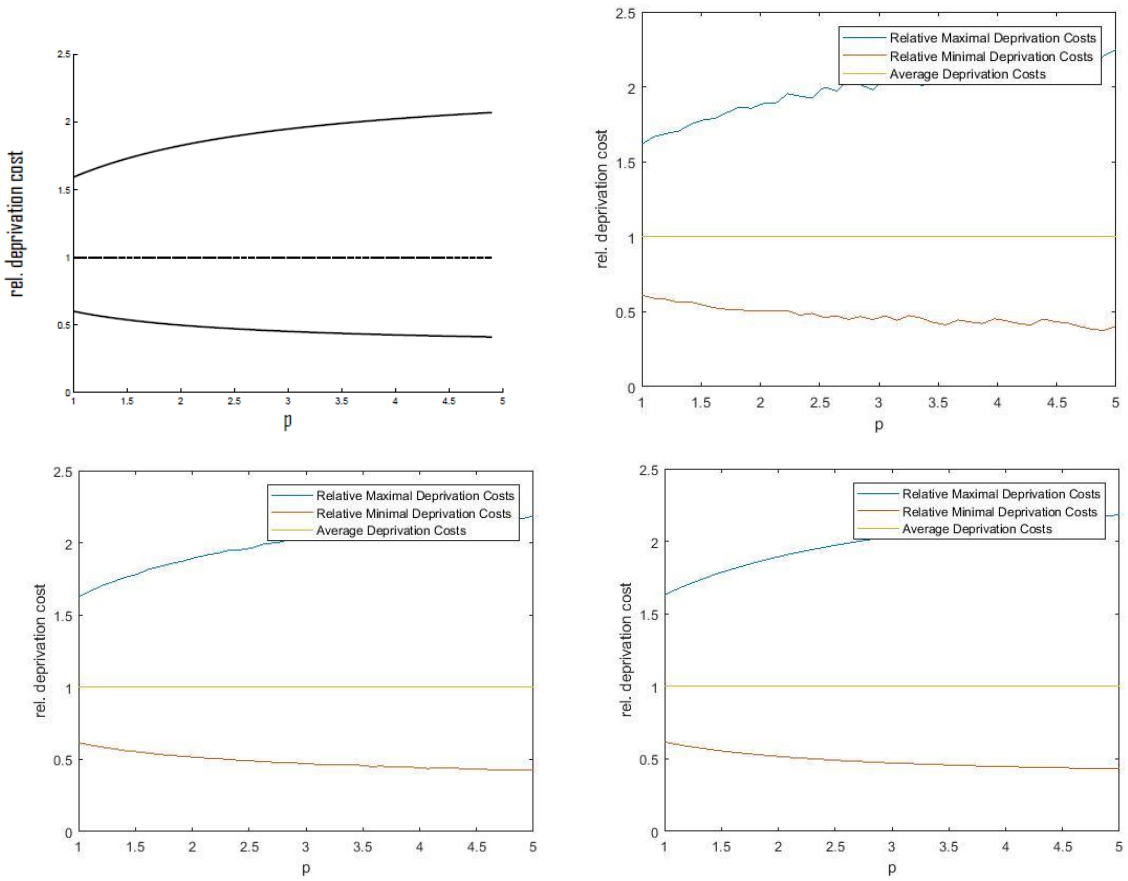

Figure 10: (a) Left upper picture: Relative Deprivation Cost in dependence of the exponent *p* of the DIF solved by PSO [3]. (b) Right upper picture: GS-HSD with *tol* = 10^{-3} and *maxit* = 10,000. (c) Left lower picture: GS-HSD with *tol* = 10^{-5} and *maxit* = 10,000. (d) Right lower picture: GS-HSD with *tol* = 10^{-8} and *maxit* = 100,000.

6.2 The Inequity-Averse Solution

In this part of our thesis we will examine the problem (P_λ), where the objective function is the minimization of the average deprivation cost with Gini´s Mean Absolute Difference. Our aim is to solve this problem by gradient based method and then compare the results with the Particle Swarm Optimization (PSO) metaheuristic from [3].

The difficulties posed by the objective function are non-convexity as noted by [3] and non-differentiability in finite many points due to the presence of absolute value in the Gini Index term. The non-convexity implies the possibility of multiple local optima. We addressed the problem of non-differentiability by using several reformulations and introducing new help functions σ and $\bar{\sigma}$ in the subsection 2.3.

Our intent is to compare our results for the inequity-averse solution with the results from [3]. We analyze the behavior of the gradient search on the Nepal test case. We will use the data from the Nepal test case as well as problem defining parameters as in the subsection 6.1. Additionally, we consider $\lambda \in [0, 0.5]$ uniformly discretized with the step of 0.05. Note that $\lambda = 0$ correspondents to the utilitarian solution. We will implement two choices of the initial solution:

- The Symmetric Initial Solution: $y_0 = \frac{\mathbf{1}}{|K|}$.
- Randomized Multistart: a list of initial solutions that are uniformly distributed on the unit simplex.

The value of $\eta = 2$ was used for calculations done by GS-HSD.

6.2.1 The Symmetric Initial Solution

First of all, we obtain the results of both methods – GS-B and GS-HSD, where the tolerance is equal to 10^{-8} and the maximum of iteration is set to 10,000. Table 3 describes the numerical results by comparing with the results of PSO from [3]. It is clear from the results that both methods do not achieve as good results as PSO metaheuristic. If we take a look at the numerical result in Table 3, we can observe that GS-HSD returns much better results than the GS-B. The values $f(x)$ of GS-HSD are very close to the values of PSO. For all values of λ, 10,000 iterations of GS-HSD were used. This indicates that number of iterations is insufficient for GS-HSD. On the other hand, the GS-B stopped at the 36th iteration for $\lambda = 0.5$. This fact suggests changing the stopping criterion. The t_{all} stands for the total time in seconds consuming by the algorithm.

