Aalto University

Mat-2.4177 Seminar on Case Studies in Operation Research
Final report

Modeling Logistics Network in a Crisis Situation

Client: Defense Forces Technical Research Centre

6.5.2011

Research team:
Kaikkonen, Ilkka (Project manager)
Forsberg, Tero
Kullberg, Niklas
Männistö, Toni

Table of Contents

Table of figures

1 Introduction

In a crisis situation, it is important to provide help to people in distress, and it is also extremely challenging to do that. After serious catastrophe, like an earthquake, to get food, medicine and other relief supplies distributed to people in area requires well-planned operations and lots of resources. Because resources in crisis areas are often very scarce, in order to make help to reach as many people as possible, operations need to be efficient. This adds another level of complexity to the issue of efficient logistic operation, which has been studied intensively. The idea of this project was to provide a first-cut tool for our client: Defense Forces Technical Research Centre to help them in planning this kind of operations

This work is inspired by the earthquake in Haiti on January 2010. The earthquake affected many people, and humanitarian operations on an island were massive. The case example that we are going to simulate takes place in an island state, like Haiti, where serious earthquake happens destroying large part of country's infrastructure and leaving people in trouble. Several countries and humanitarian organizations start immediately planning how to distribute relief supplies to catastrophe area, and how to take care that help will reach as many people as possible. Because of damaged infrastructure, it is beneficial to send not only relief supplies, but also repair forces who can repair destroyed road, food and medicine distribution facilities, harbors, airports and so on. For example, broken road connections can be repaired by repair forces, which speed up distribution network, and more relief flights are able to land on a repaired airport.

We have chosen to concentrate on a single hub network problem with one product and one mode of transport. The model can be used to simulate breaking and repairing of roads at deterministic time. From different decision making situations we have chosen three as a demonstration of the capabilities of the model: 1) Determining critical roads in the network 2) Adjusting truck driver's decision criteria and 3) use of repair forces. We created a process which can be used to determine the critical roads systematically and we have included in this report. This report is structured so that first we go through the model definition then we present the simulated cases and then we provide conclusions and future improvement areas.

2 Simulation model

The developed simulation model simulates performance of a logistics network in a catastrophe region over a certain time period which is determined by a user in the beginning of the simulation.[1] The software consists of five distinctive building blocks which together define geography of the crisis region, characteristics of population areas in the region and resources which are available to carry out a relief operation. A general description of these five blocks is given in Figure 1

[1] The software keeps track on the simulation time by the aid of the counter *k* which counts rounds (= days) from the beginning of the simulation.

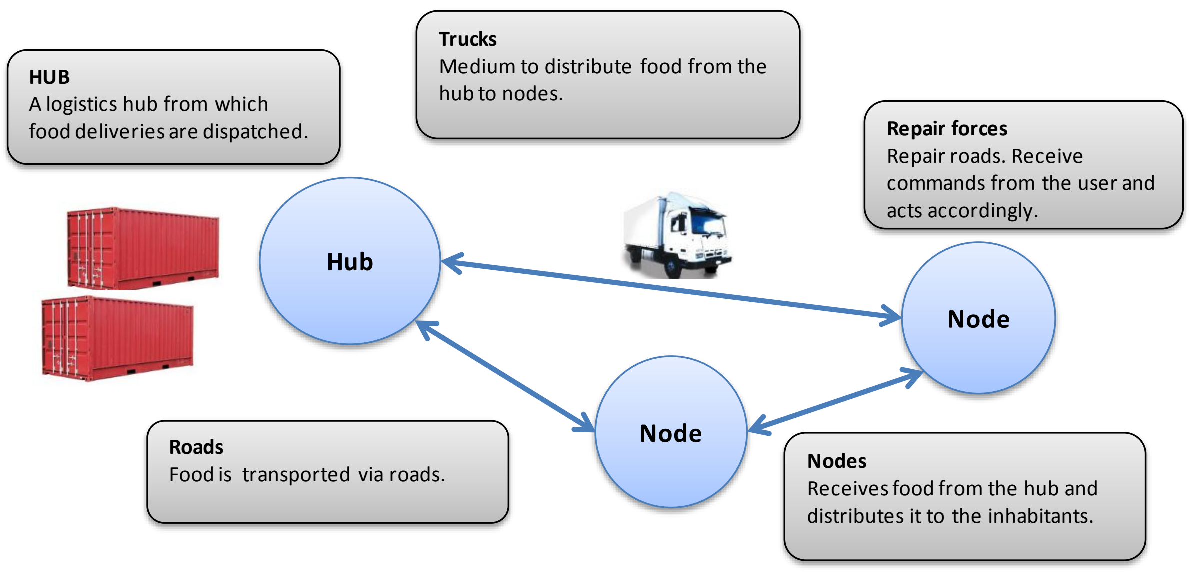

HUB
A logistics hub from which food deliveries are dispatched.
Trucks
Medium to distribute food from the hub to nodes.
Repair forces
Repair roads. Receive commands from the user and acts accordingly.
Hub
Node
Node
Roads
Food is transported via roads.
Nodes
Receives food from the hub and distributes it to the inhabitants.

Figure 1: General description of the simulation model

2.1 Setting up simulation

2.1.1 Geography of region – hubs, nodes and connecting road network

Before any simulation can be carried out, the user has to initiate the simulation by inserting certain parameters. First, the user places logistics hubs, cities and villages on the map of the crisis region. The user is asked to fill in the number of “nodes” (villages, cities and a logistics hub) and distances between these nodes in a time delay matrix in the “distance” sheet of the user interface excel. These inputs determine the map of the crisis region and define the road network, which connects the nodes and the hub. Figure 2 presents a map of a crisis region which will be used as an illustrative example in this report. The map has one logistics hub and five nodes where the population in the region lives. These nodes are connected to each other and the logistics hub by roads. The user is asked to fill travel times for a truck between directly connected hubs and nodes to the excel sheet. To determine the travel time from a hub to a node the user has to consider distance between the locations and average speed a truck can drive on the road connecting the nodes. For instance in the map illustrated in the figure 3, travel time between nodes 3 and 4 is seven times longer than between nodes 3 and 1 even though lengths of the both connections are roughly the same. This is due to the fact that the road between nodes 1 and 3 might be fast highway whereas the road from the node to the node 3 is a slow road over mountains. Not only the distance between two locations affect travelling time but also type of a road between the locations. For sake of simplicity, the simulation model supposes that travel time from a node A to B equals travelling time from B to A.

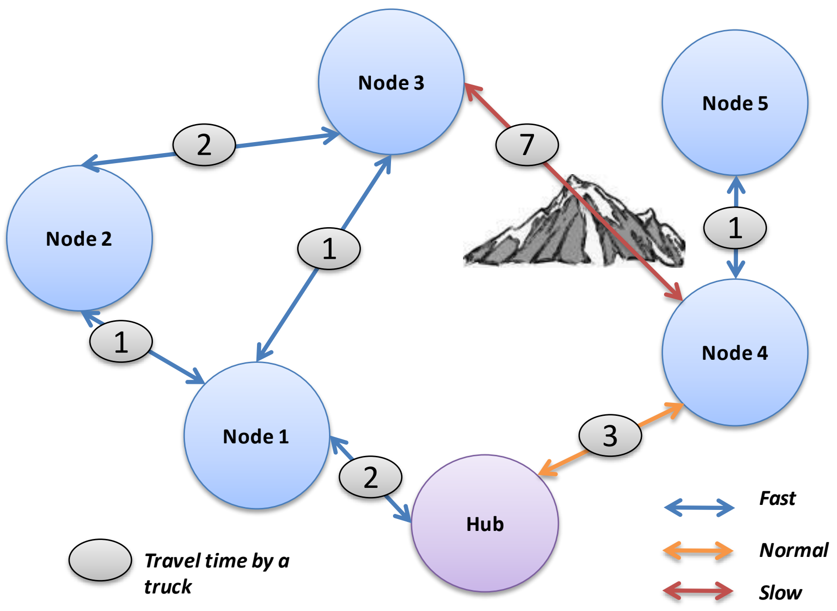
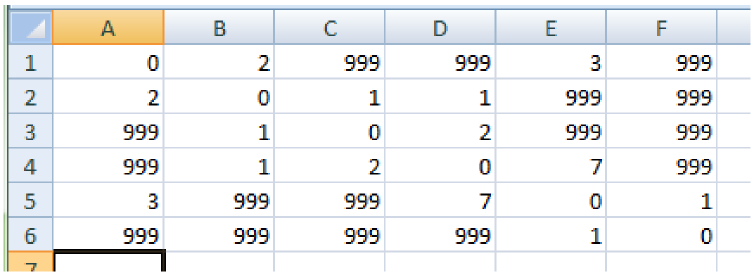


Figure 2: Example map of the crisis region

In case the user wants to simulate a relief operation in a region presented in the figure 2 which has one logistics hub and five population areas i.e. nodes and a road network, he should insert similar inputs like in the Table 1 below too the “distance” sheet in the user information excel. Each cell contains the number of hours needed to travel by a truck from a hub / node i to a node j. One should notice that if there is not a direct route between two nodes or a hub and a node, user fills in 999 in the corresponding cell.

Table 1: Example inputs of the time delay matrix



	A	B	C	D	E	F
1	0	2	999	999	3	999
2	2	0	1	1	999	999
3	999	1	0	2	999	999
4	999	1	2	0	7	999
5	3	999	999	7	0	1
6	999	999	999	999	1	0
7						

Each node consumes a certain amount of supplies per day. This consumption is proportional to the number of inhabitants in the node and is constant over the simulation period. Consumptions in the nodes are reported in “consumption units” [cu] which resembles average daily consumption of 5 000 inhabitants. Each node has also a certain amount of supplies in stock which can be distributed to the inhabitants. This stock level is also reported in consumption units [cu]. The user inserts initial daily consumption and initial inventory levels for each node to the “other_initial_values” sheet in the user interface excel sheet.

One of the nodes is a hub, which in reality represents a major harbor or an airport which receives supplies outside of the crisis region. Hub is also a central warehouse and a logistics center from where trucks which deliver supplies to the nodes are dispatched.

2.1.2 Resources – trucks and repair forces

Trucks are the medium to transport food from the hub to distressed people living in the nodes. Logically, the bigger the fleet of the trucks is the more food can be distributed to the people during a relief operation. The user fills the number of the trucks available for the relief operation in the "Truck" worksheets in the user interface excel.

Trucks use the defined road network of the crisis region to transport food from the hub to the nodes. Because infrastructure of the crisis region is often damaged after an earthquake, flood and so on, some roads in the network are broken and thus useless for the truck fleet at the outset of the relief operation. Moreover, some roads can also get broken in the aftermaths of the crisis (because of additional landslips etc.). In the simulation model, the user can determine which roads will get broken during the simulation period. Commands to break roads are given in user interface excel in the "Breaks_and_fixes" sheet.

The user has a certain number of repair forces on his disposal. These repair forces can be used to fix broken roads. In practice, after the user has inserted commands to break roads in the Break_and_fixes sheet, the user fills in the round when the broken roads are again back in order. This enables the user to easily change efficiency of the repair forces and the number of the repair units at hand by changing how long it takes to fix a road. For instance, with one repair unit which is capable of fixing one road in 7 rounds, inputs of the user to the "breaks_and_fixes" excel sheet should look like this:

Breaks on round:	From node:	To Node:	Will be fixed
5	1	2	12
5	5	6	19

Now if there are two units of repair forces available the same situation looks slightly different. In this time both roads can be fixed simultaneously (observe that node 1 is the hub node n>1 is demand node n-1):

Breaks on round:	From node:	To Node:	Will be fixed:
5	1	2	12
5	5	6	12

The user has a great flexibility to manage the repair forces. For example, if the user wants to prioritize the one road over the other, he can assign both of the repair units to work on the prioritized road. In that case user inputs to the "breaks_and_fixes" excel sheet would look this:

Breaks on round:	From node:	To Node:	Will be fixed:
5	1	2	10
5	5	6	15

The Table 2 below compiles the inputs which are asked from the user before the actual simulation begins. It presents also units in which the required parameters should be inserted in the user interface excel.

Table 2: Values entered by the user

Element:	**Inputs from the user [unit]:**	**Sheet in the user interface excel:**	**Fixed attribute:**
Hub (node number one)	-	-	Infinite stock level
Road network	Number of nodes [n] Time delay matrix [hour]	Distances	Infinitive delivery capacity

Nodes	Consumption [cu] Stock level [cu]	Other_initial_values	-
Trucks	Number of trucks [m]	Trucks	Capacity / truck = 1 cu
Repair forces	Breaks on round [round] From node [node number] To node [node number] Will be fixed [round number]	Breaks_and_fixes	
+ Round counter	Number of simulation rounds	Round	-

2.2 How simulation works?

After the required initial values are given by the user, simulation can begin. Before the first simulation round, inputs of the user are entered to the "main program" which is the kernel of the simulation program. The main program carries out the actual simulation by the aid of calculations which sub-programs have completed. Sub-programs determine shortest route between the hub and nodes and determine routing of delivery trucks over one simulation rounds based on food levels in the nodes, travel time between the hub and the nodes in a given round. The main program aggregates information, carries out simulation and prints out graphs about inventory levels in the nodes and cumulative penalty function of the logistics network over the overall simulation period. Figure 3 illustrates the interplay between the user interface, the main simulation program and the supportive sub-programs.

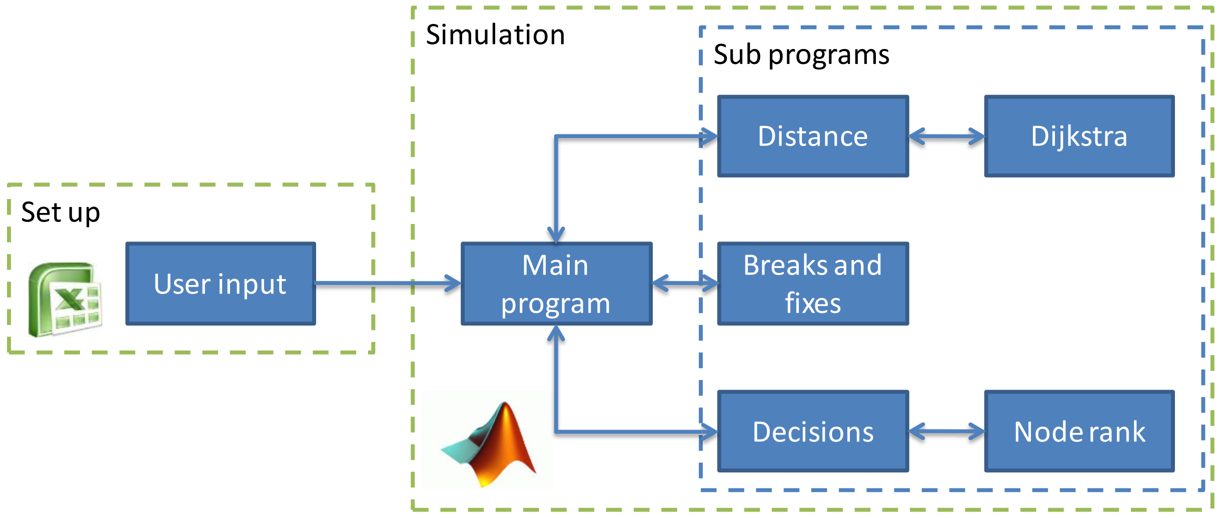


Figure 3 Interplay of the sub-programs

2.2.1 Shortest paths between hub and nodes

The shortest paths between the hub and nodes must be calculated before every simulation round because the time delay matrix changes from round to round when roads get broken and fixed eventually by the repair forces. "Breaks and fixes" sub-program breaks and fixes roads according decision of the user which he inserted in the "breaks_and_fixes" sheet in the user interface excel at the outset of the simulation.

Consequently, the sub-program creates a new time delay matrix for an ongoing simulation round which includes information about broken roads.

The new time delay matrix is fed via the main program to the "distance" sub-program which calculates new shortest paths between the hub and the nodes by the aid of Dijkstra algorithm. The shortest paths are returned back to the main program which sends them to the "decision" sub-program.[2]

2.2.2 Decision program – how trucks determine their destination?

The decision sub-program is the brain of the simulation program. It determines where trucks deliver food during one simulation round. The key idea of the decision sub-program is to rank the nodes according to a priority function so that on each decision round the first truck picks the highest ranking node, second truck picks the second highest ranking node and so forth. The priority function is presented in the form of an additive value function. Additive value function allows acknowledging several decision attributes, allows using different weights to reflect their relative importance in decision making and easily scalable to accommodate more decision making attributes.

Additive value function consists of decision making attributes (x_1^*,x_2^*,…,x_n^*), value functions (v1, v2,… , vi) and weights (w1,w2,…, wi) where n is the number of nodes and i is the number of different attributes. Value functions Vi(xi) assigns a value between one and zero to each node. The node which is ranked the first receives the value one and the least receives zero. We use these two values to gain the slope coefficient for the first degree polynomial (affine transformation). Using the polynomial attribute specific values between one and zero can be calculated for each node and each attribute. The next step is to sum up the attribute specific values. Weights are used to reflect the preferences of the decision maker which enables mapping each function on a scale from 0 to 1. The formal form of additive value function is:

$$\boldsymbol{V^n(x)} = \sum_{\boldsymbol{i=1}}^{\boldsymbol{n}} \boldsymbol{w_i v_i^n(x_i)}$$

We chose to use following decision making attributes:

i. Travel time between the hub and the node
ii. Inventory level in the node
iii. Penalty function in the node

Length from hub to node is chosen because all things remaining equal the truck should choose the closest node. Thus the node farthest from the hub receives the value 1 and the node nearest to the hub receives 0. Days inventory lasts is calculated by dividing inventory level by consumption. It was chosen as an attribute because since it combines the two variables in a meaningful way. If the warehouse levels are same in all the nodes the truck should go there where they are consumed the fastest. Thus the smallest value receives 1 and the largest value receives 0. The penalty function is used because the truck should go there where the penalty function is highest if all other parameters are on the same level. Naturally the highest penalty function value is assigned 1 and the smallest zero.

[2] We used a code written by: Author: Jorge Ignacio Barrera Alviar which can be found at http://www.mathworks.com/matlabcentral/fileexchange/14661-dijkstra-very-simple

How the weights are assigned is based on decision maker's preferences. We have, however, used the weights to optimize the functionality of the network. The optimization can be done using an optimization algorithm, but we did it manually by testing the functionality with different attribute weights. Then we conducted sensitivity analysis on weights. The optimization must be repeated each time the network is changed.

2.2.3 Main program simulates and draws results

The main program includes the simulation algorithm which compiles information from the sub-programs, simulates the logistics network on the basis of this information and draws figures which illustrate the successfulness of the relief operation. The algorithm plots inventory levels in the nodes for each simulation rounds and the number of days nodes have been without food. These two measures provide relevant information about have successful could be carried out in a crisis region with certain resources.