	GS-HSD			GS-B			The Particle Swarm Optimization (PSO) Metaheuristics		
tol	**10^{-8}**			**10^{-8}**					
maxit	**10,000**			**10,000**					
λ	**μ*10^{-7}**	**G (δ)**	**f(x)*10^{-7}**	**μ*10^{-7}**	**G (δ)**	**f(x)*10^{-7}**	**μ*10^{-7}**	**G (δ)**	**f(x)*10^{-7}**
0.00	2.5631	0.1908	2.5631	2.5631	0.1899	2.5631	2.5630	0.1898	2.5630
0.05	2.5657	0.1689	2.6090	2.5658	0.1686	2.6091	2.5657	0.1686	2.6090
0.10	2.5733	0.1481	2.6495	2.5739	0.1470	2.6496	2.5739	0.1470	2.6496
0.15	2.5863	0.1268	2.6847	2.6120	0.1503	2.7298	2.5977	0.1249	2.6950
0.20	2.6044	0.1054	2.7142	2.6674	0.1476	2.8249	2.6081	0.1016	2.7141
0.25	2.6262	0.0854	2.7383	2.7257	0.1517	2.9325	2.6348	0.0781	2.7377
0.30	2.6473	0.0690	2.7569	2.7354	0.1509	2.9831	2.6658	0.0560	2.7554
0.35	2.6590	0.0608	2.7722	2.7709	0.1477	3.0573	2.6953	0.0385	2.7679
0.40	2.6722	0.0523	2.7840	2.8169	0.1490	3.1526	2.7206	0.0225	2.7696
0.45	2.6834	0.0456	2.7935	2.8261	0.1486	3.2040	2.7422	0.0161	2.7819
0.50	2.6938	0.0398	2.8010	2.9261	0.1700	3.4236	2.7605	0.0089	2.7851
t_{all} [s]	13.47415			202.26778					

Table 3: Numerical comparison of GS-HSD and GS-B with PSO for the inequity-averse case, where *tol* = 10^{-8} and *maxit* = 10,000.

So, we decrease the tolerance constant to 10^{-10} for the GS-B and the maximum of iteration will be set to 200,000 for GS-HSD to see, whether we get better results. This changing results in clearly improvement for GS-HSD shown in Table 4. The results for λ smaller than ≈ 0.3, have the relative error smaller than 0.01. As above, all 200,000 iterations were used for GS-HSD. The results of the GS-B remain still unsatisfactory and algorithm broke at the 68th iteration for $\lambda = 0.5$.

	GS-HSD			GS-B			The Particle Swarm Optimization (PSO) Metaheuristics		
tol	**10^{-8}**			**10^{-10}**					
maxit	**200,000**			**10,000**					
λ	**$\mu*10^{-7}$**	**G (δ)**	**$f(x)*10^{-7}$**	**$\mu*10^{-7}$**	**G (δ)**	**$f(x)*10^{-7}$**	**$\mu*10^{-7}$**	**G (δ)**	**$f(x)*10^{-7}$**
0.00	2.5631	0.1899	2.5631	2.5631	0.1899	2.5631	2.5630	0.1898	2.5630
0.05	2.5657	0.1686	2.6090	2.5658	0.1686	2.6090	2.5657	0.1686	2.6090
0.10	2.5739	0.1470	2.6496	2.5733	0.1501	2.6505	2.5739	0.1470	2.6496
0.15	2.5874	0.1253	2.6847	2.6000	0.1380	2.7076	2.5977	0.1249	2.6950
0.20	2.6063	0.1035	2.7142	2.6591	0.1508	2.8195	2.6081	0.1016	2.7141
0.25	2.6286	0.0832	2.7379	2.7196	0.1457	2.9178	2.6348	0.0781	2.7377
0.30	2.6559	0.0627	2.7558	2.7663	0.1519	3.0184	2.6658	0.0560	2.7554
0.35	2.6660	0.0559	2.7703	2.8066	0.1571	3.1152	2.6953	0.0385	2.7679
0.40	2.6762	0.0495	2.7822	2.8000	0.1731	3.1878	2.7206	0.0225	2.7696
0.45	2.6863	0.0435	2.7915	2.8118	0.1455	3.1801	2.7422	0.0161	2.7819
0.50	2.6963	0.0380	2.7988	2.8164	0.1477	3.2324	2.7605	0.0089	2.7851
t_{all} [s]	250.18			242.97					

Table 4: Numerical comparison of GS-HSD and GS-B with PSO for the inequity-averse case, where *tol* = 10^{-8} and *maxit* = 200,000.

Due to unsatisfactory results of the GS-B, we consider relaxing the stopping criterion and compute 10,000 iterations for GS-B, 100,000 for GS-HSD and for each value λ. Table 5 summarizes those results. We can see that both methods return much better results with increased amount of iterations. When we compare the values of the objective function for $\lambda = 0.5$, we can observe that the relative difference between the result obtained by GS-B and PSO is only 0.5%. These findings indicate that it is desirable to modify the stopping criterion.

	GS-HSD			GS-B			The Particle Swarm Optimization (PSO) Metaheuristics		
maxit	**100,000**			**10,000**					
λ	**$\mu*10^{-7}$**	**G (δ)**	**$f(x)*10^{-7}$**	**$\mu*10^{-7}$**	**G (δ)**	**$f(x)*10^{-7}$**	**$\mu*10^{-7}$**	**G (δ)**	**$f(x)*10^{-7}$**
0.00	2.5631	0.1899	2.5631	2.5631	0.1910	2.5631	2.5630	0.1898	2.5630
0.05	2.5657	0.1687	2.6090	2.5654	0.1705	2.6091	2.5657	0.1686	2.6090
0.10	2.5738	0.1471	2.6495	2.5726	0.1498	2.6497	2.5739	0.1470	2.6496
0.15	2.5873	0.1254	2.6846	2.5863	0.1269	2.6848	2.5977	0.1249	2.6950
0.20	2.6062	0.1035	2.7141	2.6013	0.1087	2.7144	2.6081	0.1016	2.7141
0.25	2.6284	0.0834	2.7380	2.6262	0.0856	2.7386	2.6348	0.0781	2.7377
0.30	2.6557	0.0629	2.7559	2.6505	0.0669	2.7569	2.6658	0.0560	2.7554
0.35	2.6658	0.0561	2.7705	2.6617	0.0596	2.7727	2.6953	0.0385	2.7679
0.40	2.6759	0.0497	2.7823	2.6763	0.0507	2.7849	2.7206	0.0225	2.7696
0.45	2.6861	0.0437	2.7917	2.6968	0.0391	2.7917	2.7422	0.0161	2.7819
0.50	2.6962	0.0381	2.7989	2.7116	0.0324	2.7995	2.7605	0.0089	2.7851
t_{all} [s]	125.6830			747.4324					

Table 5: Numerical comparison of GS-HSD and GS-B with PSO for the inequity-averse case without stopping criterion.