The most intuitive measure for keeping track on the performance of the logistics network is human suffering in the crisis region. People in the nodes suffer if they do not get food. This scenario realizes every time the stock in a node is empty (stock level = zero consumption units). Distress in the nodes increases every day the node is without supplies. This dynamic has been taken into account in the penalty function P which illustrates the human suffering. The penalty function grows exponentially after each sequential time steps when a node is without food. The penalty function[3] for a node *i* at the round k is defined as:

$P_{i,k}$ = consumption in the node *i* * (sequential days without food)$^{\mathbf{2}}$

Sum of the penalty function values in each node indicate the total human suffering in the catastrophe region at the given round k.

$$\sum_{i=1}^{n} p_{i,k}$$

Cumulated penalty function modeling human suffering over the duration of the simulation (K rounds) **evaluates** the success of the relief operation.

$$\sum_{k=1}^{K} \sum_{i=1}^{n} p_{i,k}$$

The main program plots inventory levels and penalty functions for individual nodes and the network as a whole. These graphs provide invaluable insight about the dynamics of the crisis logistics. The user can see which nodes were suffering most in the current situations.

In general, current version of the simulation model can print out following graphs and illustrations[4]:

- Cumulative penalty function (graph)
- Individual penalty function values for each node (bar chart)

[3] In the MatLab code the penalty function is referred as "penalty" function.
[4] More graphs and relevant information about the performance of the logistics network can be printed out of the program rather easily by adjusting MatLab code in the main program.

- Inventory levels in nodes (bar chart)
- Days without food in nodes (bar chart)
- Deliveries to the nodes (bar chart)

When the user has considered the results, they can run the simulation again with different parameters. He could break more roads or carry out the operation with a smaller number of trucks and simulate how the changes affect to the cumulative penalty function of the network. The user gets a good understanding of the crisis region after a dozen of well-adjusted simulations.

3 Simulation cases

3.1 Introduction

This chapter demonstrates functionalities of the simulation software with three simulation cases. The first case shows how criticality of individual roads in a road network can be determined. The second case elaborates how weights in the priority function affect dynamics of the simulation (trucks make decision based on values of the priority function). The third case illustrates how repair forces function.

All simulations have been carried out with following initial parameters unless otherwise stated (simulation 2):

- Simulation rounds: 90
- Road network defined is the same as in the figure 2
- Number of trucks: 5[5]
- Consumptions in the nodes: $c_1 = 9$, c_2 4, $c_3 = 2$, $c_4 = 1$, $c_5 = 4$
- Inventory level in each node is zero in the beginning of the simulation
- Penalty function weights: 0.1 for length; 0.45 for stock level; and 0.45 for penalty[6]

3.2 Simulation one – determining critical roads

3.2.1 Process of identifying critical roads

Finding the critical roads is not a trivial task. This is because there is more than one factor affecting the level of criticality. Each node has a different consumption which can be derived from the number of people living in the node. The consumption level has been taken into account in the penalty function which models the level of human suffering. Thus, if a road leads to a node with high consumption it can be considered as more critical than a road which leads to a node which has a lower consumption level.

Another significant factor affecting the criticality of a road is the number of alternative routes leading to a destination node. So if the destination node has several roads leading to it, the criticality of a single road decreases. In cases where there are several roads leading to a node, the road which fastest deliveries from

[5] This is the size of the truck fleet which is required to satisfy demand of all nodes when the road network remains intact over the simulation.

[6] We tested many variants, and this seems to give quite realistic results.

the hub to the node can be considered as the most critical road among the alternatives. Therefore, also a road leading to a node with low consumption can be critical if the node has several outgoing roads - the node functions as a gateway node. Thus the number of outgoing roads affects the level of criticality of ingoing roads.

The process for determining the critical roads in a network is:

1) Optimize the logistics network when all roads remain intact over the simulation. This allows the user to find the minimal level of penalty function.
2) Break all the roads at a certain time step so that only one road is broken per simulation. Record values of the penalty function in each simulation.
3) Sort the roads from largest to smallest according to the values of the penalty function.
4) Determine which roads are taken into closer observation depending on capacity limitations and tolerated level of suffering

This process is demonstrated in the 5 parts of the simulation 5. In the first part the performance of the road network is studies when none of the roads in the network gets broken. The remaining four parts of simulation one show what happens to the penalty function when different road between hub / node and a node breaks at round 5.

3.2.2 Part 1 - Roads remain intact

In the first part of the simulation 1, all roads in the road network function over the simulation period. With this settings the cumulative penalty function which evaluates the successfulness of the relief operation remains zero meaning that there are no days without food at any node. Figure 4 shows the number of deliveries arrived in the nodes over the simulation period, and it illustrates that there is not much variance in the deliveries. It can be seen that each node is served properly. The bulk of the deliveries are sent logically to node one, which is the biggest city and thus has the highest consumption. The average delivery per day is close to the daily consumption in the nodes, except for the node one, which is served a bit more than it consumes. Figure 5 strengthens this observation, as it shows that inventory levels increase slightly in all the nodes, but most in the node 1.

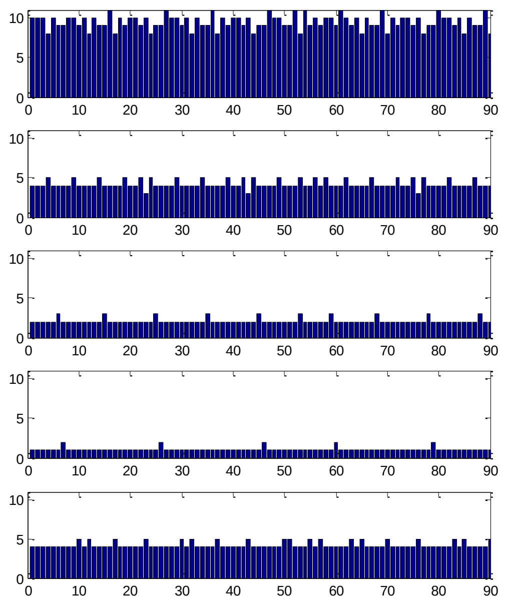


Figure 4: Deliveries to nodes (simulation 1, part 1). Node number one at uppermost.

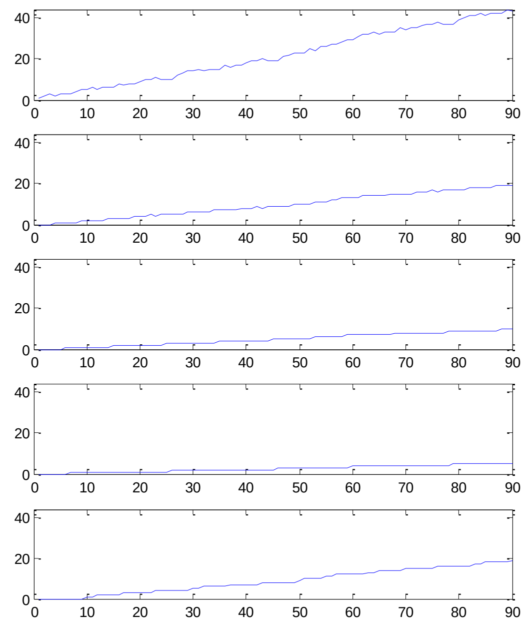


Figure 5: Inventory level (simulation 1, part 1)

3.2.3 Part 2: Road between hub and node 1 breaks

In the second part of the simulation one, the road between the hub and the node 1 breaks which causes a significant increase in the cumulative penalty function. The value of the penalty function is staggering over 4.000.000 which signifies that people in the crisis region are having hard time. It can be concluded that if the road between the hub and the node 1 is not fixed, a lot of human lives will be lost.

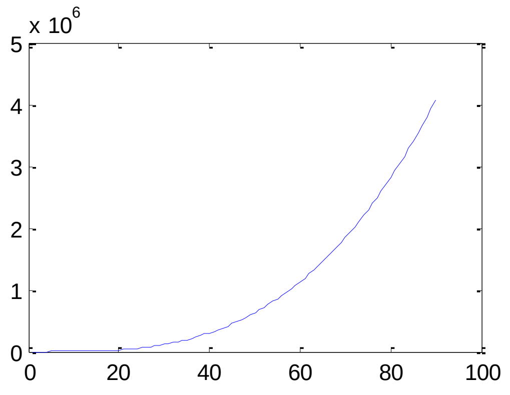


Figure 6: Cumulative penalty when road between hub and node 1 breaks at t = 5 (simulation 1, part 2)

3.2.4 Part 3: Road between hub and node 4

When the road between the hub and node 4 gets broken at round five, the cumulative penalty function increases slightly reaching value of approximately 12.000 in the end of the simulation. The road is also important, because penalty function increases. However, impact of the break of this road is only marginal when comparing it to scenario presented in the second part of the simulation 1.

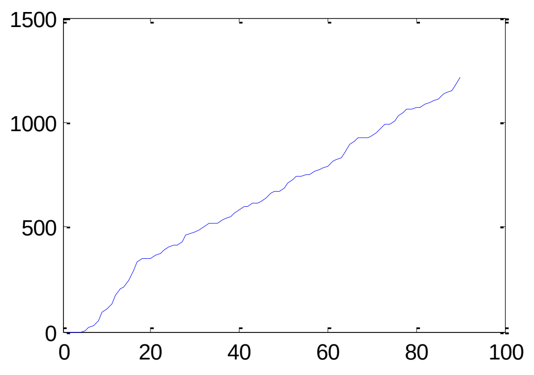


Figure 7: Cumulative penalty function when road between hub and node 4 breaks at t = 5 (simulation 1, part 3)

3.2.5 Part 4: Road between node 1 and node 2:

In the fourth part of the simulation one, the road between node 1 and node 2 brakes at round 5. The breakdown disrupts deliveries but still, cumulative penalty function values remain within tolerable limits (reaches 340 at round 90). Impact of the breakdown of the road between node 1 and node 2 cause only small impact in comparison to the two previous breakdown scenarios

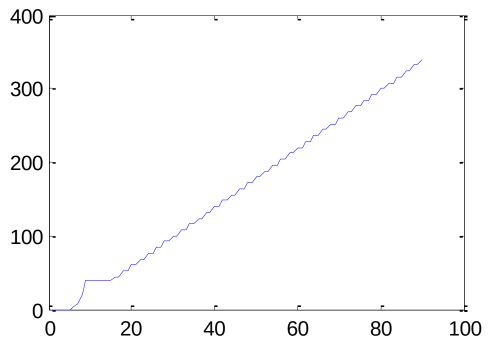


Figure 8: Cumulative penalty function when road between node 1 and node 2 breaks at t = 5 (simulation 1, part 4)

3.2.6 Part 5: Road between node 1 and node 3

Now the value of the cumulative penalty function after 90 rounds is 28, denoting that the influence of the breakdown of this road is rather small.

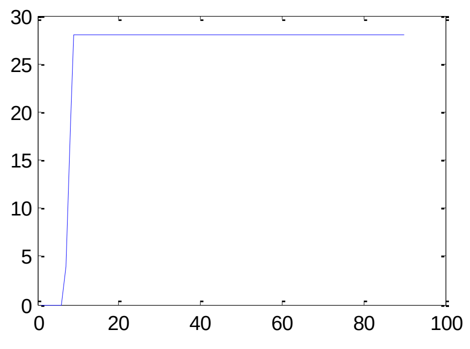


Figure 9: Cumulative penalty function when road between node 1 and node 3 breaks at t = 5 (simulation 1, part 5)

3.2.7 Part 6: Road between node 2 and node 3:

From node 2 to node 3 it takes two hours when using the straight road, and two hours when travelling via node 1. Because the travel times equal, breaking the road between node 2 and node 3 does not have any impact. Thus the results in the 6th part are exactly the same as in the first part which simulated the performance of the logistics operation when none of the roads went broken.

3.2.8 Part 7: Road between node 3 and node 4:

When all other roads are in order, there is no need to use the road between node 3 and node 4. Hence, breaking this road does not have any impact on the results of the simulation in comparison to the first part of the simulation.

3.2.9 Part 8: Road between node 4 and node 5:

In this scenario, the cumulative penalty function soars after the breakdown of the road between node 4 and node 5. The cumulative penalty function after round 90 is 860.000. Impact of the breakdown of this road is significant because it blocks the only access to the node 5.

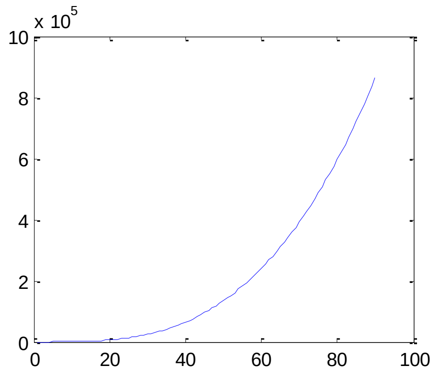


Figure 10: Cumulative penalty function when road between node 4 and node 5 breaks at t = 5 (simulation 1, part 2)

In conclusion, we can determine that the road between hub and the node 1 is the most critical. The second most critical road is the road between the node 4 and node 5. Since these cumulative penalty function for these two roads is significantly higher than for the other roads all mitigation actions should be taken on these two roads.

3.3 Simulation two – adjusting weights in priority function

The second simulation is about testing how the simulation model behaves with different weights of the prioritization function. When recalling the second chapter of this report, the priority function is used to determine which nodes trucks serve during one simulation round. The prioritization function has following formula

$$V^n(x) = \sum_{i=1}^{n} w_i v_i^n(x_i)$$

, where v_i are decision variables and w_i their corresponding weights.

The true nature of the used decision variables are presented in table 3 below.

Table 3: Nature of the decision variables

Weight number	Decision variable
w_1	Travel time from hub to node
w_2	Inventory level in the node
w_3	Consumption in node
w_4	Node's individual penalty function value

In the first simulation the used weights were $w_1 = 0,1$, $w_2 = 0,45$, $w_3 = 0$[7] and $w_4 = 0,45$. In this second simulation we conduct a sensitivity analysis in case where the road between the hub and node 4 break after round 5. This particular case is chosen for the sensitivity analysis because in this scenario the value of the cumulative penalty function is not extremely high or insignificantly small (value = 1.218).

3.3.1 Part 1: weights are equal ($w_1 = w_2 = w_4 = \frac{1}{3}$)

In the first part of the simulation, the cumulative penalty function value is remarkable high (833.410). Node 5 does not get enough deliveries due to its long distance from the hub. As figure 11 illustrates, there are some problems with deliveries to node 5 already before the breakdown of the road, when distance (w_1).has this as large weight as inventory (w_2).and node's penalty function value (w_4). This kind of delivery allocation is not realistic and far away from an optimal solution.

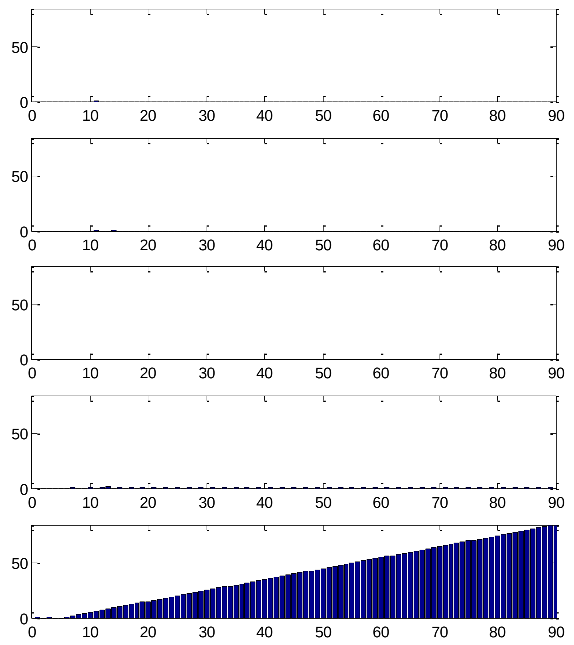


Figure 11: Days without food with priority function weights $w_1 = w_2 = w_4 = \frac{1}{3}$ (Simulation 2, part 1)

3.3.2 Part 2: Small weight for inventory ($w_1 = 0,45$, $w_2 = 0,1$, $w_4 = 0,45$)

When the weight of distance and a node's individual penalty functions is increased in the detriment of the importance of inventory levels in the nodes, the cumulative penalty function is not more than 1.033. With

[7] (w_3 was eliminated from the model because consumption has already impact on w_2 and w_4

this set up of the weights, cumulative penalty function remains low because No node has to be without food longer than three days (figure12). Every node suffers from days without food but on the other hand, every node can expect to receive food deliveries before the situation gets really bad.

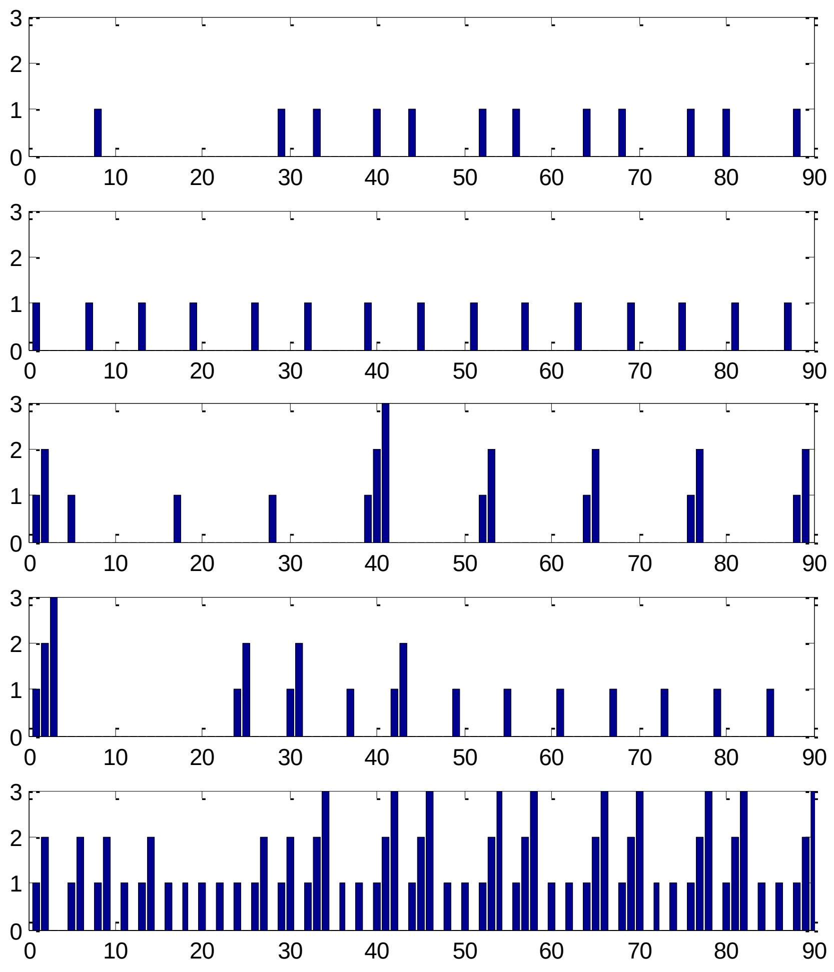


Figure 12: Days without food in the nodes (w1 = 0,45, w2 = 0,1, w4 = 0,45) (Simulation 2, part 2)

3.3.3 Part 3: Small weight for penalty function value ($w_1 = 0{,}45$, $w_2 = 0{,}45$, $w_4 = 0{,}1$)

When the penalty function value has only a small weight, cumulative penalty function rises to alarming levels (748.676). As figure 13 shows, now there is again a serious lack of food in node 5. Even though the penalty function value in the node four rises strongly, its long distance from the hub as well as low consumption rate keeps its priority low, when the penalty function value has only 10 % weight. This kind of delivery allocation is not realistic and far away from an optimal solution.