6.2.1.1. Detecting Local Minimum

In order to find out, why the GS-B returns unsatisfying results, it is interesting to examine the drop of the objective function during the iterative process. From this behavior, we can identify whether the algorithm is stuck at the local optimum. Figure 11 displays two possible courses of the iterative process, the left picture with the tolerance equal to 10^{-3} and with the maximum of iteration set to 1,000, and the right one with the strengthen tolerance and the maximum number of iterations, *tol* = 10^{-8} and *maxit* = 10,000, respectively. It is obvious from the figure that the significant change comes somewhere in the range of 0-100 iterations. Then the drop of the objective function is almost zero. Note, we only present the results for the value $\lambda = 0.5$.

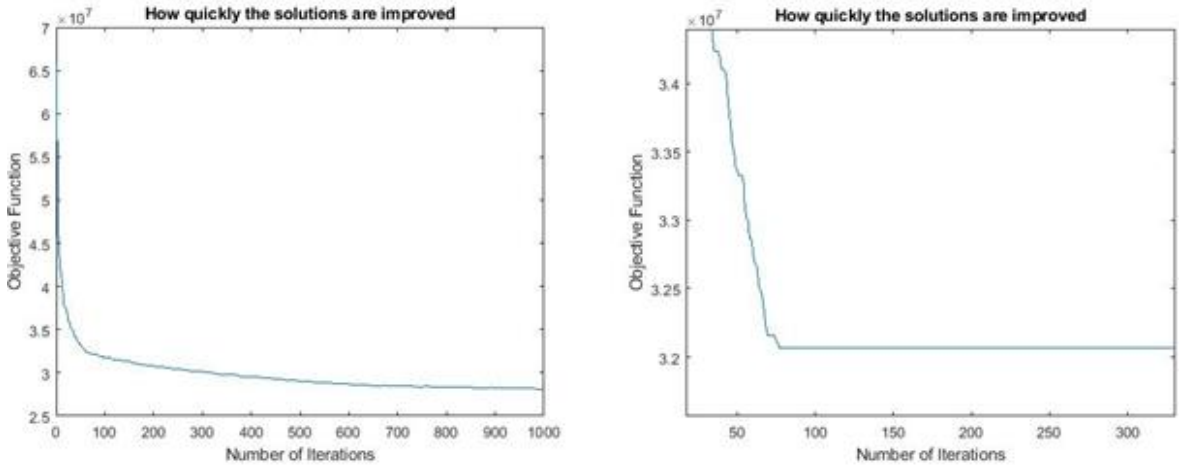

Figure 11: (a) Left picture: The Objective Function in dependence of the number of iterations with *tol* = 10^{-3} and *maxit* = 1,000. (b) Right picture: The Objective Function in dependence of the number of iterations with *tol* = 10^{-8} and *maxit* = 10,000, where λ = 0.5.

We consider the drop of the objective function between two consecutive iterations. The drop of the objective function is plotted on Figure 12 with *tol* = 10^{-4}, *maxit* = 10,000 and $\lambda = 0.5$. If the tolerance is set to 0.0001, then the GS-B will be stopped near to 30th iteration, but as it shows there can still some progress be made. This indicates that our actual stopping criterion is not appropriate because of the fact that there are some step sizes which could have significant effect on the solution, if the algorithm continued despite the stopping criterion. Our present stopping criterion is as follows:

$$\left\| x_{k+1} - x_k \right\|_2 < tol$$

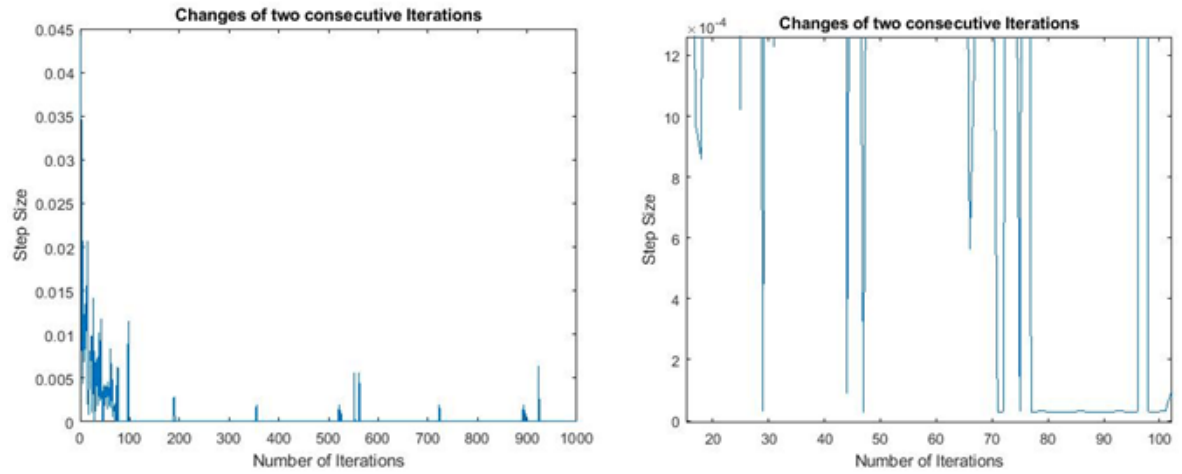

Figure 12: (a) Left picture: Changes of two consecutive iterations with *tol* = 10^{-4} and *maxit* = 10,000, where λ = 0.5. (b) Right picture: The same changes of two consecutive iterations in detail.

6.2.2. Sensitivity Analysis of η

We show how the constant η in GS-HSD method affects the results by using the symmetric initial solution. Furthermore, we relax the stopping criterion.

It is interesting to check the influence of the constant η in GS-HSD. We limit our inquiry to the case $\lambda = 0.5$. Based on Figure 13, we can deduce the following: the value of the objective function depends on η if the number of iterations is small. On the other hand, it depends less on η if the number of iterations is higher. In case, the maximum of iterations is equal to 4,000, the value of the objective function is almost constant, starting from $\eta = 0.5$. This observation confirms our decision to set $\eta = 2$.

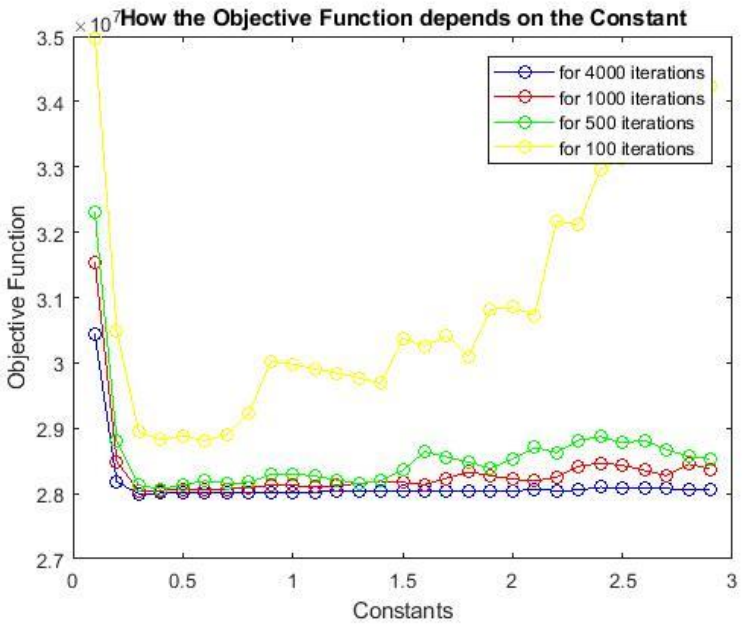

Figure 13: The influence of the constant η in Harmonic Step Decrease with λ = 0.5 without stopping criterion.

6.2.3. Randomized Multistart

In this section, we consider multistart for both GS-B and GS-HSD. The initial solution is uniformly distributed on unit simplex. We set the number of replications equal to 100. Additionally, according to the observation in the subsection 6.2.1.1, we try to get better results using another stopping criterion. The following code shows how another stopping criterion works:

```
if (norm(alpha(k)*S(:,k)) < tol)
   k_max = k_max+1; %in case it is under the tolerance
else
   k_max = 0; %in case it is above the tolerance
end
if k_max == 10 %if k_max is equal to 10 than stop the function opt_search
   break
end
```

If $\|x_{k+1} - x_k\|_2 < tol$ is satisfied *k_max* times consecutively, then break the algorithm.

Firstly, we obtain the results for both methods, where

- the tolerance is set to 10^{-3},
- the maximum number of iteration is equal to 10,000,
- the *k_max* = 10.

Table 6 shows that the results of the GS-HSD Method are very similar to the PSO, even a little better for all values λ except $\lambda = 0.5$. On the other hand, the GS-B returns a bit worse results than GS-HSD, but shows much better results than using the symmetric initial solution. Note that for $\lambda = 0.5$, GS-HSD stopped at 1,996th iteration and GS-B at 184th. Furthermore, we can see that the GS-B takes again more time than GS-HSD.

	GS-HSD			GS-B			The Particle Swarm Optimization (PSO) Metaheuristics		
tol	10^{-3}			10^{-3}					
maxit	10,000			10,000					
λ	**μ*10^{-7}**	**G (δ)**	**f(x)*10^{-7}**	**μ*10^{-7}**	**G (δ)**	**f(x)*10^{-7}**	**μ*10^{-7}**	**G (δ)**	**f(x)*10^{-7}**
0.00	2.5631	0.1883	2.5631	2.5638	0.1870	2.5638	2.5630	0.1898	2.5630
0.05	2.5648	0.1674	2.6078	2.5653	0.1666	2.6081	2.5657	0.1686	2.6090
0.10	2.5706	0.1460	2.6457	2.5708	0.1445	2.6451	2.5739	0.1470	2.6496
0.15	2.5813	0.1253	2.6783	2.5792	0.1241	2.6752	2.5977	0.1249	2.6950
0.20	2.5961	0.1031	2.7031	2.5924	0.1000	2.6961	2.6081	0.1016	2.7141
0.25	2.6130	0.0844	2.7233	2.6043	0.0811	2.7100	2.6348	0.0781	2.7377
0.30	2.6296	0.0709	2.7414	2.6171	0.0703	2.7274	2.6658	0.0560	2.7554
0.35	2.6433	0.0633	2.7603	2.6324	0.0601	2.7432	2.6953	0.0385	2.7679
0.40	2.6524	0.0543	2.7675	2.6503	0.0649	2.7880	2.7206	0.0225	2.7696
0.45	2.6629	0.0476	2.7770	2.6623	0.0575	2.8001	2.7422	0.0161	2.7819
0.50	2.6745	0.0417	2.7860	2.6684	0.0656	2.8435	2.7605	0.0089	2.7851
t [s]	264.1742			2,527.5					

Table 6: Numerical comparison of GS-HSD and the GS-B with PSO for the inequity-averse case for randomized multistart, where *k_max* = 10.

Secondly, we increase the statement *k_max* following:

- the *k_max* = 100,
- the *tol* = 10^{-3},
- the *maxit* = 10,000.