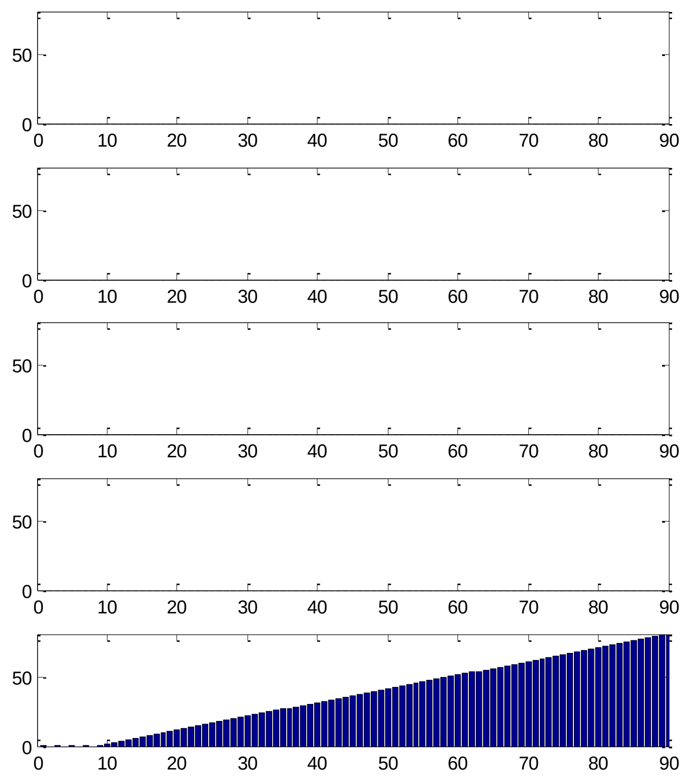


Figure 13: Days without food with weights w1 = 0,45, w2 = 0,45, w4 = 0,1 (Simulation 2 part 3)

3.4 Simulation three – repair units on duty

The third simulation takes demonstrates functionality of the repair units. In this simulation, two important roads between the hub and node 1 and between node 4 and node 5 get break on round 5. In the first part, there are only one repair unit available which is capable of fixing one broken road in seven rounds. In the second part, the user has two units of repair forces at hand. First we assume that there is only one repair unit that is able to repair one link in one week seven rounds. Next, we add to the model another repair unit, so both the roads can be fixed at the same time.

3.4.1 Part 1 – One repair unit

In the first part of the third simulation, the repair unit fixes first the most important road between the hub and node 1. After the repair unit has completed the first assignment, they move on to the second road between node 4 and 5. In this scenario, the cumulative penalty function rises strongly as soon as the roads get broken on round 5. However, the logistics system returns back to a stable state soon after the repair unit has fixed both roads. Cumulative penalty function value is 6318 in the end of the simulation on round 90.

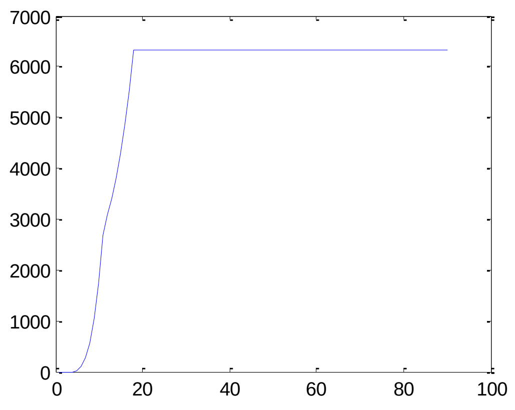


Figure 14: Cumulative penalty with one repair unit (Simulation 3, part 1)

A more detailed illustration (figure 15), shows that every node suffers from lack of food to some extent. Right after the first road between the hub and node 1 has been fixed, demand of nodes 1-4 can be fulfilled. However, node 5 suffers most the situation. Deliveries there can be continued only after the second road has been fixed. The situation in node 5 around round 18 is critical.

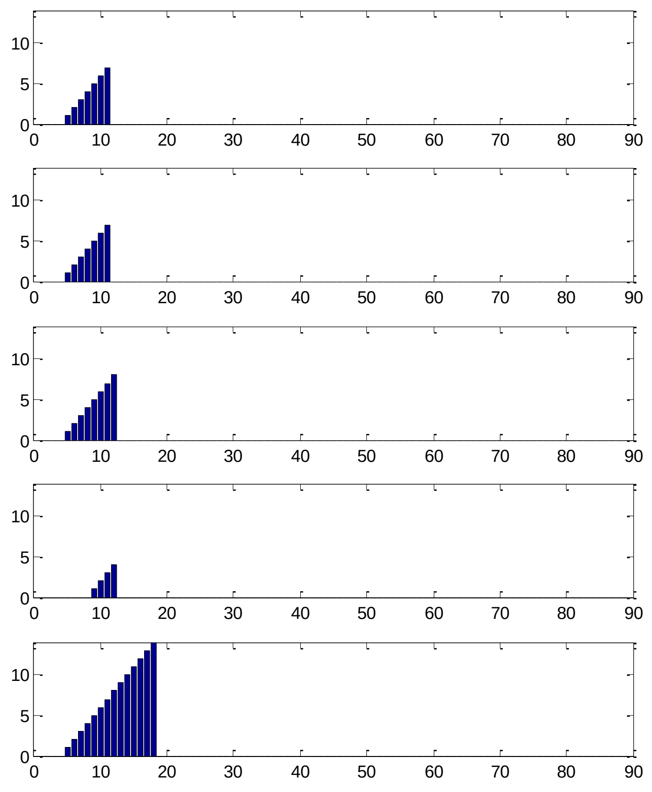


Figure 15: Days without food with one repair unit (Simulation 3, part 1)

Figure 16 show explicitly that node 5 does not receive any deliveries between the breakdown of the roads and the time when the repair forces have fixed both of the roads.

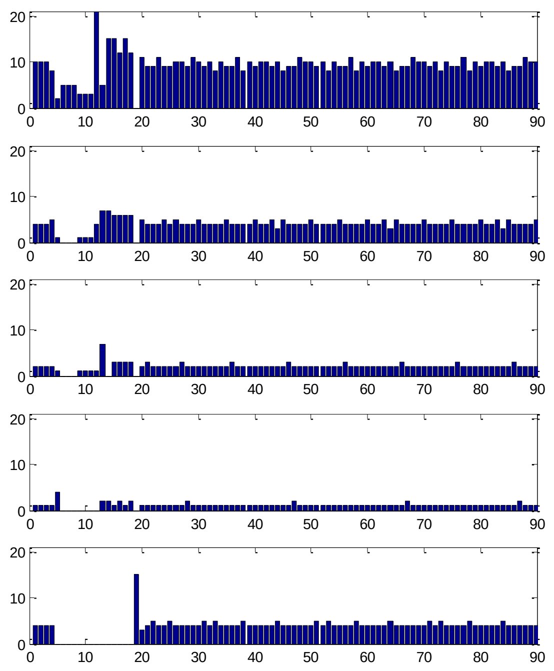


Figure 16: Deliveries to nodes with one repair unit (Simulation 3, part 1)

The similar simulation is done with repairing first the road between node 4 and 5, and then the road between the hub and node 1. The cumulative penalty function is considerably higher (21550) than in the case the resources were allocated first to road between hub and node 1.

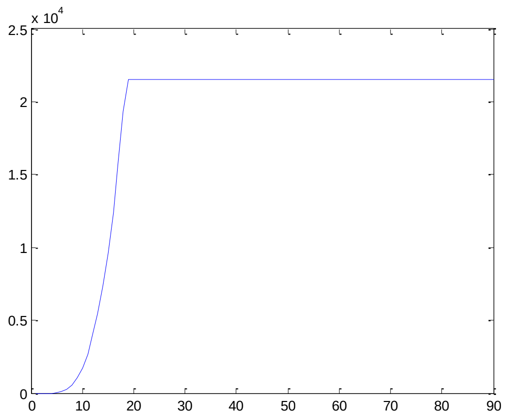


Figure 17: Cumulative penalty function when repair unit is first allocated to the road between hub 4 and 5 (Simulation 3, part 1)

All nodes except node four will suffer or remain indifferent from this resource allocation. When taking into consideration the amount of people that are left without food it can be concluded that the situation is worse than when allocating resources first to road one.

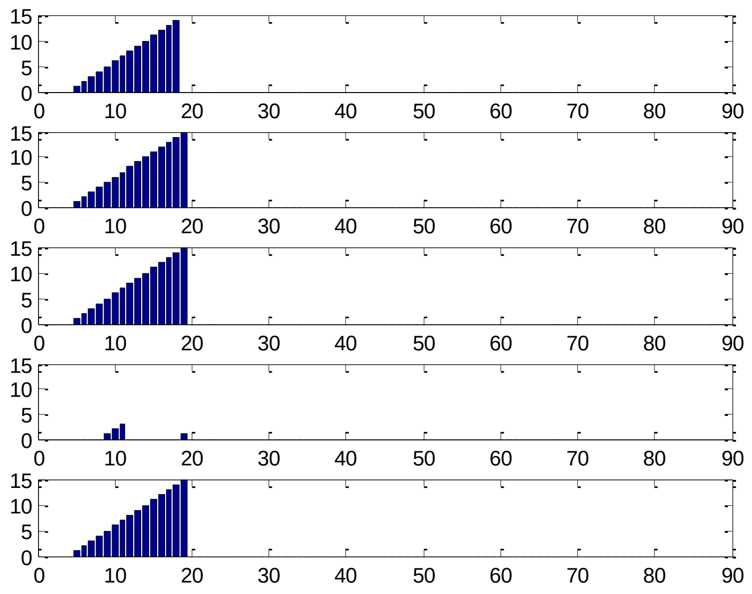


Figure 18: Days without food with one repair unit when road between node four and five repaired first (Simulation 3, part 1)

Figure 19 show explicitly that the amount of deliveries is significantly lower than when repairing first road between hub and node 1.