We can summarize some interesting findings from Table 7. If we look at the values of the objective function, it is clear that the GS-B returns better results than PSO for all λ. On the other hand, GS-HSD shows the lowest values of the $f(x)$ for $\lambda = 0.05 - 0.35$. Figure 14 clearly demonstrates this observation. In addition, this phenomenon implies the existence of multiple local optima.

Note that the values of $G(\delta)$ in different local minima differ substantially. For example, if we compare values for $\lambda = 0.5$, we can see that the relative error of the $f(x)_{0.5}$ for GS-B is smaller than 1% against PSO and the relative error of the $\mu_{0.5}$ is 2.5% smaller in favor of GS-B. On the other hand, the value of the $G(\delta)_{0.5}$ is up to 62% smaller for PSO by comparing with GS-B. Figure 15 illustrates the differences in the $G(\delta)$ for all λ values by comparing all three methods.

An especially interesting finding is that the $f(x)_{0.5}$ of GS-B obtains a lower value for cases $\lambda = 0.25 - 0.45$ as it was obtained at the particular λ, therefore the solution $\lambda = 0.5$ is better approximation of the optimal solution than ones obtained for λ in $0.25 - 0.45$. For example:

$$2.7242 = \mu(x_{0.5})\left[1 + 2*0.25*G(\delta_{0.5})\right] < \mu(x_{0.25})\left[1 + 2*0.25*G(\delta_{0.25})\right] = 2.7298$$

	GS-HSD			GS-B			The Particle Swarm Optimization (PSO) Metaheuristics		
tol	10^{-3}			10^{-3}					
maxit	10,000			10,000					
λ	**μ*10^{-7}**	**G (δ)**	**f(x)*10^{-7}**	**μ*10^{-7}**	**G (δ)**	**f(x)*10^{-7}**	**μ*10^{-7}**	**G (δ)**	**f(x)*10^{-7}**
0.00	2.5631	0.1898	2.5631	2.5631	0.1899	2.5631	2.5630	0.1898	2.5630
0.05	2.5648	0.1665	2.6075	2.5649	0.1686	2.6082	2.5657	0.1686	2.6090
0.10	2.5709	0.1473	2.6467	2.5712	0.1485	2.6476	2.5739	0.1470	2.6496
0.15	2.5799	0.1270	2.6781	2.5834	0.1251	2.6803	2.5977	0.1249	2.6950
0.20	2.5957	0.1046	2.7043	2.6007	0.1040	2.7089	2.6081	0.1016	2.7141
0.25	2.6126	0.0872	2.7265	2.6204	0.0835	2.7298	2.6348	0.0781	2.7377
0.30	2.6264	0.0719	2.7397	2.6409	0.0656	2.7449	2.6658	0.0560	2.7554
0.35	2.6390	0.0598	2.7495	2.6498	0.0545	2.7509	2.6953	0.0385	2.7679
0.40	2.6532	0.0545	2.7690	2.6673	0.0438	2.7609	2.7206	0.0225	2.7696
0.45	2.6641	0.0490	2.7815	2.6849	0.0307	2.7591	2.7422	0.0161	2.7819
0.50	2.6752	0.0418	2.7871	2.6919	0.0240	2.7565	2.7605	0.0089	2.7851
t [s]	281.5193			44,660					

Table 7: Numerical comparison of GS-HSD and GS-B with PSO for the inequity-averse case for randomized multistart, where *k_max* = 100. Note that for λ = 0.5, GS-HSD stopped at 2,159th iteration and GS-B at 10,000th.

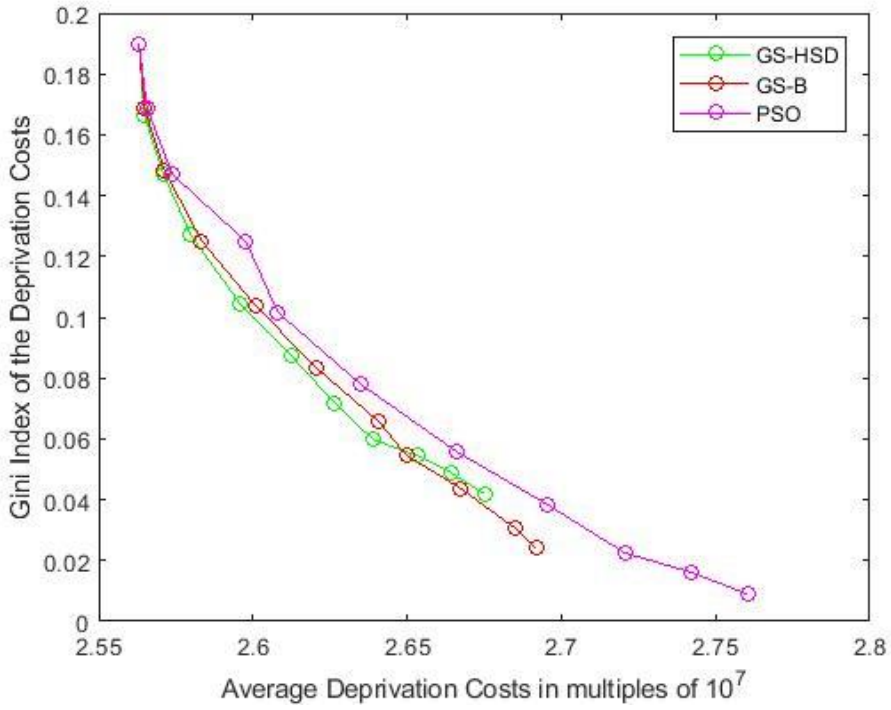

Figure 14: Tradeoff between μ and G(δ) for all three methods.

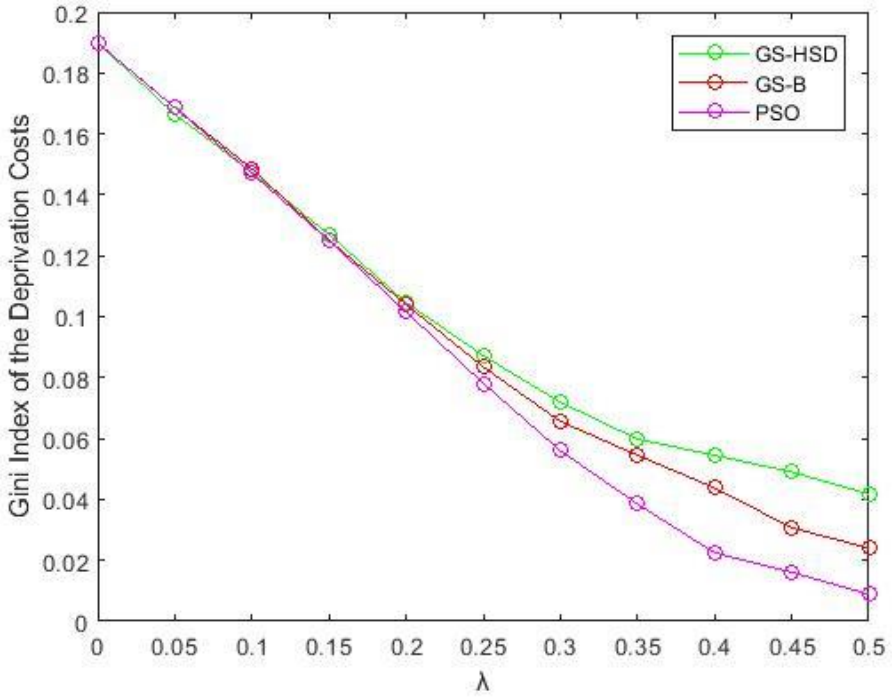

Figure 15: Tradeoff between λ and G(δ) for all three methods.

7. Conclusion

Gutjahr and Fischer [3] have shown that a model of the humanitarian logistics with penalization for inequity allows a fairer distribution of relief aid. We have shown that the use of the Gradient Search method with multistart improves numerical accuracy. It was challenging to implement the Gradient Search approach because of the fact that the objective function is non-differentiable.

The solution of this problem is presented in the third chapter, where we derived a formula for the evaluation of the derivative of the non-differentiable objective function. In the fourth chapter, we adopted the Frank-Wolfe Algorithm for the problem (P_λ). The structure of the linear program to be solved in each iteration allows a solution without using the LP solving procedure. We considered two different step sizes, the optimal step size calculated by the Bisection Method and fast heuristic approach HSD. In the sixth chapter, we solved the logistics model by gradient based method with step choices, GS-B and GS-HSD. We confirmed the existence of multiple local optima in the subsection 6.2.1.1. Moreover, we showed that the Gini Index of the deprivation costs, $G(\delta)$, can be substantially different in distinct local optima. Further, we can identify significant differences between both methods. By the using the GS-HSD, we got prompt satisfactory results. On the other hand, the slow convergence due to the non-differentiability of the objective function is probably the main reason for the low efficiency of the GS-B. The important finding is that the presented algorithms could achieve the improvement of the numerical accuracy by selecting randomized multistart comparing to the PSO.

We believe that it is possible to decrease the processing time of the GS-B or implement approximate step size like the Barzilai-Borwein. However, further algorithmic improvements of this kind are outside the scope of this thesis.

Bibliography

[1] Mangan, John, Chandra Lalwani, and Chandra L. Lalwani. Global logistics and supply chain management. John Wiley & Sons, 2016.

[2] Holguín-Veras, José, et al. "On the appropriate objective function for post-disaster humanitarian logistics models." Journal of Operations Management 31.5 (2013): 262-280.

[3] Gutjahr, Walter J., and Sophie Fischer. "Equity and deprivation costs in humanitarian logistics." European Journal of Operational Research 270.1 (2018): 185-197.

[4] Frank, Marguerite, and Philip Wolfe. "An algorithm for quadratic programming." Naval research logistics quarterly 3.1-2 (1956): 95-110.

[5] Garber, Dan, and Elad Hazan. "Faster rates for the frank-wolfe method over strongly-convex sets." arXiv preprint arXiv:1406.1305 (2014).

[6] Johnsson, Isabelle. "Tracking Relief Aid: A Spatial Analysis of Aid Distribution in Nepal After the 2015 Gorkha Earthquake." (2016).

[7] Wayne, L. WINSTON. "Operations research: applications and algorithms." Duxbury Press (1994).

[8] Burden, Richard L., and J. Douglas Faires. "Numerical analysis. 2001." Brooks/Cole, USA (2001).

[9] Bolzano, Bernard, and Hermann Hankel. Rein analytischer beweis des lehrsatzes: dass zwischen je zwey werthen, die ein entgegengesetztes resultat gewähren, wenigstens eine reelle wurzel der gleichung liege, von Bernard Bolzano.--Untersuchungen über die unendlich oft oszillierenden und unstetigen funktionen, von Hermann Hankel. No. 153. W. Engelmann, 1905.

[10] Patriksson, Michael: The Frank-Wolfe algorithm, lectures from Applied Optimization, University of Gothenburg, available online (04.02.2019).

Appendix

A The Solution of the problem (15) using Karush-Kuhn-Tucker Conditions

The Lagrange Function $L : \mathbb{R}^{|K|} \mathrm{x}\, \mathbb{R} \to \mathbb{R}$ is defined as:

$$L(x,u) = \sum_{k \in K} \omega_k x_k^{-p} + u\left(\sum_{k \in K} d_k x_k - B \right) .$$

Karush-Kuhn-Tucker Conditions:

$$\frac{\partial L}{\partial x_i} = -p\omega_i x_i^{-p-1} + u d_i = 0 \qquad \forall\ i \in K , \tag{KKT1}$$

$$\sum_{k \in K} d_k x_k - B \le 0 , \tag{KKT2}$$

$$u\left(\sum_{k \in K} d_k x_k - B \right) = 0 , \tag{KKT3}$$

$$u \ge 0 . \tag{KKT4}$$

We can consider two possible cases:

- $u = 0 \quad \Rightarrow \quad -p\omega_i x_i^{-p-1} + 0 d_i = 0$ contradiction
- $u \ne 0 \quad \Rightarrow \quad \sum_{k \in K} d_k x_k - B = 0$