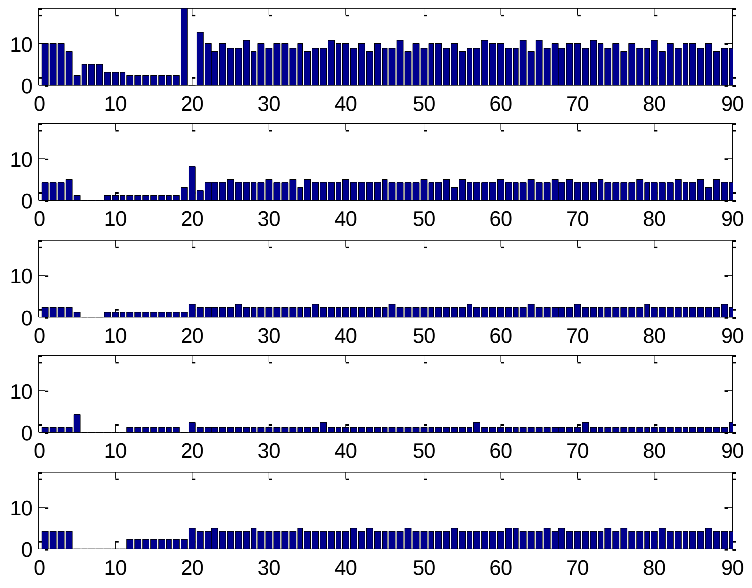


Figure 19: Deliveries to nodes with one repair unit when road between node four and five repaired first (Simulation 3, part 1)

The results indicate that it is more important to allocate resources to road between hub and node one.

3.4.2 Part 2: two repair units

The second part of the simulation three differs only slightly from the first part. Now the user has two repair units available and thus the road network can be fixed back to order faster than in the first part. The two repair units fix two broken road connection at the same time. Therefore, the road network is completely back after round 12. In this case, the situation in the node 5 does not escalate and thus the cumulative penalty function value remains at a considerably low level (3330). The cumulative penalty function for the whole logistics network is presented in the figure 20.

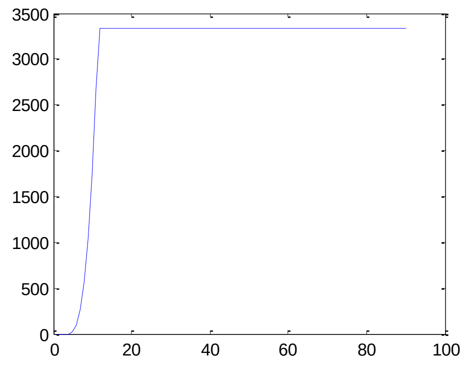


Figure 20: Cumulative penalty with two repair units (Simulation 3, part 2)

When taking a look on days without food, it can be seen that situation is worsening as fast as in the first part of the simulation but because both repairing of the both roads are completed at the same time, the logistics network returns back to the stable state seven rounds faster than in the first part of the simulation.

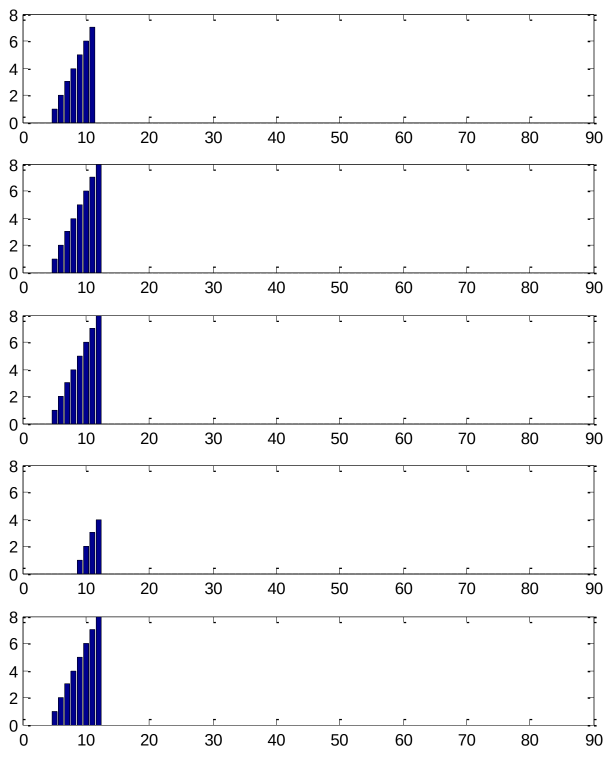


Figure 21: Days without food with two repair units (Simulation 3, part 2)

Also inventory levels in the nodes start to build up after round 12 when the both roads have been fixed (Figure 22). At the end of the 90 round situation in the crisis region seems rather good. All nodes have food but also some safety stock to survive if any disruptions in supply should occur.

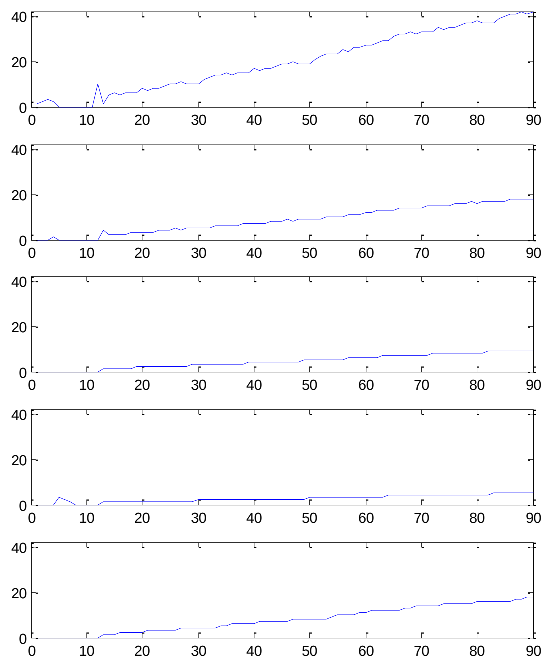


Figure 22: Inventory level with two repair units (Simulation 3, part 2)

3.5 Results and discussion

The simulation model is designed to assist decision making in a humanitarian logistics operation. As the model simulates the functionality of a logistic network with one hub, it can be used to assist in a variety of decision making situations, for example which road should be emphasized in a situation where two roads break given that the organization has only limited repairing capacity. For the purpose of demonstrating the functionality of the simulation model we have decided to study how the model can be used to assess how critical certain roads in the network are. When roads are classified according to their criticality level the decision maker can take corrective actions such as fortify weak points, such as bridges, along the road.

In the investigated scenario, we found in the first simulation that road between hub and node one is the most critical road for preventing from human suffering if no repair forces are available for a event time of 90 days. Even though the road between node four and five is the only link to node five, the amount of people and distance around the mountain result in a greater suffering level. Therefore, in this kind of situation road between the hub and node one should be given the largest attention for preventive actions. The result is intuitive, but not trivial.

In our second simulation we demonstrated how different weights affect the decision making of truck drivers when the road between the hub and node four is broken. We found that emphasizing the penalty function either with inventory or distance in decision making gives lowest penalty values. This result is not a surprise because weighing less the penalty function does not directly take into account suffering. With low priority on the penalty functions, a humanitarian catastrophe would arise in node five, and a majority of the people in the district would die without moving to another district. The results of the sensitivity analysis on weights in the priority function help the decision makers to decide on the priorities when deciding upon the coordination of deliveries.

In the third and last simulation we illustrated how different amounts of resources in the repair forces affect the levels of suffering. We can see that when repairing road one first, the suffering is concentrated to node 5, which is left without food for two weeks. When repairing road between nodes four and five first, the situation is significantly worsened and all nodes except node four will be left without food as long as when repairing the other road first. This is an important result if the resources are limited to one repair unit. When two repair force units are deployed, a significant decrease in suffering can be recognized in node five. The simulation illustrates how important it is to repair critical roads as soon as possible from the time of breakdown and helps the decision maker in resource allocation of repair forces when there is not enough of capacity to fix all damages.

4 Conclusions and future development areas

4.1 Model validation

Although the simulation software models a relief operation satisfactorily and therefore fulfills its intended purpose, the software has a set of inconsistencies which undermine its usefulness in real-world situations. First, the model is able to deal with only one commodity while in real world relief operations other commodities such as food, medicines, shelters and other commodities is distributed to people in the crisis region. Second, usually a relief operation is carried out by utilizing various modes of transportation which travel at different speed and have different transportation capacity. The current simulation model assumes unrealistically that only one kind of trucks can be used in the operation. Third, the model assumes that every truck returns back to the hub in the end of a simulation round (after 24 hours). In a real-world emergency situation trucks are not forced to return back to the main base before midnight. Fourth, the current model has only one logistics hub which receives supplies and which dispatches trucks to nodes. In reality a crisis region could have multiple logistics hubs which could vary in terms of warehousing, receiving and delivery capacity and infrastructure (a landing strip or not). Besides, there could be smaller intermediate warehouses which disassemble bigger deliveries and distribute supplies forward to the nodes in smaller quantities.