We use the (KKT1) to solve for x_i to get:

$$x_i = \left(\frac{u d_i}{p\omega_i} \right)^{\frac{1}{-p-1}} .$$

Now, we substitute the value of x_i into (KKT3):

$$\sum_{k \in K} d_k \left(\frac{u d_k}{p\omega_k} \right)^{\frac{1}{-p-1}} - B = 0$$

$$\sum_{k \in K} d_k u^{\frac{1}{-p-1}} \left(\frac{d_k}{p\omega_k} \right)^{\frac{-1}{p+1}} = B$$

$$u^{\frac{1}{-p-1}} \sum_{k \in K} d_k \left(\frac{d_k}{p\omega_k} \right)^{\frac{-1}{p+1}} = B$$

The value of the Lagrange multiplier u is:

$$u = \left(\frac{B}{\sum_{k \in K} d_k \left(\frac{d_k}{p\omega_k} \right)^{\frac{-1}{p+1}}} \right)^{-p-1} .$$

Finally, we express x_i:

$$x_i = \left(\frac{C^{-p-1} d_i}{p\omega_i} \right)^{\frac{1}{-p-1}} = C \left(\frac{p\omega_i}{d_i} \right)^q ,$$

where

$$q = \frac{1}{p+1}, \; C = \frac{B}{\sum_{k \in K} d_k^{1-q} \left(p\omega_k \right)^q} .$$

B MATLAB Code

The Objective Function

```
function [ZFval] = ZF(g1,p,w,d,B,lambda,y)          %declaration of the obj. function
N = sum(w);                                         %the sum of all receivers
Cp = g1/(2*N*N*(p+1)*B^p);
m = length(y);                                      %the number of cities
total = 0;

for i = 1:m
   for j = 1:m
      total = total + w(i)*w(j)*abs((d(i)/y(i))^p - ((d(j)/y(j))^p));
   end
end

miG = Cp*total;              %the average deprivation cost multiplied by Gini Index
mi = (g1/(B^p*N*(p+1)))*sum(w.*d.^p./y.^p);         %average deprivation cost
ZFval = mi+2*lambda*miG;                            %the value of the objective function
end
```

The Derivation of the Objective Function

```
function [DZFval] = DZF(g1,p,w,d,B,lambda,y)        %declaration of the derivation
N = sum(w);                                         %the sum of all receivers
Cp = g1/(2*N*N*(p+1)*B^p);
m = length(y);                                      %the number of cities
total = zeros(m,1);
d_mi = w.*(d.^p).*y.^(-p-1)*((-p*g1)/(N*(p+1)*B^p)); %the derivation of the deprivation
costs
D = (d./y).^p;                                  %the definition of the general help vector

for l = 1:m
   for k = 1:m
      total(l)=total(l)+sigma(l,k,D(k),D(l))*w(l)*w(k)*d(l)^p*y(l)^(-p-1);
   end
end

d_miG = 2*Cp*p*total;        %the derivation of the avg. depriv. cost and Gini Index
DZFval = d_mi + 2*lambda*d_miG;          %the derivation of the objective function
end
```

The Sigma Bar

```
function [sigma] = sigma(l,k,Dk,Dl)          %the declaration of the sigma bar
Dk = dk/yk;
Dl = dl/yl;
sigma = 0;                                   %the case, where l==k
%A: the case, where l==k´; k==k
if (l > k)
   if ((dk/yk)^p > (dl/yl)^p)                %the case for K+
      sigma = 1;
   else
```

```
        sigma = -1;
    end
end
%B: the case, where l==k; k==k´
if (l < k)
    if ((dl/yl)^p > (dk/yk)^p)                %the case for K-
        sigma = -1;
    else
        sigma = 1;
    end
end
end
```

The Gini Index

```
function [G] = GINI(w,z)                      %the declaration of the Gini Index
weight = w/sum(w);                            %the share of receivers in each demand point
m = length(z);
mi = sum(z.*weight);
total = 0;                                    %the new variable

for i = 1:m
    for j = 1:m
        total = total + weight(i)*weight(j)*abs(z(i) - z(j));
    end
end

G = 1/(2*mi)*total;                           %the definition of the Gini Index
end
```

The Bisection Method

```
function [result,N] = bisection(A,B,f,tol)    %the declaration of the bisection
N = ceil(log(tol/(B-A))/log(0.5));            %number of iteration

for i = 1:N
    if f(A)*f(B) < 0                          %if the root is within the interval [A,B]
      x_help = (A+B)/2;                       %reduce the interval to half and set a new x_help
      if f(x_help)*f(A) < 0                   %if the root is within the interval [x_help,A]
         B = x_help;                          %then the new boundary will be set to B
      else
         A = x_help;                          %otherwise the new boundary will be set to A
      end
    else
       disp ('Wrong Initial Interval!')       %for case f(A)*f(B) > 0
      return
    end
end

result = (A+B)/2;                             %the final result of the bisection method
end
```

The Opt_Search (Frank-Wolfe Algorithm)

```
function[Y,S,k,alpha] = opt_search(g1,p,w,d,B,lambda,y0,tol,alpha_mode,maxit) %the declaration of the opt_search function
n = length(y0);                 %the number of cities
Y = zeros(n,maxit+1);           %new matrix for results in each iteration
S = zeros(n,maxit+1);           %new matrix for the direction
alpha = zeros(1,maxit+1);       %new vector for the step size
Y(:,1) = y0;                    %the initial solutions will be stored in the first row
stop = 0;       %new variable for number of times we have been under the tolerance

for k = 1:maxit                 %the for-loop for the whole opt_search procedure

   % STEP 1: find direction S
   [minimum,index] = min(DZF(g1,p,w,d,B,lambda,Y(:,k))); %find the minimum value of the DZF and its position in DZF
   xi = zeros(n,1);             %new vector for the decision variable in DZF
   xi(index) = 1;               %assign 1 on index at the vector xi
   S(:,k) = xi-Y(:,k);          %the formula for calculating the direction

   % STEP 2: find the size alpha
   switch alpha_mode
      case  'HSD'               %calculate the result using HSD where η = 2
         alpha(k) = 2/(2+k);
      case  'bisection'         %calculate the bisection method
       f=@(alpha)S(:,k)'*DZF(g1,p,w,d,B,lambda,Y(:,k)+alpha*S(:,k));
       [alpha(k),N] = bisection(0,0.999999999999,f,tol); %calculate the result using the bisection method

   % STEP 3: update the solution
   Y(:,k+1) = Y(:,k)+alpha(k)*S(:,k);       %calculate new iteration point

   % STEP 4: check the optimality
   %THE FIRST STOPPING CRITERION
   %if (norm(alpha(k)*S(:,k))<tol)
      %break
   %end
   %THE SECOND STOPPING CRITERION
   if (norm(alpha(k)*S(:,k))<tol)
      k_max = k_max+1;          %in the case, it is under the tolerance
   else
      k_max = 0;                %in the case, it is above the tolerance
   end
   if k_max == 10               %if k_max is equal to 0 than stop the function opt_search
      break
   end
end
end
```

The Results for the Inequity-Averse Case: Randomized Multistart

```
g1 = 2;
d = 1000*[0.0794; 0.6055; 0.8577; 1.0138; 0.1508; 0.4925; 0.1403; 0.2419; 0.5131; 0.3673; 0.4976; 0.4528; 0.2839; 0.7716];
w = [25; 106; 83; 79; 63; 131; 39; 59; 99; 30; 66; 27; 59; 97];
p = 2;
B = 1;
tic                         %measures the processing time of the algorithm

for j = 1: lambdaN          %the for-loop for the all λ values
lambda = lambda_vec(j);
lambda_vec = 0:0.05:0.5;    %stores 11 values of λ
lambdaN = length (lambda_vec);    %the number of λ
avg_dep_vec = zeros(1,lambdaN);   %stores 11 values of the average deprivation cost
G_DELTA_vec = zeros(1,lambdaN); %stores 11 values of the Gini of deprivation cost
result_vec = zeros(1,lambdaN);    %stores 11 values of the object. function ZF
alpha_mode = 'bisection';
%alpha_mode = 'HSD';
tol = 10^(-3);
num_replication = 100;            %the number of the replications
s = rng('default');%a function that returns a seed for the generating pseudorandom numbers
y = rand(14,num_replication);     %Randomized Multistart
y = y./sum(y);
result = zeros(num_replication,1);
G_DELTA = zeros(num_replication,1);
avg_dep = zeros(num_replication,1);
k_number = zeros(num_replication,1);
for i = 1:num_replication         %the for-loop for the number of the replication
y0 = y(:,i);
maxit = 10000;
[Y, S, k, alpha] = opt_search(g1,p,w,d,B,lambda,y0,tol, alpha_mode,maxit);
x = Y(:,k+1)./d*B;
q = 1/(p+1);
C = B/(sum(d.^(1-q).*(p*w).^q));
x_C = x/C;
delta = 2./((p + 1)*x.^p);
delta_C = delta*C^p;
G_DELTA(i) = GINI(w,delta);
avg_dep(i) = w'*delta/sum(w);
k_number(i) = k;
result(i) = ZF(g1,p,w,d,B,lambda,Y(:,k+1));
end
avg_dep_vec(j) = min(avg_dep); %find the minimum in the vector of the average deprivation costs for each value of λ
G_DELTA_vec(j) = min(G_DELTA); %find the minimum in the vector of the Gini of deprivation costs for each value of λ
result_vec(j) = min(result); %find the minimum in the vector of the values of the objective function for each value of λ
end
time = toc       %measures the processing time of the algorithm
```

Abstrakt

Es wird ein Modell betrachtet, das die humanitären Operationen in der Responsephase nach einer Katastrophe in Hinblick auf Verteilungsgerechtigkeit beschreibt. Einer der neueren methodischen Fortschritte in der Humanitären Logistik ist das von Holguín-Veras et al., eingeführtes Konzept der Deprivation Costs. In einem kürzlich erschienenen Artikel wurde dieses Konzept von Gutjahr und Fischer in Hinblick auf den Fairnessaspekt erweitert. In ihrer Analyse wurden die mittleren Deprivation Costs durch Ginis mittlere absolute Abweichung der Deprivation Costs ergänzt um den Fairnessaspekt in Betracht zu ziehen. Dieses Optimierungsproblem wurde mittels einer Metaheuristik, Particle Swarm Optimization, gelöst. Das Ziel dieser Arbeit besteht darin, eine Lösung des Optimierungsproblems von Gutjahr und Fischer mit Hilfe der Gradientensuchmethode zu versuchen. Es werden zwei Verfahren benutzt, das Bisektionsverfahren und der Harmonic Step Decrease. Die nummerische Lösung wird mit der Lösung von Gutjahr und Fischer anhand eines Anwendungsbeispiels mit Daten aus dem Erdbeben in Nepal 2015 verglichen.

Abstract

We consider the model describing humanitarian logistics in the response phase after a disaster with respect to equity. One of the fundamental advances in humanitarian logistics is the deprivation cost concept introduced by Holguín-Veras et al.. In recent article, Gutjahr and Fischer reconsidered this concept additionally, pointed out the equity issue. In their analysis, they modified the deprivation cost objective by a term proportional to Gini index for quantifying inequity and performed the Particle Swarm Optimization metaheuristic. The purpose of this thesis is to investigate the use of the Gradient based search on the same logistics model as Gutjahr and Fischer. We implement the Gradient Search approach with two line search routines, the Bisection Method and the Harmonic Step Decrease. The numeric solution is compared with the Gutjahr and Fischer metaheuristic on illustration case with data from the 2015 Nepal earthquake. It is shown that the use of the Gradient Search method with multistart improves numerical accuracy.

Key words: Humanitarian Logistics, Equity, Gini Index, Deprivation Costs, Gradient Search, Frank-Wolfe Algorithm, Optimal Step, Harmonic Step