Taking these limitations into account, the simulation software models a real world relief logistics operation satisfactorily but not perfectly. Naturally, the aforementioned inconsistencies must be settled before the model can be considered as a full-fledged simulation tool which provides high quality information to support decision making process in a real emergency situation.

4.2 Usefulness of the simulation program for the client

Although we did not manage to include all the specifications in the final Matlab code we are quite satisfied with the end result. The software is able to simulate a network which consists of one logistics hub and any number of nodes which consume one commodity (food). The nodes are served with delivery trucks which number is can be any positive integer. Roads in the crisis region can get broken and they can be fixed by the aid of repair forces. The user of the simulation software is capable of changin easily number of nodes, travel times between them, number of trucks and repair units available. Besides, the user can adjust weights in the priority function as well as change structure of the penalty function. The simulation software has a user friendly interface and the underlying code is easy to understand and designed to be as simple as possible to ensure fast execution of the code.

Taking into consideration all functionalities and beneficial attributes of the simulation software, the client gets a scalable easy-to-use simulation model which can be conveniently adjusted for various settings. The client can use the simulation software as a solid platform for future development towards a comprehensive simulation model which can be utilized in situations where real human lives are at the stake.

4.3 Future improvements

During the project several things emerged which inhibited us from fully completing the project description. To increase the usefulness of the results we give some improvement areas we think are relevant for the client.

We believe that the next step to take the model forward is to add another more hubs and thus expand the network. The key issues to solve when expanding then number of hubs relates to multi allocation or single allocation problem. If the nodes can be allocated to more than one hub the problem is more complex than in the case of single allocation when scaling up is very straight forward. However the multi allocation problem can be addressed by allocating the nodes to a single hub in the beginning of each decision round since an optimal solution can be found each time.

Second step would be to add more products. Our definitions allow this in theory, but some problems may arise. First of all the problem with different product sizes may cause unexpected network behavior. For instance delivering vaccine shots for a village of 500 people should be performed with different vehicle than delivering basic food supplies to the same village. Thus this step goes hand in hand with scaling to more than one mode of transport. Problem grows even more complicated with increased number of vehicles since one must allow changing the vehicle in some instances to make the model interesting.

Adding costs to the model would be our recommendation for the third step. Although adding cost per se is very straight forward, since length travelled can simply multiplied by some cost factor and the cost of goods can also be calculated, using them to optimize network behavior is more complex. If costs are considered as constraints, the problem could be in theory turned into a minimum cost problem. We however weren't able to construct this problem in a manner which would be interesting to the client. For instance if costs are emphasized too much the network would behave so that some nodes would not receive any deliveries. Thus we recommend that costs are simply mapped and maybe added to the decision making program to make sure that all things remaining equal the truck chooses the cheapest route.

There are also other improvement areas but these three are the ones we think are the most relevant. For instance repair forces could be more versatile and a better user interface could be created. We would like to recommend that all future improvements are made so that they address a specific decision making problem. Adding more degrees of freedom into the logistic network will decrease the accuracy of the results as the behavior becomes less predictable.

Appendix A - Main program Matlab code

```
%Initial values from parametres.xls
Rounds=xlsread('parametres.xls',4); %Simulation rounds
Trucks = xlsread('parametres.xls',3); %Number of trucks
L_orig=xlsread('parametres.xls'); %Lenght from hub to node i
[NodesTEMP,~]=size(L_orig);
Nodes=NodesTEMP-1;

%Changes all values of L_orig matrix that are over 100 to infinities
[M,N] = size(L_orig);
for m=1:1:M
    for n=1:1:N
        if L_orig(m,n)>100
            L_orig(m,n)=inf;
        end
    end
end

L=L_orig;

dwf=zeros(1,Nodes); % days without food for node i

%Intial values from parametres.xls
initials=xlsread('parametres.xls',5);
[~,N]=size(initials);
x=initials(1,1:N);
c=initials(2,1:N);
h=initials(3,1:N);

%x=zeros(1,Nodes); %Inventory level
%c=[8 5 2]; %Consumption
%h=zeros(1,Nodes); %Penalty level
cumulativeh=0; % cumulative penalty
w1=ones(1,Nodes)*0.1; % weights (distance)
w2=ones(1,Nodes)*0.45; % weights (inventory)
w3=ones(1,Nodes)*0;%(0.25/2); % weights
w4=ones(1,Nodes)*0.45; % weights (penalty)

%Information about broken roads and repairing schedule
bf=xlsread('parametres.xls',2); %Matrix that includes rounds when roads between
given nodes (2. and 3. column) breaks (1. column) and get fixed (4. column)
[M,~]=size(bf);

break_rounds=bf(1:M,1);%Vector that has information about breakdown rounds
break_roads=bf(1:M,2:3);%Matrix that has information about roads that breaks

fix_rounds=bf(1:M,4);%Vector that has information about fixing rounds
fix_roads=bf(1:M,2:3);%Matrix that has information about roads that breaks and
gets fixed

%Matrixes that collect information for plotting
plot_cumulativeh = [];
plot_h =[];
plot_p =[];
plot_x=[];
plot_dwf=[];

%Simulation loop
```

```
for j=1:1:Rounds
    L=BreaksAndFixes(j, L, L_orig, break_rounds, break_roads, fix_rounds, 
fix_roads); %Changes the L matrix if road breaks or gets fixed
    l=distance(L); %Calculates the l vector (shortest distances from hub) from L 
matrix
    p=decision(Trucks, Nodes, l, x, c, h, w1, w2, w3, w4); %Calculates the 
delivery decisions of the round
    x=x+p-c; %Calculates the inventory levels based on the delivery decisions 
and consumption
    %Calculates the number of days without food in nodes
    for i=1:1:Nodes
        if x(i)<0
            m(i)=1;
        else
            m(i)=0;
        end
        if m(i)==1
            dwf(i)=dwf(i)+1;
        else
            dwf(i)=0; %Sets the days without food calculator at zero if there is 
food in node
        end
        %Sets inventory level to zero if it is negative
        if x(i)<0
            x(i)=0; %if x below zero ==> zero since state cannot be negative
        end
    end
    %Calculates the penalty level in the nodes
    h=c.*(dwf.*dwf);
    %Calculates the cumulative penalty of the network
    cumulativeh=cumulativeh+sum(h);
    %Stores the following variables for plotting
    plot_cumulativeh=[plot_cumulativeh cumulativeh];
    plot_h=[plot_h h'];
    plot_p=[plot_p p'];
    plot_x=[plot_x x'];
    plot_dwf=[plot_dwf dwf'];
end
%Draws the figures
figure(1);
plot(plot_cumulativeh)
[sizeH,~]=size(plot_h);
[sizep,~]=size(plot_p);
[sizex,~]=size(plot_x);
[sizedwf,~]=size(plot_dwf);
for i=1:sizeH
        figure(2);
    subplot(sizeH,1,i);
    bar(plot_h(i,1:end))
    if max(max(plot_h))>0
        axis([0, Rounds, 0, max(max(plot_h))])
    else
        axis([0, Rounds, 0, 1])
    end
end
for i=1:sizep
    figure(3);
    subplot(sizep,1,i);
    bar(plot_p(i,1:end))
    axis([0, Rounds, 0, max(max(plot_p))])
end
```

```
for i=1:sizex
    figure(4);
    subplot(sizex,1,i);
    plot(plot_x(i,1:end))
    axis([0, Rounds, 0, max(max(plot_x))])
end
for i=1:sizedwf
    figure(5);
    subplot(sizedwf,1,i);
    bar(plot_dwf(i,1:end))
    if max(max(plot_dwf))>0
        axis([0, Rounds, 0, max(max(plot_dwf))])
    else
        axis([0, Rounds, 0, 1])
    end
end
```

Appendix B - Matlab code for NodeRank algorithm

```
function [priorityvector]=NodeRank(l, c, xTEMP, w1, w2, w3, w4, h)

%
% l=[3 3 2]; %Lenght from hub to node i
% c=[3 5 2]; %Consumption (units/days)
% x=[1 2 1]; % warehouse level (Tero: Käytetään tässä functiossa xTEMPiä,
% %joka päivittyy rekkojen päätösten mukaan
% xTEMP=[1 2 1];
% Nodes = 3;
% w1=ones(1,Nodes)*(0.5/3); % weights
% w2=ones(1,Nodes)*(0.5/3); % weights
% w3=ones(1,Nodes)*(0.5/3); % weights
% w4=ones(1,Nodes)*0.5; % weights
% h=[1 5 6]; % penalty function

% the node longest from the hub gets 0, nearest gets 1, others something in
% between

if max(l)==min(l)
    alpha=0;
else
    alpha=1/(min(l)-max(l));
end
beta=1-alpha*min(l);
lengthrank=alpha.*l+beta;
%Ranks them according to inventory lasting
inv=xTEMP./c; %time inventory lasts
if max(inv)==min(inv)
    alpha=0;
else
    alpha=1/(min(inv)-max(inv));
end
beta=1-alpha*min(inv);
inventoryrank=alpha.*inv+beta;
%Ranks them according to inventory lasting
%time inventory lasts (Tero: Tässä oli ongelmana,
% että jos varasto on kaikkialla samalla tasolla niin jakajaan tulee nolla)
if max(c)==min(c)
    alpha=0;
else
```

```
    alpha=1/(max(c)-min(c));
end
beta=1-alpha*max(c);
consumptionrank=alpha.*c+beta;
%ranks the nodes by their penalty (Tero: Tässä oli ongelmana, että jos
%max(h)=min(h) niin jakajaan tulee nolla)
if max(h)== min(h)
    alpha=0;
else
    alpha=1/(max(h)-min(h));
end
beta=1-alpha*max(h);
penaltyrank=alpha.*h+beta;
%we have the priorityfunction
priorityvector=w1.*lengthrank+w2.*inventoryrank+w3.*consumptionrank+w4.*...
    penaltyrank;
return
```

Appendix C - Matlab code for distance program

```
function [l]=distance(L)
% this function returns the shortest path from the hub to the nodes
% we read a distance matrix l between the hubs and nodes
% defines the size of matrix l
[M,N] = size(L);

% http://www.mathworks.com/matlabcentral/fileexchange/14661-dijkstra-very-simple

%Changes all the distances over 999 to 100 (because dijkstra -function does not
%work with infinities
for m=1:1:M
    for n=1:1:N
        if L(m,n)>999
            L(m,n)=100;
        end
    end
end

%Calculates vector LTEMP
for i=1:1:N
    LTEMP(i)= dijkstra(L, 1, i);
end

%Changes all the values over 99 to infinities
for n=1:1:N
    if LTEMP(n)>99
        LTEMP(n)=inf;
    end
end

% Changes length vector so that it excludes the hub and creates output vector l
LTEMP(:,1)=[];
l=LTEMP;
return;
```

Appendix D - Matlab code for Dijkstra algorithm

```
function [l] = dijkstratesti(L, s, d)
% This is an implementation of the dijkstra´s algorithm, wich finds the
% minimal cost path between two nodes. It´s supoussed to solve the problem on
% possitive weighted instances.

% the inputs of the algorithm are:
%farthestNode: the farthest node to reach for each node after performing
% the routing;
% n: the number of nodes in the network;
% s: source node index;
% d: destination node index;

%For information about this algorithm visit:
%http://en.wikipedia.org/wiki/Dijkstra%27s_algorithm

%This implementatios is inspired by the Xiaodong Wang's implememtation of
%the dijkstra's algorithm, available at
%http://www.mathworks.com/matlabcentral/fileexchange
%file ID 5550

%Author: Jorge Ignacio Barrera Alviar. April/2007


n=size(L,1);
S(1:n) = 0;     %s, vector, set of visited vectors
dist(1:n) = inf;   % it stores the shortest distance between the source node and
any other node;
prev(1:n) = n+1;    % Previous node, informs about the best previous node known
to reach each  network node

dist(s) = 0;



while sum(S)~=n
    candidate=[];
    for i=1:n
        if S(i)==0
            candidate=[candidate dist(i)];
        else
            candidate=[candidate inf];
        end
    end
    [u_index u]=min(candidate);
    S(u)=1;
    for i=1:n
        if not(dist(u)+L(u,i)>100)

        if(dist(u)+L(u,i))<dist(i)
            dist(i)=dist(u)+L(u,i);
            prev(i)=u;
        end
        end
    end
end


sp = [d];
```

```
while sp(1) ~= s
    if prev(sp(1))<=n
        sp=[prev(sp(1)) sp];
    else
        error;
    end
end;
l = dist(d);
```

Appendix E – Matlab Code for breaks and fixes function

```
function [L] = BreaksAndFixes(j, L, L_orig, break_rounds, break_roads,
fix_rounds, fix_roads)
%Sizes of break/fix vectors
[M,~]=size(break_rounds);
[N,~]=size(fix_rounds);


%Breaking
for k=1:1:M
    if break_rounds(k)==j
        L(break_roads(k,1),break_roads(k,2))=inf; %Changes the distance between
nodes to infinity
        L(break_roads(k,2),break_roads(k,1))=inf; %Changes the distance between
nodes to infinity (opposite way)
    end
end

%Repairing
for k=1:1:N
    if fix_rounds(k)==j
        L(fix_roads(k,1),fix_roads(k,2))=L_orig(fix_roads(k,1),fix_roads(k,2));
%Returns the distance between nodes to original values
        L(fix_roads(k,2),fix_roads(k,1))=L_orig(fix_roads(k,2),fix_roads(k,1));
%Returns the distance between nodes to original values (opposite way)
    end
end
return
```

Appendix F – Brief outlook of literature on topic

The aim of this appendix is to present a brief outlook on the mathematical definition of networks, provide an example of what literature has presented in the field of humanitarian logistics network simulation as well as describe a real crisis logistics simulation from Haiti.

The **review provides only good-to-know** information on the topic but it does not anyhow help a reader to understand the simulation model presented in this report. Because of this, the literature review has been excluded from the main body of the report and presented as a standalone appendix.

General definitions of networks

The common definition of *network* is a physical or conceptual structure that includes a set of points that are connected to each other by a set of line segments (arcs). An example of a physical network is streets and intersections. A conceptual network can be a set of people and information lines between them. The links are often associated with directions of flow, and hence they are called flow networks. [1]

In mathematical terms flow networks are often called *directed graphs* (stemming from graph theory) which are defined as a set of N *nodes* and a set of A pairs of separate nodes, i.e. *arcs*. An arc (i,j) is defined to be outgoing from node i and incoming from node j (i.e. i is the starting point and j is the ending point for the flow). Directed graphs involve *paths* and *cycles*. Paths are sequences of nodes whereas cycles are special cases of paths where the starting and ending points are the same. Networks consist of *flows* (e.g. material flow in a logistics network) which are scalars that show the quantities flowing through the arcs. *Flow vectors* are sets of flows in a given network and *divergence* is a measure of the difference between the outgoing and incoming flows in a specific node. [2]

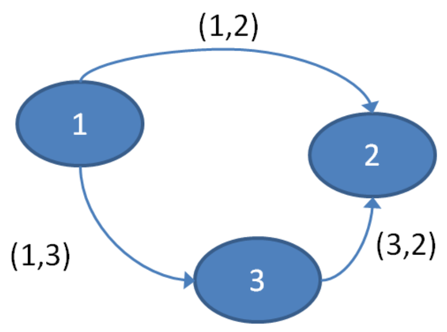


Figure 23: Example of an directed graphs with nodes 1, 2, 3 and arcs (1,2), (1,3), (3,2)

Networks can also be shown in bipartite graphs , which consist of two sets N1 and N2 of nodes without common elements so that each arc or edge in the graph has one end in N1 and one end in N2. That means that the graph has no arcs with both of the endpoints only in N1 or N2. Bipartite graphs can be used for solving assignment problems, for example matching workers in an optimal way to working tasks. [3]

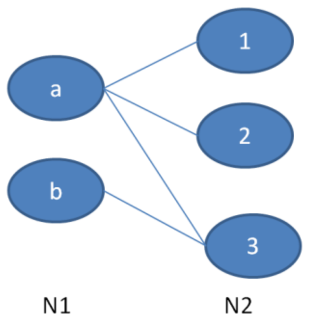


Figure 24: Simple example of bipartite graph where node a can serve nodes 1, 2, 3 and node b can serve node 3.

Humanitarian logistics network simulation – an example [4]

Humanitarian logistics network simulations have been rarely presented in academic journals. However, in a study Ekici, Keskinocak and Swann have simulated a facility location problem for humanitarian logistics in a situation of a pandemic influenza. Their objective was to determine locations for food delivery facilities by minimizing total costs and satisfying demand at different geographical locations.

Even though this model differs dramatically from our model, it can describe what kind of variables and classes can be used for model building in crisis logistics simulation. The time unit in this model is defined to be in number of weeks. The model consists of four different types of nodes: original supply nodes, major facility nodes, distribution center nodes and demand nodes. Demand nodes are placed at each census locations and the amount of demand is determined by number of individuals or households, and is deterministic. The supply nodes have a defined maximum amount of meals that they can send to distribution centers for processing and furthermore to distribution centers for distribution. The two latter have a maximum capacity constraint. The model consists also of distances between nodes, transportation costs per distance units, material handling costs and facility opening and closing costs.

Table 4: Notations used in the model (Source: [3], table 4)

T	number of weeks (time horizon)
N1	set of supply nodes
N2	set of major facility locations
N3	set of distribution centers
N4	set of demand nodes
Si	amount of meals that can be supplied by the supply node at node i for i ∈ N1
Fj	fixed cost incurred if the facility at node j is open during a week for j ∈ N2 UN3
f j	cost of opening the facility at node j for j ∈ N2 UN3
gj	cost of closing the facility at node j for j ∈ N2 UN3
c1o	unit material handling cost at a major facility
c2o	unit material handling cost at a distribution center
Cj	capacity of the facility that can be opened at node j for j ∈ N2 UN3
Dkt	demand of demand node k in period t for k ∈ N4, t ∈ T
di j	distance (in miles) between node i and node j for i ∈ Nk, j ∈ Nk+1, k ∈ {1,2,3}
c1u	unit transportation cost from a supply point to a major facility per mile
c2u	unit transportation cost from a major facility to a distribution center per mile
cindividual	unit transportation cost from a distribution center to a demand node per mile

[1] Sheffi, Y.: Urban transportation networks: equilibrium analysis with mathematical programming methods. Prentice-Hall, 1984 URL: http://web.mit.edu/sheffi/www/selectedMedia/sheffi_urban_trans_networks.pdf

[2] Bertsekas, D.P.: Network optimization: continuous and discrete methods, Athena Scientific, 1998

[3] Kreyszig, E: Advanced Engineering Mathematics 8th edition, John Wiley & Sons, 1999

[4] Ekici, A.; Keskinocak, P.; Swann, J.L.; , Pandemic influenza response, Simulation Conference, 2008. WSC 2008. Winter , vol., no., pp.1592-1600, 7-10 Dec. 2008, URL: http://ieeexplore.ieee.org.libproxy.tkk.fi/stamp/stamp.jsp?tp=&arnumber=4736242&isnumber=4736042

[5] Logcluster Cholera Operation: Haiti Cholera Response, Concept of Operations 29 October 2010, URL: http://www.logcluster.org/ops/hti10a/concept_of_operations_cholera_101029